COMPARISON OF A NOVEL DOMINANCE-BASED DIFFERENTIAL EVOLUTION METHOD WITH THE STATE-OF-THE-ART METHODS FOR SOLVING MULTI-OBJECTIVE REAL-VALUED OPTIMIZATION PROBLEMS

Mustafa Tuncay
Department of Computer Engineering
mustafa.tuncay@std.gau.edu.tr

Ali Haydar
Department of Computer Engineering
ahaydar@gau.edu.tr

1Girne American University
Mersin 10 Turkey, Girne, North Cyprus, 99428

Abstract
Differential Evolution algorithm (DE) is a well-known nature-inspired method in evolutionary computations scope. This paper adds some new features to DE algorithm and proposes a novel method focusing on ranking technique. The proposed method is named as Dominance-Based Differential Evolution, called DBDE from this point on, which is the improved version of the standard DE algorithm. The suggested DBDE applies some changes on the selection operator of the Differential Evolution (DE) algorithm and modifies the crossover and initialization phases to improve the performance of DE. The dominance ranks are used in the selection phase of DBDE to be capable of selecting higher quality solutions. A dominance-rank for solution X is the number of solutions dominating X. Moreover, some vectors called target vectors are used through the selection process. Effectiveness and performance of the proposed DBDE method is experimentally evaluated using six well-known benchmarks, provided by CEC2009, plus two additional test problems namely Kursawe and Fonseca & Fleming. The evaluation process emphasizes on specific bi-objective real-valued optimization problems reported in literature.

Likewise, the Inverted Generational Distance (IGD) metric is calculated for the obtained results to measure the performance of algorithms. To follow up the evaluation rules obeyed by all state-of-the-art methods, the fitness evaluation function is called 300,000 times and 30 independent runs of DBDE is carried out. Analysis of the obtained results indicates that the performance of the proposed algorithm (DBDE) in terms of convergence and robustness outperforms the majority of state-of-the-art methods reported in the literature.

Keywords: Differential Evolution, Multi-objective Problems, Multi-objective Algorithms, Optimization, Inverted Generational Distance.

DOI: 10.21303/2461-4262.2021.001857

1. Introduction
Nature-inspired multi-objective optimization (MOO) algorithms have been extensively used to solve complex MOO problems. The process of simultaneous optimization of a collection of objective functions is called MOO or vector optimization [1]. MOO is being widely studied in many areas of science and engineering applications. Naturally, the MOO problems exist in most disciplines and their solutions have been a significant challenge for researchers [2].

Many algorithms have been developed for solving single- and multi-objective optimization problems during recent decades. The multi-objective algorithms are being complicated and robust at the same time to be dealing with the NP-hard problems. To efficiently solve a multi-objective optimization problem, selecting an appropriate and convenient algorithm for the given problem is a must. Therefore all recently proposed algorithms should be taken into account. To measure the quality of the obtained outputs and extracted Pareto-front, the IGD values are calculated.
The main goal of MOO is to find the best solution set (Pareto-front) in the solution space by taking all objectives into account. Moreover, MOO is a common research area for engineers, in majority of engineering applications. Differential Evolution (DE) algorithm is successful in solving both single-objective and multi-objective optimization problems. Even though there are many MOO algorithms to solve MOO problems, DE is fast and effective. Many researchers have been working on DE algorithm to improve the performance and find the better non-dominated solutions or optimal Pareto-Front.

The optimal Pareto-front for a minimization problem consists of two objective functions F1 and F2, represented in Fig. 1.

DE algorithms are developed as a randomly-determined direct search method. Phase of DE algorithms are inspired by the real nature including Initialization phase alongside Mutation, Crossover and Selection operators. According to the literature, the most preferred improved algorithms are DE/rand/1 and DE/rand/2 mutation strategies [3]. In this paper, DE/rand/1 strategy is used which is a fast and effective way to reach the feasible solutions. Most of the improved DE algorithms employ many techniques to find the optimal solution. Apart from the fact that improvements on DE increase the performance of the algorithm, they also increase the complexity of the algorithm.

Compared to other methods, the proposed algorithm is simple in terms of implementation and is efficient in terms of the quality of extracted solutions. The suggested algorithm makes a significant change over the selection operator based on the dominance rank values. Likewise, the algorithm applies an improved version of target vector selection mechanism in which they are selected through the vectors of the mutation operator. The suggested selection method improves the algorithm in terms of speed and quality of solution. This paper contributes the following aspects:

– the paper proposes a dominance rank method which is a beneficial mechanism to deal with many problem types;
– in the proposed method, no additional target vector is needed to compare with the trial vector. However, it applies the same vectors with ones selected for the mutation operator.

The suggested algorithm is quite appropriate to be applied on multi-objective optimization problems. In evaluation section, the effectiveness of the proposed algorithm is tested over two-objective optimization problems. However, the evaluation of the algorithm over three- or many-objective problems is considered as future works.

In this work, a modified DE algorithm, called as DBDE, is proposed and its performance is compared with the well-known algorithms in the literature. In the following section of this paper, the structure of the DE algorithm is first briefly stated. Thereafter, the proposed algorithm is described in detail. Simulation results of DBDE are compared with those of other MOO algorithms and the graphical representation of the solutions of DBDE algorithm is highlighted. The obtained results are compared based on IGD values. It should be noticed that to calculate the IGD values, the optimal Pareto-fronts of the problems must be known. Finally, the deductions are summarized in the conclusions.
2. Materials and Methods

2.1. Differential evolution algorithm

DE algorithm was introduced in 1995 as a vector population-based stochastic optimization approach [4]. DE is an evolutionary programming method developed by Kenneth Price and Rainer Storm [5] for solving optimization problems over continuous domains, where each variable is represented by a real number. DE algorithm has a simple structure and can be implemented very easily. Likewise, DE is one of the best structures for using real variables in which they can be used in many different fields of science. Different aspects of DE have been widely used for solving many interactive problems without requiring too much expert knowledge. The DE algorithm has four operators:

2.1.1. Initialization operator

An initial population must be generated randomly to represent the initial solution space. Before initialization of the population, the upper and lower bounds for each parameter must be specified [5].

2.1.2. Mutation operator

A mutation operator is required to increase the diversity of solutions and probability of finding a correct solution [6]. In this phase, three different random vectors are selected from the population.

Let’s assume that \( R_1, R_2, \) and \( R_3 \) are random numbers, \( R_1 \neq R_2 \neq R_3 \). Random numbers are randomly selected from the population, \( R_1, R_2, R_3 \in [1 - NP] \). \( NP \) represents the number of population.

The formula is shown below, used for mutating the vectors,

\[
V = P(R_1) + F \cdot (P(R_2) - P(R_3)).
\]  

\( P(R_1) \) is the \( R_1 \)’th solution in the population \( P \), \( F \) is the constant factor between 0 and 2. The mutation creates a mutant vector \( (V[x]) \).

2.1.3. Crossover operator

Crossover operator is implemented to increase the variety of disrupted vectors. Crossover Rate (CR) value is specified by the user, which controls the fraction of the parameter values copied from the mutant vector \( (V[x]) \). CR can be between 0 and 1. RandX is a random value, used to distribute parameters randomly to create a trial vector \( (U[x]) \). If \( RandX \) is less than or equal to CR value, the trial parameter is copied from the mutant vector \( (V[x]) \), otherwise, the parameter is copied from the target vector \( (Matrix\ A[i][x]) \). \( i \) represents the \( i \)th index of the population and it is different from \( R_1, R_2 \) and \( R_3 \). The implementation of the Crossover operator is represented in Fig. 2.

```c
for (x = 0; x <= Dimension - 1; ++x)
{
    RandX = randfrom(0,1); jrand = rand(0, Dimension)
    if(CRValue >= RandX || x==jrand) // CRValue is crossover value.
        U[x] = V[x];
    else U[x] = MatrixA[i][x];
}
```

Fig. 2. Implementation of Crossover operator

2.1.4. Selection operator

This operator checks if the target vector is replaced by the trial vector or not. If the fitness value of the trial vector is smaller than the target vector, then the target vector is equated to the trial vector. In this operator, the target vector is the fourth randomly selected vector, which is \( R_4 \).
2.2. The proposed dominance-based differential evolution (DBDE) method

In DBDE, the Initialization, Crossover, and Selection operators are modified to improve the performance of the optimization while for the mutation operator the standard methods are applied. The proposed method starts with the initialization phase in which a population of size 1000 is randomly generated. Afterwards, from the randomly generated population, the best 100 solutions are selected as the first population-based on dominance rank values.

In Fig. 3 the dominance-rank values are illustrated for four solutions.

In this way, the first population most likely consists of the high-quality and more promising solutions. It can be said that the lower dominance rank values correspond to better solutions and the solutions with dominance-rank equal to zero are the best solutions.

The Crossover Rate (CR) value as a control parameter and mutation scale factor (FValue) value are selected between 0 and 1. Meanwhile, the $R_1$, $R_2$, and $R_3$ values are assumed to be the randomly chosen integer numbers between 1 and population size (100). For the mutation operator, the following formula is applied:

$$V = P[R_1] + F \cdot (P[R_2] - P[R_3]),$$

where $P[R_1]$ is the target vector.

The crossover and mutation operators are repeated until a trial vector different from the target vector is obtained. It is found that some of the trial vectors do not change after the process because when the value of the element becomes greater than upper bound or smaller than lower bound, the value of the element of the trial vector is kept as is. To avoid loss of iteration, the process repeats itself until a change in the trial vector occurs.

In the selection phase, $P[R_1]$, $P[R_2]$ and $P[R_3]$ are considered as target vectors depending on the selected parameters. The selection phase has 3 parts as follows:

1. If the trial vector has smaller dominance rank than the target vector or if the trial vector dominance rank is equal to zero, then the trial vector replaces the first selected ($P[R_1]$) target vector.

2. Otherwise, the trial vector is compared with the second randomly selected target vector ($P[R_2]$) according to their dominance rank. If the trial vector has smaller dominance rank than $P[R_2]$, it replaces $P[R_2]$. This part will be performed after 200,000 function evaluations.

3. In the case when the trial vector does not change in the first two steps, then it is compared with $P[R_3]$. If the trial vector has smaller dominance rank than the third random selection ($P[R_3]$) target vector, it replaces $P[R_3]$. This part is also performed after 200,000 function evaluations.

The record of fitness values is kept in a pool after every 500 iterations. The pool is implemented by the standard array. At the end of 300,000 iterations, the Pareto front of the pool is calculated and non-dominated solutions are found. According to the literature, in the evaluation phase, the maximum number of fitness function evaluations is considered as 300,000. Likewise, the inverted Generational Distance (IGD) metric is computed for the obtained Pareto-front’s and used to compare the methods. For each test problem, the mean and standard deviation values are also calculated for 30 runs of the algorithm.

The proposed algorithm is implemented in C++ Builder 10.4 (RadStudio) and is shown as pseudo code in Fig. 4.
2.3. The test problems description

The description of all test problems and benchmarks are given in this section. In Table 1 the test problems, used in this paper, are given in detail. The true Pareto front of each test problem can be also found in [7, 8].

Table 1 shows the test problems that are used to evaluate the performance of the DBDE algorithm. The six benchmark problems (UF1-UF6) are taken from CEC 2009 [7] and are compared with the NS-MFO [9], MOEA/D [10], INMTLBO [11], MOEADGM [12], MTS [13], DMOEA-DD [14], HNSGA [15], NSGA-II-SQP [16], OW-MOSADE [17], Clustering MOEA [18], ALMALGAM [19], OMOEA-II [20], NSGA-II [21], (original) MOSADE [17] algorithms.

Additionally, the MOEA/D [10], MSMO/2D [22], NSGA-II [23], MOCLPSO [23] algorithms are applied to solve Fonseca-Fleming problem [24]. Finally the MSMO/2D [22], MOPSO [25], MicroGA [26], MGA [27], NSGA-II [21], PAES [28], MOEA/D [10] algorithms are used to solve Kursawe [29] problem. Test problems are collected from the state-of-the-art literature.

---

**Fig. 4.** The Pseudo code of DBDE algorithm

```plaintext
1. Initialization (1000 random solutions).
   Matrix A[i][j] = rand(lower bound, upper bound);
2. Calculate Fitness (F1[i] and F2[i])
3. Select 100 Best solutions as the first population.
   While (iteration > 0)
   { 
4. Apply Mutation operator
   V1 = MatrixA[R1][x] + (MatrixA[R2][x]*FValue - MatrixA[R3][x]*FValue);
5. Apply Crossover operator
      for (x = 0; x <= Dimension - 1; ++x)
      {
        RandX = randfrom(0,1); jrand=rand(0, Dimension)
        if (CRValue >= RandX || x==jrand)
        {
          if (exceed the lower or upper bound) U[i]= MatrixA[R1][x];
          else U[x] = V[x];
        }
        else U[x] = MatrixA[R1][x];
      }
      If (U[x] == MatrixA[R1][x]) repeat 4th and 5th step.
6. Calculate Fitness (F1[New] and F2[New])
7. Apply Selection operator
   a. If ( Dominated[R1] > Dominated[New] || Dominated[New] == 0 )
      F1[R1] = F1[New] and F2[R1] = F2[New]
   b. Else If ( Dominated[R2] > Dominated[New] && iteration<100,000 )
      F1[R2] = F1[New] and F2[R2] = F2[New]
   c. Else If ( Dominated[R3] > Dominated[New] && iteration<100,000 )
      F1[R3] = F1[New] and F2[R3] = F2[New]
   Count=Count+1;
      If (count==500) // every 500 iteration
      {
        Add non-dominated fitness values (F1, F2) to the general Pool.
        Count=0;
      }
      iteration= iteration-1;
} 
8. When the iteration is finished
Select all non-dominated solutions from General Pool and represent them on the plot.
```
Table 1

| Name            | Problems                                                                 | Search Space |
|-----------------|-------------------------------------------------------------------------|--------------|
| Fonsecaand Fleming | $f_1 = 1 - \exp \left( -\frac{1}{\sqrt{n}} \sum_{i=1}^{n} \left( x_i - \frac{1}{\sqrt{n}} \right)^2 \right)$, $f_2 = 1 - \exp \left( -\frac{1}{\sqrt{n}} \sum_{i=1}^{n} \left( x_i + \frac{1}{\sqrt{n}} \right)^2 \right)$ | $-4 \leq x_i \leq 4, 1 \leq i \leq n$ |
| Kursawe          | $f_1 = \sum_{i=1}^{2} -10 \exp \left( -0.2 \sqrt{x_i^2 + x_{i+1}^2} \right)$, $f_2 = \sum_{i=1}^{3} \left| x_i \right| + 5 \sin \left( x_i^3 \right)$ | $-5 \leq x_i \leq 5, 1 \leq i \leq 3$ |
| UF1              | $f_1 = x_1 + \frac{2}{\left| J \right|} \sum_{j \in J} \left[ x_j - \sin \left( 6 \pi x_j + \frac{\pi}{n} \right) \right]^2$, $f_2 = 1 - \sqrt{x_1} + \frac{2}{\left| J \right|} \sum_{j \in J} \left[ x_j - \sin \left( 6 \pi x_j + \frac{\pi}{n} \right) \right]^2$ | $[0,1] \times [-1]^{n-1}$ |
| UF2              | $f_1 = x_1 + \frac{2}{\left| J \right|} \sum_{j \in J} y_j^2$, $f_2 = 1 - \sqrt{x_1} + \frac{2}{\left| J \right|} \sum_{j \in J} y_j^2$ | $[0,1] \times [-1]^{n-1}$ |
| UF3              | $f_1 = x_1 + \frac{2}{\left| J \right|} \sum_{j \in J} \left( 4 \sum_{j \in J} y_j^2 - 2 \prod_{j \in J} \cos \left( \frac{20 \pi y_j}{\sqrt{j}} \right) + 2 \right)$ | $[0.1]^{n}$ |
| UF4              | $f_1 = x_1 + \frac{2}{\left| J \right|} \sum_{j \in J} h(y_j)$, $f_2 = 1 - x_1^2 + \frac{2}{\left| J \right|} \sum_{j \in J} h(y_j)$ | $[0,1] \times [-2.2]^{n-1}$ |
| UF5              | $f_1 = x_1 + \frac{1}{2N^2 + \varepsilon} \left( \sin \left( 2N \pi x_1 \right) \right)$ + $\frac{2}{\left| J \right|} \sum_{j \in J} h(y_j)$, $f_2 = 1 - x_1 + \frac{1}{2N^2 + \varepsilon} \left( \sin \left( 2N \pi x_1 \right) \right)$ + $\frac{2}{\left| J \right|} \sum_{j \in J} h(y_j)$ | $[0,1] \times [-1.1]^{n-1}$ |
| UF6              | $f_1 = x_1 + \frac{2}{\left| J \right|} \sum_{j \in J} \left( 4 \sum_{j \in J} y_j^2 - 2 \prod_{j \in J} \cos \left( \frac{20 \pi y_j}{\sqrt{j}} \right) + 2 \right)$ | $[0,1] \times [-1.1]^{n-1}$ |

2.4. Performance metrics and ranking method

The Inverted Generational Distance (IGD) metric [30] is used to evaluate the proposed algorithm and compare it with its competitors. IGD is the converse of Generational Distance metric [31]. IGD calculates the average Euclidean distance between two Pareto-fronts. The mathematical description of IGD is as follows:

$$IGD = \frac{\left( \sum_{j=1}^{P} \frac{1}{d_j^p} \right)^{1/\gamma}}{\left| P^* \right|}.$$  (3)

The formula calculates the average distance from $P^*$ to its nearest solution $P$ [11], $d_j$ value is the minimum Euclidean distance between the point $P^*$ and the solution $P$. Likewise, $|P^*|$ is the size of $P^*$. A smaller IGD value indicates a good convergence and well-spread distribution of the solutions obtained by the algorithm. In this paper, $p = 1$ is used for all test problems.

An IGD value, equal to 0 shows that all solutions are distributed on the true Pareto front and have the best possible distribution [32]. The smallest IGD value among the obtained Pareto-fronts indicates that it is the best performing algorithm. Meanwhile, the smallest value is the nearest and
well-distributed selected Pareto front. To calculate the IGD, the true Pareto front is needed for each multi-objective optimization problem. This is the limitation of the IGD for testing the algorithms; it means that if a problem does not have a true Pareto, calculation of IGD is not possible.

3. Results
Performance evaluation of the proposed DBDE and exhibition of its comparative success against state-of-the-art methods are given in this section. In the literature, in order to compare the performance of the multi-objective algorithms, the evaluation is based on some predefined values. Usually, the mean IGD values obtained by 30 independent runs of the programs and the quality of non-dominated solutions found by performing exactly 300,000 function evaluations are used to compare algorithms. In order to compare the performance of the DBDE algorithm with those reported algorithms in the literature, the same number of function evaluations and independent runs are performed.

The performance of DBDE algorithm is evaluated using the 6 test problems of CEC2009, given in Table 1. For each problem, the mean value and also the standard deviation are calculated over 30 independent runs. Thereafter, they are compared with the state-of-the-art algorithms described in the test problems section. In Table 2, it can be observed that the proposed DBDE algorithm has a comparable performance with other 14 algorithms chosen from the literature. The best results in Table 2 are shown in bold.

Table 2
Mean and standard deviation of IGD (Inverted Generational Distance) results for unconstrained multi-objective problems of CEC 2009

| Algorithm          | Abbreviation | Metrics    | UF1     | UF2     | UF3     | UF4     | UF5     | UF6     |
|--------------------|--------------|------------|---------|---------|---------|---------|---------|---------|
| DBDE               | A1           | Mean       | 3.326E-03| 4.273E-03| 2.165E-02| 2.514E-02| 4.981E-02| 4.462E-02|
|                    |              | Std. Deviation| 1.666E-04| 3.512E-04| 3.521E-03| 7.318E-04| 5.623E-03| 5.764E-03|
| INMTLBO            | A2           | Mean       | 6.193E-03| 6.997E-03| 2.399E-02| 2.666E-02| 4.981E-02| 4.462E-02|
|                    |              | Std. Deviation| 1.683E-04| 3.540E-04| 9.631E-03| 8.590E-04| 2.060E-02| 2.794E-02|
| DMOEA-DD           | A3           | Mean       | 1.038E-02| 6.791E-03| 3.337E-02| 4.269E-02| 3.145E-01| 6.673E-02|
|                    |              | Std. Deviation| 2.367E-03| 2.017E-03| 5.680E-03| 7.318E-04| 1.586E-01| 6.673E-02|
| MOEADGM            | A4           | Mean       | 6.200E-03| 6.400E-03| 4.290E-02| 4.760E-02| 1.792E+00| 5.563E-01|
|                    |              | Std. Deviation| 1.130E-03| 4.500E-04| 3.407E-02| 2.220E-03| 5.124E-02| 1.470E-01|
| OMOEAI2            | A5           | Mean       | 8.564E-02| 3.057E-02| 2.714E-01| 4.264E-02| 1.792E+00| 7.338E-02|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| MTS                | A6           | Mean       | 6.467E-03| 6.158E-03| 5.311E-02| 2.356E-02| 1.489E-02| 5.918E-02|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| MOEAD              | A7           | Mean       | 4.350E-03| 6.790E-03| 7.420E-03| 6.358E-02| 1.807E-01| 5.870E-03|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| Clustering MOEA    | A8           | Mean       | 2.990E-02| 2.280E-02| 5.490E-02| 5.850E-02| 2.473E-01| 8.710E-02|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| OW-MOSADE          | A9           | Mean       | 1.220E-02| 8.100E-02| 1.030E-01| 5.130E-02| 4.303E-01| 1.918E-01|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| (original) MOSADE  | A10          | Mean       | 9.830E-02| 6.070E-02| 3.248E-02| 9.770E-02| 6.963E-02| 3.640E-01|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| HNSGA              | A11          | Mean       | 1.124E-02| 3.897E-03| 2.875E-02| 4.121E-02| 3.972E-01| 1.508E-01|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| ALMALGAM           | A12          | Mean       | 5.799E-02| 1.322E-02| 1.365E-01| 4.102E-02| 1.718E-01| 7.855E-02|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| NSGA-II-SQP        | A13          | Mean       | 8.557E-03| 1.367E-03| 2.293E-02| 3.320E-04| 2.873E-03| 5.998E-03|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| NSGA-II            | A14          | Mean       | 9.608E-03| 9.108E-03| 6.864E-02| 5.116E-03| 1.827E-01| 1.913E-01|
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
| NS-MFO             | A15          | Mean       | 4.21E-03 | 7.620E-03| 6.720E-02| 3.29E-02 | 6.29E-02 | 4.54E-02 |
|                    |              | Std. Deviation| N/A     | N/A     | N/A     | N/A     | N/A     | N/A     |
In order to clearly observe the performance of the proposed algorithm, only the mean values of these 6 test problems are tabulated in Table 3, once more in ascending order.

In Table 3, it can be deduced that the proposed algorithm (A1) outperforms the majority of the algorithms that are used for comparison. For all 6 test problems out of 16 problems, the proposed algorithm is always among the first 2 most-effective algorithms according to their mean of 30 IGD values. It can be seen that MOEAD and MTS methods take the first place for UF1 and UF6 problems respectively. Likewise, the proposed algorithm outperforms other methods for UF1 problem. Also, it can be noticed that HNSGA, NSGA-II-SQP and NSGA-II algorithms have lower performance compared to the others. Beside this, INMTLBO algorithm powered by the NSGA methodology and A2 has better performance than other NSGA versions. It is clear that the proposed method outperforms other methods taken into consideration for majority of test cases.

Additionally, the mean and the standard deviation values of Fonseca & Fleming and Kursawe problems are shown in Table 4.

In Table 3, together with the proposed algorithm for Fonseca & Fleming, problem, five mean IGD values are compared and for Kursawe problem eight mean IGD values are compared. The mean IGD value of DBDE algorithm is in the second and fourth rank for the Kursawe and Fonseca and Fleming problems respectively. Mean IGD value of DBDE algorithm is 1.584E-02 for Fonseca & Fleming problem and is 3.42E-03 for Kursawe problem. According to the mean values, ascending order of the algorithms is shown in Table 5.
Table 5 represents the performance of algorithms in ascending order, in which the DBDE algorithm is marked in yellow. It can be seen that even though the proposed algorithm solves the Kursawe problem efficiently, it takes the forth place for the Fonseca & Fleming problem.

As a matter of fact, according to the presented results no algorithm completely outperforms the others because each algorithm has its own set of advantages and disadvantages for solving particular problems.

Fig. 5 shows a graphical representation of the non-dominated solutions obtained using DBDE algorithm for all test problems. The red dots represent the True Pareto, and the green dots represent the solution Pareto set.

Fig. 5. represents the visualization of non-dominated solutions obtained using DBDE algorithm for all test problems: a – Fonseca & Fleming function; b – Kursawe Function; c – UF1 Function; d – UF2 Function; e – UF3 Function; f – UF4 Function; g – UF5 Function; h – UF6 Function

It is seen that Fig. 5, a–d have good convergence and better IGD results than Fig. 5, e–h. Also when the complexity increases, the possible convergence issues arise.

22
4. Discussion

The experiments and comparisons are carried out based on IGD and standard deviation values, which prove the high performance of the proposed method. The IGD values are calculated at the end of 30 independent runs of the algorithm in which each run conducts 300,000 fitness function evaluations. Lowest IGD shows the best optimized results. The ascending order of the results is mentioned in Table 3 and Table 5. Depends on this results; the ranks for UF2 – UF6 are in the second order for DBDE. Meanwhile, DBDE obtains a best rank for UF1 test case. The IGD mean value for UF1 is 3.33E-03 which is obtained by A1 (DBDE) method. That is the smallest value among the selected algorithms. Smaller IGD values mean higher quality solutions. The second best mean value for UF1 is 4.21E-03 which is obtained by A15 (NS-MFO) method. Likewise the third ranked algorithm is A7 (MOEAD) which has 4.35E-03 IGD mean value.

Next, the results for the second problem are considered with regard to the results of the different comparable algorithms. UF2 IGD mean value is 3.90E-03 in A11 (HNSGA), which is the best smallest value between the selected algorithms. Second best UF2 IGD mean value is 4.27E-03 in the A1 (DBDE). The third ranked algorithm is the A6 (MTS) has 6.16E-03 IGD mean value.

Likewise, next the results for the third problem are considered with regard to the results of the different comparable algorithms UF3 IGD mean value is 7.42E-03 in A7 (MOEAD), which is the best smallest value between the selected algorithms. Second best UF3 IGD mean value is 2.17E-02 in the A1 (DBDE). The third ranked algorithm is the A2 (INMTLBO) has 2.40E-02 IGD mean value.

Next, the results for the fourth problem, UF4 IGD mean value is 2.36E-02 in A6 (MTS), which is the best smallest value between the selected algorithms. Second best UF4 IGD mean value is 2.51E-02 in the A1 (DBDE). The third ranked algorithm is the A2 (INMTLBO) has 2.67E-02 IGD mean value.

The fifth problem, UF5 IGD mean value is 1.49E-02 in A6 (MTS), which is the best smallest value between the selected algorithms. Second best UF5 IGD mean value is 4.98E-02 in the A1 (DBDE). The third ranked algorithm is the A15 (NS-MFO) has 6.29E-02 IGD mean value.

The last problem, UF6 IGD mean value is 5.87E-03 in A7 (MOEAD), which is the best smallest value between the selected algorithms. Second best UF6 IGD mean value is 4.46E-02 in the A1 (DBDE). The third ranked algorithm is the A15 (NS-MFO) has 4.54E-02 IGD mean value.

According to the obtained results, the proposed method is among the first two places out of 15 state-of-the-art algorithms for all six test problems of CEC2009.

For future research work, the following objectives are to be considered:
1. The new methods and modifications can be proposed to be integrated into DBDE to be able to solve diverse kinds of problems.
2. The proposed method can be extended to deal with three and more objective problems.

5. Conclusions

In this work, the well-known DE algorithm is enhanced to improve the quality of extracted solutions for the optimization problems. The improvement is achieved by modifying the main operators, e.g. crossover and selection operators. Furthermore, the Crossover operator is repeated...
until a trial vector different from target vector is obtained. The selection phase is also modified by choosing three target vectors \( P[R1], P[R2] \) and \( P[R3] \) to affect the proposed DBDE positively for increasing the performance. Likewise, the mutation operator is also applied on those three selected vectors. In the standard DE algorithm, the target vector is the \( i^{th} \) vector in the population but in the proposed method, in order to increase the quality of solutions, the target vector for mutation operator is selected among randomly chosen vectors. The modified DE improves the convergence quality as well.

According to the numerical and graphical experimental results presented in Tables 2–5, and Fig. 5, it can be noticed that the proposed method is effective and outclassing. The evaluation results illustrate that the proposed DBDE outperforms the majority of its competitors.

Acknowledgments

We express our gratitude to our colleagues in Engineering Faculty of GAU (North Cyprus) for their valuable suggestions in this study.

References

[1] Marler, R. T., Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization, 26 (6), 369–395. doi: https://doi.org/10.1007/s00158-003-0368-6

[2] Coello, C. A. C., Lamont, G. B., van Veldhuizen, D. A. (2007). Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, 800. doi: https://doi.org/10.1007/978-0-387-36797-2

[3] Li, Y., Wang, S., Yang, B. (2020). An improved differential evolution algorithm with dual mutation strategies collaboration. Expert Systems with Applications, 153, 113451. doi: https://doi.org/10.1016/j.eswa.2020.113451

[4] Mezura-Montes, E., Reyes-Sierra, M., Coello, C. A. C. (2008). Multi-objective Optimization Using Differential Evolution: A Survey of the State-of-the-Art. Studies in Computational Intelligence, 173–196. doi: https://doi.org/10.1007/978-3-540-68830-3_7

[5] Salomon, M., Perrin, G.-R., Heitz, F., Armstack, J.-P. (2005). Parallel Differential Evolution: Application to 3-D Medical Image Registration. Differential Evolution, 353–411. doi: https://doi.org/10.1007/3-540-31306-0_12

[6] Ma, J.-P., Zheng, Z.-B., Tong, Q.-X., Zheng, L.-F. (2003). An application of genetic algorithms on band selection for hyperspectral image classification. Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693). doi: https://doi.org/10.1109/icmlc.2003.1260030

[7] Zhang, Q., Zhou, A., Zhao, S., Suganthan, P. N., Liu, W. (2009). Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition. 2009 IEEE Congress on Evolutionary Computation (CEC 2009). Available at: https://www.researchgate.net/profile/Ponnuthurai-Suganthan/publication/265432807_Multiobjective_optimization_Test_Instances_for_the_CEC_2009_Special_Session_and_Competition/links/54b7d9940cf2c27adc473433/Multiobjective-optimization-Test-Instances-for-the-CEC-2009-Special-Session-and-Competition.pdf

[8] JMetal Web site. Available at: http://jmetal.sourceforge.net/index.html

[9] Savsani, V., Tawhid, M. A. (2017). Non-dominated sorting moth flame optimization (NS-MFO) for multi-objective problems. Engineering Applications of Artificial Intelligence, 63, 20–32. doi: https://doi.org/10.1016/j.engappai.2017.04.018

[10] Zhang, Q., Li, H. (2007). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Transactions on Evolutionary Computation, 11 (6), 712–731. doi: https://doi.org/10.1109/tevc.2007.92759

[11] Yu, D., Hong, J., Zhang, J., Niu, Q. (2018). Multi-Objective Individualized-Instruction Teaching-Learning-Based Optimization Algorithm. Applied Soft Computing, 62, 288–314. doi: https://doi.org/10.1016/j.asoc.2017.08.056

[12] Chen, C.-M., Chen, Y., Zhang, Q. (2009). Enhancing MOEA/D with guided mutation and priority update for multi-objective optimization. 2009 IEEE Congress on Evolutionary Computation. doi: https://doi.org/10.1109/cec.2009.4982950

[13] Zong, P. (2011). MTS Algorithm Based on Vague Set. 2011 International Conference on Intelligence Science and Information Engineering. doi: https://doi.org/10.1109/isie.2011.123

[14] Liu, M., Zou, X., Chen, Y., Wu, Z. (2009). Performance assessment of DMOEA-DD with CEC 2009 MOEA competition test instances. 2009 IEEE Congress on Evolutionary Computation. doi: https://doi.org/10.1109/cecc.2009.4983309

[15] Khan Mashwani, W., Salhi, A., Yeniay, O., Hussain, H., Jan, M. A. (2017). Hybrid non-dominated sorting genetic algorithm with adaptive operators selection. Applied Soft Computing, 56, 1–18. doi: https://doi.org/10.1016/j.asoc.2017.01.056

[16] Sindhya, K., Sinha, A., Deb, K., Miettinen, K. (2009). Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. 2009 IEEE Congress on Evolutionary Computation. doi: https://doi.org/10.1109/cecc.2009.4983310
[17] Huang, V. L., Qin, A. K., Suganthan, P. N., Tasgetiren, M. F. (2007). Multi-objective optimization based on self-adaptive differential evolution algorithm. 2007 IEEE Congress on Evolutionary Computation. doi: https://doi.org/10.1109/cec.2007.4424939

[18] Lin, Q., Liu, S., Wong, K.-C., Gong, M., Coello Coello, C. A., Chen, J., Zhang, J. (2019). A Clustering-Based Evolutionary Algorithm for Many-Objective Optimization Problems. IEEE Transactions on Evolutionary Computation, 23 (3), 391–405. doi: https://doi.org/10.1109/tevc.2018.2866927

[19] Vrugt, J. A., Robinson, B. A. (2007). Improved evolutionary optimization from genetically adaptive multimethod search. Proceedings of the National Academy of Sciences, 104 (3), 708–711. doi: https://doi.org/10.1073/pnas.0610471010

[20] Zeng, S., Yao, S., Kang, L., Liu, Y. (2005). An Efficient Multi-objective Evolutionary Algorithm: OMOPA-II. Evolutionary Multi-Criterion Optimization, 108–119. doi: https://doi.org/10.1007/978-3-540-31880-4_8

[21] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6 (2), 182–197. doi: https://doi.org/10.1109/4235.996017

[22] Sedarous, S., El-Gokhy, S. M., Sallam, E. (2018). Multi-swarm multi-objective optimization based on a hybrid strategy. Alexandria Engineering Journal, 57 (3), 1619–1629. doi: https://doi.org/10.1016/j.aej.2017.06.017

[23] Wu, D., Liu, Y., Zhou, K., Li, K., Li, J. (2019). A multi-objective particle swarm optimization algorithm based on human social behavior for environmental economics dispatch problems. Environmental Engineering and Management Journal, 18 (7), 1599–1607. doi: https://doi.org/10.30638/eemj.2019.150

[24] Fonseca, C. M., Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. Evolutionary Computation, 3 (1), 1–16. doi: https://doi.org/10.1162/evco.1995.3.1.1

[25] Coello, C. A. C., Pulido, G. T., Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. IEEE Transactions on Evolutionary Computation, 8 (3), 256–279. doi: https://doi.org/10.1109/tevc.2004.826067

[26] Coello Coello, C. A., Pulido, G. T. (2005). Multiobjective structural optimization using a microgenetic algorithm. Structural and Multidisciplinary Optimization, 30 (5), 388–403. doi: https://doi.org/10.1007/s00158-005-0527-z

[27] Lim, W. J., Jambek, A. B., Neoh, S. C. (2015). Kursawe and ZDT functions optimization using hybrid micro genetic algorithm (HMGA). Soft Computing, 19 (12), 3571–3580. doi: https://doi.org/10.1007/s00500-015-1767-5

[28] Knowles, J., Corne, D. (1999). The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). doi: https://doi.org/10.1109/cecco.1999.781913

[29] Kursawe, F. (1991). A variant of evolution strategies for vector optimization. Lecture Notes in Computer Science, 193–197. doi: https://doi.org/10.1007/bfb0029752

[30] Sierra, M. R., Coello Coello, C. A. (2005). Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ε-Dominance. Evolutionary Multi-Criterion Optimization, 505–519. doi: https://doi.org/10.1007/978-3-540-31880-4_35

[31] Durillo, J. J., Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software, 42 (10), 760–771. doi: https://doi.org/10.1016/j.advengsoft.2011.05.014

[32] Lwin, K. T., Qu, R., MacCarthy, B. L. (2017). Mean-VaR portfolio optimization: A nonparametric approach. European Journal of Operational Research, 260 (2), 751–766. doi: https://doi.org/10.1016/j.ejor.2017.01.005

Received date 24.03.2021
Accepted date 10.05.2021
Published date 31.05.2021

How to cite: Tuncay, M., Haydar, A. (2021). Comparison of a novel dominance-based differential evolution method with the state-of-the-art methods for solving multi-objective real-valued optimization problems. EUREKA: Physics and Engineering, 3, 14–25. doi: https://doi.org/10.21303/2461-4262.2021.001857