GDE4: The Generalized Differential Evolution with Ordered Mutation

Azam Asilian Bidgoli[1,2(✉)], Sedigheh Mahdavi[2], Shahryar Rahnamayan[2], and Hessein Ebrahimpour-Komleh[1]

[1] Department of Computer and Electrical Engineering, University of Kashan, Kashan, Iran
ebrahimpour@kashanu.ac.ir

[2] Nature Inspired Computational Intelligence (NICI) Lab, Department of Electrical, Computer, and Software Engineering, University of Ontario Institute of Technology (UOIT), Oshawa, Canada
{azam.asilianbidgoli,sedigheh.mahdavi,shahryar.rahnamayan}@uoit.ca

Abstract. Differential Evolution (DE) is one the most popular evolutionary algorithm (EA) to handle optimization problems with an efficient performance. Due to its success and popularity, it has been utilized by researchers in multi-objective optimization, so there are various multi-objective versions of DE. Similar to other population-based algorithms, DE uses a mutation operator to produce the new individual for the next generation. Although the original version of DE randomly selects three candidate solutions from the population without considering any ordering in its mutation scheme, this paper proposes ordering strategy of individuals which influences the performance of the algorithm. An enhanced version (GDE4) of Generalized Differential Evolution (GDE) with ordered mutation operator is designed. GDE is a multi-objective evolutionary algorithm based on DE. The proposed approach orders candidate individuals using popular ranking measures of multi-objective optimization problems to utilize the ordered solutions in mutation operator. The best one of three randomly selected solutions is considered as the parent, and two others are applied as second and third candidate solutions in DE mutation, respectively. Unlike most of the multi-objective methods which consider multi-objectiveness during the selection process, the proposed method improves the performance using a modification on the genetic operator. The standard benchmark functions and measures are adopted to evaluate the performance of GDE4. The conducted experiments are on 5, 10, and 15 objectives for the utilized benchmark set. The comparison results reveal that GDE4 algorithm outperforms GDE3, the last version of GDE.

Keywords: Evolutionary computation · Multi-objective optimization · Generalized differential evolution · Ordered mutation

K. Deb et al. (Eds.): EMO 2019, LNCS 11411, pp. 101–113, 2019.
https://doi.org/10.1007/978-3-030-12598-1_9

1 Introduction

Since many real-world problems involve more than one objective, solving multi-objective optimization problems is considered as an important subject in many fields of science and engineering. The main issue that makes such problems harder than single objective problems is that how it is possible to compare solutions with two or more conflicting objectives. Evolutionary computation (EC) as a powerful method has been used to solve multi-objective optimization problems. There are a wide variety of single objective evolutionary algorithms (EA's) which have been adapted for multi-objective schemes [1,2]. Differential evolution (DE) is one of them which its simplicity offers a great characteristic to apply it in single- and multi-objective optimization. Generalized differential evolution (GDE) [3] is a multi-objective version of DE. There are some research works to improve GDE to be more successful in the optimization. The third version (GDE3) [4] was proposed to handle all types of multi-objective optimization problems including non-constrained and constrained ones.

Creation of a new individual in population-based algorithms, is one of the most important steps to make the generation more progressive. So selecting parents can influence producing better population and increasing elitism in the next generations. DE mutation which uses three randomly selected individuals from the population to create a new offspring. For single objective DE, there are some designed schemes of ordered mutation based on objective function value which have shown significant improvement in obtained results [5–7].

GDE3 also uses DE mutation operator, so ordered selection can improve the results of multi-objective optimization. The difficulty of ordered selection in multi-objective optimization case compared to the single objective one is the defining strategy of ranking of three selected individuals. Since there are two or more conflicting objective values, decision making in which candidate solutions are better, is sophisticated. This paper presents a version of GDE3 with ordered mutation (GDE4) for multi-objective optimization problems. Three selected candidate solutions are sorted based on two known measures, non-dominating sorting and crowding distance [8]. The best one is used as a base vector, and two other ranked candidate solutions are considered as the second and third individuals in DE mutation of GDE3. Since optimality doesn't have a straightforward definition, most of the multi-objective algorithms consider multi-objectiveness during their selection process. They concentrate on the proposing a method to rank candidate solutions while the proposed method improves multi-objectiveness in generative operator. Experiments show an enhancement of results in GDE4 compared to GDE3 in standard benchmarks. The organization of the rest of this paper is as follows. Section 2 gives a brief background review of GDE3 algorithms. Section 3 describes the proposed scheme in detail. Section 4 presents a simple algorithm and the experimental results to support discussion on the proposed scheme. Finally, the paper is concluded in Sect. 5.

2 Background Review

Many real-world optimization problems have more than one conflicting objectives to be optimized. The definition of the optimality is not as simple as the single-objective optimization. Therefore it is necessary to make a tradeoff between objective values. There are some well-known concepts to compare two candidate solutions in the multi-objective problem space. Since this paper utilizes non-dominated sorting and crowding distance to order candidate solutions for DE mutation scheme, in this section, we define these measures in detail. A minimization multi-objective optimization problem is defined as follows:

$$Minimize\ F(x) = [f_1(x), f_2(x), ..., f_M(x)] \quad L_i \leq x_i \leq U_i, i = 1, 2, ..., d \qquad (1)$$

where M is the number of objectives, d is the number of variables (dimension) of solution vector, x_i is in interval $[L_i, U_i]$. f_i represent the objective function which should be minimized.

If $x = (x_1, x_2, ..., x_d)$ and $\acute{x} = (\acute{x}_1, \acute{x}_2, ..., \acute{x}_d)$ are two vectors in search space, x dominates $\acute{x}$ ($x \succ \acute{x}$) if and only if:

$$\forall i \in \{1, 2, ..., d\}, f(x_i) \leq f(\acute{x}_i) \wedge \exists i \in \{1, 2, ..., d\} : f(x_i) < f(\acute{x}_i) \qquad (2)$$

It defines optimality for solutions in objective space. Candidate solution x is better than $\acute{x}$ if it is not bigger than $\acute{x}$ in any of objectives and at least it has a smaller value in one of the objectives. All solutions that are not dominated using none of other solutions in the population called non-dominated solutions and they create the Pareto front set.

Non-dominated sorting is an algorithm to rank obtained solutions to different levels in the processing of multi-objective optimization. All non-dominated solutions are in the first rank and then the second rank is made of solutions which are non-dominated by removing the first rank from the population. This process is repeated until all solutions are ranked using this concept.

Crowding distance is another measure which usually completes comparison of solutions along with non-dominating sorting. It is a measure to compute the diversity of obtained solutions by calculating the distance between adjacent solutions. In the beginning, the set of solutions in the same rank are sorted according to each objective function value in ascending order. To get crowding distance, the difference between neighbors objective values of each solution is computed. This computation is done for all objectives, then the sum of individual distance values corresponding to each objective is considered as overall crowding distance. The bigger value of crowding distance for a vector in population shows less diversity around that vector.

2.1 Generalized Differential Evolution

The DE is an evolutionary algorithm originally for solving continuous optimization problems which improves initial population using the crossover and mutation operations. Creation of new generation is done by a mutation and a crossover

operator. The mutation operator for a gene, j, is defined as follows:

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

Applying this operator generates a new D dimensional vector, v_i, using three randomly selected individuals, x_{j,i_1}, x_{j,i_2}, and x_{j,i_2} from the current population. Parameter F, mutation factor, scales difference between two vectors. The crossover operator changes some or all of the genes of parent solution based on Crossover Rate (CR). Similar to other population-based algorithms, the single objective version of DE starts with a uniform randomly generated population. Next generation is created using mentioned mutation and crossover operations; then best individual (between parent and new individual) is selected based on their objective values; which is called a greedy selection. It iterates until meeting stopping criterion such as a predefined number of generations.

There are also several variants of DE algorithms for multi-objective optimization. The first version of Generalized Differential Evolution (GDE) [3] changed the DE selection mechanism for producing the next generation. The idea in the selection was based on constraint-domination. The new vector is selected if it dominates the old vector. GDE2 [9], the next version of multi-objective DE algorithm, added the crowding distance measure to its selection scheme. If both vectors are non-dominating each other, the vector with a higher crowding distance will be selected.

The third version of GDE (GDE3) extends DE algorithm for multi-objective optimization problems with M objectives and K constraints. DE operators are applied using three randomly selected vectors to produce an offspring per parent in each generation. The selection strategy is similar to the GDE2 except in two parts: 1. Applying constraints during selection process. 2. The non-dominating case of two candidate solutions. Selection rules in GDE3 are as follows: when old and new vectors are infeasible solutions, each solution that dominates other in constraint violation space is selected. In the case that one of them is feasible vector, feasible vector is selected. If both vectors are feasible, then one is selected for the next generation that dominates other. In non-dominating case, both vectors are selected. Therefore, the size of the population generated may be larger than the population of the previous generation. If this is the case, it is then decreased back to the original size. Selection strategy for this step is similar to NSGA-II algorithm [10]; it sorts individuals in the population, based on the non-dominated sorting algorithm and crowding distance measure. Similar to other population-based multi-objective algorithms, the selected individuals go to the next generation to continue optimization processing. The common point about all of these versions is the utilizing randomly selected individuals to produce a new vector using the main mutation operator of DE which will be modified in our proposed algorithm in this paper. So, even the mutation scheme would be tailored to support multi-objective optimization strategy.

2.2 Existing Single Objective Differential Evolution with Ordered Mutation

In some versions of DE algorithm, the ordering of the candidate solutions is utilized for the mutation operator to enhance the performance of DE algorithm for solving the single objective optimization problems. A new scheme of mutation operator, DE/2-Opt, was defined in [5] which sorts two first candidate solutions in the mutation operator according (for minimization case) 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':

$$v_i = \begin{cases} x_{i_1} + F \,.\, (x_{i_2} - x_{i_3}) & \text{if} \quad f(x_{i_1}) <= f(x_{i_2}) \\ x_{i_2} + F \,.\, (x_{i_1} - x_{i_3}) & \text{if} \quad f(x_{i_2}) < f(x_{i_1}) \end{cases} \tag{4}$$

'DE/2-Opt/2':

$$v_i = \begin{cases} x_{i_1} + F \,.\, (x_{i_2} - x_{i_3} + x_{i_4} - x_{i_5}) & \text{if} \quad f(x_{i_1}) <= f(x_{i_2}) \\ x_{i_2} + F \,.\, (x_{i_1} - x_{i_3} + x_{i_4} - x_{i_5}) & \text{if} \quad f(x_{i_2}) < f(x_{i_1}) \end{cases} \tag{5}$$

In [6], the winner mutation (DE/win) was proposed which uses the best candidate of three selected random candidate solutions for the base vector as follows:
'DE/win/1':

$$v_i = \begin{cases} x_{i_1} + F \,.\, (x_{i_2} - x_{i_3}) & \text{if} \quad f(x_{i_1}) <= f(x_{i_2}), f(x_{i_3}) \\ x_{i_2} + F \,.\, (x_{i_1} - x_{i_3}) & \text{if} \quad f(x_{i_2}) < f(x_{i_1}), f(x_{i_3}) \\ x_{i_3} + F \,.\, (x_{i_2} - x_{i_1}) & \text{if} \quad f(x_{i_3}) < f(x_{i_2}), f(x_{i_1}) \end{cases} \tag{6}$$

In [7], a modified DE algorithm with the order mutation scheme was proposed which 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 operator.
'DE/order/1':

$$v_i = x_{i_1} + F \,.\, (x_{i_2} - x_{i_3}) s.t. \;\; f(x_{i_1}) <= f(x_{i_2}) <= f(x_{i_3}) \tag{7}$$

Where $f(x)$ indicates the objective function. This method outperforms previous mentioned DE schemes.

3 Proposed Algorithm: The Generalized Differential Evolution with the Ordered Mutation (GDE4)

The proposed enhanced version of GDE3 method has the same components of GDE3 method except for the mutation operator in the DE algorithm. In this paper, a new mutation scheme is proposed according to a defined order for the candidate solutions involved in the mutation of the DE algorithm. GDE3 uses the

DE/rand/1/bin method to solve problems with M objectives and K constraints. The basic mutation, in the classical DE (DE/rand/1/bin) generates the mutant vector as a linear combination of three selected individual candidate solutions from the current population as follows:

$$v_i = x_{i_1} + F \,.\, (x_{i_2} - x_{i_3}) \tag{8}$$

Where i_1, i_2, i_3 are different random integer numbers within $[1, NP]$ and NP is the population size. In [7], an ordered mutation scheme was proposed to improve the performance of DE algorithm and we change this mutation scheme by defining a new order of the randomly selected candidate solutions for the problems with M objectives. In the GDE4, we propose an order mutation scheme which uses non-dominance and crowding distance measures to sort three different random candidate solutions to set as vectors in the mutation scheme. The sorted candidate solutions can be called as the best (x_b), the second best (x_{sb}), and the worst candidate (x_w) solutions.

In the following, we explain how the three randomly selected candidate solutions are sorted. First, all candidate solutions are sorted by non-dominated sorting method [10] and they are associated with their corresponding non-dominated ranks ($Rank_d$) obtained from non-dominated sorting. Random candidate solutions can be faced with four possible cases based on their non-dominance ranks:

1. In the first case, all three candidate solutions are in different Pareto fronts; therefore, they are set to x_b, x_{sb}, and x_w to their non-dominated ranks. The ordered mutation scheme ($DE/order/1$) is defined as follows: 'DE/order/1':

$$v_i = x_{i_b} + F \,.\, (x_{i_{sb}} - x_{i_w})$$
$$s.t.\ Rank_d(x_{i_b}) < Rank_d(x_{i_{sb}}) < Rank_d(x_{i_w})$$

2. In this case, two candidate solutions are in the same Pareto front, so we compute crowding distance (CD) measure to sort these solutions. The ordered mutation scheme ($DE/order/1$) is defined as two possible cases:
   (a) 'DE/order/1':

$$v_i = x_{i_b} + F \,.\, (x_{i_{sb}} - x_{i_w})$$
$$s.t.\ Rank_d(x_{i_b}) = Rank_d(x_{i_{sb}}) < Rank_d(x_{i_w})$$
$$CD(x_{i_b}) > CD(x_{i_{sb}})$$

   (b) 'DE/order/1':

$$v_i = x_{i_b} + F \,.\, (x_{i_{sb}} - x_{i_w})$$
$$s.t.\ Rank_d(x_{i_b}) < Rank_d(x_{i_w}) = Rank_d(x_{i_{sb}})$$
$$and CD(x_{i_{sb}}) > CD(x_{i_w})$$

3. If all three random candidate solutions are in the same Pareto front, they are sorted based on their crowding distance (CD) to place in the mutation scheme. The ordered mutation scheme ($DE/order/1$) is defined as follows: 'DE/order/1':

$$v_i = x_{i_b} + F\,.\,(x_{i_{sb}} - x_{i_w})$$
$$s.t.\ CD(x_{i_b}) > CD(x_{i_{sb}}) > CD(x_{i_w})$$

The proposed method uses the order mutation scheme for DE algorithm in GDE3, and other components remain untouched. Generalized Differential Evolution with the ordered mutation (GDE4) suggests that placing the best solution of three selected candidate solutions according to two measures, non-dominance and crowding distance, as the base vector causes to generate more promising trial solutions. Also, we use the worst candidate solution of three candidate solutions as the third vector in the mutation which causes the new trial candidate solution to get away from the worst candidate and move toward the second best candidate solution.

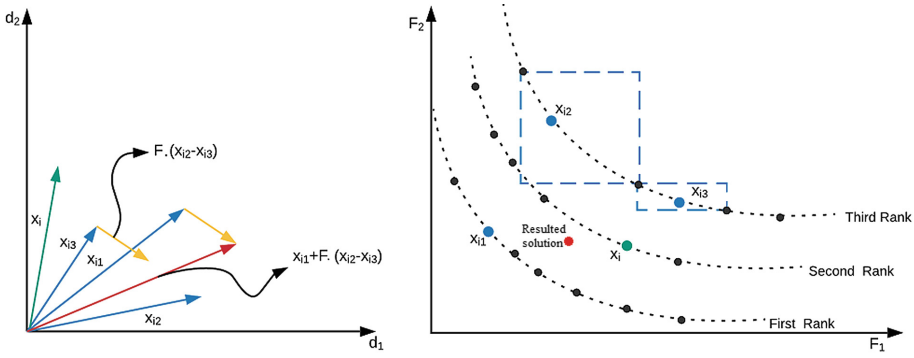

Fig. 1. An example of variable and objective spaces for ordered DE mutation.

Figure 1 presents variable and objective spaces in a case of ordered mutation and clarifies the benefits of this strategy in creating a promising new solution. As it is shown, for a parent solution, x_i, three randomly selected candidate solutions are ordered based on the proposed strategy in GDE4 algorithm. In this case, the first candidate solution, x_{i1} is in the first rank of non-dominated sorting, so it is considered as the base vector (best). x_{i2} and x_{i3} are in the same rank therefore they ordered according to crowding distance. x_{i2} has a bigger crowding distance comparing to x_{i3}, so they are ordered as second (better) and third vector (worst) in mutation operator. Right sub-figure in Fig. 1 shows the operation of mutation on selected vectors. $F\,.\,(x_{i2} - x_{i3})$ leads new vector moves toward to better solution and gets away from the worst while F is considered 1. In this example, better solution is one with a bigger crowding distance. Moving

toward this solution causes the creation of a vector in a less crowded region to have a well-distributed Pareto front. Then summation operation on x_{i1} and $F.(x_{i2} - x_{i3})$ causes the final resulted vector goes toward the best candidate solution. So it is expected to generate a more promising candidate solution.

Table 1. Main properties of the test functions [11].

Problem	Properties
MaF1	Linear
MaF2	Concave
MaF3	Convex, multimodal
MaF4	Concave, multimodal
MaF5	Convex, biased
MaF6	Concave, degenerate
MaF7	Mixed, disconnected, multimodal
MaF8	Linear, degenerate
MaF9	Linear, degenerate
MaF10	Mixed, biased
MaF11	Convex, disconnected, nonseparable
MaF12	Concave, nonseparable, biased, deceptive
MaF13	Concave, unimodal, nonseparable, degenerate
MaF14	Linear, partially separable, large scale
MaF15	Convex, partially separable, large scale

4 Experiment

GDE4 is evaluated with a set of test problems and compared to GDE3 regarding multi-objective evaluation measures. The same settings are considered for two algorithms. The mutation amplification factor (F) and crossover rate (CR) are set to 0.5 and 1, respectively. For population size and maximum evaluation number, value 100 and $3000 * D$ are considered. To evaluate the performance of the proposed algorithm, we use the inverse generational distance (IGD) metric [12–14], which measures the convergence and the diversity of the obtained Pareto-optimal solutions at the same time. The IGD metric measures the distances between each solution composing the Pareto-optimal front and the obtained solution. The IGD metric is defined as follow:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i}}{n} \tag{9}$$

Where n is the number of solutions in the Pareto-optimal front, and d_i is the Euclidean distance (measured in the objective space) between each point of the Pareto-optimal front (reference Pareto front) and the nearest member of obtained solution. Also, all algorithms were executed 51 times independently, and the best, the worst, the median, and the average results of each algorithm are reported. Additionally, the Wilcoxon's signed rank statistical test with a confidence interval of 95% is conducted to evaluate the statistical significance of the obtained results. We have utilized GDE3 algorithm in the MATLAB based MOEA platform (PlatEMO) [15] and it was modified by changing its mutation operator to the order mutation as explained for the GDE4.

In the experiments, fifteen test problems are used to evaluate the performance of the proposed algorithm from the MaF test suite which is designed for the assessment of MOEAs in the CEC 2017 competition on evolutionary many-objective optimization [11]. These benchmark functions have many properties to resemble various real-world scenarios such as multi-modal, disconnected, degenerate, and/or nonseparable, and having an irregular Pareto front shape, a complex Pareto set or a large number of decision variables. The main properties of functions are detailed in Table 1. Experiments are performed on 5, 10 and 15 objective functions.

Figure 2 illustrates the distribution of obtained solutions by GDE4 and GDE3 for MFa11 test problem in different number of objectives. The diagrams are resulted based on median value of IGD. As the figure shows both algorithms are able to find distributed solutions with same performance when the number of objectives is 5. However, as the number of objectives of the test problem increases, GDE4 performs significantly better to find well-distributed solutions. The difference between diversity of obtained solutions using GDE3 and GFE4 is more remarkable with 15 objectives.

The results of IGD metric for two comparing methods are summarized in Table 2. Better mean of IGDs are highlighted based on Wilcoxon's signed rank statistical test. It can be seen from the tables, on functions with five objectives, GDE4 can achieve the better results than GDE3 on seven functions while GDE3 is better than GDE4 on five functions, and they are similar results on three functions. On functions with ten objectives, GDE4 can achieve the better results than GDE3 on ten functions while GDE3 can obtain better results than GDE4 on three functions; and they are similar results on two functions. On functions with fifteen objectives, GDE4 outperforms GDE3 on nine functions while GDE3 can obtain better results than GDE4 on five functions; and they are similar results on one functions. Results show that by increasing the number of objectives in the many-objective functions, GDE4 preforms significantly better than GDE3 regarding statistical test. Furthermore, comparing results according to median and best IGD confirms better performance of GDE4. Median IGDs of GDE4 are better in 9, 11 and 9 out of 15 functions for 5, 10, and 15 objective problems respectively comparing to GDE3.

According to best IGDs, GDE4 achieve better results in 7, 9, and 8 out of 15 functions for 5, 10, and 15 objective problems respectively. So the order of

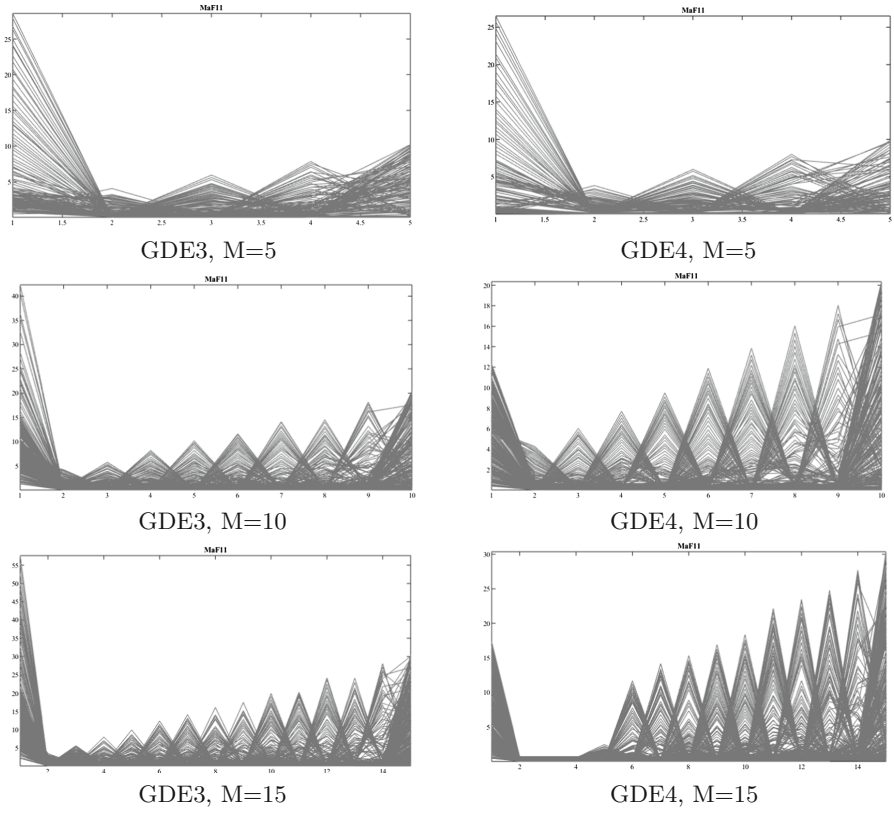

Fig. 2. Comparison of obtained Pareto fronts by GDE3 and GDE4 for MFa11 test problem in different dimensions.

solutions in DE mutation operator improves the search processing in many-objective optimization problems using generating better (non-dominated) solutions. The generated solution is expected to create in place close to the best solution in term of the rank of non-dominated sorting and the less crowded region. As another advantage of the proposed method, it can be clarified that this improvement is achieved without any extra objective function evaluation. The method needs only ordering of three existing solutions, so there isn't overhead computation for applying mutation comparing to previous version.

Table 2. Results of GDE3 and GDE4 algorithms for the functions MaF_1-MaF_{15}. The highlighted entries are significantly better.

Function		#Objectives=5		#Objectives=10		#Objectives=15	
		GDE3	GDE4	GDE3	GDE4	GDE3	GDE4
MaF_1	Mean	0.2052	**0.1696**	0.3529	**0.3015**	0.3718	**0.3220**
	Median	0.2049	0.1700	0.3542	0.3006	0.3676	0.3207
	Worst	0.2244	0.1843	0.3687	0.3096	0.4312	0.3489
	Best	0.1923	0.1570	0.3346	0.2894	0.3510	0.3038
$MaF2$	Mean	**0.1502**	**0.1414**	**0.1691**	0.1717	0.1960	**0.1639**
	Median	0.1448	0.1393	0.1691	0.1718	0.1962	0.1638
	Worst	0.2016	0.2059	0.1783	0.1802	0.2159	0.1698
	Best	0.0983	0.1006	0.1619	0.1630	0.1774	0.1580
$MaF3$	Mean	2.9848e+4	**1.1454e+4**	8.9508e+4	**4.7488e+4**	6.3820e+6	**7.6529e+4**
	Median	3.0976e+4	9.6593e+3	6.5095e+04	3.8852e+4	1.2809e+5	6.0298e+4
	Worst	5.1531e+4	2.9545e+4	2.6107e+5	1.2766e+5	1.9288e+8	2.5940e+5
	Best	303.1595	4.0487e+3	3.3775e+4	1.2146e+4	5.5970e+4	9.7001e+3
MaF_4	Mean	**185.6015**	**154.0321**	1.8330e+4	**6.5089e+3**	6.1292e+5	**1.7718e+5**
	Median	132.2845	157.4819	1.7994e+4	5.7372e+3	5.6660e+5	1.6946e+5
	Worst	558.9949	272.2185	3.8579e+4	1.6750e+4	1.2662e+6	4.3676e+5
	Best	2.8850	56.2430	73.5666	1.3418e+3	5.1928e+3	4.2660e+4
MaF_5	Mean	3.4354	**2.4941**	81.7945	**52.6467**	1.6891e+3	**1.2131e+3**
	Median	3.4658	2.5029	78.1735	52.7151	1.7065e+3	1.2137e+3
	Worst	4.1599	2.9408	130.9308	66.7699	2.2733e+3	1.6161e+3
	Best	2.9288	2.0434	58.0378	43.6443	1.3950e+3	1.0154e+3
MaF_6	Mean	0.0043	**0.0042**	0.5219	**0.2241**	**0.3858**	**0.3430**
	Median	0.0043	0.0041	0.4389	0.2496	0.3418	0.3425
	Worst	0.0045	0.0049	1.2602	0.3195	0.7446	0.3474
	Best	0.0039	0.0038	0.3101	0.0025	0.3415	0.3415
MaF_7	Mean	0.5699	**0.4674**	1.8627	**1.6517**	**2.0686**	3.2783
	Median	0.5701	0.4653	1.8428	1.5986	2.0761	3.0007
	Worst	0.6658	0.5433	2.0059	2.4190	2.1438	5.5710
	Best	0.4885	0.3954	1.7207	1.4164	1.9249	2.1578
MaF_8	Mean	**0.1352**	0.5757	**0.1420**	1.3324	**0.1414**	2.5921
	Median	0.1342	0.5621	0.1417	1.2636	0.1417	2.2632
	Worst	0.1618	0.9057	0.1505	2.5778	0.1463	7.5285
	Best	0.1214	0.3688	0.1363	0.7950	0.1349	1.2687
MaF_9	Mean	**0.7077**	1.1437	**64.3750**	**53.1231**	**0.8668**	9.3834
	Median	0.7029	1.1106	46.9070	45.8173	0.8606	12.0917
	Worst	0.7417	1.7796	173.4796	155.0931	1.0069	15.3572
	Best	0.6873	0.8050	12.3038	2.4595	0.7851	1.7868
MaF_{10}	Mean	2.3053	**1.9520**	4.0770	**3.0795**	4.8911	**4.0115**
	Median	2.2896	1.9442	4.1003	3.0895	4.9108	4.0144
	Worst	2.5162	2.0167	4.2786	3.1721	5.0849	4.1371
	Best	2.1792	1.9116	3.7678	3.0070	4.7297	3.8524
MaF_{11}	Mean	0.9947	**0.6098**	1.6504	**0.8708**	1.9490	**1.4747**
	Median	0.9777	0.5826	1.7212	1.0780	2.2806	1.8268
	Worst	1.1922	0.9363	2.2222	1.6269	3.0181	2.2427
	Best	0.8443	0.4961	0.5462	0.1776	0.7014	0.2508
MaF_{12}	Mean	**1.5934**	1.6983	5.7623	**5.4441**	8.6395	**7.7793**
	Median	1.5959	1.7319	5.7621	5.4454	8.6360	7.8252
	Worst	1.7707	1.8636	5.8966	5.6727	8.9330	8.0114
	Best	1.4322	1.5049	5.5960	4.9936	8.3844	7.3945
MaF_{13}	Mean	0.1869	**0.1209**	0.1232	**0.1071**	0.1045	**0.0953**
	Median	0.1748	0.1219	0.1204	0.1063	0.1017	0.0945
	Worst	0.2502	0.1308	0.1647	0.1235	0.1500	0.1129
	Best	0.1372	0.1045	0.1089	0.0928	0.0921	0.0845
MaF_{14}	Mean	**0.9794**	25.9004	**8.0794**	25.5172	**3.1449**	41.3429
	Median	0.9796	28.4090	8.3299	23.0875	1.0996	39.7949
	Worst	0.9796	45.2110	18.2883	48.8730	12.0864	59.5061
	Best	0.9774	8.0826	1.9220	11.4911	1.0963	27.2051
MaF_{15}	Mean	**9.9889**	**11.2497**	**56.4160**	**53.1113**	**50.6759**	72.4516
	Median	9.2002	11.2324	49.1652	51.9795	53.6135	72.7500
	Worst	15.1699	16.8106	108.9222	73.1437	90.7196	84.7248
	Best	6.5229	6.6440	32.3527	40.5497	23.5498	59.5537

5 Conclusion Remarks

This paper proposes GDE4, a new version of Generalized Differential Evolution algorithm for multi-objective optimization problems. The ordering of randomly selected candidate solutions for DE mutation operator is investigated. Method sorts three solutions at first, based on non-dominated sorting approach and then crowding distance measure to utilize as first, second and best solutions in DE mutation to generate a new individual exhibiting better fitness. DE summation and subtraction operators cause moving of new solution toward the first and

second vectors and getting away from the third vector. So ordered vectors has inherited the quality of best and better candidate solutions. The performance of the method is evaluated using standard benchmark functions of CEC 2017 competition on evolutionary many-objective optimization problems. The results indicate that the proposed algorithm outperforms GDE3 which puts solutions in mutation operator randomly in most test problems. In the future, it is intended to investigate new strategies to order candidate solutions, such as the distance of each vector from an ideal point.

References

1. Ali, M., Siarry, P., Pant, M.: An efficient differential evolution based algorithm for solving multi-objective optimization problems. Eur. J. Oper. Res. **217**(2), 404–416 (2012)
2. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer, M., et al. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45356-3_83
3. Lampinen, J.: DEs selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology, pp. 03–04 (2001)
4. Kukkonen, S., Lampinen, J.: GDE3: the third evolution step of generalized differential evolution. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 443–450. IEEE (2005)
5. Chiang, C.W., Lee, W.P., Heh, J.S.: A 2-opt based differential evolution for global optimization. Appl. Soft Comput. **10**(4), 1200–1207 (2010)
6. Yeh, M.F., Lu, H.C., Chen, T.H., Huang, P.J.: System identification using differential evolution with winner mutation strategy. In: 2014 International Conference on Machine Learning and Cybernetics (ICMLC), vol. 1, pp. 77–81. IEEE (2014)
7. Mahdavi, S., Rahnamayan, S., Karia, C.: Analyzing effects of ordering vectors in mutation schemes on performance of differential evolution. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2290–2298 (2017). https://doi.org/10.1109/CEC.2017.7969582
8. Seada, H., Deb, K.: Non-dominated sorting based multi/many-objective optimization: two decades of research and application. In: Mandal, J.K., Mukhopadhyay, S., Dutta, P. (eds.) Multi-Objective Optimization, pp. 1–24. Springer, Singapore (2018). https://doi.org/10.1007/978-981-13-1471-1_1
9. Kukkonen, S., Lampinen, J.: An extension of generalized differential evolution for multi-objective optimization with constraints. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 752–761. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30217-9_76
10. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
11. Cheng, R., et al.: A benchmark test suite for evolutionary many-objective optimization. Complex Intell. Syst. **3**(1), 67–81 (2017)
12. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: 1996 Proceedings of IEEE International Conference on Evolutionary Computation, pp. 312–317. IEEE (1996)

13. Iorio, A.W., Li, X.: Solving rotated multi-objective optimization problems using differential evolution. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 861–872. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30549-1_74
14. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014)
15. Tian, Y., Cheng, R., Zhang, X., Jin, Y.: PlatEMO: a MATLAB platform for evolutionary multi-objective optimization [educational forum]. IEEE Comput. Intell. Mag. **12**(4), 73–87 (2017)