Mixed Mutation Strategy Embedded Differential Evolution

Millie Pant, Musrrat Ali and Ajith Abraham

Abstract- **Differential evolution (DE) is a powerful yet simple evolutionary algorithm for optimizing real valued optimization problems. Traditional investigations with differential evolution have used a single mutation operator. Using a variety of mutation operators that can be integrated during evolution could hold the potential to generate a better solution with less computational effort. In view of this, in this paper a mixed mutation strategy which uses the concept of evolutionary game theory is proposed to integrate basic differential evolution mutation and quadratic interpolation to generate a new solution. Throughout this paper we refer this new algorithm as, differential evolution with mixed mutation strategy (MSDE). 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.**

Keywords: differential evolution, mutation operator, mixed strategy.

1. Introduction

Differential evolution, proposed by Storn and Price in 1995 [2] is a relatively new optimization technique compared to evolutionary algorithms (EAs) such as Genetic Algorithms, Evolutionary Strategy, and Evolutionary Programming. 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 [12], opposition based DE [11], neighborhood search DE [14], Parent Centric DE [13] etc. several recent versions of DE can be found in [15].

In all the above mentioned versions of DE, a single mutation operation is used. It is quite natural to think that a DE having more than one mutation operation may work better than the one having a single mutation operation. In this paper we propose a DE inspired by the basic concepts of game theory.

In classical game theory, we have a set of players and a set of strategies. Each player tries to improve its performance by selecting a strategy from the given set and the value of the game changes accordingly.

Based on this analogy, we refer to the particles of the DE as players and the mutation operation as the strategy. The basic DE having a single mutation operation (single strategy) is called a pure strategy DE (PSDE) and the DE having more than one mutation operation (multiple strategies) is called mixed strategy DE (MSDE.

In this research, we propose an MSDE having a set of two strategies or a set of two mutation operations for solving unconstrained global optimization problems. The concept of mixed strategies is not new to the field of EA's [8, 9], however, to the best of our knowledge it has not been used in DE.

The remainder of the paper is structured as follows. Section 2 describes the basics Differential Evolution. Section 3 presents the proposed MSDE. Experimental setting is given in Section 4. Benchmark problems are listed in Section 5. Section 6 provides comparisons of results. Finally the paper is concluded in Section 7.

2. Differential Evolution (DE)

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 follows:

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 $\{r_1, r_2, r_3\}$ 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 $F \in [0, 1+]$.

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.

3. Proposed Algorithm

In this section we describe the proposed modified version, MSDE, which uses the concept of evolutionary game theory [10]. The individuals are regarded as players in an artificial evolutionary game applying different mutation operators to generate offspring. This is in contrast with the basic DE, where all the individuals are subject to a single mutation operator. In MSDE, for every individual of the population may select any of the two strategies provided to it in order to produce a perturbed (mutant) vector $V_{i,G+1}$.

A single mutation operator is called a pure strategy in the terms of game theory. A strategy profile, vector $\vec{p}$, is a collection pure strategies such that $\vec{p} = (p_1, \ldots, p_\alpha)$, where p_i is the pure strategy used by individual *i*. The strategies taken in the present study are p_1 and p_2, where p_1 denotes the usual mutation operation given in equation (1) and p_2 denotes the quadratic interpolation. Mathematical definitions of the strategies are given in Table 4.

The second strategy p_2 denotes quadratic interpolation, which determines the point of minima of the quadratic curve passing through three selected points. The symbols have the usual meaning as described in the previous section. There is no particular rationale for choosing quadratic interpolation as the second strategy except that it is a well known gradient free, direct search optimization method and has given good results in several cases [16], [17].

At each generation, each individual chooses a mutation operator from its strategy set based on a probability distribution. This distribution over the set of pure strategies available to an individual is called the mixed strategy of individual *i.* and is represented by a vector $\vec{\lambda}_i = (\lambda_i(p_1), \ldots, \lambda_i(p_\beta))$, where *β(=2 in our case)* is the number of strategies, and $\lambda_i(a)$ is the probability of individual *i* applying pure strategy *a* in mutation. To each individual a payoff is assigned according to its performance using particular mutation strategy. An individual can adjust its mixed strategy based on the payoffs of strategies. Usually, the strategy with a better payoff will be preferred with a higher probability in the next generation.

The procedure of this algorithm is outlined as follows:

Step1: Determine the initial set S using random number generator and initially assign mixed strategy as $\vec{\lambda}_i = (\lambda_i(p_1), \lambda_i(p_1)) = (0.5, 0.5)$.

Step 2: Calculate the objective function value $f(X_i)$ for all X_i.

Step3: Set *i=0.*

Step4: *i=i*+1;

Step5: Target vector X_i (parent vector) choose strategy (mutation operator) according to probability distribution $\overline{\lambda_i}$. If probability of pure strategy p_1 is greater than the probability of strategy p_2 then go to step 6 else go to step 7.

Step6: Select three distinct points from population and generate perturbed individual V_i using equation (1) and go to step 8.

Step7: Select one best point and other two distinct points from population and generate perturbed individual V_i by quadratic interpolation.

Step8: Recombine the each target vector X_i with perturbed individual generated in step 6 or 7 to generate a trial vector U_i using equation (2).

Step9: Check whether each variable of the trial vector is within range. If not keep it within range using $u_{i,j} = 2* x_{min,j} - u_{i,j}$,if $u_{i,j} < x_{min,j}$ and $u_{i,j} = 2* x_{max,j} - u_{i,j}$, if $u_{i,j} > x_{max,j}$, otherwise go to step 10.

Step10: Calculate the objective function value for vector U_i.

Step11: Choose better of the two (function value at target and trial point) using equation (3) for next generation.

Step12: If the target vector X_i uses strategy p_{α}, where α=1, 2 and new point survive in next generation (G+1) then

$$\lambda_i^{G+1}(p_\alpha) = \lambda_i^G(p_\alpha) + (1 - \lambda_i^G(p_\alpha))\gamma$$

$$\lambda_i^{G+1}(p_\beta) = \lambda_i^G(p_\beta) - \lambda_i^G(p_\beta)\gamma \quad \beta \neq \alpha$$

Otherwise

$$\lambda_i^{G+1}(p_\alpha) = \lambda_i^G(p_\alpha) - \lambda_i^G(p_\alpha)\gamma$$

$$\lambda_i^{G+1}(p_\beta) = \lambda_i^G(p_\beta) + \lambda_i^G(p_\beta)\gamma \quad \beta \neq \alpha$$

, where γ we have taken 1/3 [8].

Step13: If *i*<population size then go to step4 else go to step14.

Step14: Check whether convergence criterion is met. If yes, stop; otherwise go to step 3.

4. Experimental Setup

In order to make a fair comparison of MSDE 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 a fixed quantity, 100. Values of F scale outside the range of 0.4 to 1.2 are rarely effective, so F=0.5 is usually a good initial choice. In general C_r should be as large as possible to speedup the convergence so in this study we have taken C_r =0.33. 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., $|f_{max} - f_{min}| \leq \varepsilon = 10^{-5}$ or when the maximum number of function evaluations (NFE =10^6) was reached.

5. Benchmark problems

The performance of the proposed algorithm is tested on a set of five benchmark problems taken from literature [11]. All the functions are multimodal in nature. Except for functions f_2 and f_5, which are of dimensions 4 and 2 respectively, the remaining test problems are scalable and are tested for dimensions 10, 20 and 50. Mathematical models of the problems are given in the Appendix.

6. Numerical results and comparisons

6.1 Comparison between DE and MSDE

This section compares MSDE with the basic DE algorithm. Table 1 gives average fitness of function values, standard deviation, t- values and average error are listed. Average error is defined as the difference between the true global optimum value and the value obtained by the algorithm. Table 2 provides number of function evaluations (NFE) and improvement in term of number of functions evaluation. In Table 3, average time of execution of algorithms is given. As it is clear from the Table 1 that in term of fitness function value and standard deviation both the algorithms give more or less similar results although in some cases MSDE performs slightly better than classical DE. On the basis of t-values, last column of the Table 1, we conclude that there is a significant difference between both the algorithms at 5% level of significance. The superior performance of the proposed MSDE is more evident from Table 2, which gives the average number of function evaluations. From Table 2 we can see that MSDE takes less number of function evaluations to achieve the required fitness in comparison to the basic DE in all cases except to noisy function (f_3), in which both the algorithms approach to the maximum number of function evaluation (NFE=10^6). In term of improvement in number of function evaluation MSDE reduces the number of function evaluation up to 77% for function f_4 of dimension 10. If we talk about overall reduction in number of function evaluation, it is more than of 50%. But for function f_3, which is a noisy function, in term of function evaluation there is no improvement, both algorithms takes maximum number of function evaluation. From Table 3, it can be seen that MSDE takes less run time in comparison to basic DE but in case of function f_3, where number of function evaluation is same, MSDE takes more time than to basic DE. This behavior of MSDE is quite expected because it spends time in updating the probabilities and also because the evaluation of second strategy p_2 takes more time in comparison to the usual mutation operator. Performance curves (convergence graphs) of few selected functions are given in Figures 1(a) – 1(d). From these illustrations, it is evident that the convergence of proposed algorithm is faster than basic DE. The performance of MSDE is shown further in Figures 2 and 3 with respect to function f_1 increasing its dimension up to

200 variables. Figure 2 depicts that with the increase of time the fitness function value converges more rapidly in case of MSDE in comparison to basic DE. In Figure 3 we illustrate the effect on fitness function value with the increase in dimension. From the illustration it can be seen that the fitness remains almost consistent for MSDE when the dimension is increased, where as for basic DE, the fitness decreases with the increase in dimension.

6.2 Comparisons between MSDE and other modified versions of differential evolution.

The performance of the proposed MSDE is further compared with two recent versions of DE; opposition based population initialization DE, ODE [11] and trigonometric mutation differential evolution, TDE [12]. Both the algorithms have reported superior performance over the basic DE. The results obtained are summarized in Tables 5 and 6. In Table 5, the results of MSDE and ODE are compared for average number of function evaluations. For ODE, we have taken the results as mentioned in [11]. It is clear from the Table 5 that MSDE takes less number of function evaluations except for the function f_2 to achieve the accuracy given in last column. Improvement column shows that there is an improvement up to 87% if we talk for overall improvement it is 45%. In Table 6, we show the comparison of MSDE with TDE. Because the data for comparison purpose is not given in [12], so we have taken the same parameter setting as given in [12] and run TDE thirty times for the function f_1 for different dimension. Mean fitness value as well as number of function evaluations obtained by MSDE is better than TDE in all cases.

6.3 Sensitivity analysis of parameter γ

The parameter γ plays a crucial role in the performance of the proposed MSDE algorithm. It acts like a weighting parameter defined by the user at the beginning of the program. In the present study we recorded the performance of γ for various values between 0 and 1. For the sake of brevity, we are giving the numerical results for five different values of γ viz. 0.001, 0.1, 0.25, 0.33 and 0.95 in terms of fitness function, standard deviation and number of functions evaluations (NFE). For scalable problems we compared the results for dimension 20. The results for different values of γ are given in Table 7. Empirical analysis of results shows that smaller values of γ results in slower convergence thereby increasing the number of functions evaluations. Values between 0.25 to 0.95 are most suited for the optimization problems taken in the present study. We have considered the value of γ as 0.33 for all the numerical test functions taken in the present study.

7. Discussion and conclusions

In this paper we proposed a modified version of basic DE called MSDE by incorporating a mixed mutation strategy. 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. We have also compared our results with other two algorithms ODE and TDE which showed that mixed mutation strategy is beneficial in comparison to single strategy. The set of problems considered though small and limited show the promising nature of MSDE. One apparent drawback of proposed MSDE is that for noisy functions like f_3 it takes more time than the basic DE, although the number of function evaluation is same. However, we would like to maintain that the work is still in the preliminary stages and making any concrete conclusion about it do not sound justified. In this paper we have taken only two strategies we intend to work with more strategies in future and shall apply it for more complex problems and compare its performance with other versions of DE and with other optimization algorithms. The concept of mixed strategy can be applied to population generation and crossover rates also.

Table 1: Mean fitness, standard deviation of functions in 30 runs and t-valve.

Fun.	Dim.	Mean fitness (Std)		Average error		t-value
		DE	MSDE	DE	MSDE	
f_1	10	4.89922e-006 (9.3006e-007)	3.65155e-007 (4.20151e-007)	4.89922e-006	3.65155e-007	23.92
	20	1.02463e-005 (2.00048e-006)	3.01699e-006 (1.35368e-006)	1.02463e-005	3.01699e-006	16.12
	50	2.31386e-005 (2.21348e-006)	6.72181e-006 (1.04039e-006)	2.31386e-005	6.72181e-006	36.16
f_2	4	4.5118e-008 (2.32542e-008)	7.83712e-010 (1.15383e-009)	4.5118e-008	7.83712e-010	10.25
f_3	10	0.000120068 (4.75295e-005)	1.77752e-005 (1.22154e-005)	0.000120068	1.77752e-005	11.23
	20	0.000803444 (0.000154331)	0.000121088 (4.24884e-005)	0.000803444	0.000121088	22.96
	50	0.00692452 (0.00125875)	0.00043024 (0.000126203)	0.0003418	0.00002384	27.65
f_4	10	9.48435e-007 (3.33338e-007)	5.85615e-008 (4.37959e-008)	9.48435e-007	5.85615e-008	14.25
	20	3.68394e-006 (9.27743e-007)	5.40822e-007 (2.69767e-007)	3.68394e-006	5.40822e-007	17.52
	50	9.66678e-006 (1.00454e-006)	1.69896e-006 (4.23073e-007)	9.66678e-006	1.69896e-006	39.36
f_5	2	-1.03163 (7.93442e-009)	-1.03163 (1.28681e-014)	1.55038e-006	0.000268453	0.00

Table 2: Number of functions evaluation, % improvement of functions in 30 runs.

Fun.	Dim.	No of function Eva.		% Improvement
		DE	MSDE	
f_1	10	31040	14350	53.76
	20	57830	20390	64.74
	50	154490	39260	74.58
f_2	4	76160	58780	22.82
f_3	10	1e+006	1e+006	0.00
	20	1e+006	1e+006	0.00
	50	1e+006	1e+006	0.00
f_4	10	60880	13980	77.03
	20	52690	16990	67.75
	50	121930	30080	75.33
f_5	2	7190	2400	66.62

Table 3: Average time (sec) in 30 runs.

Fun.	Dim.	Average Time (Sec)	
		DE	MSDE
f_1	10	1.3	1.1
	20	3.1	2.9
	50	20.1	17.8
f_2	4	1.9	2.0
f_3	10	25	86.4
	20	49.7	176.1
	50	134.2	287.1
f_4	10	1.7	1.5
	20	2.9	3.8
	50	16.9	14.9
f_5	2	0.1	0.1

Table 4: Strategies used in the proposed algorithm.

Strategies used	**Definition**
P1	$X_{r3,G} + F * (X_{r1,G} - X_{r2,G})$
P2	$\frac{1}{2}\frac{(X_{r1,G}^2 - X_{r2,G}^2)f(X_{r3,G}) + (X_{r2,G}^2 - X_{r3,G}^2)f(X_{r1,G}) + (X_{r3,G}^2 - X_{r1,G}^2)f(X_{r2,G})}{(X_{r1,G} - X_{r2,G})f(X_{r3,G}) + (X_{r2,G} - X_{r3,G})f(X_{r1,G}) + (X_{r3,G} - X_{r1,G})f(X_{r2,G})}$

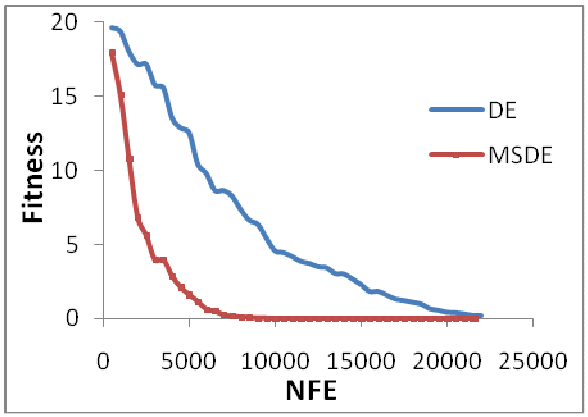

Fig 1(a): Performance curves of DE vs. MSDE for function f_1 , dimension 20.

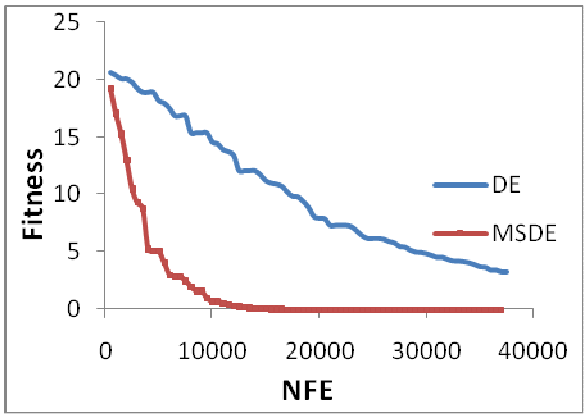

Fig 1(b): Performance curves of DE vs. MSDE for function f_1, dimension 50.

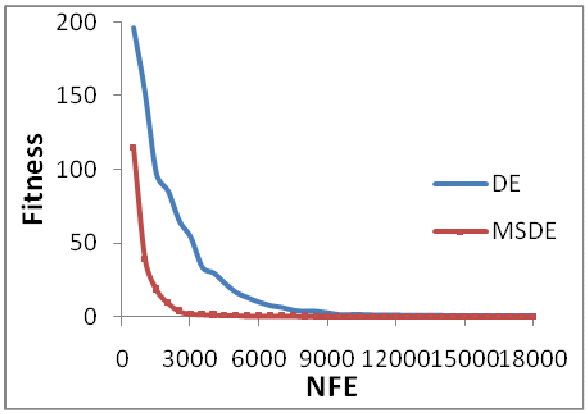

Fig 1(c): Performance curves of DE vs. MSDE for function f_4, dimension 20.

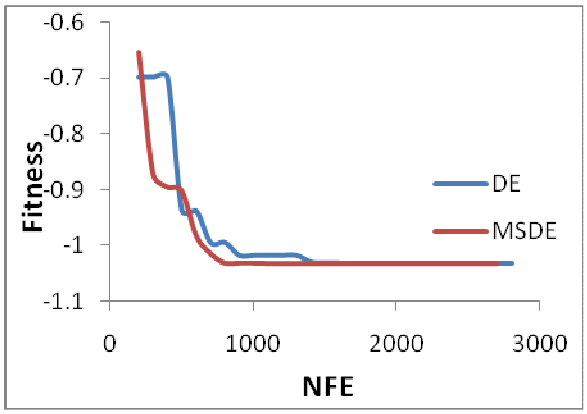

Fig 1(d): Performance curves of DE vs. MSDE for function f_5

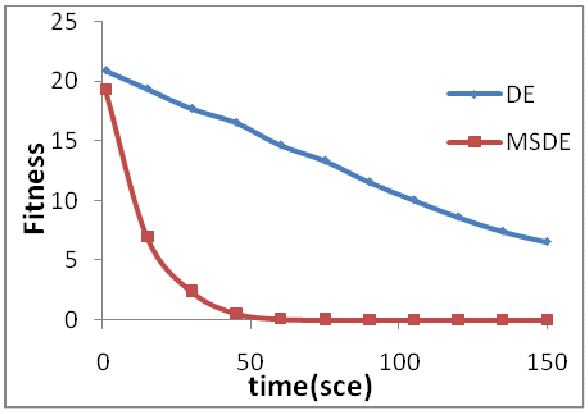

Fig 2: Fitness Vs time for function f_1 for 200 Dim.

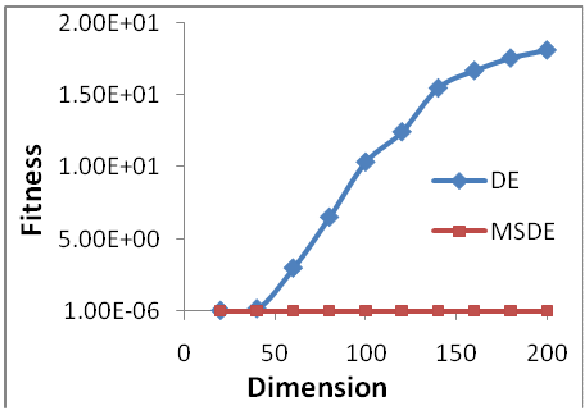

Fig 3 Fitness Vs dimension for function f_1.

Table 5: Average number of functions evaluation in 30 runs and % improvement.

Fun.	Dim.	No of function Eva.		% Improvement	To achieve Accuracy[11]
		ODE[11]	MSDE		
f_1	30	51619	12500	75.78411	10^{-1}
f_2	4	7959	22890	0.00	10^{-1}
f_3	30	24248	19040	21.47806	10^{-1}
f_4	30	53311	6660	87.50727	10^{-1}
f_5	2	5155	2740	46.84772	10^{-7}

Table 6: Mean fitness, standard deviation and average of functions evaluations in 30 runs for function f_1.

Dim.	Mean fitness		Std		No of fun. Evaluation	
	TDE	MSDE	TDE	MSDE	TDE	MSDE
10	3.49344e-006	3.65155e-007	8.35737e-007	4.20151e-007	39820	14350
30	1.26726e-005	4.48145e-006	2.20241e-006	1.13972e-006	84210	26790
20	8.19955e-006	3.01699e-006	1.24685e-006	1.35368e-006	63140	20390
50	2.40794e-005	6.72181e-006	3.08538e-006	1.04039e-006	127400	39260

Table 7: Sensitivity of parameter γ

Fun.		γ= .001	γ= .1	γ= .25	γ= .33	γ= .95
f_1 *Dim 20*	Fitness	3.29362e-006	2.5314e-006	3.14209e-006	3.01699e-006	2.94767e-006
	Std.	6.17117e-007	4.80274e-007	8.58226e-007	1.35368e-006	9.79393e-007
	NFE	21820	21900	21420	20390	20490
f_2 *Dim 4*	Fitness	1.58156e-009	5.98808e-010	1.17071e-009	7.83712e-010	9.45272e-010
	Std.	2.08799e-009	1.44467e-009	1.13281e-009	1.15383e-009	1.00746e-009
	NFE	63130	55960	53030	58780	49760
f_3 *Dim 20*	Fitness	0.000126695	0.000152264	0.0001191	0.000121088	1.16693e-005
	Std.	0.000114664	0.000130133	0.00011491	4.24884e-005	0.000383002
	NFE	1000000	1000000	1000000	1000000	1000000
f_4 *Dim 20*	Fitness	4.30757e-007	4.21027e-007	4.65772e-007	5.40822e-007	6.02965e-007
	Std.	1.9568e-007	1.42966e-007	1.55814e-007	2.69767e-007	2.87743e-007
	NFE	17920	17280	16970	16990	16360
f_5 *Dim 2*	Fitness	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163
	Std.	2.68149e-014	9.85413e-015	1.87432e-014	1.28681e-014	1.6407e-014
	NFE	2960	2510	2750	2400	2670

References:

[1] K. Price. "An introduction to differential evolution", *New Ideas in Optimization*, 1999, pp 79-108.

[2] 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.

[3] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization", *Evol. Comput,* 10(3), 2002, pp 207-234.

[4] R. Thomsen, "Multimodal optimization using crowding-based differential evolution", *In Proceedings of the 2004 Congress on Evolutionary Computation,* volume 2, 2004, pp 1382-1389,.

[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]. H.Dong, J.He, H.Huang, W.Hou, "Evolutionary programming using a mixed mutation strategy", *Information Science 177*, 2007 pp 312-327.

[9]. J.He, X.Yao, "A game theoretic approach for designing mixed mutation strategy", *LNCS 3612*, 2005 pp 279-288, 2005.

[10]. J.W. Weibull, "Evolutionary Game Theory", *MIT press, Cambridge, MA,* 1995.

[11].Shahryar Rahnamayan, H.R. Tizhoosh, M.M.A.Salama, "A novel population initialization method for accelerating evolutionary algorithms", *computer and applied mathematics with application* (53), 2007 pp 1605-1614.

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

[13] 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.

[14] 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.

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

[16] M. Pant, R. Thangaraj and A. Abraham, "A New PSO Algorithm Incorporating Reproduction Operator for Solving Global Optimization Problems", 7th International Conference on Hybrid Intelligent Systems (HIS'07), Kaiserslautern, Germany, IEEE Computer Society press, USA, ISBN 07695-2662-4, pp. 144-149, 2007.

[17] Mohan C and Shanker K, "A Controlled Random Search Technique For Global Optimization using Quadratic Approximation", Asia-Pacific Journal of Operational Research, Vol. 11, pp. 93-101, 1994.

Appendix

1. Ackley's function:

· $f_1(X) = -20 * \exp\left(-.2\sqrt{1/n\sum_{i=1}^{n} x_i^2}\right) - \exp\left(1/n\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$, With $-32 \le x_i \le 32$, min $f_1(0,...,0) = 0$

It is a multimodal function. the presence of an exponential term makes its surface covered with several local minima.

2. Colville function:

$f_2(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2 + 90(x_4 - x_3^2)^2 + (1-x_3)^2 + 10.1((x_2-1)^2 + (x_4-1)^2) + 19.8(x_2-1)(x_4-1)$

With $-10 \le x_i \le 10$, min $f_2(1,1,1,1) = 0$

It's a Unimodal function. Its global optimum functions resides inside a long, narrow, and parabolic-shaped flat valley.

3. Quartic function:

$f_3(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$ With $-1.28 \le x_i \le 1.28$, min $f_3(0,...,0) = 0$ It is a noisy function, 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.

4. Griewenk function:

$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$ With $-600 \le x_i \le 600$, min $f_4(0,...,0) = 0$

It is a continuous multimodal function considered difficult to optimize because of its non-separable nature.

5. Six hump Camel back function:

$f_5(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$ With $-5 \le x_i \le 5$, min

$f_5(0.0898, -0.7126)/(-0.0898, 0.7126) = -1.0316285$

It is a multimodal function with two global minima and four local minima.