Differential Evolution with Laplace Mutation Operator

Millie Pant, Radha Thangaraj, Ajith Abraham and Crina Grosan

Abstract— **Differential Evolution (DE) is a novel evolutionary approach capable of handling non-differentiable, non-linear and multi-modal objective functions. DE has been consistently ranked as one of the best search algorithm for solving global optimization problems in several case studies. Mutation operation plays the most significant role in the performance of a DE algorithm. This paper proposes a simple modified version of classical DE called MDE. MDE makes use of a new mutant vector in which the scaling factor *F* is self adaptive. *F* is a random variable following Laplace distribution. The proposed algorithm is examined on a set of ten standard, nonlinear, benchmark, global optimization problems having different dimensions, taken from literature. The preliminary numerical results show that the incorporation of the proposed mutant vector helps in improving the performance of DE in terms of final convergence rate without compromising with the fitness function value.**

I. Introduction

EVOLUTIONARY Algorithms (EAs) [1] are a broad class of stochastic optimization algorithms inspired by biology and, in particular, by those biological processes that allow populations of organisms to adapt to their surrounding environments: genetic inheritance and survival of the fittest. EAs have a prominent advantage over other types of numerical methods, among which the following two are the most important [2]:

- They can be applied to problems that consist of discontinuous, non-differentiable and non- convex objective functions and/or constraints.
- They can easily escape from local optima

EAs have been applied to a wide range of functions and real life problems [3] – [6]. Some common EAs are Genetic Algorithms (GA), Evolutionary Programming (EP), Particle Swarm Optimization (PSO), Differential Evolution (DE) etc. In the present research paper, we have concentrated our work to DE, which is comparatively a newer addition to the class of population based search techniques. DE is a stochastic, population based search strategy developed by Storn and Price [7] in 1995. It is a novel evolutionary approach capable of handling non-differentiable, non-linear and multimodal objective functions. DE has been designed as a stochastic parallel direct search method, which utilizes concepts borrowed from the broad class of EAs. The method typically requires few, easily chosen control parameters. Experimental results have shown that performance of DE is better than many other well known EAs [8], [9]. While DE shares similarities with other EAs, it differs significantly in the sense that in DE, distance and direction information is used to guide the search process [10].

Despite several attractive features, it has been observed that DE sometimes does not perform as good as the expectations. Empirical analysis of DE has shown that it may stop proceeding towards a global optimum even though the population has not converged even to a local optimum [12]. The situation when the algorithm does not show any improvement though it accepts new individuals in the population is known as stagnation. Besides this, DE also suffers from the problem of premature convergence. This situation arises when there is a loss of diversity in the population. It generally arises when the objective function is multi objective having several local and global optimums. Like other EA, the performance of DE deteriorates with the increase in dimensionality of the objective function. Several modifications have been made in the structure of DE to improve its performance. Some interesting modifications include parameter adaption strategy for DE by Zaharie [13], Abbas [14] proposed a self adaptive crossover rate for multiobjective optimization problems, Omran et al. [15] introduced a self adaptive scaling factor parameter *F*, Brest et al. [16] proposed SADE, which encoded control parameters F and Cr into the individuals and evolved their values by using two new probabilities. Das et al. [17] introduced two schemes for the scale factor *F* in DE. some other recent modified versions include Opposition based DE (ODE) by Rahnamayan et al. [18], a hybridization of DE with Neghborhood search by Yang et al. [19], Fittest Individual refinement [FIR] method by Noman and Iba [20]. Several recent developments in DE algorithm design and application can be found in [21].

In continuation to the techniques of improving the performance of DE, in the present study we present a modified version of DE called MDE. The proposed MDE is a semi adaptive type DE in which the scaling factor F takes value according to the Laplace Distribution. The scaling factor F plays a significant role in the generation of perturbed mutant vector. The presence of a good scaling factor may help in preserving the diversity by enhancing the exploration and exploiting capabilities of the population.

The structure of the paper is as follows: in Section 2, we briefly explain the Differential Evolution Algorithm, in Section 3; we have defined and explained the proposed MDE algorithm. Section 4 deals with experimental settings, Sections 5 and 6 give the benchmark problems and their

M. Pant is with the Indian Institute of Technology Roorkee, Saharanpur - 247001, India (phone: +91-9759561464; e-mail: millifpt@iitr.ernet.in)

R. Thangaraj is with the Indian Institute of Technology Roorkee, Saharanpur – 247001, India (e-mail: t.radha@ieee.org)

A. Abraham is with the Center of Excellence for Quantifiable Quality of Service, Norwegian University of Science and Technology, Norway and Machine Intelligence Research Labs -MIR Labs (e-mail: ajith.abraham@ieee.org)

C. Grosan is with the Department of Computer Science,Babes-Bolyai University, Romania (e-mail: cgrosan@cs.ubbcluj.ro)

numerical results respectively and finally the paper conclude with Section 7.

II. Differential Evolution

DE shares a common terminology of selection, crossover and mutation operators with GA however it is the application of these operators that make DE different from GA. Whereas, in GA crossover plays a significant role, it is the mutation operator which effects the working of DE [11]. The working of DE may be described as follows:

For a D-dimensional search space, each target vector $x_{i,g}$, a mutant vector is generated by

$$v_{i,g+1} = x_{r_1,g} + F*(x_{r_2,g} - x_{r_3,g}) \quad (1)$$

where $r_1, r_2, r_3 \in \{1,2,....,NP\}$ are randomly chosen integers, must be different from each other and also different from the running index i. F (>0) is a scaling factor which controls the amplification of the differential evolution $(x_{r_2,g} - x_{r_3,g})$. In order to increase the diversity of the perturbed parameter vectors, crossover is introduced [8]. The parent vector is mixed with the mutated vector to produce a trial vector $u_{ji,g+1}$,

$$u_{ji,g+1} = \begin{cases} v_{ji,g+1} & if \ (rand_j \le CR) \ or \ (j = j_{rand}) \\ x_{ji,g} & if \ (rand_j > CR) \ and \ (j \ne j_{rand}) \end{cases} \quad (2)$$

where j = 1, 2,……, D; $rand_j \in [0,1]$; CR is the crossover constant takes values in the range [0, 1] and $j_{rand} \in (1,2,....,D)$ is the randomly chosen index.

Selection is the step to choose the vector between the target vector and the trial vector with the aim of creating an individual for the next generation.

III. Modified DE Algorithm

Initially Storn and Price proposed ten versions of DE. In the present study, we have embedded the proposed mutant vector in the DE/rand/1/bin version [8], which is perhaps the most commonly used version. The performance of DE depends largely on the selection of control parameters. The control parameters generally take the fixed values as decided by the user. If these values are taken probabilistically then the user may be saved from the trouble of undergoing rigorous sensitivity analysis for deciding the appropriate value of parameters. Various continuous probability distributions are available in literature which may be taken for deciding the behavior of control parameters. In the present article we propose a 'semi adaptive' type of DE in which one of the control parameters F is generated probabilistically while the sensitivity analysis is done for the other parameter CR. Instead of taking a fixed value of F the proposed MDE algorithm takes random variable following Laplace distribution. The Probability Density Function (pdf) of Laplace distribution is similar to that of normal distribution however, whereas the normal distribution is expressed in terms of squared difference from the mean, Laplace density is expressed in terms of absolute difference from the mean. As a result Laplace distribution has a fatter tail than normal distribution. The presence of a fatter tail in turn implies that a random variable having Laplace distribution will be able to control differential vectors more effectively and will probably help in preventing premature convergence by maintaining the diversity.

A C++ style computational code for the proposed algorithm may be given as:

```
//Initialize the population and calculate the fitness value for each particle
Do
For i = 1 to number of particles
  // Mutation
$v_{i,g+1} = x_{r_1,g} + £* | x_{r_1,g} - x_{r_2,g} |$
//where £ is the random variable having Laplace Distribution
       Do Crossover and Selection
End for.
Until some stopping criteria is reached.
```

IV. Experimental Settings

In order to make a fair comparison of DE and MDE algorithms, we fixed the same seed for random number generation so that the initial population is same for both the algorithms. The population size is taken as 100 for all the test problems. The crossover rate and scaling factor F, for classical DE, are fixed at 0.2 and 0.9 respectively. For MDE we did a sensitivity analysis for various crossover rates varying it from 0.1 to 0.9 (please also see Table IV) for all the test problems and observed that the crossover rate of 0.2 is most suitable. The scaling factor F for MDE follows Laplace distribution given as:

$$f(x|\theta,\mu) = \frac{1}{2\mu} exp\left(\frac{-|x-\theta|}{\mu}\right), -\infty \le x \le \infty$$

$$= \frac{1}{2\mu} \begin{cases} exp\left(-\frac{x-\theta}{\mu}\right) & if \ x < \theta \\ exp\left(-\frac{\theta-x}{\mu}\right) & if \ x > \theta \end{cases}$$

$\mu > 0$ is the scale parameter.

For each algorithm, the maximum number of iterations allowed was set to 5000 and the error goal was set as 1*e-04. A total of 30 runs for each experimental setting were conducted and the average fitness along with the average number of function evaluations (NFE), time taken and number of generations (GNE) of the best solutions throughout the run were recorded. The algorithms were programmed using Developer C++ and were executed on a Pentium IV PC.

V. Benchmark Problems

For the present study we considered a test bed of 10 benchmark problems given in Table I. Though this test bed is rather narrow, we have tried to include problems having different characteristics. Except for the last two functions; f_9 and f_{10}, all the problems are solved for dimension 50. In this section we describe briefly the properties of these functions.

- Rastringin's function's contour is made up of a large number of local minima which increases with the increase in the dimensionality of the problem.

- The second function is a simple sphere function which is strictly convex and unimodal and is generally considered as a good starting point for testing an optimization algorithm.
- Griewank function is a continuous multimodal function considered difficult to optimize because of its non-separable nature.
- The search space of Rosenbrock function is dominated by a large gradual slope which is raised along one edge to a fine point. Though it looks simple, it is notoriously hard for some optimization algorithms because of the extremely large search space combined with relatively small global minima.
- Noisy function is constructed by adding a uniformly distributed random noise to a quartic function. Due to the presence of noise the global optimum keeps on shifting from one position to another.
- The surface of Schwefel function consists of a large number of peaks and valleys. Also for this function the global minimum is near the bounds of the domain.
- In Ackley function, the presence of an exponential term makes is surface covered with several local minima.
- The eighth function is again a multimodal function having several local and global minima.
- Himmelblau's function is also a multimodal function with one global minimum and four identical local minima.
- Shubert's function has 760 local minima out of which 18 are global minima.

TABLE I. NUMERICAL BENCHMARK PROBLEMS

Function	Function Definition	Range	Min.Value
Rastringin Function	$f_1(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	[-5.12,5.12]	0
Spherical Function	$f_2(x)=\sum_{i=1}^{n}x_i^2$	[-5.12,5.12]	0
Griewank Function	$f_3(x)=\frac{1}{4000}\sum_{i=0}^{n-1}x_i^2+\sum_{i=0}^{n-1}\cos(\frac{x_i}{\sqrt{i+1}})+1$	[-600,600]	0
Rosenbrock Function	$f_4(x)=\sum_{i=0}^{n-1}100(x_{i+1}-x_i^2)^2+(x_i-1)^2$	[-30,30]	0
Noisy Function	$f_5(x)=(\sum_{i=0}^{n-1}(i+1)x_i^4)+rand[0,1]$	[-1.28,1.28]	0
Schewefel Function	$f_6(x)=-\sum_{i=1}^{n}x_i\sin(\sqrt{\mid x_i\mid})$	[-500,500]	-8379.658
Ackley Function	$f_7(x)=20+e-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))$	[-32,32]	0
Function f_8	$f_8(x)=-\sum_{i=1}^{n}\sin(x_i)(\sin(i\frac{x_i^2}{\pi}))^{2m}$, $m=10$	[-π,π]	---
Himmelblau Function	$f_9(x)=(x_2+x_1^2-11)^2+(x_1+x_2^2-7)^2+x_1$	[-5,5]	-3.78396
Shubert Function	$f_{10}(x)=\sum_{j=1}^{5}j\cos((j+1)x_1+j)\sum_{j=1}^{5}j\cos((j+1)x_2+j)$	[-10,10]	-186.7309

TABLE II. MDE VS. DE (MEAN FITNESS, STANDARD DEVIATION)

Function	MDE		DE	
	Fitness	Standard deviation	Fitness	Standard deviation
f_1	7.09739	2.40624	102.747	6.36517
f_2	6.15276e-05	1.12258e-05	8.2473e-05	1.2193e-05
f_3	6.4616e-05	1.05123e-05	8.40502e-05	1.56117e-05
f_4	1.86068	0.694999	43.9651	1.70131
f_5	0.008664	0.0018241	0.012731	0.002349
f_6	-20862.3	101.117	-20669.3	171.058
f_7	0.000169	2.57312e-06	0.000212	2.46469e-05
f_8	-47.7664	0.493178	-36.0939	0.621769
f_9	-3.6221	0.361939	-3.00776	1.2093
f_{10}	-186.731	2.50912e-07	-186.731	1.12765e-07

TABLE III. MDE VS. DE IN TERMS OF NUMBER OF FUNCTION EVALUATIONS (NFE), GENERATIONS (GNE) AND TIME

Function	MDE			DE		
	NFE	GNE	Time (sec)	NFE	GNE	Time (sec)
f_1	118740	1186	12.76	500100^+	5000^+	55.33
f_2	43160	430	4.8	67696	675	7.1
f_3	61253	611	7.5	97823	977	12.3
f_4	352907	3528	98.36	500100^+	5000^+	152.2
f_5	500100^+	5000^+	51.77	500100^+	5000^+	52.03
f_6	40556	810	1.267	171432	3427	5.33
f_7	70873	707	8.03	114207	1141	13.16
f_8	105657	2112	47.133	500100^+	5000^+	175.4
f_9	4278	84	0.067	4765	94	0.1
f_{10}	3910	77	0.033	12170	242	0.067
$\sum$	801334	9545	231.72	1468293	26556	473.02

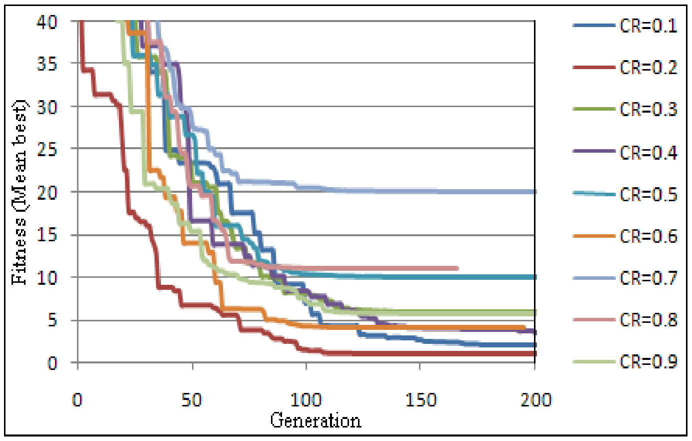

Fig 1 (a). Rastringin Function

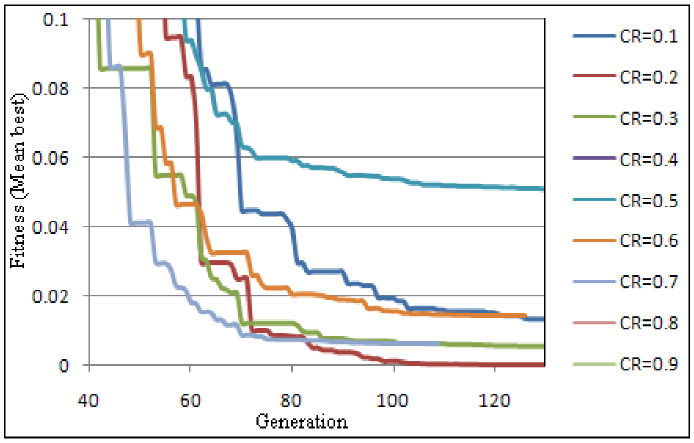

Fig 1 (b). Sphere Function

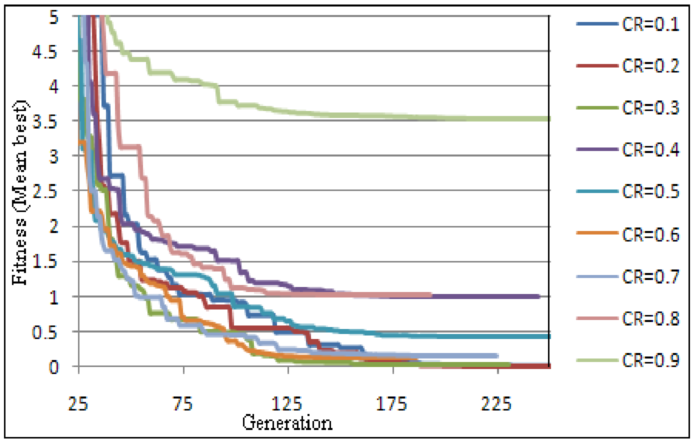

Fig 1 (c) Griewank Function

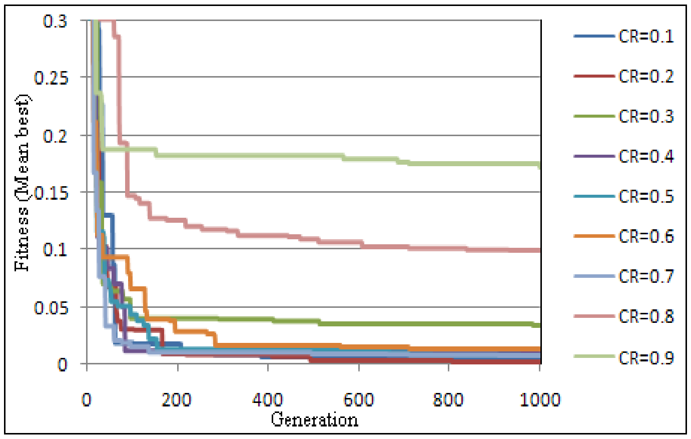

Fig 1 (d) Noisy Function

Fig 1(a) –1(d); Sensitivity analysis of MDE with respect to the various crossover rates for selected benchmark problems

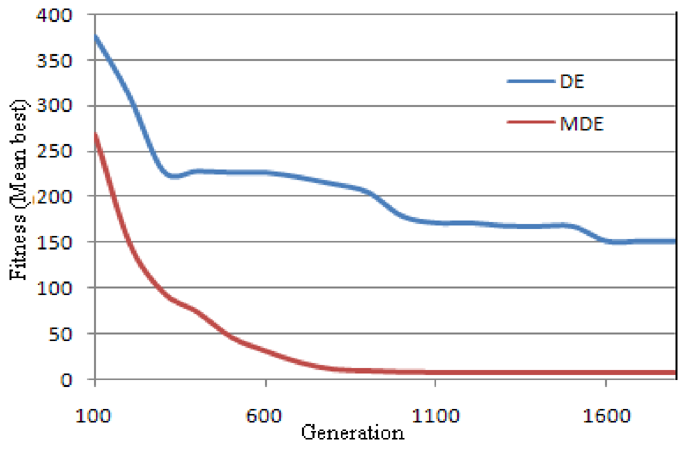

Fig 2 (a). Rastringin function

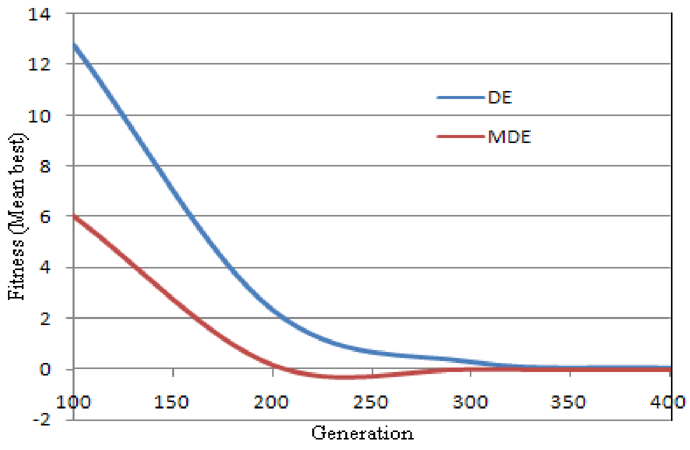

Fig 2 (b). Sphere function

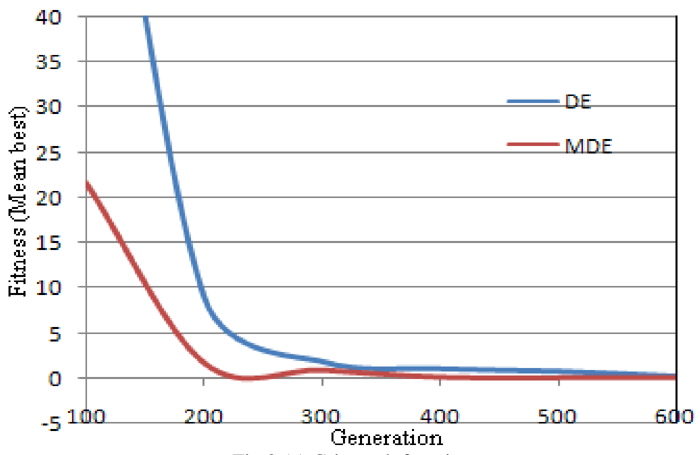

Fig 2 (c) Griewank function

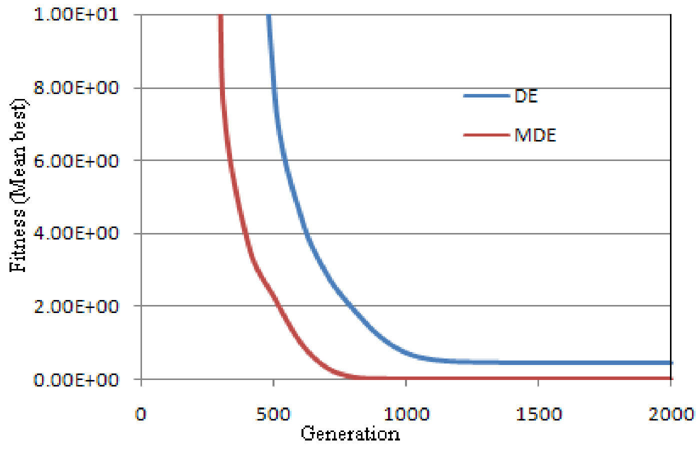

Fig 2 (d) Rosenbrock function

Fig 2 (a) – 2(d). Performance of DE and MDE for selected benchmark problems

VI NUMERICAL RESULTS AND COMPARISONS

The MDE algorithm is compared with the classical DE in terms of Average fitness function value, number of function evaluations (NFE), average number of iterations (GNE) and run time. In Table II, we have shown the numerical results of benchmark problems in terms of average fitness function value and standard deviation. Table III gives the number of function evaluations, number of generations and time taken. From Table II, it can be seen that for Rastringin function (f_1), the difference in the average fitness function values for DE and MDE is quite visible. The true global minimum for Rastringin function is located at 0.0. None of the algorithms were able to reach this value for the dimension 50. However MDE gave a much better value in comparison to DE. Similarly for Rosenbrock function, the proposed MDE gave a value much closer to the true optimum (0.0) in comparison to DE. For all other functions both the algorithms gave more or less similar values quite near to the true optimum value.

The better performance of MDE is more visible from Table III, where number of function evaluations, number of generations and time are reported. From this Table it is clear that the proposed MDE converges much faster than the classical DE. The total number of function evaluations for solving 10 test problems comes out to be 801334 for MDE in comparison to 1468293 as obtained by DE. Similarly, for the total time taken by MDE is 231.72 whereas the total time taken by DE is 473.02. In case of number of generations, MDE required 9545 generations and DE took 26556 generations. Thus, the overall percentage improvement in terms of NFE, GNE and Time taken for solving the 10 benchmark problems is around 50%.

$$(\%\text{age improvement})_{NFE}=\left(\frac{1468293-801334}{1468293}\right)*100=45.424$$

$$(\%\text{age improvement})_{GNE}=\left(\frac{26556-9545}{26556}\right)*100=64.05709$$

$$(\%\text{age improvement})_{Time}=\left(\frac{473.02-231.72}{473.02}\right)*100=51.01233$$

The performance curves of MDE vs. DE for selected benchmark problems are shown in Fig 1(a) – Fig 1(d). Performance curves of MDE using different crossover rates are given in Figures 2 (a) – 2(d).

TABLE IV. SENSITIVITY ANALYSIS OF MDE FOR DIFFERENT CROSSOVER RATES

CR	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Fun									
f_1	2.38	**0.36**	4.05	3.16	7.02	7.19	10.51	13.52	15.68
	1.7151	**0.6013**	1.5208	1.5826	2.734	2.8548	4.0836	6.3352	8.9189
	302	**205**	218	217	213	217	221	232	256
f_2	0.0260	**0.0007**	0.0528	0.0864	0.2107	0.1897	0.3227	0.6466	1.5256
	0.0945	**0.0023**	0.1102	0.1727	0.3980	0.2682	0.5283	0.6269	1.4390
	165	154	151	152	149	146	150	176	169
f_3	0.0937	**0.0922**	0.1423	0.3639	0.3475	1.052	2.2736	2.8714	6.2459
	0.3967	**0.1445**	0.3711	0.3983	0.3684	1.3840	2.4206	2.7900	5.2566
	548	**408**	581	558	530	520	514	525	545
f_4	5.0408	**1.523**	4.497	2.088	2.426	9.921	9.422	6.456	6.523
	0.2378	**0.1256**	0.2765	0.4523	0.4635	0.5342	0.4376	0.2248	0.6532
	991	818	856	864	873	852	857	889	875
f_5	0.0072	**0.0060**	0.0076	0.0068	0.0113	0.0311	0.0293	0.0465	0.1268
	0.0044	**0.0030**	0.0071	0.0056	0.0102	0.0551	0.0398	0.0575	0.1294
	1000^+	1000^+	1000^+	1000^+	1000^+	1000^+	1000^+	1000^+	1000^+
f_6	-4106.7	**-4138.7**	-4079.3	-4007.7	-3968.2	-3956.4	-3885.3	-3681.6	-3356.0
	115.01	**71.082**	129.10	142.31	208.54	171.93	246.29	404.06	394.83
	305	**227**	237	243	251	242	275	275	298
f_7	**0.7164**	1.0222	1.2261	1.0676	2.1165	2.7051	4.4418	6.4287	8.3859
	0.8649	1.0796	1.5979	1.0056	1.7611	2.0251	1.9437	2.5419	2.5998
	228	231	233	298	275	273	273	279	305
f_8	-9.1158	**-9.3339**	-9.1517	-9.0319	-8.7116	-8.8325	-8.4146	-8.0267	-6.6724
	0.4124	**0.1672**	0.2982	0.4760	0.5127	0.4515	0.6963	0.8349	1.1851
	363	**220**	235	312	299	272	275	273	263
f_9	-1.0324	-1.6135	-1.0411	**-2.2646**	-1.1164	-0.3789	-0.8485	-0.0613	0.9223
	2.9545	2.5664	3.0129	**2.1002**	2.9736	3.0504	2.9915	3.1689	3.2310
	86	107	63	66	79	75	49	54	50
f_{10}	**-186.73**	-186.73	-186.73	-186.72	-186.72	-186.66	-186.66	-186.63	-186.29
	4.1e-05	0.0058	0.0039	0.0295	0.0399	0.2884	0.2164	0.3883	1.222
	80	68	67	63	62	62	66	62	73

VII CONCLUSIONS

In the present study we proposed the use of scaling factor depending on Laplace Distribution for generating a mutant vector. The proposed MDE algorithm is tested on 10 benchmark problems and the results are compared with the classical DE. The numerical results show that the use of random variable having Laplace distribution as a scaling factor F, improves the performance of classical DE significantly. Although, we have not done any theoretical analysis but from the empirical results it can be seen that instead of fixing scaling factor it is better to take it in an adaptive manner. Also we would like to add that though we have tried to take a diverse set of bench mark problems, it is still a narrow test bed and we are continuing it to solve more complex problems and compare its performance with other existing EA for global optimization. The proposed work is still in the preliminary stage and several improvements can be added to it; like an adaptive crossover rate. Also, the work can be extended for other distributions like Cauchy and Levy distributions which have shown promising results in Evolutionary Programming.

REFERENCES

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., Handbook of EvolutionaryComputation. New York: Inst. Phys. and Oxford Univ. Press, 1997.

[2] J. Zhang, J. Xu and Q. Zhou, "A New Differential Evolution for Constrained Optimization Problems", In Proc. of the sixth Int. conf. on Intelligent Systems, Design and Applications, 2006, pp. 1018-1023.

[3] M. Pant, R. Thangaraj and A. Abraham, "Optimization of a Kraft Pulping System: Using Particle Swarm Optimization and Differential Evolution", In Proc. of 2nd Asia Int. Conf. on Modeling and Simulation, Malaysia, IEEE Computer Society Press, USA, pp. 637 – 641, 2008.

[4] A. Abbasy and S. H. Hosseini, "A Novel Multi-Agent Evolutionary Programming Algorithm for Economic Dispatch Problems with Non-Smooth Cost Functions, In Proc. of IEEE Power Engineering Society General Meeting, pp. 1 -7, 2007.

[5] M. Pant, R. Thangaraj and V. P. Singh, "Efficiency Optimization of Electric motors: A Comparative Study of Stochastic Algorithms",

World Journal of Modeling and Simulation, Vol. 4(2), pp.140 – 148, 2008.

[6] E. Cao and M. Lai, “An Improved Differential Evolution Algorithm for the Vehicle Routing Problem With Simultaneous Delivery and Pick-up Service”, In Proc. of Third International Conference on Natural Computation, Vol. 3, pp. 436 – 440, 2007.

[7] R. Storn and K. Price, “Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces”, Technical Report, International Computer Science Institute, Berkley, 1995.

[8] R. Storn and K. Price, “Differential Evolution – a simple and efficient Heuristic for global optimization over continuous spaces”, Journal Global Optimization. 11, 1997, pp. 341 – 359.

[9] R. Stom, “System design by constraint adaptation and differential evolution”, IEEE Transactions on Evolutionary Computation, Vol. 3, pp. 22-34, 1999.

[10] A.P. Engelbrecht, “Fundamentals of Computational Swarm Intelligence,” John Wiley & Sons Ltd, 2005.

[11] D. Karaboga and S. Okdem, “A simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm”, Turk J. Elec. Engin. 12(1), 2004, pp. 53 – 60.

[12] J. Lampinen and I. Zelinka, “On stagnation of the differential evolution algorithm,” in: Pavel Ošmera, (ed.) *Proc. of MENDEL 2000, 6th International Mendel Conference on Soft Computing*, pp. 76 – 83, June 7–9. 2000, Brno, Czech Republic.

[13] D. Zaharie, “Control of population diversity and adaptation in differential evolution algorithms,” In D.Matousek, P. Osmera (eds.), *Proc. of MENDEL 2003, 9th International Conference on Soft Computing,* Brno, Czech Republic, pp. 41-46, June 2003.

[14] H. Abbass, “The self-adaptive pareto differential evolution algorithm,” in *Proc. of the 2002 Congress onEvolutionary Computation*, 831-836, 2002.

[15] M. Omran, A. Salman, and A. P. Engelbrecht, “Self-adaptive differential evolution, computational intelligence and security,” *PT 1, Proceedings Lecture Notes In Artificial Intelligence* 3801: 192-199, 2005.

[16] J. Brest, S. Greiner, B. Boškovic, M. Mernik, and V. Žumer, “Self-adapting Control parameters in differential evolution: a comparative study on numerical benchmark problems,” *IEEE Transactions on Evolutionary Computation*, Vol. 10, Issue 6, pp. 646 – 657, 2006.

[17] S. Das, A. Konar, and U. K. Chakraborty, “Two improved differential evolution schemes for faster global search,” *ACM-SIGEVO Proceedings of GECCO*, Washington D.C., pp. 991-998, June 2005.

[18] S. Rahnamayan, H.R. Tizhoosh, and M. M. A. Salama, “Opposition-Based Differential Evolution,” *IEEE Transactions on Evolutionary Computation*, Vol. 12, Issue 1, pp. 64 – 79, 2008.

[19] 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.

[20] N. Noman and H. Iba, “Enhancing differential evolution performance with local search for high dimensional function optimization,” in *Proc. of the 2005 Conference on Genetic and Evolutionary Computation*, pp. 967–974, June 2005.

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