https://doi.org/10.48048/tis.2021.17

Multi-Objective Differential Evolution Algorithm with a New Environmental Parameter based Mutation for Solving Optimization Problems

Avjeet Singh[*], Lekhraj, Alok Kumar and Anoj Kumar

Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology Allahabad, Prayagraj 211004, India

([*]Corresponding author's e-mail: 2016rcs01@mnnit.ac.in)

Received: 26 February 2021, Revised: 26 May 2021, Accepted: 31 May 2021

Abstract

Simultaneous optimization of two or more objectives is an instance of multi-objective optimization (MOO). However, for most of the Multi-Objective Problems (MOPs), no single solution can optimize all the objective functions simultaneously. Evolutionary algorithms are a type of optimization algorithm used for constructing a well-distributed optimal front more quickly than a much more efficient approach. One of the most commonly used algorithms in this regard is differential evolution (DE). To solve the problem of non-dominated sorting for Multi-Objective DE (MODE), an approach is proposed that reduces the time complexity. DE is commonly recognized as one of the best methods for solving the MOPs. Several versions of DEs have been proposed for MOPs. This article proposes a novel mutation technique, named Environmental Parameter-based Multi-Objective Differential Evolution (EP-MODE) that acquires two additional environmental parameters to preserve diversity and accelerate the convergence. The proposed approach also reduces the time complexity of non-dominated sorting for Multi-Objective DE (MODE). The proposed mutation's performance has been evaluated on DTLZ series and ZDT series benchmark test functions on the Pareto-optimal front and compared with existing multi-objective algorithms (MODE and MODE-RMO). The solution comparatively verifies that the proposed EP-MODE outperforms most of the MODEs for lower dimension functions and higher functions.

Keywords: Multi-objective optimization, Meta-heuristic algorithms, Evolutionary computational.

Introduction

Multi-objective optimization is used in many areas of science and technology. For example, whenever we buy a car, we try to get more features at a low cost. Hence, we have to fulfill two conflicting objectives: maximize the features and minimize the cost. DE was initially proposed by Storn and Price [17]. DE is an evolutionary algorithm that is conventionally used to solve Single Objective Optimization (SOO) problems. In SOO, there is only 1 objective function, so we can quickly evaluate a solution's fitness. While in MOO, there are many conflicting objectives. Hence, 1 solution may fit one of the objective functions while being unfit for another objective function. The algorithm proposed in Storn and Price [17] is designed for MOPs that increase the convergence rate. To solve test problems, Babu and Jehan [11] proposed a DE for multipurpose optimization in 2003. Here, the penalty function method and the weighting factor method were used to finding Pareto optimum set. Chen *et al.* [5], proposed Pareto-based MODE, which implemented the differential vectors and extended the DE algorithm for the MOPs. The proposed algorithm had fast convergence due to adaptive parameter setting. Singh *et al*. [3] proposed a novel mutation for DE (NEPDE) called new environment-based parameters in 2020 [3]. To solve the stagnation problem and maintain diversity, 2 variants were proposed for best-case and current-to-best cases.

This article improves the convergence rate by reducing the time complexity of non-dominated sorting through a new version of MODE, named EP-MODE. The proposed mutation technique uses two additional environmental parameters, maintaining diversity and improving the convergence rate. We have evaluated the performance of the modified algorithm using the 12 benchmarking functions (ZDT1 - ZDT6 and DTLZ1 - DTLZ7). The ZDT family is also known as the benchmark function of the Bi-objective, and the DTLZ family is known as the benchmark function of the Tri-objective.

The rest of the paper is structured as follows. The rest of the paper is organized as follows. Background details are discussed in Section II, while the related work is discussed in Section III. Section IV elaborates the proposed work. The experimental results are discussed in section V. At last, the discussion of the conclusion and future work are included in Section VI.

Background details

DE is one of the most popular techniques to solve complex optimization problems. Inspired by the biological phenomenon of natural selection, Storn and Price [17] proposed a differential algorithm in 1995. The logic behind differential evolution is simple, and the algorithm effectively reaches optimum global values. The DE algorithm updates the randomly generated initial population on its own with the help of 3 operators mutation, crossover, and selection. Till now, a lot of optimization problems have been solved using differential evolution and its variants. The original DE algorithm is conventionally used for SOO, i.e., in optimization problems with only one objective function. Many of the real-world optimization problems have been solved successfully by the DE algorithm with some modifications in mutation techniques. DE has improved significantly over the last few decades with some changes in DE operators. They are Population size (NP), Scaling factor (F), and Crossover (CR).

DE initialize with random population as defined according to Eq. (1) [1-5];

$$\vec{\mathrm{X}}_{i,\mathrm{g}} = \{x_{i,1,g}, x_{i,2,g}, x_{i,3,g}, \ldots, x_{i,D,g}\} \quad (1)$$

where $i = (1, 2, 3, \ldots, \acute{N}P)$, dimension of the objective function (D) with generation (g).

Algorithm 1 describes the basic DE algorithm. Here, the essential operation of DE operators is discussed. First, initialize the population randomly. Second, evaluate the fitness value corresponding to the mutation strategy. Third, assess the donor vector, where crossover strategy is used to calculate the donor vector. Lastly, the selection method is applied to evaluate the optimal solution. If conditions are satisfied then the procedure is terminated. Otherwise, the process is repeated to evaluate the fitness value. Here, we discuss the original DE algorithm.

DE has the following steps:

a) Initialization: Population is initialized randomly.

b) Mutation: Some parameters of randomly chosen vectors are changed.

c) Crossover: New vectors are generated by using the existing population.

d) Selection: New generation of the population is finalized by selecting vectors using a fitness function.

Algorithm 1: Standard DE

Start the procedure

Initialize random population (P^0)

To calculate the fitness value

Apply mutation technique to generate the donor vector:

$$^{k}_{i}\vec{\mathrm{V}}_{g} = \vec{\mathrm{X}}_{R_1^i,\mathrm{g}} + F_1^i\,(\vec{\mathrm{X}}_{R_2^i\,\mathrm{g}} - \vec{\mathrm{X}}_{R_3^i\,\mathrm{g}})$$

Apply crossover technique to generate trail vector:

$$\vec{U}_{j,i,\mathrm{g}} = \begin{cases} \vec{V}_{i,\mathrm{j,g}}, if \;\; rand\,(j,i) \leq CR \; or \; j = j_{rand} \\ {}^{k}_{i}\vec{X}_{i,j,g} \qquad\qquad \text{otherwise} \end{cases}$$

Selection technique is used:

$$^{k}_{i}\vec{\mathrm{X}}_{i,g+1} = \begin{cases} \vec{U}_{i,g}, & \text{if } f(\vec{U}_{i,g}) \leq f({}^{k}_{i}\vec{\mathrm{X}}_{i,g}) \\ {}^{k}_{i}\vec{\mathrm{X}}_{i,g} & \text{if } f(\vec{U}_{i,g}) > f(\vec{\mathrm{X}}_{i,g}) \end{cases}$$

If condition satisfied, then terminate the procedure; otherwise go to step 2

End procedure

Initialization

For establishing a starting point for the process of optimization, an initial population P^0 is created. Generally, in P^0, each $\hat{j}^{th}$ component ($\hat{j}$ = 1, 2, 3, ..., D) of the i^{th} individuals (i = 1, 2, 3, ..., $\acute{N}P$) is obtained as per Eq. (2) [18];

$$\chi^0_{i,j} = \chi_{j,L} + rand.(\chi_{j,U} - \chi_{j,L}) \quad (2)$$

where D denotes the number of dimensions, $\acute{N}P$ is population size, and rand returns random real value in the interval [0,1].

$\chi_{j,U}$, $\chi_{j,L}$ these are the variables bounds named upper bound and lower bound.

Mutation

In order to create a mutant vector ${}^k_i\vec{V}_g$ for each target vector, the equation that is used to construct these mutant vectors is as follows [1-5];

$${}^k_i\vec{V}_g = \vec{X}_{R_1^i,g} + F_1^i\,(\vec{X}_{R_2^i\,g} - \vec{X}_{R_3^i\,g}), \quad (3)$$

$R_1 \neq R_2 \neq R_3 \neq i,$

where indices R_1, R_2, $R_3 \in \{1, 2, ..., \acute{N}P\}$ *chosen randomly.* Here, F_1^i(scaling factor) belongs to a real number, and it controls the amplification of the vector ($\vec{X}_{R_2^i\,g} - \vec{X}_{R_3^i\,g}$). On the rarest occasion, a part of the mutant vector violates the limits of the search space. For such cases, another value is computed utilizing the initialization equation.

Crossover

When 1 or more values in the target vector are crossed with 1 or more values in the mutated vector in the crossover process, the target vector's values are recombined to yield the trial vector [1-5];

$$\vec{U}_{j,i,g} = \begin{cases} \vec{V}_{i,j,g}, if \;\; rand\,(j,i) \leq CR \; or \; j = j_{rand} \\ {}^k_i\vec{X}_{i,j,g} \qquad\qquad \text{otherwise} \end{cases} \quad (4)$$

where $\vec{U}_{j,i,}$ is the trail vector at generation g, and $rand_{j,i}$, ($j \in [1,D]$ and $i \in [1,\acute{N}P]$) is a random real number in [0,1], CR $\in$[0,1] is called the crossover rate.

Selection

A better selection scheme is implemented after completing the crossover operation. The resulting offspring should be chosen if the objective value of the offspring is smaller or equal to the value of the offspring. DE has a greedy selection method, which helps in exploitation. $\vec{V}_{j,i,g}$ is set to $\vec{V}_{j,i,g+1}$, If the trial vector $\vec{V}_{j,i,g}$ gives a optimum fitness function than $\vec{X}_{R_1^i\,g}$,Otherwise, the old vector $\vec{X}_{R_1^i,g}$ is held. The choice plan is as per the following equation [1-5];

$${}^k_i\vec{X}_{i,g+1} = \begin{cases} \vec{V}_{i,g}, & \text{if } f(\vec{V}_{i,g}) \leq f({}^k_i\vec{X}_{i,g}) \\ {}^k_i\vec{X}_{i,g} & \text{if } f(\vec{V}_{i,g}) > f(\vec{X}_{i,g}) \end{cases} \quad (5)$$

DE for multi-objective optimization problem

Two or more objective functions that occur in multi-objective optimization work simultaneously, but for most multi-objective problems (MOPs), no single solution is capable of optimizing all objective functions simultaneously. MOPs can be characterized as the problem of finding all $(\vec{f}_1, \vec{f}_2, \ldots, (\vec{fn}))^T$ having p equality and m inequality constraints satisfy the following Eqs. (6) - (7) [6-11];

$$G_i(\vec{x}) \leq 0; \qquad i = 1, 2, ..., m \quad (6)$$

$$H_j(\vec{x}) = 0; \qquad j = 1, 2, ..., p \quad (7)$$

$G_i(\vec{x})$ defines the set of inequality constraints, $H_j(\vec{x})$ defines the set equality constraints. The vector of decision variables defines as $(\vec{x}) = [x_i, x_2, x_3, \dots, x_n]^T$

$Min\ ((f_1(\vec{x}), f_2(\vec{x}),..., f_m(\vec{x}))^T$

The objective is to find the decision vectors that have all of the constraints fulfilled to optimize the objective function vector.

Pareto dominance

To use DE in MOO, only few modifications are required. Single objective optimization (SOO) deals with only 1 objective function, whereas multi-objective deals with a minimum of 2 objective functions. Hence, we can easily compare 2 solutions based on the obtained results from the objective functions. In multi-objective optimization, we cannot do this as there are multiple conflicting objective functions. We utilize the idea of Pareto dominance to analyze the 2 solutions. A solution χ_1 is said to rule other solution χ_2 in case of the minimization problem, if the accompanying conditions are fulfilled [13];

For all values (1, 2, 3, ...n): $\dot{F}_i(\chi_1) \leq \dot{F}_i(\chi_2)$
For some values (1, 2, 3, ...k): $\dot{F}_i(\chi_1) < \dot{F}_i(\chi_2)$

If a solution contains any objective better then that solution dominates the other solution, not worsen in any of them.
Hence, while looking at 2 arrangements χ_1 and χ_2, there are 3 prospects:

dominates χ_2
χ_2 dominates χ_1 χ_1
Both are non-dominated

Fast non-dominated sorting

Two substances are determined for every solution of set Ś: Firstly, evaluate the number of solutions (assume count n_i) that can dominate the solution i. After that, calculate the set of dominated solutions ($Ś_i$). For each solution in the first non-dominated front F_1, all the solutions behind it in the dominance table can be counted as 0 at the end of this process. Currently, for every solution i with $\eta_j = 0$, it visits every member of its set $Ś_i$ and lessens its domination check by 1. If a member's count reaches to 0, they are placed separately. This is the second non-dominated front F_2. This process is repeated for all the members of F_2 to find the front F_3. Until all fronts are established, this process is conducted again and again [13].

Crowding distance

In order to determine the crowding distance, the population is sorted according to the different objective functions' values, and it is ensured that the population is in increasing order of magnitude. Likewise, to provide an infinitely long boundary to each of the functions with the smallest and largest values, an infinite distance value is given to each objective function. This estimation is further extended with other objective functions. The total crowding distance value is determined as the total of each of the objective's distance values. All of which are compared to the overall objective [12].

Related work

Storn and Price [17] proposed the differential evolution algorithm in 1995; it worked on simple logic and took a reasonable time to reach optimum global values. During the past few years, several instances of evolutionary algorithms have been employed to help solve MOPs. Numerous researchers have additionally applied the standard differential evolution algorithm to tackle MOPs. As used in SOO and regular DE operators, these operators can also be used in multi-objective optimization. While comparing the two solutions, we can use the Pareto dominance property. So, because of this, the DE general algorithm can be used to optimize multi-objective problems by modifying DE operators that are a little bit different.

In 2008, a novel constraint-handling method was proposed by Gong and Cai. [7] to enhance the local search ability. This proposed approach was introduced using the orthogonal design technique and the crossover operator to establish the initial population. A multi-objective Genetic algorithm for rank-based fitness assignment was proposed by Fonseca and Fleming [16]. Here, an extended version of the

traditional niche formation methods for multipurpose was proposed and allowed the external decision-makers direct intervention for the modified fitness assignment method.

For solving the constraints MOPs, Kukkonen and Lampinen [10] proposed an extension of Generalized DE (GDE) in 2004. The proposed approach improved the diversity and execution time of GDE2. As one sequence, generalized differential evolution GDE3 was presented by Kukkonen and Lampinen [9]. It was accompanied by another iteration of the process known as GDE2. It is the improved version of the previous GDE in the case of MOPs. To solve the multi-objective problems, Deb *et al.* [13] introduced in 2002 a Non-dominated Sorting Genetic Algorithm II (NSGAII). This proposed algorithm obtains the non-dominated front after comparing the other Multi-Objective Evolutionary Algorithms (MOEAs) - SPEA and PAES. PAES found better convergence and maintained diversity by controlling crowding of solutions in the search space of pre-specified numbers of similar size cells. The single objective DE for multi-purpose optimization has been expanded by Madavan [12] in 2002, using a Pareto-based approach. The description here is that the results are very effective, which maintains diversity in the Pareto set.

For solving the MOPs, Wang *et al*. [6] proposed an algorithm named MOSADE in 2010, including the Pareto dominance. This proposed method is committed to preserving the non-dominated solutions found during the evolutionary process by maintaining an external elite set. Additional diversity is also accomplished by means of a new crowding entropy diversity estimation technique that secures the diversity of Pareto optimality. MOSADE was compared to GDE, and the given result showed that it performs well in diversity and convergence rate. In order to measure the quality of the individual and improve the diversity, Cheng *et al.* [1] proposed a differential development algorithm in 2021. The proposed approach that was being developed included an algorithm called FDDE which had a fitness and diversity ranking-based mutation operator. FDDI has been compared with rank-jDE, rank-SHADE, jDE, SHADE and LSHADE and found that the proposed algorithm gave better results in CSE-2005, CEC-2013 and 2014 benchmark functions.

In an attempt to find a better optimization technique, Bhaumik, and Maity. [2] in 2020, introduced multi-response optimization with gray relationship analysis using EDM parameters to make the optimization processes more scalable. This research aims to evaluate the most efficient process parameters setting for increasing machining effectiveness. Peng et al. [8] proposed a novel approach for solving the MOPs called opposition-based MODE (OMODE) in 2008. The proposed algorithm used a self-adaptive approach to control the scaling factor, and OMODE aims to generate an initial population. In this order, Bidgoli et al. [4] suggested an opposition-based binary MODE for feature selection in 2019. To overcome binary optimization problems, the proposed solution proposed a continuous metaheuristic algorithm to solve the problems over time.

Proposed work

The standard DE algorithm is widely used in a number of fields, such as science and engineering. However, it suffers from many drawbacks, such as stagnation problems, low convergence rate, etc. This paper aims to implement similar changes to DE for MOPs and check whether the performance of existing algorithms can be increased since the mutation operator has a very low success rate for dealing with the problem of stagnation. To overcome this problem, mutations can be combined with some vector operators. Initially, the base vector is selected as an individual; the mutation's influence depends on the arrangement of the mutation (i.e., base vector, difference of the start and end vector). The base vector occupies a prominent place in the mutation strategy. It is the most significant because once the base vector decides where the search should start and in which regions mutation individuals should be found, it then uses it to plan where to search and look for individuals with mutations.

It can be guaranteed that if the fitness value is assumed, it will achieve exploitation, while exploration can be increased by emphasizing diversity. Like many DE algorithms, MODE uses the regular DE operators, such as mutation, crossover, and selection. In order to sort the solutions, the MODE algorithm uses fast non-dominated sorting and crowding distance techniques. In order to sort the solutions, the MODE algorithm uses fast non-dominated sorting and crowding distance techniques.

Here, we present the modification to the MODE algorithm, which introduced a novel mutation (Environmental based parameter) to increase the algorithm's overall efficiency. Several greedy approaches have been used in MODE, such as DE/rand, DE / best, and DE / rand/2, DE / best/2, DE / current to best to maintain diversity and improve convergence rates.

Subsequently, a novel DE mutation named a New Environmental Parameter based on Multi-Objective Differential Evolution (EP-MODE) was proposed in this article. Here, 2 mutant vectors (namely $t1$ and $t2$) have been introduced to maintain diversity during the initial generation. The first

temporal vector considers the current as the best and the current worst vectors, which are the first and last vectors of the population, then arrange to use increasingly non-dominated sorting and crowding distances. The second temporal vector is like a standard DE algorithm. To increase the exploration capabilities of the algorithm, this vector helps because it is random. The results have been compared with the standard MODE algorithm and with the ranking-based mutation operator.

Proposed mutation

In the proposed mutation technique, we add an alternative technique that has been integrated into our modification to the MODE algorithm called an Environmental-based parameter. To maintain diversity during the initial generation, two mutant vectors (*t1* and *t2*) were introduced. The first temporal vector considers the current best and worst vectors, which are the first and last vectors of the population, then organize to utilize more non-dominated sorting and crowding distances. The second temporal vector is similar to a traditional DE algorithm, which increases the algorithm's effectiveness. Two mutant vectors namely t_1 and t_2 are generated as follows;

$$\overset{k}{i}\vec{V}_g = \vec{X}_{R_1^i,g} + F_1^i\,(\vec{X}_{R_2^i\,g} - \vec{X}_{R_3^i\,g}), \tag{8}$$

$$t_1 = \vec{X}_{R_1^i,g} + r1.(\vec{X}_{best,\,g} - \vec{X}_{R_1^i,g}) - r2.(\vec{X}_{best,\,g} - \vec{X}_{R_1^i,g}) + r3.(\vec{X}_{worst,g} - \vec{X}_{R_1^i,g}), \tag{9}$$

$$t_2 = \vec{X}_{R_4^i,g} + F_2^i\,(\vec{X}_{R_5^i,g} - \vec{X}_{R_6^i,g}),\ r4 \neq r5 \neq r6 = \acute{\imath} \tag{10}$$

*w*here *r1, r2, r3* are randomly chosen indices in the range [1, *ÑP*] while, $\vec{X}_{R_1^i,g}$, $\vec{X}_{best,\,g}$, $\vec{X}_{worst,g}$ are the current vector, the first vector, and the last vector after the non-dominated sorting and crowding distance adjustment of the population.

Now, if t_2 dominates t_1, then t_2 becomes the mutant vector; otherwise, t_1 becomes the mutant vector.

$$\overset{k}{i}\vec{V}_g = \begin{cases} t_2, & \text{if } t_2 \ dominates\ t_1 \\ t_1 & otherwise \end{cases} \tag{11}$$

Selection

The selection in the case of MODE is not as simple as the standard DE algorithm. When the candidate vector $\vec{V}_{i,g}$ which is created after mutation and crossover is compared to its parent $= \vec{X}_{R_1^i,g}$, the following cases arise;

If $\vec{X}_{R_1^i,g}$ dominates $\vec{V}_{i,g}$, then $\vec{V}_{i,g}$ is discarded
If $\vec{V}_{i,g}$ dominates $\vec{X}_{R_1^i,g}$, then $\vec{V}_{i,g}$ replaces $\vec{X}_{R_1^i,g}$
If $\vec{X}_{R_1^i,g}$ and $\vec{V}_{i,g}$ are non-dominated, then $\vec{V}_{i,g}$ is added to the population.

Proposed algorithm

All the operators except the mutation are the same as that of the MODE algorithm. The mutation has been modified according to the mathematical equations that are given above. Algorithm 2 describes the EP-MODE algorithm. The essential operation of MODE operators is discussed here. First, randomly initialize the population, evaluate the fitness value corresponding to the mutation strategy, and then evaluate the rank based on non-dominance sorting and crowding distance. The final selection method is implemented to determine which vector dominates and replaces each other. If the conditions are satisfied, then terminate the process. Otherwise, repeat the process back to evaluate the fitness value. The proposed algorithm is as follows:

Algorithm2: EP-MODE

Generate initial random population $\vec{X}_{i,\,g}$, $\forall_i$, *í = 1, ..., ÑP*
fitness value $f(\vec{X}_{i,\,g})$ $\forall_i$, *í = 1, ..., ÑP*
Rank the population based on non-dominated sorting and crowding distance
***For G** = 1 to G = no of iterations*
***For í** = 1 to í = ÑP*

For $\hat{j}$ = *1 to* $\hat{j}$ = *dimensions*
If ($rand_{i,l}[0, 1] < CR$ *or* $j = \hat{j}_{rand}$) *then*
$t_1 = \vec{X}_{R_1^i,g} + r1.(\vec{X}_{best,g} - \vec{X}_{R_1^i,g}) - r2.(\vec{X}_{best,g} - \vec{X}_{R_1^i,g}) + r3.(\vec{X}_{worst,g} - \vec{X}_{R_1^i,g})$
$t_2 = \vec{X}_{R_4^i,g} + F_2^i\,(\vec{X}_{R_5^i,g} - \vec{X}_{R_6^i,g})$
If $(t_2\ dominates\ t_1)$ then
${}_i^k\vec{V}_g = t_2$
Else
${}_i^k\vec{V}_g = t_1$
End If
Else
${}_i^k\vec{V}_g = {}_i^k\vec{X}_i$
End If
End For
If $(({}_i^k\vec{V}_g)$ *dominates* $({}_i^k\vec{X}_i))$ then
$({}_i^k\vec{V}_g)$ replaces $({}_i^k\vec{X}_i)$ in next generation
Else If $(({}_i^k\vec{X}_i)$ *dominates* $({}_i^k\vec{V}_g))$ then
$({}_i^k\vec{V}_g)$ is discarded in the next generation
Else
$({}_i^k\vec{V}_g) = {}_i^k\vec{V}_{g+1})$, add into next generation
End If
End For
Truncate the population to ŃP members after non-dominated sorting and crowding distance calculation
End For
End

Flow of proposed algorithm

Figure 1 demonstrates the flow of the algorithm and the order in which it is followed. Initially, a random population is generated after evaluating the fitness and proposed mutant vector, followed by generating the trail vector and target vector. After that, check if the trail vector dominates the target vector and vice versa. If somebody dominates the other then replace it and sort the population through fast dominant sorting and crowding distance. Lastly, the condition criteria were checked if they were satisfied; otherwise, they would repeat until the condition was not fulfilled.

The overall algorithm is made up of 4 separate operators, which are initialization, mutation, crossover, and selection. All the operations except the mutation are kept the same as MODE, whereas the mutation has been modified. The mutant vector can be any of the two temporary vectors. The first temporary vector considers the current best and worst vectors, namely the first and last vectors of the population after sorting it through fast non-dominated sorting and crowding distance.

The second temporary vector is the same as that of the standard DE algorithm. This vector helps in increasing the exploration capabilities of the algorithm as it is random. If the first temporary vector dominates by the second vector, then the second vector becomes the mutant vector; otherwise, the first temporary vector becomes the mutant vector. There are two terms involving the current best point in the population. They are multiplied with random values, and there is an equal probability that these terms will move towards the best solution. Hence, there is equalization in the exploitation and exploration capabilities of the algorithm. The term involving the current worst point helps in the exploration of the search space. Hence, problems like premature convergence are avoided. As two temporary mutations are compared, there is a better chance of reaching the Pareto front. The proposed algorithm contains the population size '*NP*' (100), maximum numbers of iteration (200), Scaling Factor lies between 0.1 to 2, and the crossover rate lies between 0.1 to 1.

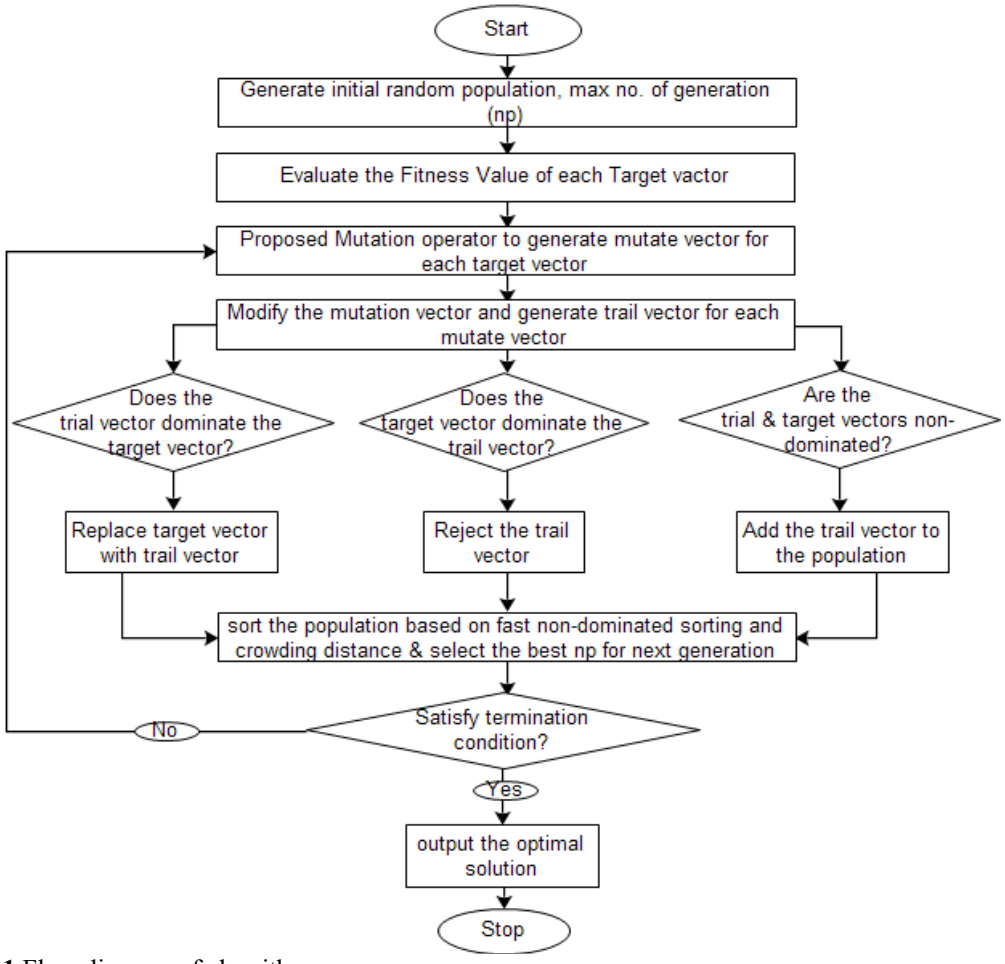

Figure1 Flow diagram of algorithm.

Experimental results analysis

Experimental setup

For experimental analysis, we have used the following hardware and software specifications:

The software specifications of the setup are as follows: The proposed algorithms have been implemented in MATLAB. The Version of MATLAB used is MATLAB R2018b with version 9.5.0, 878302. The inbuilt plotting functions of MATLAB have been used to plot the figures. The operating system used is Windows 10 with Version 1803 Build 17134.523

The hardware specifications of the setup are as follows: Processor: Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz, GPU: NVIDIA GeForce 940MX, and RAM: 8.00 GB DDR3 Memory.

Evaluation criteria

The modified algorithm has been tested on the well-known ZDT (Bi-objective test functions) and DTLZ (Tri-objective test functions) functions. The performance metrics used in this paper are the Generational Distance (GD) [14] and Inverted Generational Distance (IGD) [15].

$$\mathrm{GD}\,(P, P^*) = \sqrt{\frac{\sum_{v \in p} d(v,P^*)^2}{|P|}} \tag{12}$$

where P^* is the true objective vector of Pareto obtained in the true P of the MOP, and P is the set of objective vectors of Pareto which have been obtained. P^* is the objective vector of Pareto in the P of the MOP. The minimum Euclidean distance between v and the nearest point in P^* is $d\,(v, P^*)$. $|P|$ is the number of Pareto objective vectors obtained by EA's in P [14].

$$\text{GD}(P^*, P) = \sqrt{\frac{\sum_{v \in p} d(v,P)^2}{|P^*|}} \quad (13)$$

where $d(v, P)$ is the Euclidean distance between v and the nearest point in P, $|P^*|$ is the count of points in P^* [15].

These metrics calculate the distance of the approximated front with a uniformly distributed optimal front. Hence, a better-performing algorithm has a lesser value of the generational distance and inverted generational distance performance metric. The mathematical expressions (objective functions with constraint functions) of bi-objective text functions with their parameter domains are described in **Table 1**, and the mathematical expressions (objective functions with constraint functions) of tri-objective text functions with their parameter domains are described in **Table 2**.

Table 1 ZDT test functions (bi-objective text functions).

Name	Objective function	Parameter domains
ZDT1	$f_i(x) = x_i$ $g(x) = 1 + 9(\sum_{i=1}^{k} z_i / k)$ $h(x) = 1 - \sqrt{\frac{f_i(x)}{g(x)}}$	[0,1]
ZDT2	$f_i(x) = x_i$ $g(x) = 1 + 9(\sum_{i=1}^{k} z_i / k)$ $h(x) = 1 - (\frac{f_i(x)}{g(x)})^2$	[0,1]
ZDT3	$f_i(x) = x_i$ $g(x) = 1 + 9(\sum_{i=1}^{k} z_i / k)$ $h(x) = 1 - (\frac{f_i(x)}{g(x)}) \sin(10\pi f_i)$	[0,1]
ZDT4	$f_i(x) = x_i$ $g(x) = 1 + 10k + (\sum_{i=1}^{k} z_i^2{}_i - 10\cos(4\pi z_i))$ $h(x) = 1 - \sqrt{\frac{f_i(x)}{g(x)}}$	$x_i \in [0,1]$ $z_i, z_2, \ldots, z_k \in [-5, 5]$
ZDT6	$f_i(x) = 1 - \exp(-4x_i) \sin^6(6\pi x_i)$ $g(x) = 1 + 9(\sum_{i=1}^{k} z_i / k)^{.25}$ $h(x) = 1 - (\frac{f_i(x)}{g(x)})^2$	[0,1]

Table 2 DTLZ test functions (tri-objective text functions).

Name	Objective function	Parameter domains
DTLZ1	$f_i(x) = (1 + g(x)) * 0.5 (\prod_{i=1}^{M-1} x_i)$ $f_{m=2;M-1} = (1 + g(x)) * 0.5 (\prod_{i=1}^{M-m} x_i) * (1 - x_{M-m+1})$ $f_m(x) = (1 + g(x)) * 0.5 (1 - x_i)$ $g(x) = 100 [k + (\sum_{i=1}^{k} (z_i - 0.5)^2 - \cos 20\pi (z_i - 0.5))]$	[0,1]
DTLZ2	$f_i(x) = (1 + g)(\prod_{i=1}^{M-1} \cos(x_i \pi/2))$ $f_{m=2;M-1} = 1 + g(x) + (\prod_{i=1}^{M-m} \cos(x_i \pi/2))(\sin(x_{M-m+1} \pi/2))$	[0,1]

Name	Objective function	Parameter domains
	$f_M(x) = (1 + g(x)) \sin(x_i \pi/2)$ $g(x) = \sum_{i=1}^{k} (z_i - 0.5)^2$	
DTLZ3	As DTLZ2, except the equation for g is replaced by the one form DTLZ1	[0,1]
DTLZ4	As DTLZ2, except all $x_i \in x$ are replaced by x_i^{α} , where $\alpha > 0$	[0,1]
DTLZ5	As DTLZ2, except all $x_1, x_2, \ldots, x_{M-1} \in x$ are replaced by $\frac{1+2(gx_i)}{2(1+g(x))}$	[0,1]
DTLZ6	As DTLZ5, except the equation for g is replaced by $g = \sum_{i=1}^{k} z_i^{0.1}$	[0,1]
DTLZ7	$f_{m=2;M-1} = x_m$ $f_M = (1 + g(x)) * [M - \sum_{i=1}^{M-1} (\frac{f_i}{(1+g(x)} (1+ \sin(3\pi f_i(x))))]$ $g = 1 + 9 (\sum_{i=1}^{k} (z_i / k)$	[0,1]

Experimental results

Experimental results have been evaluated with different MODE text functions (bi-objective and tri-objective), and both of these efficiency metrics (IGD and GD) have been employed to compare algorithms. To demonstrate the efficacy of the proposed solution and the other MODE algorithms, a mean and standard deviation calculation was performed on the algorithms.

The performance variants GD and IGD have been calculated for all variants. The algorithm's proposed results have been compared to the MODE algorithm and the MODE algorithm with a ranking-based mutation operator algorithm, and the results were shown to be comparable. Several other MODEs have been measured based on their mean and standard deviation, and the proposed algorithm has been compared to these other MODEs with different dimensions.

Table 3 contains the comparison of generational distance values with bi-objective (i.e ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6) and tri-objective test functions (i.e DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5 and DTLZ6) with different dimensions (i.e. 30D, 50D and 70D). **Table 4** contains the comparison of inverted generational distance values with bi-objective (i.e ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6) and tri-objective test functions (i.e DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5 and DTLZ6) with different dimensions (i.e. 30D, 50D, and 70D).

Figures 2 - 4 describe the convex Pareto-optimal front, which shows the graphical comparison with other MODEs in the ZDT1 test function with 30D, 50D, and 70D. Figures **5 - 7** describe the non-convex Pareto-optimal front, showing the graphical comparison with other MODEs in the ZDT2 test function. **Figures 8 - 10** add a discreteness feature to the front. Its Pareto-optimal front consists of several noncontiguous convex parts, which show the graphical comparison with other MODEs in the ZDT3 test function. **Figures 11 - 13** describe the local Pareto-optimal front, showing the graphical comparison with other MODEs in the ZDT4 test function. **Figures 14 - 16** describe the Pareto-optimal fronts, which are non-uniformly distributed along the global Pareto front and also show the graphical comparison with others MODEs in the ZDT6 test function with 30D, 50D, and 70D.

Figures 17 - 19 show the graphical comparison with others MODEs in the DTLZ1 test function with 30D, 70D, and 120D. **Figures 20 - 22** show the graphical comparison with others MODEs in the DTLZ2 test function. **Figures 23 - 25** show the graphical comparison with others MODEs in the DTLZ3 test function. **Figures 26 - 28** show the graphical comparison with other MODEs in the DTLZ4 test function. **Figures 29 - 31** show the graphical comparison with other MODEs in the DTLZ5 test function. **Figures 32 - 34** show the graphical comparison with other MODEs in the DTLZ6 test function. Lastly, **Figures 35** show the graphical comparison with others MODEs in the DTLZ7 test function in 30D.

The proposed algorithm usually outperforms the MODE and MODE (RMO) algorithms in most of the functions. EP-MODE is seen to perform better on all dimensions of the ZDT functions. In the case of DTLZ functions, the proposed algorithm performs better on the higher dimensions. Multiple readings have been taken to measure performance metrics. The function DTLZ7 is tested on one dimension, whereas other functions are tested on multiple dimensions. The results are as follows:

Table 3 Comparison of GD values.

Function	Dimension	MODE		MODE-RMO		EP-MODE	
		Mean	StdDev	Mean	StdDev	Mean	StdDev
ZDT1	30	0.0050996	**0.0003510**	0.0038296	0.0004733	**0.0032775**	0.0036644
	50	0.0257171	0.0027408	0.0227154	0.0005902	**0.0016726**	**0.0003898**
	70	0.0512469	0.0024280	0.0447564	0.0025927	0.**0038416**	0.**0016418**
ZDT2	30	0.0106149	0.0018493	0.0071729	**0.0002935**	0.**0030919**	0.0027828
	50	0.0491727	0.**002134**	0.0402188	0.0026933	0.0045062	0.0024422
	70	0.0908123	0.0045683	0.0767683	0.0091966	0.0048849	0.**0005035**
ZDT3	30	0.0072951	0.0014513	0.0048654	0.0005968	0.**0012489**	0.**0003293**
	50	0.0323419	0.0030036	0.0247812	0.0027943	**0.0029991**	0.**0018525**
	70	0.0546471	0.0048956	0.0485958	0.0031356	0.**0007067**	0.**0000774**
ZDT4	30	0.0072951	0.0014513	15.8828413	0.5094700	0.**0003530**	**0.0000195**
	50	0.0323419	0.0030036	37.0554198	0.3679918	0.**0007819**	0.**0003555**
	70	0.0546471	0.0048956	59.7347932	0.**4264647**	0.**5313274**	0.4854828
ZDT6	30	0.3309942	**0.0084861**	0.3083662	0.0099254	0.**2708610**	0.0095527
	50	0.4746727	0.0089370	0.4519638	0.**0089108**	0.**2743374**	0.0127305
	70	0.5274105	0.0079258	0.5159756	0.0065984	0.**2968758**	0.0188281
DTLZ1	30	**50.6969484**	5.8146139	53.2943813	**2.0926112**	735.0846099	133.9393146
	70	208.6264083	6.0246123	200.1275131	**1.8754451**	**146.6315092**	2.5128044
	120	400.613572	10.8624199	385.6247373	**5.5206018**	**266.1048385**	12.3019438
DTLZ2	30	0.0031763	0.0004513	0.**0030363**	**0.0003095**	0.0132009	0.0029794
	70	0.0179406	0.0007520	0.0149286	**0.0006016**	0.**0081481**	0.0035949
	120	0.0708418	0.0045333	0.0704844	0.0121317	0.**0148780**	0.**0024099**
DTLZ3	30	128.5415897	12.3323591	117.1763553	11.4581514	**117.3359666**	**6.1701184**
	70	442.7748506	**10.8673471**	448.5303993	14.4965128	**375.5174806**	27.9078387
	120	914.0440512	914.0440512	897.0508815	**21.0231502**	**625.2881388**	23.8866059
DTLZ4	30	0.0028101	0.0004345	**0.0021597**	0.**0002083**	0.0077095	0.0057735
	70	0.0208106	0.**0012676**	0.**0104229**	0.0119907	0.0387747	0.0164883
	120	0.1022045	0.**0061516**	0.0954379	0.0125115	0.**0587744**	0.0320255
DTLZ5	30	0.0002891	0.0000401	0.**0001616**	**0.0000055**	0.0003565	0.0000956
	70	0.0086133	0.0005236	0.0082159	0.0002949	0.**0012802**	0.**0001924**
	120	0.0783565	0.0084853	0.0760735	0.0029018	0.**0054372**	0.**0054372**
DTLZ6	30	2.3542042	0.**0033284**	2.3558221	0.0045906	**2.2499245**	0.0330471
	70	5.958473	0.0171278	5.9524208	**0.0082762**	**5.6105173**	0.2227938
	120	10.5023923	0.0340580	**5.9486490**	0.**0159106**	9.8161195	0.1656597

Table 4 Comparison of IGD values.

Function	Dimension	MODE		MODE-RMO		(EP-MODE)	
		Mean	StdDev	Mean	StdDev	Mean	StdDev
ZDT1	30	0.0021164	0.**0001637**	0.0016268	0.0002043	0.**0006513**	0.0002029
	50	0.0097957	0.0006882	0.0086227	0.0001842	0.**0007574**	0.**0001606**
	70	0.0186186	0.0007867	0.0168519	0.0006132	0.**0010173**	0.**0000520**
ZDT2	30	0.0044874	0.0007116	0.0031378	0.**0001422**	0.**0012601**	0.0009197
	50	0.0214965	0.0011155	0.**0001422**	0.**0008132**	0.001778	0.0009855
	70	0.0393484	0.001307	0.0329999	0.0040464	0.**0018444**	0.**0003271**

Function	Dimension	MODE		MODE-RMO		(EP-MODE)	
		Mean	StdDev	Mean	StdDev	Mean	StdDev
ZDT3	30	0.0036169	0.0003194	0.0030094	0.**0002164**	0.**0007665**	0.0002873
	50	0.0096386	0.0004813	0.0086504	0.0005675	0.**0011157**	0.**0004334**
	70	0.016514	0.0010833	0.0149696	0.0005261	0.**000531**	0.**0000787**
ZDT4	30	0.0036169	0.0003194	4.7783395	0.2396983	0.**0002586**	0.**0000084**
	50	0.0096386	0.0004813	12.7316509	0.2907907	0.**0004135**	0.**0001406**
	70	0.016514	0.0010833	22.3308135	0.8650338	0.**0004584**	0.**0001102**
ZDT6	30	0.1515706	0.**0053251**	0.1384164	0.0034282	0.**1160654**	0.0077217
	50	0.2128466	0.**0036208**	0.2027558	0.0036781	0.**1155832**	0.0066553
	70	0.2380626	0.0056036	0.**0036781**	0.**0052425**	0.1343534	0.0109848
DTLZ1	30	**3.1173601**	0.**2278579**	3.3305183	0.4283513	32.9541978	5.276202
	70	19.8604139	0.**6656523**	17.1503022	0.7026122	**4.8720915**	1.6958783
	120	40.2391206	2.3573877	41.0334758	1.7817998	**9.6781528**	**1.308647**
DTLZ2	30	0.0010955	**4.45e-05**	0.0010961	2e-06	0.**0012005**	6.78e-05
	70	0.0025962	8.1e-05	0.0022639	4.09e-05	0.**0012534**	**1.97e-05**
	120	0.0086763	**0.0005553**	0.0089447	0.0014922	0.**0015176**	9.29e-05
DTLZ3	30	8.0945495	0.**854539**	8.1200336	0.9143742	**3.6547758**	1.0067295
	70	49.969007	2.0212062	48.4322866	**1.4991671**	**11.5786654**	3.4348376
	120	107.5883221	1.**4380856**	104.6445329	7.81295	**29.7758266**	3.8799994
DTLZ4	30	0.0010757	2.48e-05	0.**0010712**	5.75e-05	0.0022131	0.**0002438**
	70	0.0029543	0.0001618	0.**0018743**	0.0012027	0.0041032	0.**000503**
	120	0.0125720	0.**0009269**	0.0117704	0.0014284	0.**0067901**	0.0022677
DTLZ5	30	0.0001119	2e-06	0.1041725	0.**0034684**	0.**0001139**	7.2e-06
	70	0.**001041**	5.83e-05	0.0009881	4.25e-05	0.0002147	**3.02e-05**
	120	0.0084769	0.001262	0.0082116	2.62e-05	0.**0005944**	0.**000113**
DTLZ6	30	0.3136651	0.0028916	0.317116	0.**0008425**	0.**2853261**	0.0028157
	70	0.8158995	0.0026413	0.8182297	0.**0014049**	0.**7266716**	0.0321836
	120	1.447474	0.0067975	0.**8143451**	0.**0008848**	1.2695113	0.0292717

Graphical representation of results

The following are the comparative graphs showing the Pareto fronts with different test functions (bi-objective and tri-objective test functions):

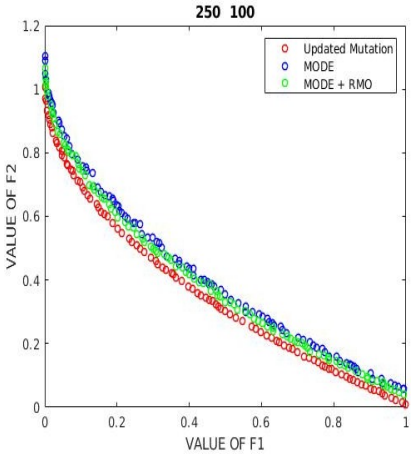

Figure 2 ZDT1, D = 30.

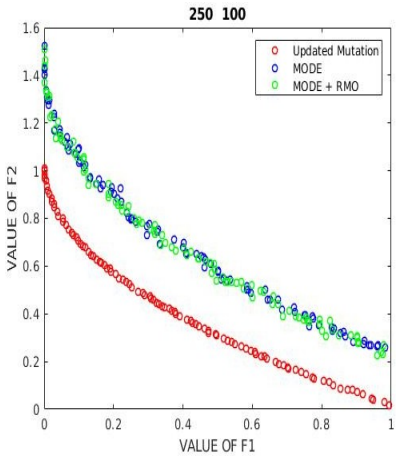

Figure 3 ZDT1, D = 50.

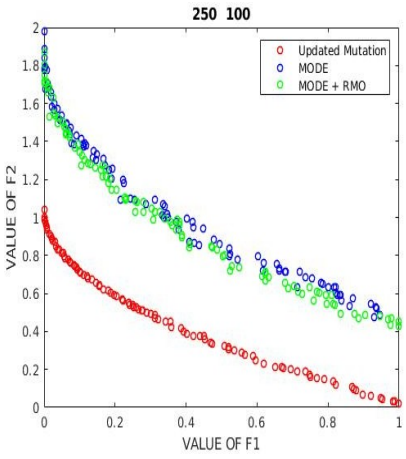

Figure 4 ZDT1, D = 70.

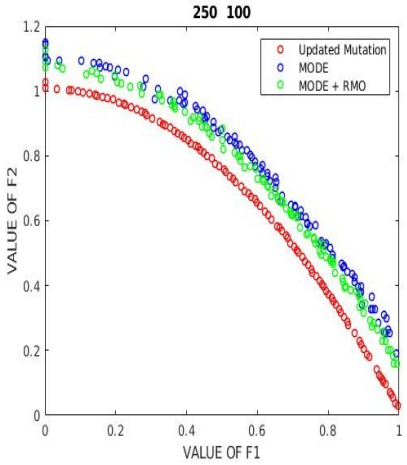

Figure 5 ZDT2, D = 30.

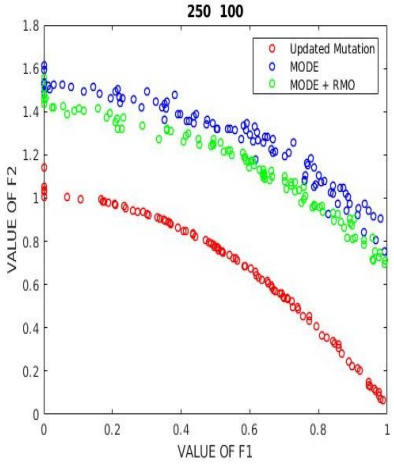

Figure 6 ZDT2, D = 50.

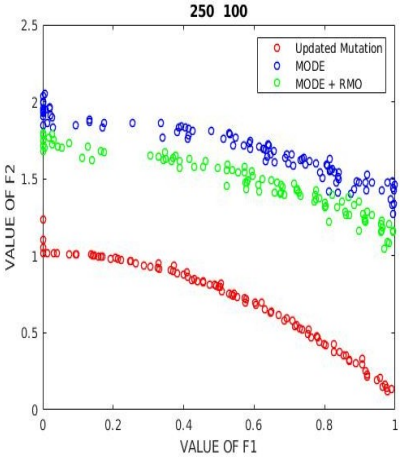

Figure 7 ZDT2, D = 70.

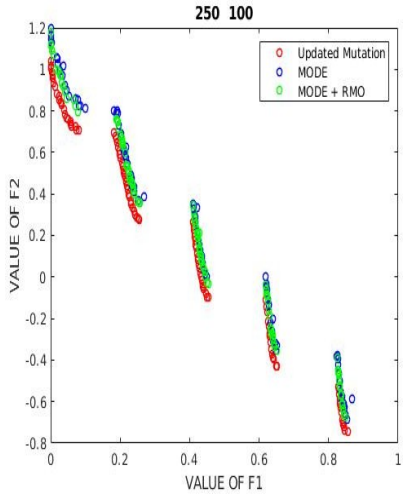

Figure 8 ZDT3, D = 30.

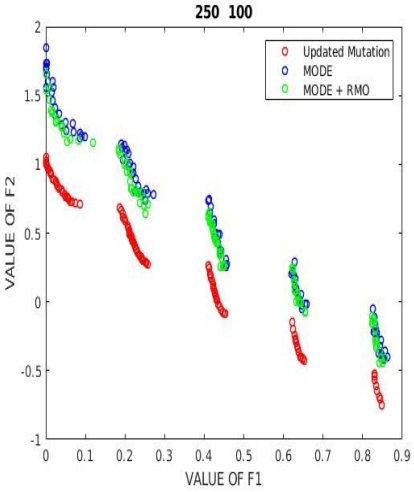

Figure 9 ZDT3, D = 50.

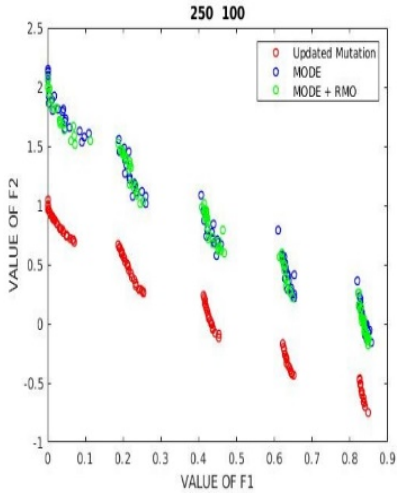

Figure 10 ZDT3, D = 70.

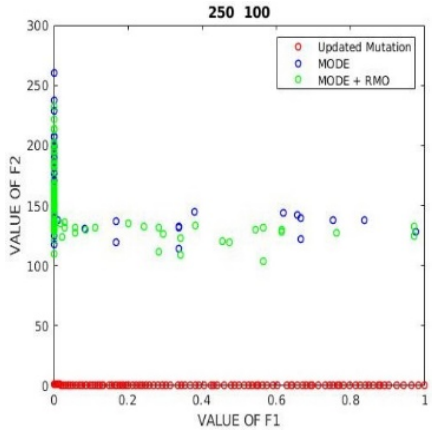

Figure 11 ZDT4, D = 30.

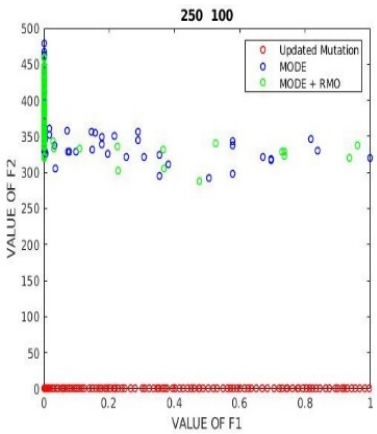

Figure 12 ZDT4, D = 50.

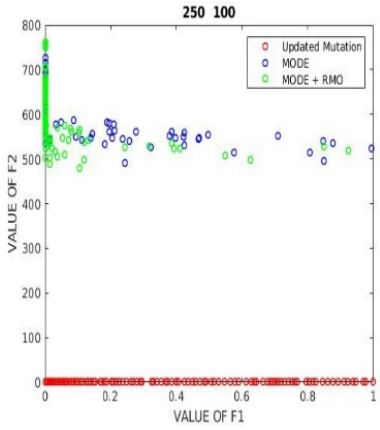

Figure 13 ZDT4, D = 70.

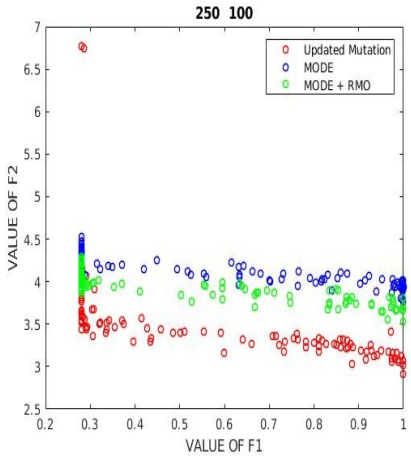

Figure 14 ZDT6, D = 30.

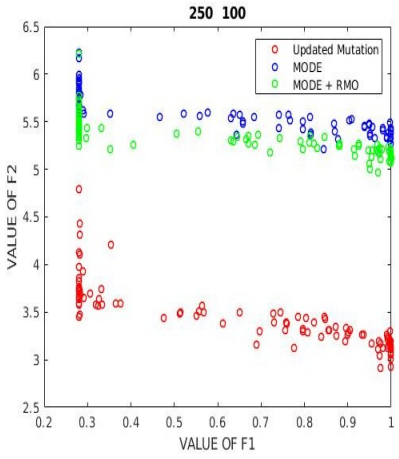

Figure 15 ZDT6, D = 50.

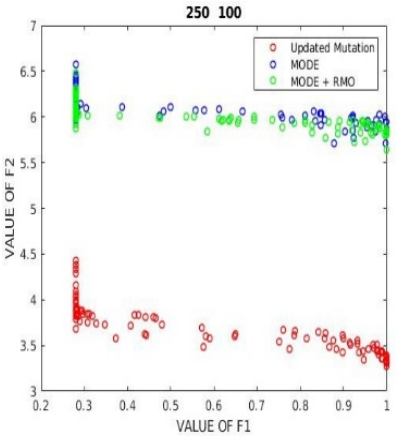

Figure 16 ZDT6, D = 70.

Graphical representation of tri-objective functions:

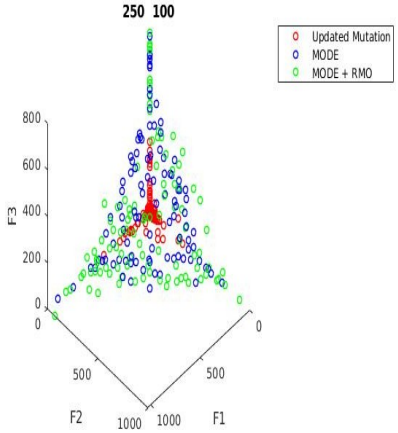

Figure 17 DTLZ1, D = 30.

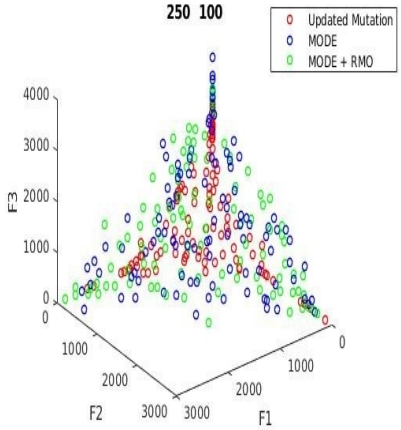

Figure 18 DTLZ1, D = 70.

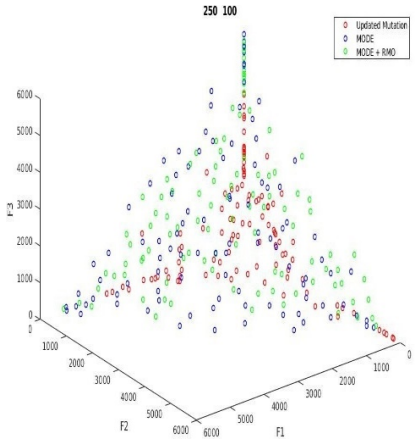

Figure 19 DTLZ1, D = 120.

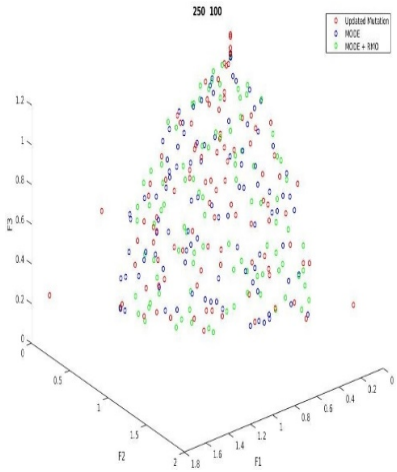

Figure 20 DTLZ2, D = 30.

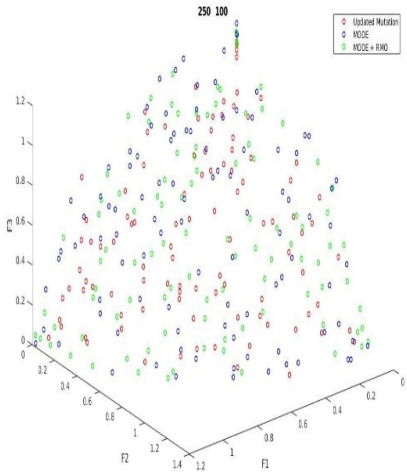

Figure 21 DTLZ2, D = 70.

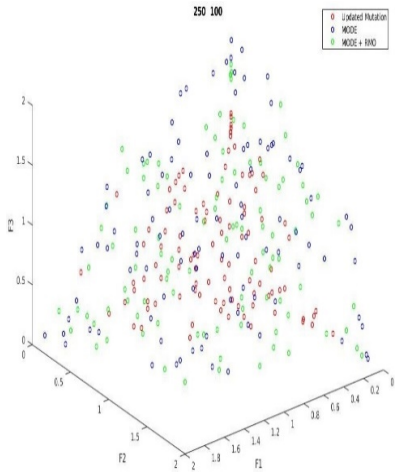

Figure 22 DTLZ2, D = 120.

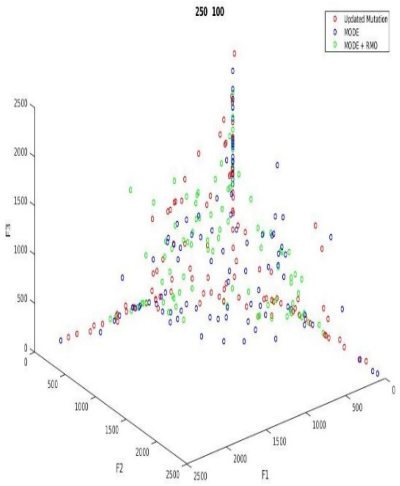

Figure 23 DTLZ3, D = 30.

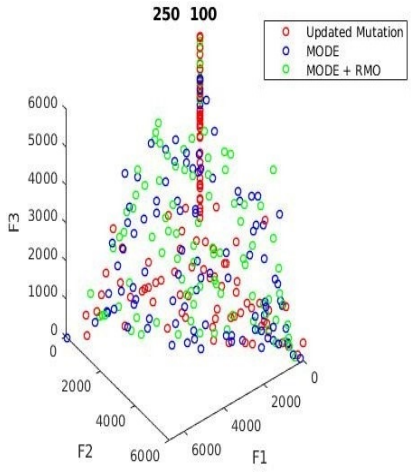

Figure 24 DTLZ3, D = 70

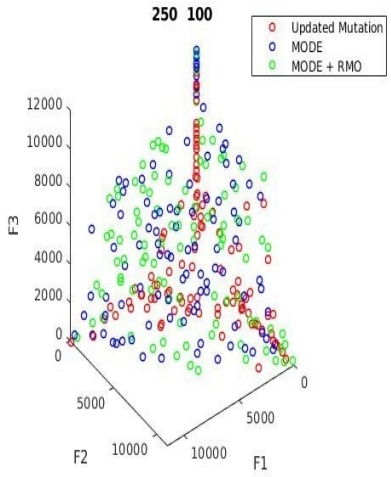

Figure 25 DTLZ3, D = 120.

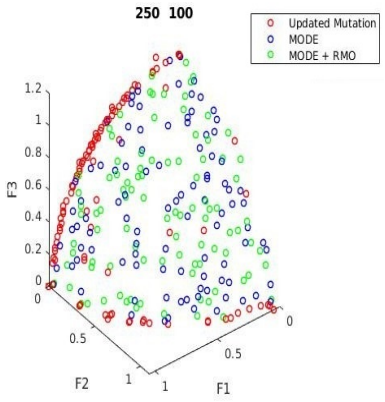

Figure 26 DTLZ4, D = 30.

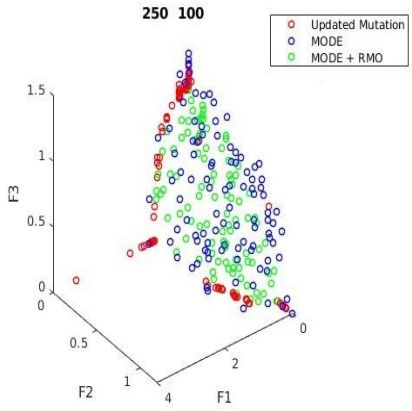

Figure 27 DTLZ4, D = 70.

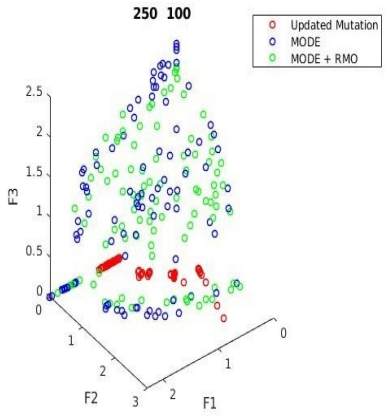

Figure 28 DTLZ4, D = 120.

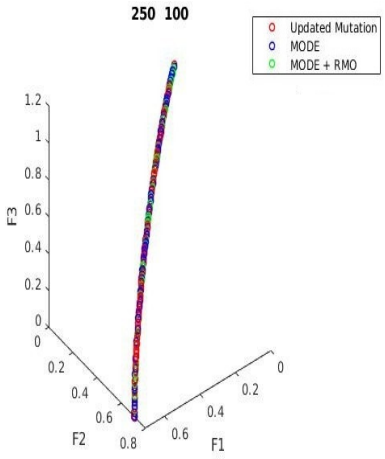

Figure 29 DTLZ5, D = 30.

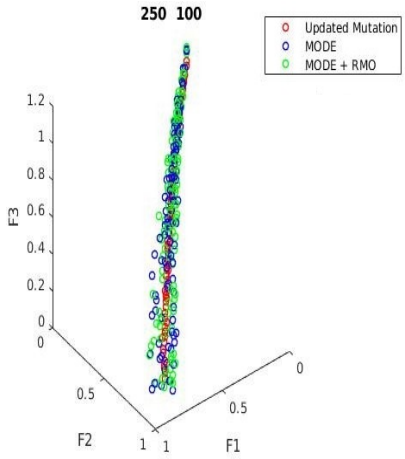

Figure 30 DTLZ5, D = 70.

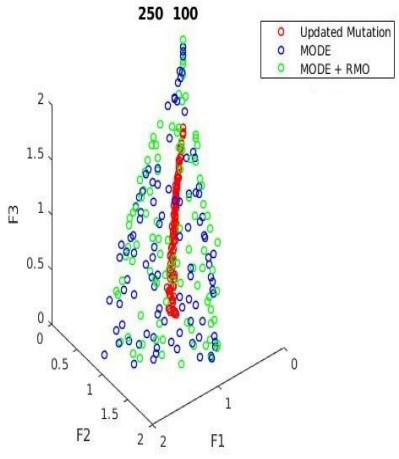

Figure 31 DTLZ5, D = 120.

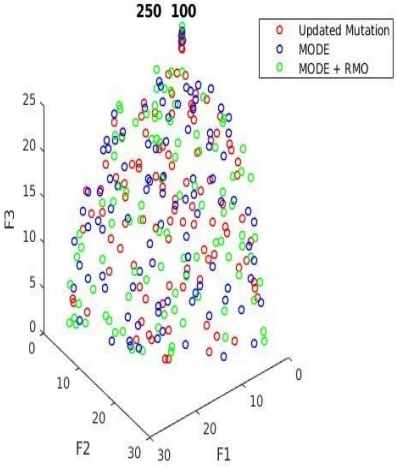

Figure 32 DTLZ6, D = 30.

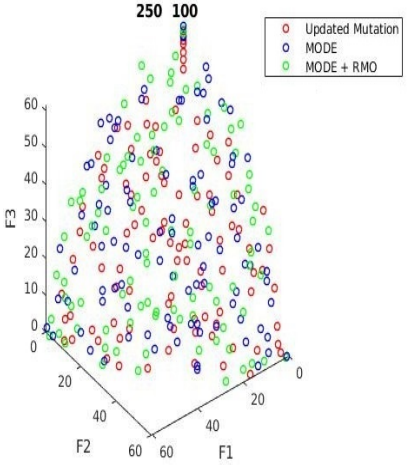

Figure 33 DTLZ6, D = 70.

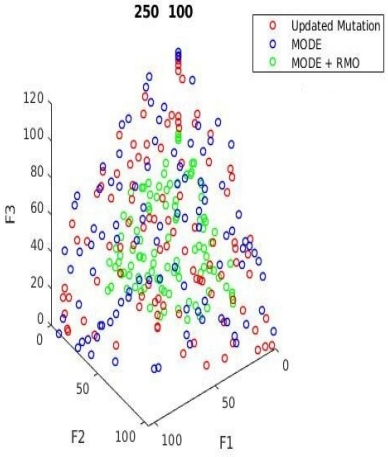

Figure 34 DTLZ6, D = 120.

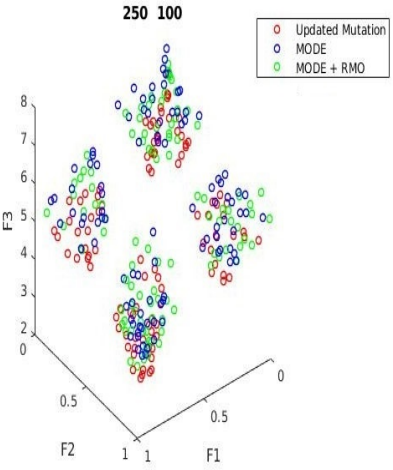

Figure 35 DTLZ7, D = 30.

Conclusions

Although Evolutionary Algorithms are special optimization algorithms used to construct a well-distributed optimal front more quickly, these algorithms are not able to produce optimal solutions globally. DE is capable of resolving various MOPs, but it can be more optimized to improve the solution. So, in this regard, new environmental-based mutations, known as Environment-based Parameters for MODE (EP-MODE), have been presented. EP-MODE successfully maintains diversity, improves performance in terms of high convergence rate, and solves the stagnation problem. The authors in the paper have shown that the performance of the MODE algorithm is successfully increased. In the paper, EP-MODE has been tested on ZDT and DTLZ benchmark functions. The proposed algorithm's efficiency is higher than that of the MODE algorithm and the MODE algorithm with a ranking-based mutation operator. On lower dimensions, the other algorithms can sometimes perform better than the proposed algorithm. The proposed algorithm outperforms better with higher dimension functions in comparison to lower dimension functions. According to the experimental outcomes, EP-MODE can be used to generate Pareto-optimal fronts that are appropriate in terms of convergence and diversity, can be achieved.

In the future, we intend to work on real-life problems involving numerical optimization. This algorithm can be used in many problems that have concerned with MOPs. The proposed algorithm can also be used in the areas of machine learning and artificial intelligence. The updated differential evolution algorithm can be used to optimize the error functions in real-life problems. Despite all these challenges, research has shown that differential evolution can be effectively used in MOO. Still, there is a scope for

improvement in the performance of the algorithm. In the future, the proposed algorithm can also be compared with more evolutionary algorithms.

References

[1] J Cheng, Z Pan, H Liang, Z Gao and J Gao. Differential evolution algorithm with fitness and diversity ranking-based mutation operator. *Swarm Evol. Comput.* 2021; **61**, 100816.

[2] M Bhaumik and K Maity. Multi-response optimization of EDM parameters using grey relational analysis (GRA) for Ti-5Al-2.5 Sn titanium alloy.*World J. Eng.* 2020; **18**, 50-7.

[3] A Singh, A Kumar and A Kumar. *An enhanced differential evolution algorithm with new environmental-based parameters for solving optimization problems*. *In*: A Hassanien, R Bhatnagar and A Darwish (Eds.). International conference on advanced machine learning technologies and applications. Springer, Singapore, 2020, p. 119-29.

[4] AA Bidgoli, S Rahnamayan and H Ebrahimpour-Komleh. *Opposition-based multi-objective binary differential evolution for multi-label feature selection. In*: K Deb, E Goodman, CAC Coello, K Klamroth, K Miettinen, S Mostaghim and P Reed (Eds.). International conference on evolutionary multi-criterion optimization. Springer, Cham, 2019, p. 553-64.

[5] X Chen, W Du and F Qian. Multi-objective differential evolution with ranking-based mutation operator and its application in chemical process optimization. *Chemometr. Intell. Lab. Syst.* 2014; **136**, 85-96.

[6] YN Wang, LH Wu and XF Yaun. Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure. *Soft Comput.* 2010; **14**, 193-209.

[7] W Gong and Z Cai. A multiobjective differential evolution algorithm for constrained optimization. *In*: Proceedings of the IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, China. 2008, p. 181-8.

[8] L Peng, Y Wang and G Dai. *A novel opposition-based multi-objective differential evolution algorithm for multi-objective optimization*. *In*: L Kang, Z Cai, X Yan and Y Liu (Eds.). Advances in computation and intelligence. Springer, Berlin Heidelberg, 2008, p. 162-70.

[9] S Kukkonen and J Lampinen. GDE3: The third evolution step of generalized differential evolution. *In*: Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK. 2005, p. 443-50.

[10] S Kukkonen and J Lampinen. *An extension of generalized differential evolution for multi-objective optimization with constraints*. *In*: X Yao, EK Burke, JA Lozano, J Smith, JJ Merelo-Guervós, JA Bullinaria, JE Rowe, P Tiňo, A Kabán and HP Schwefel (Eds.). Parallel problem solving from nature - PPSN VIII. Springer, Berlin Heidelberg, 2004, p. 752-61.

[11] BV Babu and MML Jehan. Differential evolution for multi-objective optimization. *In*: Proceedings of the Congress on Evolutionary Computation, Canberra ACT, Australia. 2003, p. 2696-703.

[12] NK Madavan. Multiobjective optimization using a Pareto differential evolution approach. *In*: Proceedings of the Congress on Evolutionary Computation (Cat. No. 02TH8600), Honolulu HI, USA. 2002, p. 1145-150.

[13] K Deb, A Pratap, S Agarwal and T Meyarivan. A fast and elitist multiobjective GA: NSGA-II. *IEEE Trans. Evol. Comput.* 2002; **6**, 182-97.

[14] DAV Veldhuizen and GB Lamont. On measuring multiobjective evolutionary algorithm performance. *In*: Proceedings of the Congress on Evolutionary Computation (Cat. No. 00TH8512), La Jolla CA, USA. 2000, p. 204-11.

[15] E Zitzler and L Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* 1999; **3**, 257-71.

[16] CM Fonseca and PJ Fleming. Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. *In*: Proceedings of the 5th International Conference on Genetic Algorithms, Urbana-Champaign IL, USA. 1993, p. 416-23.

[17] RM Storn and K Price. *Differrential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. International Computer Science Institute, Report no. TR-95-012, Berkeley, 1995.

[18] Q Fan and X Yan. Multi-objective modified differential evolution algorithm with archive-base mutation for solving multi-objective p-xylene oxidation process. *J. Intell. Manuf.* 2018; **29**, 35-49.