Enhanced Mutation Strategy for Differential Evolution

Pravesh Kumar
DPT, Indian Institute of Technology, Roorkee, India
praveshtomariitr@gmail.com

Millie Pant
DPT, Indian Institute of Technology, Roorkee, India
millidma@gmail.com

***Abstract*—It is well known that mutation plays a very important role in the successful performance of Differential Evolution (DE) algorithm. The proposed scheme named Modified Random Localization (MRL) is based on strategically selecting the individuals from the entire search space rather than choosing them randomly as in basic DE. The corresponding DE variant named MRL-DE is analyzed on a set of 8 traditional benchmark functions and 6 nontraditional shifted functions. Numerical and statistical results indicate the competence of the proposed MRL-DE for solving unconstrained global optimization problems.**

Keywords-differential evolution, mutation, perturbed vector, randomized localization.

I. INTRODUCTION

Differential Evolution (DE), a kind of evolutionary algorithm (EA), was proposed by Storn and Price [1] in 1995. It has emerged as a simple and powerful algorithm for global optimization over continuous spaces. It has many attractive characteristics, such as compact structure, ease of use, good convergence speed and robustness [2]. Its effectiveness and efficiency has been successfully demonstrated in many application fields such as pattern recognition [3], chemical engineering [4], and many other science and engineering fields [5].

Although, DE shares the similar operators like mutation, crossover and selection as that of Genetic Algorithms (GA), it is the working of these operators that makes DE different from GA and also from other population based EAs. Mutation is the main operator of DE as it guides the movement of solution vectors towards the global optimum. Initially, a single mutation strategy was suggested by Storn and Price. Later on they suggested nine more strategies for DE. Considering the significant effect of the mutation operator on the performance of DE, researchers have suggested enhanced mutation schemes to further improve the performance of DE. Some modified mutation schemes available in literature are Trigonometric Mutation (TDE) [6], Cauchy Mutation [7], Mixed mutation strategy based DE [8], DE with Laplace mutation [9], DE with random localization, (DERL) [10], Self adaptive DE (SaDE) [11], adaptive DE with optional external archive (JADE) [12], DE with global and local neighborhood (DEGL) [13], two latest versions in which modifications are done in mutation operator are [14],[15].

A recent literature survey of various DE variants can be found in [16] and [17].

In the present study we have proposed a simple scheme called Modified Random Location (MRL) for selection of three different vectors to perform mutation operation. In MRL we divide the search space into three regions on the basis the fitness of the solution vectors. From these regions we select the candidates which will take part in the mutation process to generate the mutant vector. By selecting the candidates from different regions we try to extract maximum information from the search space. This is likely to be beneficial in real life scenarios where no a-priori information is available about the search space.

It is expected that the proposed scheme will enhance the searching capabilities of basic DE.

The rest of paper is organized as follows: Section II provides a compact overview of DE. Section III presents the proposed MRL-DE algorithm with graphical description. Benchmark problems and experimental settings are given in Section IV. Results and comparisons are reported in Section V, Statistical analysis is shown in Section VI and finally the conclusions derived from the present study are drawn in Section VII.

II. BASIC DIFFERENTIAL EVOLUTION (DE)

DE is a stochastic, population-based direct search method for optimizing real-valued functions of continuous variables. The working of DE is as follows: First of all, the individuals are initialized with uniformly distributed random numbers and are evaluated using the fitness function provided. Then the following will be executed until a stopping a criterion is met;

a) *Mutation*: For a *D*-dimensional search space, for each target vector X_i^G at the generation *G*, its associated mutant vector is generated via a certain mutation strategy. The most often used mutation strategy implemented in the DE is given by:

$$DE/rand/1/bin: V_i^{G+1} = X_{r1}^G + F*(X_{r2}^G - X_{r3}^G) \quad (1)$$

where $r_1, r_2, r_3 \in \{1,2,....,NP\}$ are randomly chosen integers, different from each other and also different from the running index *i*. Here *NP* represents the population size. *F* (>0) is a scaling factor which controls the amplification of the difference vector $(X_{r2}{}^G - X_{r3}{}^G)$

b) *Crossover*: Once the mutation phase is over, crossover is performed between the target vector and the mutated vector to generate a *trial* vector for the next generation. The mutated individual, $V_i^{G+1} = (v_{1,i}{}^{G+1}, v_{2,i}{}^{G+1} \ldots, v_{D,i}{}^{G+1})$ and the current population member (target vector), $X_i^G = (x_{1,i}{}^G, x_{2,i}{}^G \ldots, x_{D,i}{}^G)$ are then subject to the crossover

operation, that finally generates the population of candidates, or trial vectors, $U_i^{G+1} = (u_{1,i}^{G+1}, u_{2,i}^{G+1} \ldots, u_{D,i}^{G+1})$ as follows:

$$u_{j,i}^{G+1} = \begin{cases} v_{j,i}^{G+1} & if \quad rand_j \le Cr \vee j = k \\ x_{j,i}^{G} & otherwise \end{cases} \tag{2}$$

where j, $k \in \{1,\ldots, D\}$ k is a random parameter index, chosen once for each i, C_r is the crossover probability parameter whose value is generally taken as $C_r \in [0, 1]$.

c) *Selection*: The final step in the DE algorithm is the selection process. Each individual of the temporary (trial) population is compared with its target vector in the current population. The one with the lower objective function value survives the tournament selection and goes to 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.

$$X_i^{G+1} = \begin{cases} U_i^{G+1} & if \quad f(U_i^{G+1}) \le f(X_i^{G}) \\ X_i^{G} & otherwise \end{cases} \tag{3}$$

III. PROPOSED MRL-DE

We know that the mutation operation for basic DE usually follows no rule of selection of three random vectors X_{r1}, X_{r2} and X_{r3} from population except for the fact that these should be mutually different from each other and also from the target vector X_i. Here, we are not sure about the position of these vectors. These vectors may be selected from either a small cluster or may be selected very far from each others. This procedure may lead to the loss of some important information about the search space.

In the present study, we propose a new mutation scheme where instead of having a random selection we make use of localized selection where each solution vector represents a particular region of the search space.

The proposed strategy is very simple. After sorting the initial population according to the fitness function value, we divide it into three regions say *R-I, R-II* and *R-III*.

R-I represent the region having the fittest individuals or the elite individuals.

R-II represents the set of next best individuals and

R-III represents the remaining.

Now, we select the three candidates for mutation; X_{r1}, X_{r2} and X_{r3} from *R-I R-II* and *R-III* respectively.

We can easily see that this scheme, tries to cover the maximum of the search space making it more exploratory in nature.

The size of *R-I R-II* and *R-III* are taken as $NP*\alpha\%$, $NP*\beta\%$ and $NP*\gamma\%$ respectively. Where α, β *and* γ are integers to be decided by the user. The proposed strategy is named Modified Random localization (MRL) and the corresponding DE variant is called MRL-DE.

The proposed MRL is in contrast to [10], because here the candidate vectors are selected strategically so as to cover most of the search space. Further, in MRL-DE we select X_{r1} (base vector) from *R-I* which is the fittest part of the population implying that X_{r1} is always fitter than X_{r2} and X_{r3} and as suggest by Kaelo and Ali [10], if base vector is fitter from the difference vectors then convergence speed of DE will be better.

Since, X_{r1} (base vector) always taken from $\alpha\%$ part of population. So the base vector will be near to global best vector of population at any generation. Here we would like to mention that once we have decided the size of R1, the remaining regions R1 and R2 are taken as 50% each of the remaining population. That is to say, if R1 consists of 20 individuals, R2 and R3 will consist of 40 individuals each for a population size of 100.

In Fig-1 the graphical description of proposed selection scheme X_{r1}, X_{r2} and X_{r3} is given and in TABLE I, pseudo code is given;

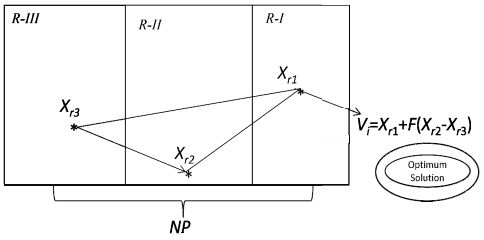

Figure 1. Graphical description of proposed selection scheme

TABLE I. PSEUDO-CODE

```
    Begin
2   Generate uniformly distribution random population P= {X_i^G,
    i=1, 2,...NP}.
3   X_i^G = X_lower +(X_upper –X_lower)*rand(0,1), where i =1, 2,..,NP
4   Evaluate f(X_i^G)
5   Sort(f(X_i^G))
6   While (Termination criteria is not met )
7   {
8     For i=1:NP
9       {
10        Divide P into 3 regions say R-I, R-II ans R-III where
    size of R-I, R-II and R-III are
    (NP*α%), (NP*β%) and (NP*γ%) respectively
11          Now Select three vectors X_r1^G, X_r2^G and X_r3^G as;
12          do{r1 = (int)(rand(0,1)*α)}while(r1 == i)
            do{r2 = (int)(α + rand(0,1)*β)}while(r2 == i)
            do{r3 = (int)(β + rand(0,1)*γ)}while(r3 == i)
13          Perform mutation operation as defined by "(1)"
14          Perform crossover operation as defined by "(2)"
15          Evaluate f(U_i^G)
16          Select fittest vector from X_i^G and U_i^G+1
            next generation by using "(3)"
17        }
18    Generate new population Q={X_i^G+1 , i=1,2,...NP}
19  } /* end while loop*/
20  END
```

IV. BENCHMARK PROBLEMS AND EXPERIMENTAL SETTINGS

We have tested the proposed MRL-DE algorithm on 8 standard benchmark problem from [12], [18] and with 6 nontraditional shifted benchmark problems from [21], [22].

These test problems are given in TABLE-II.

TABLE II. TEST PROBLEMS

Standard Benchmark Function		Shifted Benchmark Function	
F_1	Sphere Function	SF_1	Shifted Sphere Function
F_2	Ackley Function	SF_2	Shifted Schwefel 2.21 Function
F_3	Rastrigin Function	SF_3	Shifted Rosenbrock Function
F_4	Rosenbrock Function	SF_4	Shifted Rastrigin Function
F_5	Noise Function	SF_5	Shifted Griewank Function
F_6	Schwefel 1.2 Function	SF_6	Shifted Ackley Function
F_7	Schwefel 2.22 Function		
F_8	Griewank Function		

In order to investigate the performance of the proposed MRL-DE we compared it with the basic DE algorithm (SDE) and other modified variants of DE as; DERL [10]. SaDE [11], JADE [12], LeDE [18], and jDE [19].

Basic DE and proposed MRL-DE are implemented in Dev-C++ and the experiments are conducted on a computer with 2.00 GHz Intel (R) core (TM) 2 duo CPU and 2- GB of RAM.

Parameter settings and performance criteria are taken as follows:

1) Parameter Settings: The parameter setting is taken after consulting various literatures. This setting is kept same in order to maintain uniformity of results.

TABLE III. PARAMETER SETTINGS

Pop size (*NP*)	100 [2], [18], [20]
Dimension (*D*)	30 [2], [18]
Scale Factor (*F*), and Crossover rate (*Cr*)	0.5 and 0.9 respectively [12], [18]
α	20,30,40 and 50
β , γ	$(NP - \alpha)/2$
Value to reach (*VTR*)	10^{-08} except for F_5 where *VTR* 10^{-02} [12], [18].
Max NFE	150000 for sphere and Ackley, 200000 for Schwefel 2.22, and Griewank. 300000 for Restrigin and Noisy function. 500000 for Schwefel 1.2 and Rosenbrock. [12], [18], D*10000 for Special Benchmark Problems [18].

2) Performance Criteria: Four performance criteria are selected from the literature [20],[21] to evaluate the performance of the algorithms. These criteria are:

a) Error: The average error $|f_{opt} - f_{globqal}|$ is recorded by using predefined maximum NFEs, in each run. Also the average and standard deviation of the fitness values are calculated.

b) NFEs : The number of fitness function evaluations (NFEs) is recorded when the VTR is reached before to reach maximum NFE. i.e we set the termination criteria as $|f_{opt} - f_{globqal}| \le VTR$ and record average NFE of successful run over 50 runs.

*c) Convergence Graph :*The convergence graphs show the mean fitness performance of the total runs, in the respective experiments.

d) Acceleration rate (AR) in %: This criterion is used to compare the convergence speeds between MRL-DE and other algorithms *[2], [20]*. It is defined as follows: $AR = \frac{NFE_A - NFE_B}{NFE_A}\%$,.

where *A* and *B* are different algorithms.

V. NUMERICAL RESULTS AND COMPARISONS

A. Comparison with DE and DERL

In TABLE IV, comparisons of MRL-DE are given with basic DE and DERL in term of average NFE.

For MRL-DE it is very important to choose the size of *R-I* i.e. value of α should be chosen in a manner such that it is neither very small (or it will be like a greedy selection strategy) nor it is very large (or it will behave like basic DE). After conducting a series of experiments, we observed that when α is taken as 20% of the total population (NP), the results are fairly good.

From the TABLE IV we can see that for every benchmark problem MRL-DE (at α =20) takes lesser NFE to achieve the desired accuracy in comparison of DE and DERL [17].

The total NFEs are taken by DE and DERL are 1514400 and 840500 respectively while total NFEs taken by MRL-DE are 568410, 780450, 1000160 and 1246480 at α =20, 30, 40 and 50 respectively. Hence acceleration rate of DERL with respect to DE is 44.49% while the acceleration rate of MRL-DE with respect to DE are 62.46%, 48.46%, 9.99% and 1.76% α =20, 30, 40 and 50 respectively.

So here we can see that as we increase the size of *R-I* (i.e. α =30, 40 and 50) the NFEs taken by MRL-DE also increases and at α =30 it behaves like DERL and at α =50, it behaves like basic DE algorithm.

TABLE IV. EXPERIMENTAL RESULTS AND COMPARISONS WITH DE AND DERL IN TERM OF AVERAGE NFE OF 50 RUNS FOR STANDARD BENCHMARK PROBLEMS

F	DE	DERL	MRL-DE			
			α =20	α =30	α =40	α =50
F_1	103530	54880	40150	55350	73660	91840
F_2	163990	87240	62050	86510	113200	141600
F_3	NA	NA	NA	NA	NA	NA
F_4	442780	263050	146400	183470	229000	280130
F_5	110640	72480	58280	86020	102770	117000
F_6	410720	212550	151390	216520	283540	365460
F_7	173740	92210	68120	94620	122910	156070
F_8	109000	58090	42020	57960	75080	94740
$\sum$	*1514400*	*840500*	*568410*	*780450*	*1000160*	*1246840*

B. Comparison with jDE, SaDE, JADE and LeDE

In TABLE-V comparisons of MRL-DE is given with jDE, SaDE, JADE and LeDE in term of average error and standard deviation of 50 runs. Although, comparison with these algorithms may not sound fair because they have a different structure, these algorithms are taken for comparison because these are some of the recently proposed variants of DE and have given good results.

Here all results are taken as mention in [18]. From the table we can see that MRL-DE gives better result in case of F_2, F_4, F_7 and F_8 than all others while in case of F_5 and F_6 it performs better than only jDE, and SaDE. In case of F_1 MRL-DE gave better result than others except in comparison to JADE which gave best result for F1.

TABLE V. EXPERIMENTAL RESULTS AND COMPARISONS WITH SADE, JDE JADE AND LEDE IN TERM OF AVERAGE ERROR AND STANDARD DEVIAVITION (S.D.) OF 50 RUNS

F	Max NFE	SaDE	jDE	JADE	LeDE	MRL-DE α =20
F_1	$15x10^4$	1.03E-37 (1.86E-37)	1.16E-28 (1.24 E-28)	**1.30E-54 (9.20E-54)**	2.19E-34 (1.18E-34)	2.7E-43 (2.5E-43)
F_2	$15x10^4$	3.55E-15 (0.0E+00)	8.33E-15 (1.86E-15)	4.40E-15 (0.0E+00)	5.89E-15 (0.0E+00)	**3.4E-15 (0.0E+00)**
F_3	$30x10^4$	**0.0E+00 (0.0E+00)**	**0.0E+00 (0.0E+00)**	**0.0E+00 (0.0E+00)**	**0.0E+00 (0.0E+00)**	3.7E+01 (2.5E+01)
F_4	$50x10^4$	2.66E-01 (1.03E+00)	1.63E-01 (7.89E-01)	3.20E-01 (1.10E+00)	0.0E+00 (0.0E+00)	**0.0E+00 (0.0E+00)**
F_5	$30x10^4$	1.21E-03 (3.55E-03)	3.26E-03 (7.74E-04)	**6.8E-04 (2.5E-04)**	1.1E-03 (5.58E-04)	1.43E-03 (3.23E-04)
F_6	$50x10^4$	2.42E-35 (6.07E-35)	1.64E-13 (3.65E-13)	**6.0E-87 (1.9E-86)**	1.16E-38 (2.28E-38)	1.8E-37 (2.6E-36)
F_7	$20x10^4$	1.54E-26 (1.40E-26)	9.98E-24 (7.59E-24)	3.90E-22 (2.70E-21)	1.09E-24 (4.09E-25)	**3.78E-29 (1.79E-29)**
F_8	$20x10^4$	**0.0E+00 (0.0E+00**	**0.0E+00 (0.0E+00**	2.00E-04 (1.4E-03)	**0.0E+00 (0.0E+00**	**0.0E+00 (0.0E+00**

C. Results of Shifted Benchmark Problems

In TABLE-VI results of shifted functions is given. These functions are specially designed to analyze the efficiency and robustness of population based global optimization algorithms. Dimension of all these function are taken is 30 (i.e. D=30) and max-NFEs= D*10000. Here results have been taken in terms of average error and standard deviation of 50 runs.

Once again from the numerical results we can see that the proposed MRL-DE is quite competent for solving numerical optimization problems of the type considered in this study.

TABLE VI. EXPERIMENTAL RESULTS AND COMPARISONS WITH DE IN TERM OF AVERAGE NFE OF 50 RUNS FOR SHIFTED BENCHMARK PROBLEMS

F	DE	MRL-DE			
		α =20	α =30	α =40	α =50
SF_1	2.2E-14 (2.7E-14)	**0.0E+00 (0.0E+00)**	**0.0E+00 (0.0E+00)**	**0.0E+00 (0.0E+00)**	**0.0E+00 (0.0E+00)**
SF_2	6.7E-04 (1.7E-04)	**1.8E-13 (4.2E-14)**	5.4E-12 (4.4E-12)	1.8E-08 (1.9E-08)	1.6e-006 (1.8E-06)
SF_3	1.8E+01 (1.2E+00)	**4.5E-14 (2.2E-14)**	**4.5 E-14 (2.2 E-14)**	7.9E-14 (7.7E-14)	2.5E-09 (2.0E-09)
SF_4	9.4E+01 (8.2E+00)	**2.6E+01 (1.9E+01)**	1.0E+02 (5.0E+01)	1.4E+02 (1.6E+01)	1.4E+02 (1.5E+01)
SF_5	5.6E-15 (1.1E-14)	**0.0E+00 (0.0E+00)**	**0.0E+00 (0.0E+00)**	**0.0E+00 (0.0E+00)**	**0.0E+00 (0.0E+00)**
SF_6	3.9E-14 (1.3E-14)	**2.8E-14 (0.0E+00)**	**2.8E-14 (0.0E+00)**	**2.8E-14(0.0E+00)**	**2.8E-14 (0.0E+00)**

Next in Fig-2 and Fig-3, convergence graph of F_1 and SF_1 are given;

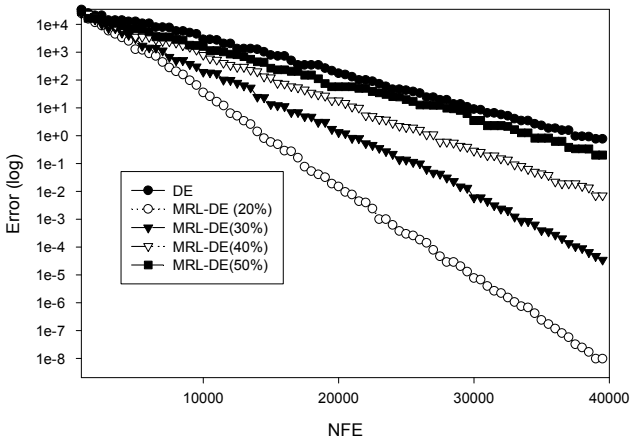

Figure 2. Convergence graph of function F_1 for average Error and NFE.

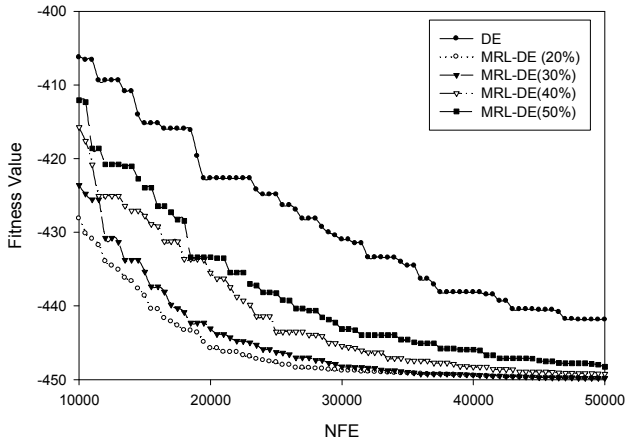

Figure 3. Convergence graph of shifted function SF_2 for fitness valus and NFE

VI. STATISTICAL ANALYSIS

In this section we have given a statistical analysis of proposed MRL-DE in order to check whether the results of MRL-DE are significant in the comparisons of other enhance variants of DE.

TABLE VII shows the results of applying Friedman test in order to see whether there are global differences in the results of TABLE IV. Given that the *p*-value of Friedman test is lower than the level of significance considered a = 0.05 and 0.1, so there are significant differences among the observed results.

Attending to these results, a post-hoc statistical analysis is done to detect concrete differences among algorithms. First of all, we employed Bonferroni-Dunn's test to detect significant differences for the control algorithm MRL-DE. TABLE VIII summarizes the ranking of algorithms based on their NFEs as given in TABLE IV obtained by Friedman's test and the critical difference (CD) of Bonferroni-Dunn's procedure. Bonferroni-Dunn's procedure to calculated CD value is given in [23]. In Fig 4, Bonferroni-Dunn's graphic illustrates difference among rankings obtained for each algorithm. In this, we draw a horizontal cut line which represents the threshold for the best performing algorithm, the one with the lowest ranking bar, in order to consider it better than other algorithms. A cut line is drawn for each level of significance considered in the study at height equal to the sum of the ranking of the control algorithm and the corresponding Critical Difference computed by the Bonferroni-Dunn method. The bars which exceed this line are associated to an algorithm with worse performance than the control algorithm. So by the application of Bonferroni-Dunn's test we can see that MRL-DE at α =20 is better than all of the algorithms also we see that the bars of DE, MRL-DE (at α =40, 50) goes outside the horizontal lines which are correspondence to significant level a=0.05 and 0.1. so we will consider these algorithms give worst performance than others.

A similar statistical analysis for SADE, JADE, LeDE, jDE and MRL-DE (α =20) is given in TABLE IX and Fig-5.

From Fig-5 we can see no rank bar exceeds from horizontal lines, so there is no significant difference between all algorithms. But we would like to point out that probably from the programming point of view; the proposed MRL-DE is easiest.

TABLE VII. RESULTS ON FRIEDMAN TEST BASED ON NFE GIVEN IN TABLE IV

N	Friedman value	df	*p*-value
8	32.714	5	<0.001

N is no of functions and *df*: degree of freedom

TABLE VIII. RANKING OBTAINED BY FRIEDMAN TEST AND CRITICAL DIFFERNCE (CD) CALCULATED THROUGHT BONFERRONI-DUNN'S PROCEDURE

Algorithms	Ranking
DE	5.56
DERL	2.69
MRL-DE (α =20)	1.31
MRL-DE (α =30)	2.69
MRL-DE (α =40)	3.81
MRL-DE (α =50)	4.94
CD value at *a*=0.1	**2.17**
CD value at *a*=0.05	**2.40**

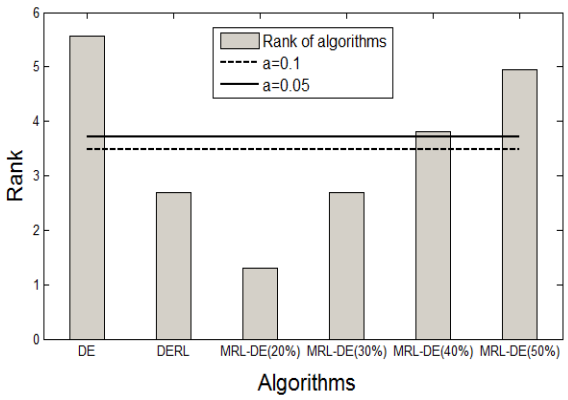

Figure 4. Bonferroni-Dunn's graphic corresponding to:NFE as given in TABLE VIII

TABLE IX. RANKING OBTAINED BY FRIEDMAN TEST AND CRITICAL DIFFERNCE (CD) CALCULATED THROUGHT BONFERRONI-DUNN'S PROCEDURE

Algorithms	Ranking
SADE	2.88
jDE	4.00
SADE	2.94
LeDE	2.69
MRL-DE (α =20)	2.50
CD value at *a*=0.1	**1.77**
CD value at *a*=0.05	**1.97**

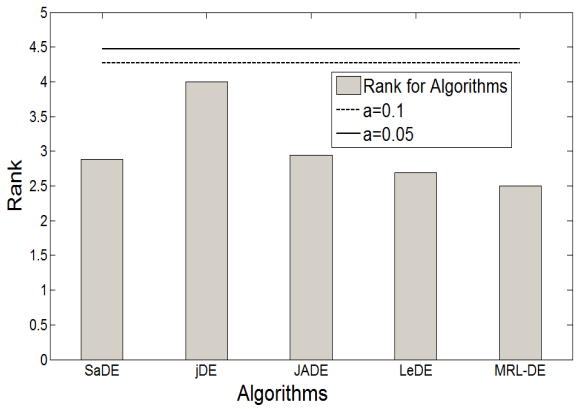

Figure 5. Bonferroni-Dunn's graphic corresponding to:average error as given in TABLE IX

VII. CONCLUSONS

In the present study a modified selection strategy is suggested for selecting the candidates to undergo mutation operation. In this strategy, named Modified Random Location (MRL) the search space is divided into three regions from which the candidates are selected. This is done to enhance the exploratory feature of the basic DE. The corresponding MRL-DE was tested on a set of 8 standard benchmark problems and 6 nontraditional shifted functions. It was observed that at α =20, the proposed algorithm best results. Numerical and statistical results showed that the proposed scheme although simple in nature improves the functioning of basic DE and performs either better or at par with some of its recent variants.

ACKNOWLEDGMENT

The first author acknowledges with thanks the financial assistance provided by MHRD.

REFERENCES

[1] R. Storn and K. Price, "Differential Evolution—A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces," J Glob Optim, vol 11(4), 1997, pp-341–359.

[2] P.Kumar, M. Pant and V.P. Singh, "Information Preserving Selection Strategy for Differential Evolution Algorithm", Proc IEEE World Congress on Information and Communication Technology (WICT 2011), Mumbai, India, 2011, pp 466-470

[3] J. Ilonen, J-K Kamarainen, and J. Lampinen, "Differential Evolution Training algorithm for Feed-Forward Neural Networks", Neural Process Lett 17(1), 2003, pp-93–105

[4] Y. Wang, J. Zhang, and G. Zhang, "A Dynamic Clustering Based Differential Evolution Algorithm for Global Optimization", Eur J Oper Res, 183(1), 2007, pp 56–73

[5] V. Plagianakos, D. Tasoulis,and M. Vrahatis, "A Review of Major Application Areas of Differential Evolution", In: Advances in Differential Evolution, Springer, Berlin, vol 143, 2008., pp 197 238

[6] H. Fan, and J. Lampinen, "A Trigonometric Mutation Operation to Differentia Evolution", Journal of Global Optimization. 27, 2003, pp 105-129.

[7] M. Ali and M. Pant, "Improving the Performance of Differential Evolution Algorithm Using Cauchy Mutation", Soft Computing, 2010. doi:10.1007/s00500-010-0655-2.

[8] M. Pant, M. Ali and A. Abraham, "Mixed Mutation Strategy Embedded Differential Evolution", Proc IEEE Congress on Evolutionary Computation (CEC-09), 2009, pp 1240-1246.

[9] M. Pant, R. Thangaraj, A. Abraham, C. Grosan, "Differential Evolution with Laplace Mutation Operator", Proc IEEE Congress on Evolutionary Computation, Norway, 2005, 2841-2849.

[10] P. Kaelo and M.M. Ali, "A Numerical Study of Some Modified Differential Evolution Algorithms", European Journal of Operational Research. 169, 2006, pp 1176-1184.

[11] A.K. Qin, V.L. Huang and P.N. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization", IEEE Transaction of Evolutionary Computing. 13 (2), 2009, pp 398–417.

[12] J. Zhang and A.C. Sanderson, "JADE: Adaptive Differential Evolution With Optional External Archive:, IEEE Transaction of Evolutionary Computing. 13(5), 2009, pp 945–958.

[13] S. Das, A. Abraham, U.K. Chakraborty and A. Konar, "Differential Evolution Using a Neighborhood Based Mutation Operator", IEEE Transaction of Evolutionary Computing, 13(3), 2009, 526–553.

[14] B. Dorronsoro, P. Bouvry, "Improving Classical and Decentralized Differential Evolution with New Mutation Operator and Population Topologies", IEEE Trans. Evol. Comput. 15 (1) , 2011, 67–98.

[15] M.G. Epitropakis, D. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, "Enhancing Differential Evolution Utilizing Proximity-Based Mutation Operators", IEEE Trans. Evol. Comput. 15 (1), 2011, 99–119.

[16] F. Neri and V. Tirronen, "Recent Advances in Differential Evolution: A Survey and Experimental Analysis", Artif Intell Rev. 33 (1–2), 2010, 61–106

[17] S. Das and P.N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art", IEEE Transaction of Evolutionary Computing, 15(1), 2011. pp 4-13.

[18] Y. Cai, J. Wang and J. Yin, "Learning-Enhanced Differential Evolution for Numerical Optimization", Springer-Verlag, Soft Comput, 2011, DOI 10.1007/s00500-011-0744-x

[19] J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, "Self Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems", IEEE Transaction of Evolutionary Computing. 10(6), 2006, pp 646–657.

[20] S. Rahnamayan, H.R. Tizhoosh and M.M.A. Salama, "Opposition Based Differential Evolution", IEEE Transaction of Evolutinary Computing. 12(1), 2008, pp 64–79.

[21] P.N. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization", Nanyang Technol Universiy, Singapore, 2005, pp 1–50

[22] K.Tang, X.Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, and Z.Yang, "Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization", Technical Report CEC-08, 2008, pp 1-18.

[23] J. Demsar, "Statiscally Comparisons of Classifier over Multiple Date Set", Journal of Machine Learning Research, 7, 2006, 1-30.