A Modified Differential Evolution Algorithm and Its Application to Engineering Problems

Musrrat Ali and Millie Pant
Department of Paper Technology
Indian Institute of Technology Roorkee, Saharanpur campus, Saharanpur, India.
E-mail: {musrrat.iitr, millidma}@gmail.com

Ajith Abraham
Machine Intelligence Research Labs (MIR Labs),
Scientific Network for Innovation and Research Excellence, P.O. Box 2259, Auburn, Washington-98071-2259, USA.
E-mail: ajith.abraham@ieee.org

***Abstract*—In the present study a Modified Differential Evolution (MDE) algorithm is proposed. This algorithm is different in three ways from basic DE. For initialization it utilizes opposition-based learning while in basic DE uniform random numbers serve this task. Secondly, in basic DE mutant individual is random while in MDE it is tournament best and finally MDE utilizes only one set of population as against two sets as used in basic DE. The performance of proposed algorithm is investigated and compared with basic differential evolution. The experiments conducted shows that proposed algorithm outperform the basic DE algorithm in all the benchmark problems and real life applications**

Keywords: differential evolution, mutation operator, opposition based learning.

I. INTRODUCTION

Differential evolution, proposed by Storn and Price in 1995 [1] is a relatively new optimization technique compared to evolutionary algorithms (EAs) such as Genetic Algorithms [2], Evolutionary Strategies [3], and Evolutionary Programming [4]. Within a short span of around thirteen years, DE has emerged as one of the most popular techniques for solving optimization problems. However, it has been observed that the convergence rate of DE do not meet the expectations in cases of highly multimodal problems. Several variants of DE have been proposed to improve its performance. Some of the recent versions include greedy random strategy [5], preferential mutation operator [6], self adaptive DE [7], Trigonometric DE [8], opposition based DE [9], neighborhood search DE [10], Parent Centric DE [11], modified differential evolution [12], differential evolution with random localization [13] etc. several recent versions of DE can be found in [14].

In all the above mentioned versions of DE, other than [9], modifications are done in mutation or in update processes.

The proposed MDE algorithm is inspired by three ideas; (1) use of *opposition based learning* to generate the initial population (2) use of tournament best process to generate mutant vector to explore the region around the tournament best individual x_{tb} (say) for each mutated point and finally (3) use of a single set population in contrast to the two set population as in basic DE.

The concept of *opposition based learning* (OBL) was first given in [9] to generate the initial population for a basic DE algorithm. The main idea behind OBL is the simultaneous consideration of an estimate and its corresponding opposite estimate in order to achieve a better approximation for the current candidate solution. In fact, a mathematical proof has been proposed to show that, in general, opposite numbers to the initial set of random numbers are more likely to be closer to the optimal solution than purely random one [15]. Use of a single set of population for DE was suggested in [12] where it was shown use of a single set population helps in reducing the computational time of the DE algorithm.

Motivated by the successful implementation of the above mentioned modifications we decided to club these features together to develop a modified algorithm which we have named as modified DE or MDE.

The remainder of the paper is structured as follows. Section II describes the basics Differential Evolution. Section III presents the proposed MDE. Experimental settings are given in Section IV. Benchmark problems and real life application problems are listed in Section V. Section VI provides comparisons of results. Finally the paper is concluded in section VII

II. DIFFERENTIAL EVOLUTION

Throughout the present study we shall follow *DE/rand/1/bin version* of DE and shall refer to it as basic version. This particular scheme is briefly described as:

DE starts with a population of NP candidate solutions: $X_{i,G}$, $i = 1, \ldots, NP$, where the index i denotes the population and G denotes the generation to which the population belongs. The three main operators of DE are mutation, crossover and selection.

Mutation: The mutation operation of DE applies the vector differentials between the existing population members for determining both the degree and direction of perturbation applied to the individual subject of the mutation operation. The mutation process at each generation begins by randomly selecting three individuals $\{X_{r1}, X_{r2}, X_{r3}\}$ in the population set of (say) NP elements. The i^{th} perturbed individual, $V_{i,G+1}$, is generated based on the three chosen individuals as follows:

$$V_{i,G+1} = X_{r3,G} + F * (X_{r1,G} - X_{r2,G}) \quad (1)$$

Where, $i = 1 \ldots NP$, $r_1, r_2, r_3 \in \{1 \ldots NP\}$ are randomly selected such that $r_1 \neq r_2 \neq r_3 \neq i$,

F is the control parameter such that $F \in [0, 1+]$.

978-0-7695-3879-2/09 $26.00 © 2009 IEEE
DOI 10.1109/SoCPaR.2009.48

Crossover: once the mutant vector is generated, the perturbed individual, $V_{i,G+1} = (v_{1,i,G+1}, \ldots, v_{n,i,G+1})$, and the current population member, $X_{i,G} = (x_{1,i,G}, \ldots, x_{n,i,G})$, are then subject to the crossover operation, that finally generates the population of candidates, or "trial" vectors, $U_{i,G+1} = (u_{1,i,G+1}, \ldots, u_{n,i,G+1})$, as follows:

$$u_{j,i.G+1} = \begin{cases} v_{j,i.G+1} \; if \; rand_j \le C_r \vee j = k \\ x_{j,i.G} \qquad\qquad otherwise \end{cases} \tag{2}$$

Where, j = 1. . . n, k ∈ {1, . . . , n} is a random parameter's index, chosen once for each *i*. The crossover rate, Cr ∈ [0, 1], is set by the user.

Selection: The selection scheme of DE also differs from that of other EAs. The population for the next generation is selected from the individual in current population and its corresponding trial vector according to the following rule:

$$X_{i.G+1} = \begin{cases} U_{i.G+1} \; if \; f(U_{i.G+1}) \le f(X_{i.G}) \\ X_{i.G} \qquad\qquad otherwise \end{cases} \tag{3}$$

Thus, each individual of the temporary (trial) population is compared with its counterpart in the current population. The one with the lower objective function value will survive from the tournament selection to the population of the next generation. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. In DE trial vector is not compared against all the individuals in the current generation, but only against one individual, its counterpart, in the current generation.

III. PROPOSED DE ALGORITHM

In this section we describe the proposed MDE, which uses the concepts of opposition based learning, random localization and one population set. The basic operators of MDE are same as basic DE but still it is different from it three points:

1. MDE differs from basic DE in the initialization phase where MDE utilizes opposition based learning method while DE uses uniform random numbers for initialization of population.
2. In mutation step MDE uses best individual of three points as mutant individual while in DE it is random (there is an equal chance of all these three for being selected as mutant individual).
3. MDE maintain one population set while DE maintains two population sets, one current population and second advanced population (for next generation). The population is updated as the better individual is found. Also the newly found individual can take part in generation of new individual in current generation.

A point to point comparison of two algorithms is given in Table I.

IV. EXPERIMENTAL SETUP

In order to make a fair comparison of MDE and basic DE, we have used C++ rand *( )* function to generate initial population for both the algorithms. The number of individuals in the population is taken 10*n. Value scaling factor F is taken as 0.5 which is neither too high nor low and therefore maybe considered as a good initial choice. Very small values of crossover constant C_r makes the convergence very slow whereas large values of C_r may end up in premature convergence. In the present study we have taken C_r =0.5. All the algorithms are executed on a PIV PC, using DEV C++, thirty times for each problem. In every case, a run was terminated when the function values of all points in population S were identical to an accuracy of five decimal places, i.e., $|f_{max} - f_{min}| \le 10^{-4}$ or when the maximum number of function evaluations (NFE =10^6) was reached.

V. BENCHMARK PROBLEMS

The performance of the proposed algorithm is tested on a set of ten benchmark and two application problems taken from literature [9]. First five functions f_{EP}, f_{CB6}, f_{GP}, f_{H3} and f_{CV} are with fixed dimension while second five f_{RB}, f_{ACK}, f_{SWF}, f_{GW} and f_{ZA} are scalable in nature. Scalable problems are tested for dimensions 10, 30 and 50. Real life application problem is taken from [16].

VI NUMERICAL RESULTS AND COMPARISONS

A. Comparison between DE and MDE

In this section we compare MDE with the basic DE algorithm in terms of average fitness of function values, standard deviation, and t- values for which the results are listed in Table II. For scalable problems the dimension is taken as n=30. Table III provides number of function evaluations (NFE), percentage improvement in terms of number of functions evaluations and average time taken for the execution of algorithms. As it is clear from the Table II that in term of fitness function value and standard deviation both the algorithms give more or less similar results although in some cases MDE performs slightly better than classical DE. On the basis of t-values, last column of the Table II, we can conclude that there is a significant difference between both the algorithms at 5% level of significance. The superior performance of the proposed MDE is more evident from Table III, which gives the average number of functions evaluations from which we can see that MDE takes less number of function evaluations to achieve the required fitness in comparison to the basic DE in all cases except for Rosenbrock function (f_{RB}), in which both the algorithms approach to the maximum number of function evaluation (NFE=10^6). In terms of improvement in number of function evaluation MDE reduces the number of function evaluation up to 44.5% for function f_{CB6}. If we talk about overall reduction in number of function evaluation, it is more than of 29.99%. But for function f_{RB}, in terms of function evaluation there is no improvement, both algorithms take maximum number of function evaluation. Also from Table III, it can be seen that MDE takes less run time in comparison to basic DE but in case of function f_{RB}, where number of function evaluation is same, MDE takes more time than basic DE. Performance curves (convergence

graphs) of few selected functions are given in Fig (1) – Fig (4). From these illustrations also it is evident that the convergence of proposed algorithm is faster than basic DE.

B. *Influence of Dimensionality.*

The performance of the proposed MDE is further compared with the basic DE for scalable problems of dimension 10, 30 and 50. The results obtained are summarized in Tables IV which gives the results of MDE and DE algorithms in terms of average fitness and average number of function evaluations. According to results obtained, MDE surpasses DE on 11 cases while DE outperforms MDE in 4 cases out of 15 cases in term of average fitness. If we talk in term of average number of function evaluation (NFE) it is less in all cases for MDE except for f_{RB} and f_{SWF} where it is same for both the algorithms.

C. *Numerical results of Application problems*

In order to further validate the performance of MDE algorithm we used it for solving two real life problems; Transformer design [16] and transistor modeling [16]. Out of these problems, the first problem is constrained in nature, while the second is unconstrained. For handling constraints, we have used the method proposed by Deb [17].

The numerical results of the real life problems are given in Table V. experimental settings for real life problems are same as that of benchmark problems. A run is terminated when an accuracy of 10^{-04} i.e. $|f_{min} - f_{max}| \leq 10^{-4}$ is reached and then fitness standard deviation NFE and time is stored in Table V. Once again from this Table we can observe the superior performance of the proposed MDE algorithm in terms of NFE and time which are quite less than the basic DE in all the cases

VII DISCUSSION AND CONCLUSIONS

In this paper we proposed a modified version of basic DE called MDE. The simulation of results showed that the proposed algorithm is quite competent for solving problems of different dimensions in less time and less number of function evaluations without compromising with the quality of solution. The set of problems considered, though small and limited show the promising nature of MDE. Only for Rosenbrock function f_{RB} MDE took more time than the basic DE, although the number of functions evaluations are same. However, the work is still in the preliminary stages and more modifications may be added to it to make it more robust.

TABLE I. COMPARISON OF TWO ALGORITHMS.

DE	MDE
Initialization: Construct an initial population S of NP individuals, dimension of each vector being n, using the following rule: $X_{i,j}= X_{min,j} + rand(0, 1)(X_{max,j}-X_{min,j})$, Where $X_{min,j}$ and X_{max} are lower and upper bound for j^{th} component respectively and rand(0,1) is a uniform random number between 0 and 1.	*Initialization:* Randomly construct a population P of NP individuals, dimension of each vector being n, using the following rule: $X_{i,j}= X_{min,j} + rand(0, 1)(X_{max,j}-X_{min,j})$, Where $X_{min,j}$ and X_{max} are lower and upper bound for j^{th} component respectively and rand(0,1) is a uniform random number between 0 and 1. Construct another population *OP* of NP individuals using the following rule: $y_{i,j} = X_{min,j} + X_{max,j} - P_{i,j}$ Where $P_{i,j}$ are the points of population P. Construct initial population S taking NP best individuals from union of these two populations.
Mutation: Select randomly three distinct individuals X_{r1}, X_{r2} and X_{r3} from population S and perform mutation using formula: $V_i = \boldsymbol{X_{r1}} + F \times (X_{r2} - X_{r3})$ Where individual X_{r1} is random (i.e. it may be any one from these three individuals).	*Mutation:* Select randomly three distinct individuals X_{r1}, X_{r2} and X_{r3} from population S and perform mutation using formula: $V_i = \boldsymbol{X_{tb}} + F \times (X_{r2} - X_{r3})$ Where individual X_{tb} is best of these three individuals and X_{r2}, X_{r3} are the remaining two.
Crossover: Perform crossover according to equation (2).	*Crossover:* Perform crossover according to equation (2).
Selection: Calculate the objective function value at new generated individual. Choose better of the two (function value at target and trial point) using equation (3) for next generation's population.	*Selection:* Calculate the objective function value at new generated individual. If it is better than target individual then replace target individual by this new individual in current population.

TABLE II. MEAN FITNESS, STANDARD DEVIATION OF FUNCTIONS IN 30 RUNS AND T VALUE.

Fun	Fitness		Standard deviation		t-value
	DE	MDE	DE	MDE	
f_{EP}	-0.99999	-0.99999	6.98197e-07	6.21603e-07	0.00
f_{CB6}	-1.03163	-1.03163	8.17617e-07	6.71679e-07	0.00
f_{GP}	3.00000	3.00000	1.07046e-06	5.40168e-07	0.00
f_{H3}	-3.86230	-3.86230	1.46942e-06	8.85213e-07	0.00
f_{CV}	1.65825e-06	2.51160e-06	1.86572e-06	2.61837e-06	1.45
f_{RB}	13.83440	6.91061	8.55350e-02	6.24724e-02	358.03
f_{ACK}	1.42800e-04	1.35818e-04	1.79218e-05	1.24837e-05	1.75
f_{SWF}	7.28960e-04	7.30199e-04	3.85744e-06	8.73741e-06	0.71
f_{GW}	4.62272e-05	4.71135e-05	9.03396e-06	8.84722e-06	0.38
f_{ZA}	4.50199e-05	4.15536e-05	7.48947e-06	1.09776e-05	1.42

TABLE III. NUMBER OF FUNCTIONS EVALUATION, % IMPROVEMETS AND AVERAGE TIME IN SECONDS

Fun	NFE		% Improvement	Time	
	DE	MDE		DE	MDE
f_{EP}	833	568	31.812	0.10	0.10
f_{CB6}	1020	566	44.509	0.11	0.10
f_{GP}	970	630	35.051	0.11	0.10
f_{H3}	1170	843	27.948	0.10	0.10
f_{CV}	12716	8844	30.449	0.2	0.10
f_{RB}	1000000	1000000	0.000	33.12	34.23
f_{ACK}	259410	176670	31.895	18.90	14.70
f_{SWF}	366570	249720	31.876	5.20	4.30
f_{GW}	224910	150480	33.093	17.10	12.50
f_{ZA}	214890	143220	33.351	44.10	28.30

TABLE IV. MEAN FITNESS AND AVERAGE OF FUNCTION EVALUATIONS IN 30 RUNS FOR FUNCTIONS.

Fun	Fitness and NFE(n=10)		Fitness and NFE(n=30)		Fitness and NFE(n=50)	
	DE	MDE	DE	MDE	DE	MDE
f_{RB}	1.88258e-02 843100	3.27520e-06 162810	1.38344e+01 1000000	6.91061e+00 1000000	4.28553e+01 1000000	3.92105e+01 1000000
f_{ACK}	5.72106e-05 25910	8.96282e-05 16500	1.42800e-04 259410	1.35818e-04 176670	2.09244e-04 917750	1.97978e-04 629800
f_{SWF}	2.42898e-04 26170	2.38916e-04 18130	7.28960e-04 366570	7.30199e-04 249720	4.50132e+03 1000000	4.68756e-03 1000000
f_{GW}	1.20440e-05 96790	1.36853e-05 62840	4.62272e-05 224910	4.71135e-05 150480	8.04424e-05 764150	7.64880e-05 519200
f_{ZA}	1.52691e-05 17170	1.37965e-05 11400	4.50199e-05 214890	4.15536e-05 143220	8.57237e-05 851000	7.83553e-05 580900

TABLE V. NUMERICAL RESULTS OF REAL LIFE APPLICATION PROBLEMS.

Solutions found of transformer design problem				
	DE	MDE	NFE	Time
x1	5.23203	5.23257	DE=344292 MDE=225090	DE=3.1 MDE=2.0
x2	4.72749	4.7292		
x3	10.0924	10.0906		
x4	13.618	13.6168		
x5	0.827257	0.827318		

x6	0.736074	0.73587		
g1	-3.77599e-05	8.05321e-05		
g2	9.97301e-05	9.99378e-05		
f(X)	86.6225	86.6225		
Solutions found of transistor modeling problem				
	DE	MDE	NFE	Time
x1	0.901341	0.901336	DE=780768 MDE=458388	DE=17.3 MDE=9.9
x2	0.891174	0.891053		
x3	3.87757	3.87933		
x4	3.94643	3.94662		
x5	5.32623	5.32509		
x6	10.6239	10.6162		
x7	0.0	0.0		
x8	1.08914	1.08881		
x9	0.705575	0.706727		
f(X)	0.0543713	0.0543658		

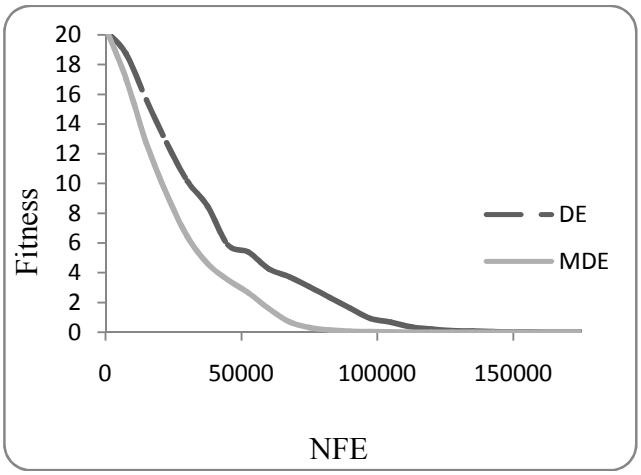

Figure 1. Performance curves of Ackley function.

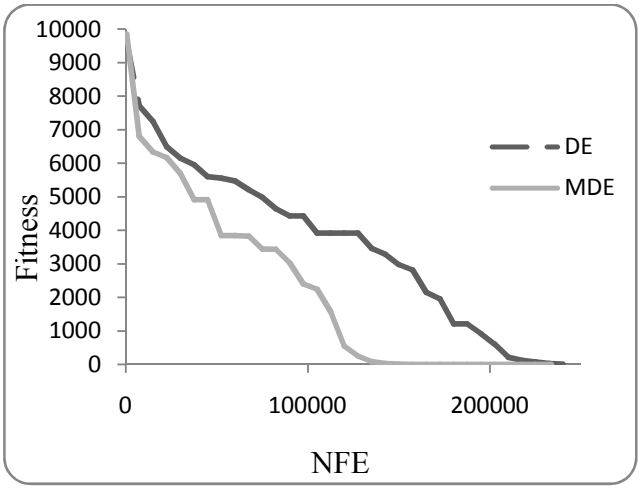

Figure 2. Performance curves of Schawefel function.

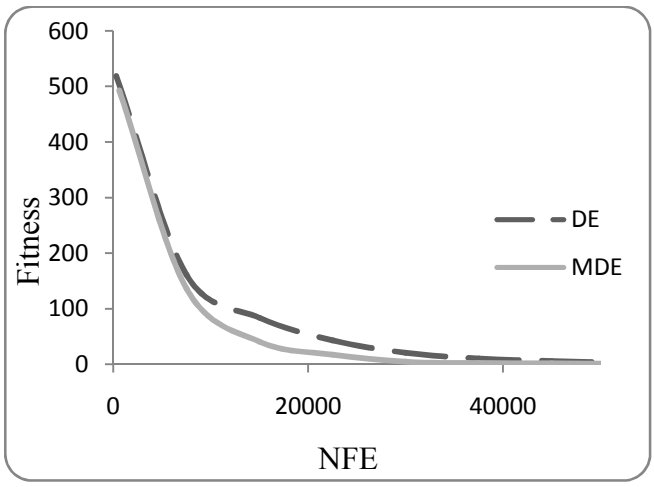

Figure 3. Performance curves of Griewenk function.

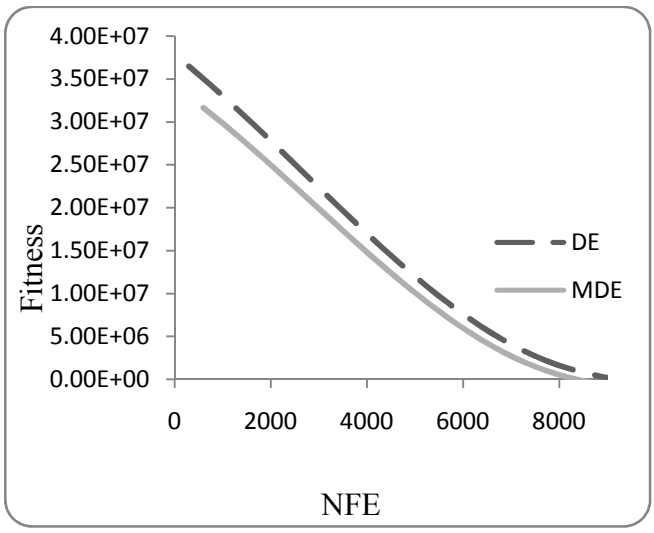

Figure 4. Performance curves of Zakharov function.

REFERENCES

[1] R. Storn and K. Price, Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, *Technical Report TR-95-012, Berkeley, CA*, 1995.

[2] D. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley. 1989.

[3] T.Back, F. Hoffmeister, H.Schwefel, A survey of evolution strategies. In: Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications, 1991.pp. 2–9.

[4] L.Fogel, Evolutionary programming in perspective: The top-down view. In: Zurada, J.M., Marks, R. Jr., Robinson, C. (Eds.), Computational Intelligence: Imitating Life. IEEE Press, Piscataway, NJ, USA. 1994.

[5] Paul K. Bergey, Cliff Ragsdale, Modified differential evolution: a greedy random strategy for genetic recombination, *Omega The International Journal of Management Science 33*, 2005, pp 255-265.

[6] M.M.Ali, Differential evolution with preferential crossover, *European Journal of Operation Research 181*, 2007 pp.1137-1147.

[7] A.Salman, A.P.Engelbrecht, M.G.H.Omran, "Empirical analysis of self adaptive differential evolution", *European Journal of operational research 183,* 2007 pp 785-804.

[8] Hui-Yuan Fan, Jouni Lampinen, "A Trigonometric Mutation Operation to Differential Evolution," *Journal of Global Optimization* 2003, 27:105-129.

[9] Shahryar Rahnamayan, H.R. Tizhoosh, M.M.A.Salama, opposition based differential evolution, *IEEE transactions on evolutionary computation,* 2007 pp 1-16.

[10] Z. Yang, J. He, and X. Yao, *Making a Difference to Differential Evolution*, in *Advances in Metaheuristics for Hard Optimization*, Z. Michalewicz and P. Siarry (eds.), pp 415-432, Springer, 2007.

[11] Millie Pant, Musrrat Ali and V.P. Singh, "Differential Evolution with Parent Centric Crossover", *Second UKSIM European Symposium on Computer Modeling and Simulation* 2008, 141 – 146.

[12] B.V.Babu and R.Angira, modified differential evolution (MDE) for optimization of non-linear chemical processes, computer and chemical engineering 30, 2006, 989-1002.

[13] P.Kaelo and M.M.Ali, a numerical study of some modified differential evolution algorithms, European journal of operational research 169, 2006, 1176-1184.

[14] U. K. Chakraborty (Ed.) *Advances in Differential Evolution*, Springer-Verlag, Heidelberg, 2008.

[15] Shahryar Rahnamayan, H.R. Tizhoosh, M.M.A.Salama, opposition versus randomness in soft computing techniques, *Applied soft computing,* 2006.

[16] W.L.Price, global optimization by controlled random search, journal of optimization theory and applications 40(3), 333-348, 1983.

[17] K. Deb, an efficient constraint handling method for genetic algorithm, computer method in applied mechanics and engineering, 186(2/4), pp 311-338, 2000.