International Journal of Innovative Computing, Information and Control
Volume **7**, Number **8**, August **2011**
ICIC International ©2011 ISSN 1349-4198
pp. 5063–5084

DIFFERENTIAL EVOLUTION USING MIXED STRATEGIES IN COMPETITIVE ENVIRONMENT

Musrrat Ali[1], Millie Pant[1], Ajith Abraham[2,3] and Vaclav Snasel[3]

[1]Department of Paper Technology
Indian Institute of Technology Roorkee
Roorkee Uttarakhand 247667, India
{ musrrat.iitr; millidma }@gmail.com

[2]VSB Technical University
17. listopadu 15/2172, Ostrava-Poruba 70833, Czech Republic
vaclav.snasel@vsb.cz

[3]Machine Intelligence Research Labs (MIR Labs)
Scientific Network for Innovation and Research Excellence
P.O. Box 2259, Auburn, Washington 98071-2259, USA
ajith.abraham@ieee.org

Received September 2009; revised January 2010

Abstract. *Differential evolution (DE) is a powerful yet simple evolutionary algorithm for optimizing real valued optimization problems. Traditional investigations with DE 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 the present study, a mixed mutation strategy which uses the concept of evolutionary game theory is proposed integrating the basic differential evolution mutation and quadratic interpolation based mutation to generate a new solution. Throughout of this paper, we refer this new algorithm as Mixed Strategy Differential Evolution (MSDE). The performance of proposed MSDE is investigated and compared with basic differential evolution and some other modified versions of DE available in literature. The experiments conducted show the competence of the proposed MSDE algorithm.*
Keywords: Differential evolution, Mutation operator, Pure strategy, Mixed strategy

1. **Introduction.** In the past few decades, Evolutionary Algorithms (EAs) have become the center of attention for solving complex global optimization problems which are otherwise difficult to solve by traditional methods. These algorithms have been successfully applied to a wide range of single and multi-objective optimization problems [1-4].

Some common EAs available in literature include Genetic Algorithms [5], Evolutionary Strategies [6], Evolutionary Programming [7], Particle Swarm Optimization [8], Differential Evolution [9], etc.

In the present study, we focus on DE, proposed by Storn and Price in 1995 [9], which is relatively a new addition to the class of EAs. Within a short span of around fifteen years, DE has emerged as one of the most popular techniques for solving optimization problems. DE has been successfully applied to solve a wide range of real life application problems arising in the field of science and engineering. Some of the areas where DE has been applied successfully include aerodynamic shape optimization [10], optimization of radial active magnetic bearings [11], automated mirror design [12], optimization of fermentation by using high ethanol tolerance yeast [13], clustering [14], neural network

[15], unsupervised image classification [16], digital filter design [17], optimization of non-linear functions [18], global optimization of non-linear chemical engineering processes [19] and multi-objective optimization [20], etc. Also, it has reportedly outperformed other optimization techniques [21-23].

Despite several positive features, it has been observed that DE sometimes does not perform as good as the expectations. Some of the drawbacks of DE are premature convergence which causes the entire population to converge to a point which may not even be a local minimum. Secondly, there may be a problem of stagnation in the population. In this situation, the population stops proceeding towards the global optimum though it allows new individuals to enter the population [24]. Moreover, like other population based search techniques, the performance of DE gradually deteriorates with the increase in the number of variables. These problems become more persistent when the objective function is multimodal in nature having several local and global optima. Several modifications have been made in the structure of DE to improve its performance. Some interesting modifications include: the suggestion of parameter adaption strategy for DE by Zaharie [25], use of a self adaptive crossover rate for multiobjective optimization problems by Abbas [26]. Omran et al. [27] introduced a self adaptive scaling factor parameter F. Brest et al. [28] proposed a Self Adaptive Differential Evolution (SADE), which encoded control parameters F and C_r into the individuals and evolved their values by using two new probabilities. Das et al. [29] introduced two schemes for the scaling factor, F, in DE. Some other recent modified versions include Opposition based DE (ODE) by Rahnamayan et al. [30], a hybridization of DE with Neighborhood search by Yang et al. [31], Fittest Individual refinement [FIR] method by Noman and Iba [32], Differential Evolution with Preferential Crossover (DEPC) and Differential Evolution with refined local search (DERL) by M. M. Ali [33], Trigonometric Differential Evolution (TDE) by Fan and Lampienen [34], Differential Evolution with parent centric crossover (DEPCX) by Pant et al. [35], Bare Bone DE or BBDE by [36], a greedy random strategy for genetic recombination by Bergey and Ragsdale [37]. Zhang et al. suggested a new constrained handling method for DE [38]. Many other recent developments in DE algorithm design and application can be found in [39].

In all the above mentioned versions of DE, a single mutation operation is used. However, it is possible that a particular mutation operator may not be suitable for all types of problems. For example, a mutation operator which gives good results in case of a Unimodal may have difficulty in tracking the optimum of multimodal functions. For multimodal functions, the mutation operator should be such that it can easily facilitate the exploratory capacities of a DE algorithm. An obvious solution to the problem of deciding an appropriate mutation operator which is well suited for all types functions can provide the DE individuals not just one mutation operator but a collection of mutation operators and manipulate them in an order to achieve the maximum benefit. This is the central theme of the present research.

In this paper, we propose a variant of DE having more than one mutation strategy (mixed strategy). We have borrowed the concept of mixed strategy from the classical game theory [40,41], which consists of 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 the present study, we consider a set of two strategies or a set of two mutation operations for all the members (or particles) of the population. One strategy is the mutation strategy of the basic DE and the other strategy is the quadratic interpolation mutation strategy. Each particle may select any one of the two strategies provided to it solving unconstrained global optimization problems. A detailed description of the proposed MSDE algorithm is given in Section 3.

Here we would like to mention that a preliminary version of this paper has already been published in conference proceedings [42]. However, in this paper, we present an elaborated version of [42]. We have included more test functions and compared the proposed MSDE algorithm with other recently modified versions of DE available in literature. Besides using the common performance measures like average fitness function value, standard deviation, number of function evaluations, CPU time, etc. for comparing the algorithms, we have also done a rigorous non parameter statistical analysis of the numerical results.

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 MSDE with basic DE. Comparison of MSDE with other modified versions of DE is made in Section 7. In Section 8, an analysis of applying mixed strategy to the family of DE is made. Comparison of algorithms on real life problems is given in Section 9. Finally, the conclusions based on the present study are drawn in Section 10.

2. **Differential Evolution (DE).**

In this section, we describe the DE/rand/1/bin scheme. It starts with a population of NP candidate solutions: $X_{i,G}$, $i = 1, \ldots, NP$, where the index i denotes the i^{th} individual of 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:

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

where, $i = 1 \ldots NP$, $r_1, r_2, r_3 \in \{1, \cdots, NP\}$ 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, $X_{i,G} = (x_{1,i,G}, \cdots, x_{n,i,G})$, are then subject to the crossover operation, that finally generates the population of candidates, or "trial" vectors, $U_{i,G+1} = (u_{1,i,G+1}, \cdots, u_{n,i,G+1})$, as follows:

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

where, $j = 1, \cdots, n$, $k \in \{1, \cdots, n\}$ is a random parameter's index, chosen once for each i. The crossover rate, $C_r \in [0, 1]$, 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:

$$X_{i.G+1} = \begin{cases} U_{i.G+1}, & \text{if } f(U_{i.G+1}) \leq f(X_{i.G}) \\ X_{i.G}, & \text{otherwise} \end{cases} \tag{3}$$

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.

2.1. **Complete family of DE.** Storn and Price suggested a total of 10 different strategies of DE [43]. the general convention used to describe the type of strategy is given as DE/X/Y/Z where DE stands for Differential Evolution, X denotes the type of vector to be perturbed and Y denotes the number of difference vectors considered for perturbation of X. Besides this, each strategy is combined with exponential (exp) or binary (bin) type of crossover which is denoted as Z. These strategies are listed as follows:
(i) DE/best/1/exp (ii) DE/rand/1/exp (iii) DE/rand to best/1/exp (iv) DE/best/2/exp (v) DE/rand/2/exp (vi) DE/best/1/bin (vii) DE/rand/1/bin (vii) DE/rand to best/1/bin (ix) DE/best/2/bin (x) DE/rand/1/bin

In the previous section, we have described the last scheme listed here. This is perhaps the most frequently used version of DE.

3. **Proposed MSDE Algorithm.** In this section, we describe the proposed modified version, MSDE, which uses the concept of evolutionary game theory [40,41]. In MSDE algorithm, 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, every individual of the population may select any one 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 $\bar{p}$, is a collection of pure strategies such that $\vec{p} = (p_1, \ldots, p_\alpha)$, where p_i is the pure strategy used by individual i. In the present study, only two mutation strategies p_1 and p_2 are considered where p_1 denotes the usual mutation operation as given by Equation (1) and p_2 is defined as:

$$\begin{aligned} p_{\downarrow}2 = 1/2(&(X_{\downarrow}(r1,G)^{\uparrow}2^{|}X_{\downarrow}(r2,G)^{\uparrow}2)f(X_{\downarrow}(r3,G)) \\ &+ (X_{\downarrow}(r2,G)^{\uparrow}2^{|}X_{\downarrow}(r3,G)^{\uparrow}2)f(X_{\downarrow}(r1,G)) \\ &+ (X_{\downarrow}(r3,G)^{\uparrow}2^{|}X_{\downarrow}(r1,G)^{\uparrow}2)f(X_{\downarrow}(r2,G))) \\ &/((X_{\downarrow}(r1,G)^{|}X_{\downarrow}(r2,G))f(X_{\downarrow}(r3,G)) \\ &+ (X_{\downarrow}(r2,G)^{|}X_{\downarrow}(r3,G))f(X_{\downarrow}(r1,G)) + (X_{\downarrow}(r3,G)^{|}X \end{aligned} \tag{4}$$

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 method that makes use of gradient in a numerical way. It is a direct search optimization method and has given good results in several cases [44-46].

At each generation, every 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 $\vec{\lambda}_i = (\lambda_i(p_1), \cdots, \lambda_i(p_\beta))$, 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 MSDE is outlined as follows:

Pseudo code of MSDE	
Step1:	Determine the initial set S using random number generator and initially assign mixed strategy as: $\vec{\lambda}_i = (\lambda_i(p_1), \lambda_i(p_2)) = (0.5, 0.5)$
Step2:	Calculate the objective function value $f(X_i)$ for all X_i.
Step3:	Set $i=0$.
Step4:	$i=i+1$.
Step5:	For Target vector X_i (parent vector) choose strategy (mutation operator) according to probability distribution $\vec{\lambda}_i$. If probability of pure strategy p_1 is greater than the probability of strategy p_2 then go to step 6 otherwise 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 given in Equation (4).
Step8:	Recombine 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:	If the trial vector is within the given range then go to step 10 otherwise bring the trial vector 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 $u_{i,j} > x_{\max,j}$ and 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 $\alpha = 1, 2$ and new point survive in next generation $(G + 1)$ then Otherwise
Step13:	If $I <$ population size then go to step 4 else go to step 14.
Step14:	Check whether convergence criterion is met. If yes, stop; otherwise go to step 3.

The mixed strategy scheme is applied to the family of DE algorithms given in Section 2.1 using binomial crossover. However, while comparing it with other algorithms, we have made use of the last scheme described in Section 2.1 because as mentioned earlier this is the most commonly used version of DE and all the algorithms taken for comparison in this study follow the same.

4. **Experimental Setup.** The main parameters of DE are population size, scaling factor and the crossover rate. After conducting several experiments for deciding the optimal choice of parameters, we considered the following parameter setting. The number of individuals in the population is taken as a fixed quantity, 100. Values of scaling factor F outside the range of 0.4 to 1.2 are rarely effective, so the value of F is taken as 0.5, which is generally considered good initial choice. The crossover rate C_r is taken as 0.33. The proposed MSDE algorithm has an additional parameter γ, for which the value is taken as 1/3 [41].

All the algorithms are executed on a PIV PC, using DEV C++, thirty times for each problem. In every case, a run is terminated when the function values of all points in population S were identical to an accuracy of five decimal places, i.e., $|f_{\max} - f_{\min}| \le \varepsilon = 10^{-5}$ or when the maximum number of function evaluations (NFE $= 10^6$) was reached, whichever occurred first.

5. **Benchmark Problems.** The performance of the proposed algorithm is tested on a set of eleven benchmark problems taken from literature [30]. All the functions are multimodal in nature except for the functions f_{CV} and f_{RB} which are unimodal. Functions f_{CV}, f_{PAT}, f_{RB}, f_{CB6} and f_{H3} are of fixed dimensions 4, 5, 10, 2 and 3 respectively and the remaining test problems are scalable in nature. The scalable problems are tested for dimensions 10, 20 and 50. Thus the total number of cases considered is 23. Abbreviation and name of the problems are as:

Table 1

f_{SWF}:	Schwefel,	f_{RB}:	Rosenbrock,	f_{QU}:	Quartic,
f_{ACK}:	Ackley,	f_{GW}:	Griewenk,	f_{PAT}:	Pathological,
f_{CV}:	Colville,	f_{H3}:	Hartman 3,	f_{RG}:	Rastrigin,
f_{PN1}:	Generalized penalized1,			f_{CB6}:	Six hump Camel back

6. Numerical Results and Comparisons.

6.1. **Comparisons between DE and MSDE.** This section compares MSDE with the basic DE algorithm. Table 2 gives average fitness of function values, standard deviation, t-values and average error. Average error is defined as the difference between the true global optimum value and the value obtained by the algorithm. Table 3 provides number of function evaluations (NFE), improvement in term of number of functions evaluation of algorithms. As it is clear from the Table 2 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 2, 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 3, which gives the average number of function evaluations. From Table 3, 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 Quartic function (f_{QU}), in which both the algorithms approach to the maximum number of function evaluation (NFE $= 10^6$). Only in case of Rosenbrock function (f_{RB}), MSDE took more number of function evaluation than basic DE. In terms of percentage improvement in number of function evaluations, MSDE reduces the number of function evaluation up to 96% for the function (f_{RG}) of dimension 50 which is the maximum reduction in number of function evaluations in all the cases. The total NFE taken by DE is 6659200 whereas in case of MSDE, the total NFE is 4441250 only. An overall acceleration rate (AR) [30], for the proposed MSDE algorithm is about AR = 33.30%. This implies that the proposed MSDE algorithm is more than 30% faster than the basic DE algorithm.

Performance curves (convergence graphs) of few selected functions are illustrated in Figures 1(a)-(d). From these graphs also we can see that the convergence of proposed algorithm is faster than basic DE.

6.2. **Influence of dimensionality.** It is often observed that the performance of an algorithm may deteriorate with the increase in the dimension of the problem. Therefore, in order to check the influence of dimensionality on the proposed MSDE we increased the dimensions of the scalable problems from 50 to 200 (taking an interval of 50).

Here, we fixed the maximum NFE as 10^5 and recorded the corresponding results in Table 4 in terms of average fitness and standard deviation. All the problems considered are

TABLE 2. Mean fitness, standard deviation of functions in 30 runs and t-value

Fun.	Dim.	Mean fitness (Std)		Average error		t-value
		DE	MSDE	DE	MSDE	
f_{SWF}	10	-4189.83 (4.1263e-007)	-4189.83 (3.00514e-007)	0.000128469	0.000128368	0
	20	-8379.66 (9.66001e-007)	-8379.66 (8.31499e-007)	0.000258515	0.000257919	0
	50	-20949.1 (1.29067e-006)	-20949.1 (1.25273e-006)	0.00064596	0.000645155	0
f_{ACK}	10	4.89922e-006 (9.3006e-007)	3.65155e-007 (4.20151e-007)	4.89922e-006	3.65155e-007	24.33
	20	1.02463e-005 (2.00048e-006)	3.01699e-006 (1.35368e-006)	1.02463e-005	3.01699e-006	16.39
	50	2.31386e-005 (2.21348e-006)	6.72181e-006 (1.04039e-006)	2.31386e-005	6.72181e-006	36.76
f_{CV}	4	4.5118e-008 (2.32542e-008)	7.83712e-010 (1.15383e-009)	4.5118e-008	7.83712e-010	10.42
f_{QU}	10	1.20068e-004 (4.75295e-005)	1.77752e-005 (1.22154e-005)	0.000120068	1.77752e-005	11.41
	20	8.03444e-004 (0.000154331)	1.21088e-004 (4.24884e-005)	0.000803444	0.000121088	23.34
	50	6.92452e-003 (0.00125875)	4.3024e-004 (0.000126203)	0.0003418	0.00002384	28.11
f_{PAT}	5	5.14828e-004 (0.000262477)	0.000000 (0.000000)	0.000514828	0.000000	10.74
f_{RG}	10	1.62983e-006 (5.62915e-007)	3.70693e-008 (5.54947e-008)	1.62983e-006	3.70693e-008	15.42
	20	3.30759e-006 (5.62531e-007)	5.15673e-007 (2.33515e-007)	3.30759e-006	5.15673e-007	25.10
	50	1.56347e+002 1.55847	1.7742e-006 3.51253e-007	156.347	1.7742e-006	549.47
f_{RB}	10	1.81887e-006 (1.77091e-006)	5.57968e-005 (0.000157562)	1.81887e-006	5.57968e-005	1.87
f_{GW}	10	9.48435e-007 (3.33338e-007)	5.85615e-008 (4.37959e-008)	9.48435e-007	5.85615e-008	14.49
	20	3.68394e-006 (9.27743e-007)	5.40822e-007 (2.69767e-007)	3.68394e-006	5.40822e-007	17.81
	50	9.66678e-006 (1.00454e-006)	1.69896e-006 (4.23073e-007)	9.66678e-006	1.69896e-006	40.03
f_{PN1}	10	9.38479e-007 (1.77917e-007)	1.8547e-007 (3.77841e-008)	9.38479e-007	1.8547e-007	22.67
	20	3.80868e-006 (8.9966e-007)	1.55568e-006 (4.87461e-007)	3.80868e-006	1.55568e-006	12.06
	50	9.30713e-006 (1.2164e-0060)	6.37824e-006 (1.25104e-006)	9.30713e-006	6.37824e-006	12.82
f_{CB6}	2	-1.03163 (7.93442e-009)	-1.03163 (1.28681e-014)	1.55038e-006	0.000268453	0
f_{H3}	3	-3.8623 (4.34641e-008)	-3.8623 (6.36759e-010)	0.000482381	0.000482339	0

TABLE 3. Number of functions evaluation % improvement and average time of functions in 30 runs

Fun.	Dim.	No of function Eva.		% Improvement
		DE	MSDE	
f_{SWF}	10	26710	19690	26.28229
	20	56090	40450	27.88376
	50	190700	141520	25.7892
f_{ACK}	10	31040	14350	53.76
	20	57830	20390	64.74
	50	154490	39260	74.58
f_{CV}	4	76160	58780	22.82
f_{QU}	10	1e+006	1e+006	0.00
	20	1e+006	1e+006	0.00
	50	1e+006	1e+006	0.00
f_{PAT}	5	1e+006	27200	97.28
f_{RG}	10	52850	14060	73.3964
	20	345100	23160	93.2889
	50	1e+006	36580	96.342
f_{RB}	10	141340	888730	0.00
f_{GW}	10	60880	13980	77.03
	20	52690	16990	67.75
	50	121930	30080	75.33
f_{PN1}	10	20600	8070	60.82524
	20	45250	13610	69.92265
	50	213080	28880	86.44641
f_{CB6}	2	7190	2400	66.62
f_{H3}	3	5270	3070	41.74573
		Σ 6659200	**Σ** 4441250	**AR**=33.3065

multimodal in nature where the complexity increases with the increase in dimensionality of the problem. We can observe that up to dimension 50, the two algorithms are comparable only in case of f_{SWF} and f_{QU}. For all the other problems there is a significant difference in the fitness function value for dimension 50 particularly in case of f_{RG} and f_{PN1}, where there is 100% improvement in the function value. For dimensions higher than 50, except for f_{SWF}, there is more than 90% improvement in the fitness function value for all the test cases.

The results clearly indicate that MSDE surpasses DE in all the cases. The superior performance of MSDE is also evident from Figure 2 and Figure 3 with respect to function

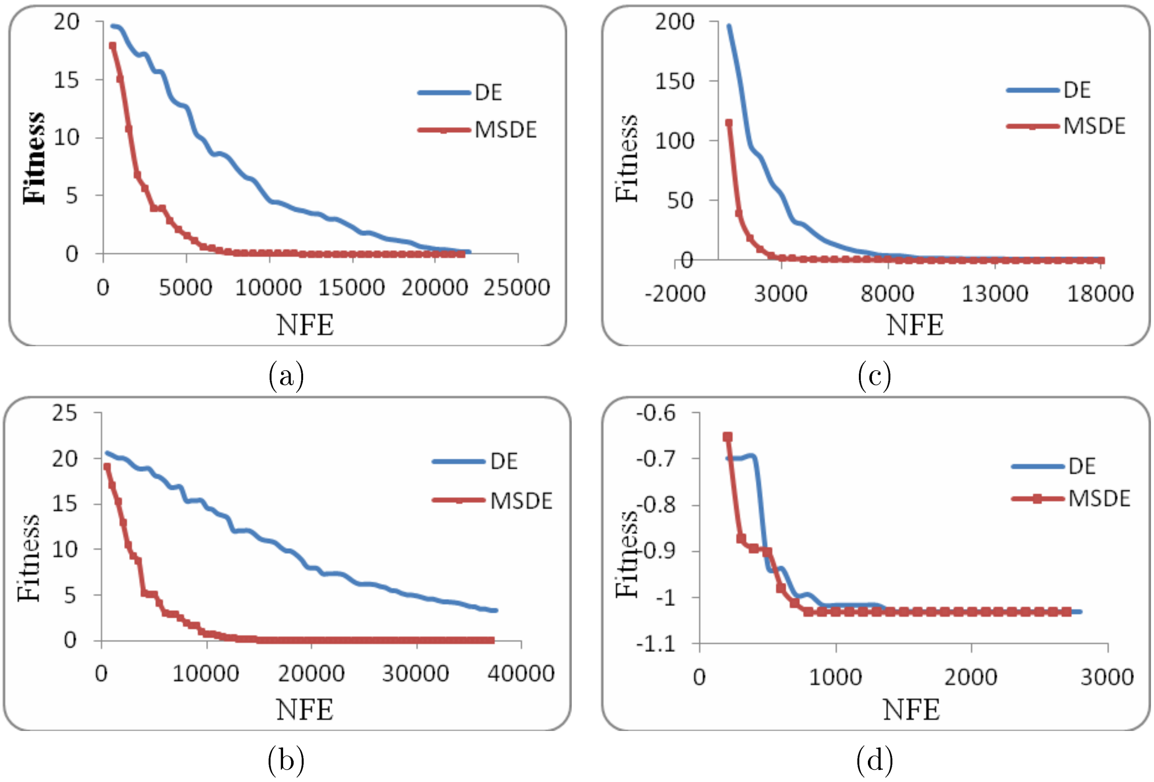

FIGURE 1. (a): performance curves of DE vs. MSDE for function f_{ACK}, dimension 20; (b): performance curves of DE vs. MSDE for function f_{ACK}, dimension 50; (c): performance curves of DE vs. MSDE for function f_{GW}, dimension 20; (d): performance curves of DE vs. MSDE for function f_{CB6}

f_{ACK}. Figure 2 shows 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 show the effect on fitness function value with the increase in dimension. From the graph, it can be seen that the fitness remains almost consistent for MSDE when the dimension is increased, whereas the performance of basic DE, deteriorates with the increase in dimension of the problem.

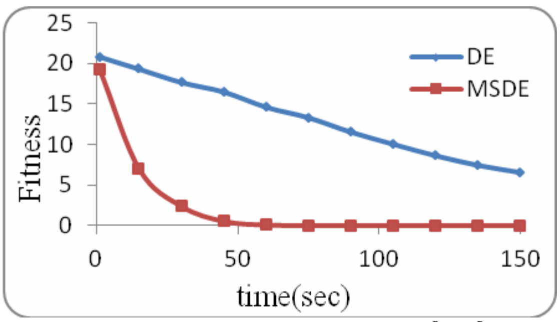

FIGURE 2. Fitness Vs time for function f_{ACK} for 200 Dim

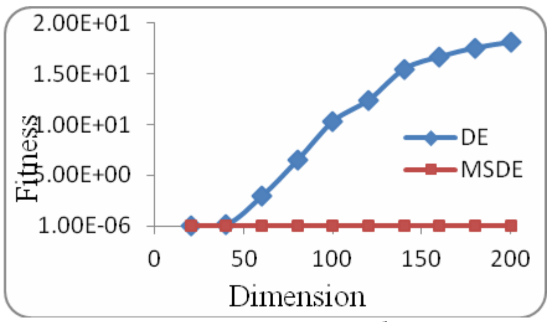

FIGURE 3. Fitness Vs dimension for function f_{ACK}

7. **Comparison of MSDE with Other Modified Versions of DE.** The performance of the proposed MSDE algorithm is further assessed by comparing it with five recent modified versions of DE available in literature namely: Trigonometric mutation differential evolution, TDE [34], DEahcSPX [47], bare bone differential evolution, BBDE [36], differential evolution with random localization, DERL[48] and differential evolution with

TABLE 4. Mean fitness and standard deviation of scalable function in 30 runs

Fun.	Dim.	Mean fitness		Std	
		DE	MSDE	DE	MSDE
f_{SWF}	50	-20337.8	-20924.8	836.725	48.2742
	100	-19599.9	-17510	771.349	407.162
	150	-22422.4	-20581.3	520.802	700.372
	200	-25550.6	-23108.3	397.804	423.306
f_{ACK}	50	0.00390277	6.98221e-007	0.000390698	4.52771e-008
	100	3.81244	8.12885e-007	0.070367	5.95926e-007
	150	10.5221	9.29643e-007	0.215458	1.07938e-007
	200	15.6727	1.98694e-006	0.150102	7.44813e-007
f_{QU}	50	0.0741305	0.00441708	0.0112104	0.00110143
	100	0.980489	0.0141425	0.147694	0.00333821
	150	45.2345	0.0201761	3.21039	0.0260641
	200	408.698	0.0211157	27.574	0.0165785
f_{RG}	50	268.947	1.75744e-007	9.41583	3.38434e-008
	100	841.086	2.70473e-007	11.1428	4.53267e-008
	150	1463.33	2.75742e-007	32.5051	1.72157e-008
	200	2202.76	4.91036e-007	32.3485	3.28364e-007
f_{GW}	50	0.000612059	2.30494e-007	0.000248435	3.77458e-008
	100	2.64676	2.57621e-007	0.185725	4.50721e-008
	150	98.2393	3.09592e-007	3.78594	8.87582e-008
	200	535.535	4.11414e-007	14.579	3.17781e-007
f_{PNI}	50	62787.6	7.07231e-007	44504.5	1.66856e-007
	100	2.12673e+007	7.84554e-007	1.17872e+006	1.26001e-007
	150	1.75373e+008	1.40968e-006	1.87007e+007	2.78722e-007
	200	6.82924e+008	4.05958e-006	3.93012e+007	3.06801e-006

preferential crossover, DEPC [48]. In this section, we give a brief description of these algorithms.

In order to compare the proposed MSDE algorithm with the algorithms, we selected the test problems common to the present study and to the algorithm with which MSDE is being compared. Also in order to be fair, we have taken the same comparison criteria as mentioned in the literature of these algorithms. The results obtained are summarized in Tables 5, 6, 7 and 8.

In Table 5, we show the comparison of MSDE with TDE. Because the data for comparison purpose is not given in [34], we have taken the same parameter setting as given in [34] and run TDE thirty times for each function, here, we have taken dimension twenty for scalable problems. The numerical results show that the mean fitness value as well as number of function evaluations obtained by MSDE is better than TDE in all cases.

Comparison between MSDE and DEahcSPX [47] is made in Table 6. Following the comparison criteria given in [47]; we first fixed the number of function evaluations (NFE) as $3{*}10^5$ and executed the proposed MSDE thirty times up to maximum NFE and recorded the average fitness function value. We then fixed the accuracy as 10^{-6} and recorded the maximum NFE required to obtain the desired accuracy. From the results given in Table 6, it can be seen that MSDE surpasses DEahcSPX in 4 cases while DEahcSPX outperforms MSDE in the remaining 2 cases out of the total 6 cases considered for comparison.

Comparison between MSDE and BBDE [36] is given in Table 7. In this case, the number of test problems common to both the papers is 11. As in [36], comparison is made in terms of average fitness function value only we executed the proposed MSDE algorithm thirty times up to maximum NFE $= 5 * 10^4$ and recorded the fitness. The corresponding results show that MSDE surpasses BBDE in 7 cases while BBDE outperformed MSDE in 3 cases and in the remaining case, there was a tie between BBDE and MSDE.

In Table 8, a comparison is made between MSDE, DERL [48] and DEPC [48]. Here, we made comparison in term of NFE, for this, we run our algorithm MSDE thirty times to achieve an accuracy 10^{-4} and recorded the NFE, here, we have taken the dimension of scalable problems as 10. From the corresponding results given in Table 8, we can easily observe that MSDE surpasses DELR and DEPC in 5 cases while DERL outperform MSDE in 3 cases out of 11 cases.

Table 5. MSDE Vs TDE in term of fitness value, standard deviation and average of function evaluation in 30 runs for dimension 20

Fun	Mean fitness		Standard deviation (Std)		No of fun. Evaluation	
	TDE	MSDE	TDE	MSDE	TDE	MSDE
f_{SWF}	-8379.66	-8379.66	2.61108e-007	8.31499e-007	679320	**40450**
f_{ACK}	8.38861e-006	**3.01699e-006**	2.44146e-006	1.35368e-006	62800	**20390**
f_{CV}	1.56403e-007	**7.83712e-010**	1.07845e-007	1.15383e-009	30420	58780
f_{QU}	0.000413125	**0.000121088**	0.000166186	4.24884e-005	1e+006	1e+006
f_{PAT}	0.00201115	**0.000000**	0.00264581	0.000000	368240	**27200**
f_{RG}	10.5827	**5.15673e-007**	2.83003	2.33515e-007	1e+006	**23160**
f_{RB}	4.85175e-006	3.51466	2.95541e-006	0.378219	242140	**1e+006**
f_{GW}	0.00148201	**5.40822e-007**	0.00295796	2.69767e-007	54320	**16990**
f_{PN1}	2.61676e-006	**1.55568e-006**	5.65888e-007	4.87461e-007	125660	**13610**
f_{CB6}	-1.03163	-1.03163	1.11409e-008	1.28681e-014	9040	**2400**
f_{H3}	-3.8623	-3.8623	2.0025e-008	6.36759e-010	6500	**3070**

8. **Analysis of Mixed Strategy on Family of DE Algorithms Using Non Parametric Test.** As mentioned in Section 2.1, Storn and Price suggested ten versions of DE collectively known as the family of DE algorithms. These versions typically differ from each other in the manner in which mutation and crossover operators are applied. In this section, we analyze the application of mixed strategy on the last five versions mentioned

Table 6. MSDE Vs DEachSPX in term of mean fitness value, standard deviation and average of function evaluation in 30 runs for dimension 30

Fun.	Mean fitness after NFE=300000		Std		NFE to achieve an accuracy 10^{-6}		t-value
	DEahcSPX	MSDE	DEahcSPX	MSDE	DEahcSPX	MSDE	
f_{SWF}	4.70e+02	**3.82e-04**	2.96e+02	6.12e-06	--	**63600**	8.69
f_{ACK}	2.66e-15	**1.45e-16**	0.00e+00	6.28e-17	129211	**30370**	219.35
f_{RG}	2.14e+01	**0.00e+00**	1.23e+01	0.00e+00	--	**30080**	9.52
f_{RB}	4.52e+00	**2.19e+01**	1.55e+01	1.05e-01	299913	--	-6.14
f_{GW}	2.07e-03	**0.00e+00**	5.89e-03	0.00e+00	121579	**22990**	1.92
f_{PNI}	2.07e-02	**1.35e-19**	8.46e-02	9.68e-27	96149	**21950**	1.34

Table 7. MSDE Vs BBDE in term of mean fitness value, standard deviation and t-value

Fun.	Dim	Mean fitness after NFE=5000		Std		t-value
		BBDE	MSDE	BBDE	MSDE	
f_{SWF}	30	-11649.008729	**-12569.5**	272.707782	0.0167741	18.48
	100	-34746.152554	-15032.4	3750.927593	486.798	-28.54
f_{ACK}	30	0.0	5.28125e-012	0.0	3.78586e-012	-7.64
	100	0.0	0.000136263	0.000001	7.03266e-005	-10.61
f_{RG}	30	37.551246	**0.00**	15.254959	0.00	13.48
	100	616.194754	**2.68118e-005**	38.115845	1.87538e-005	88.54
f_{RB}	30	47.857080	**26.0205**	31.835408	0.120493	3.75
	100	312.632070	97.1313	195.546311	0.0283949	6.03
f_{GW}	30	0.000657	0.00	0.002583	0.00	1.39
	100	0.001640	2.23292e-006	0.005296	2.09332e-006	1.69
f_{CB6}	2	-1.031628	-1.031628	0.0	2.22045e-016	0

in Section 2.1. These versions make use of binary crossover and are more popular than the other five versions.

The usual parametric tests like two tailed student t-test that are commonly used for analyzing two algorithms cannot be used when we are simultaneously comparing more than two algorithms. In a recent study, performed by Garcia et al [49], it was suggested with the help of several examples and statistical tests that the parametric statistical analysis is not be appropriate specially when dealing with multiple problems results. In multiple problem analysis, they proposed the use of non-parametric statistical tests given that they are less restrictive than parametric ones and they can be used over small size sample results.

Table 8. MSDE Vs DERL and DEPC in term of mean of function evaluation and mean time

Fun.	No of Function Evaluation to achieve accuracy 10^{-4}			Time (in sec)		
	DERL	DEPC	MSDE	DERL	DEPC	MSDE
f_{SWF}	21738	24046	**19690**	0.86	0.37	**0.20**
f_{ACK}	21983	29825	**14350**	0.75	0.80	1.10
f_{RG}	96428	26927	**14060**	2.11	0.42	**0.20**
f_{RB}	198584	512165	888730	7.28	10.52	14.88
f_{GW}	15231	47963	**13980**	0.67	0.75	**0.30**
f_{PN1}	9568	13732	**8070**	0.28	0.19	0.20
f_{CB6}	766	911	2400	0.04	0.04	**0.01**
f_{H3}	1032	1354	3070	0.03	0.03	**0.02**

The following analysis is used to determine in a rigorous manner whether there is a statistically significant difference in the results, average error taken from Table 2, indicating that the proposed algorithms outperforms the basic DE algorithm. Parametric statistical tests to measure significance (e.g., the t-test or ANOVA) assume samples are nearly normally distributed. However, in practice samples are often not normally distributed [50] and applying parametric methods on such samples can lead to incorrect statistical inferences [51]. The significance of using non-parametric analysis for the comparison of experimental data, which is not uniformly distributed, has been shown by many researchers. A detailed study on the use of non-parametric tests for analyzing the evolutionary algorithms is given in [49]. Other examples on the application of non-parametric tests can be found in [51-54].

We first examined whether test results (scores) are nearly normally distributed. For this analysis, we used the two-tailed Kolmogorov-Smirno and Shapiro-Wilk test which provides test statistics and significant-values. The null and alternative hypotheses are:

$$H_0 : \; F(x) = G(x) \tag{5}$$

$$H_1 : \; F(x) \neq G(x) \tag{6}$$

where $F(x)$ is the unknown distribution function of a set of results and $G(x)$ is the Gaussian one, respectively. Table 12 provides the test statistic and significant values of the normality testes over the samples results obtained by DE and MSDE. Figures 5 and 6 represent the corresponding histograms and Q-Q plots for such samples. Now, from Table 12, we can see that the null hypothesis H_0 is rejected at both significance levels $\alpha = 0.05$ and $\alpha = 0.10$, because significant values in Table 12 are less than significance levels and hence assume the results are not normally distributed. This implies that nonparametric methods will be more efficient. We then compared the behaviour of the two algorithms by means of pair wise statistical tests:

- The p-value with a paired t-test is $p = 0.328$. The paired t-test does not consider the existence of difference in performance between the algorithms.

- The p-value with a Wilcoxon test is $p = 0.002$. The Wilcoxon t-test also does not consider the existence of difference in performance between the algorithms, but it considerably reduces the minimal level of significance for detecting differences.

Next, we examined whether a statistically significant difference exist between any of the sets of results of Table 11.

Values in Table 11 allow us to carry out a rigorous statistical study. Our study is focused on the algorithm that had the lowest average error rate in comparison, MSDE/rand/1/bin. We studied the behavior of this algorithm with respect to remaining ones and determined if the results it offered are better than the ones offered by the rest of algorithms, computing the p-values for each comparison. Table 13, shows the result of applying Friedman's test in order to see whether there are global differences in the results. Given that the p-values of Friedman is lower than the level of significance considered $\alpha = 0.05$, there are significant differences among the observed results. Attending to these results, a post-hoc statistical analysis could help us to detect concrete differences among algorithms. For this, we will employ Bonferroni-Dunn's test to detect significant differences for the control algorithm. Table 14 summarizes the ranking obtained by Friedman's test and the critical difference of Bonferroni-Dunn's procedure. Figure 7 display graphical representations (including the rankings obtained for each algorithm). In a Bonferroni-Dunn's graphic, the difference among rankings obtained for each algorithm is illustrated. In them, we can draw a horizontal cut line which represents the threshold for the best performing algorithm, that one with the lowest ranking bar, in order to consider it better than other algorithm. 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. Those bars which exceed this line are associated to an algorithm with worse performance than the control algorithm. The application of Bonferroni-Dunn's test informs us that *MSDE/rand/1/bin* is better than *MSDE/best/1/bin* and *MSDE/target-to-best/1/bin* with $\alpha = 0.05$ and $\alpha = 0.10$ (2/5 algorithms) whereas its performance is at par with the remaining two DE versions.

9. **Comparative Performance of Algorithms on Real Life Problems.** The performance of proposed MSDE algorithm is further analyzed on two interesting real life problems which are common in various fields of engineering designs [55]. These are: ($F1$) frequency modulation sound parameter identification and ($F2$) the spread spectrum radar poly-phase code design problem. The first problem is of fixed dimension, six. The second problem $F2$ is tested for two sets of dimensions; 19 and 20. Both the problems considered in the present study are highly nonlinear in nature. Mathematical models of the problems are given in the Appendix.

Numerical results to test the performance measures of algorithms for the real life problems are given in Table 15. From this Table, we can see that in terms of average fitness function value, the proposed algorithm gives better solution in comparison to other algorithms. Standard deviation of MSDE is lesser than other algorithms for frequency modulation sound parameter identification problem while for the spread spectrum radar poly-phase code design problem its standard deviation is more or less similar to other algorithms.

10. **Discussion and Conclusions.** In this paper, we proposed a modified version of basic DE called Mixed Strategy Differential Evolution (MSDE) based on evolutionary game theory. In MSDE algorithm, a choice of two mutation strategies is given to the individuals by incorporating a mixed mutation strategy. The simulation of results show

that the proposed algorithm is quite competent for solving problems of different dimensions in lesser time and lesser number of function evaluations without compromising with the quality of solution. Numerical results show that while using MSDE, there is an improvement of more than 30% in convergence rate in comparison to basic DE. Further for higher dimensions also the proposed MSDE surpassed the basic DE quite significantly in terms of fitness function value.

The performance of MSDE is also compared with five other recently modified versions of DE algorithm available in literature. In order to give an advantage to the algorithms with which we were comparing the MSDE algorithm, we changed the parameter settings of DE as that of other algorithms. The corresponding numerical results showed that even under the changed parameter settings, the MSDE algorithm performed better than other algorithms in most of the cases.

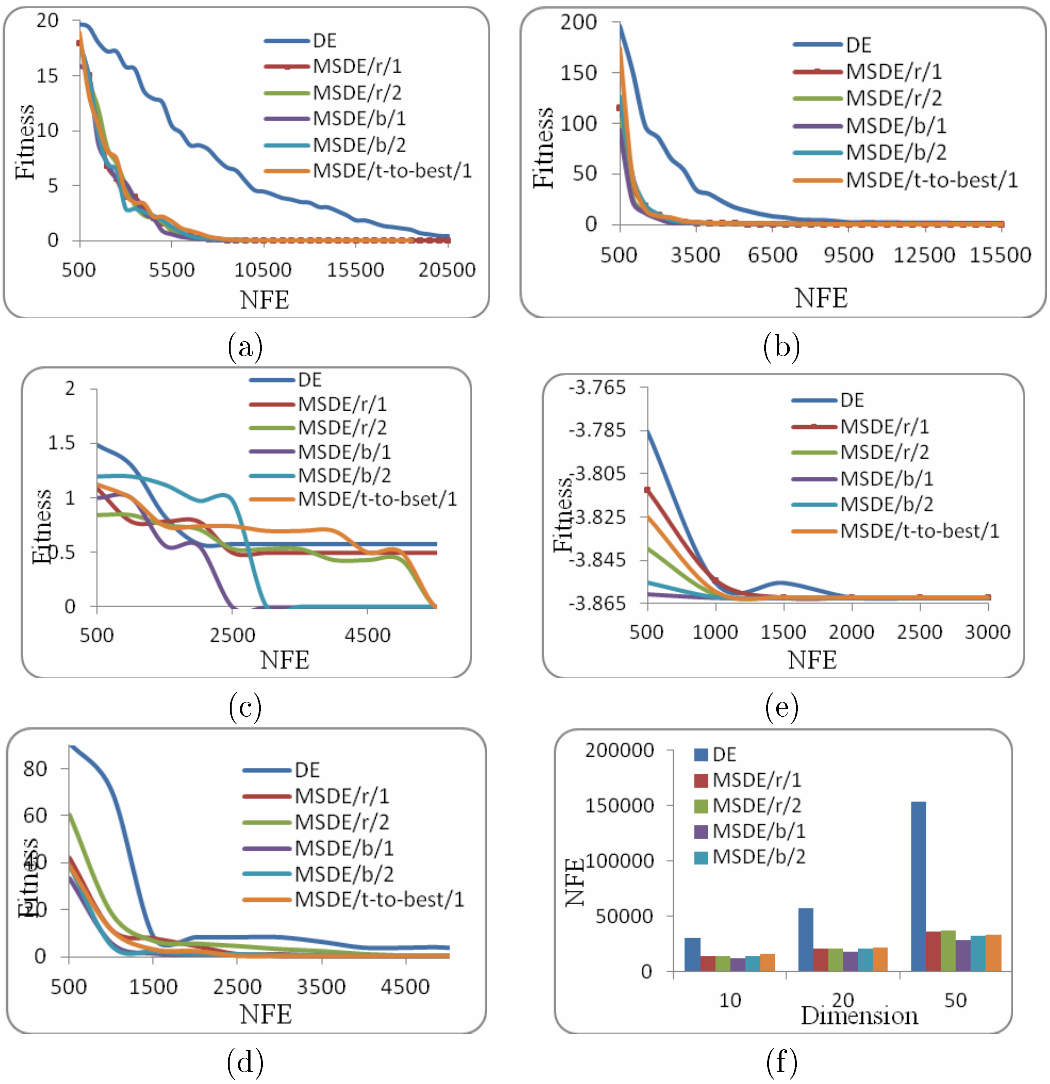

FIGURE 4. (a): performance curves of DE vs. MSDE for function f_{ACK}, dimension 20; (b): performance curves of DE vs. MSDE for function f_{GW}, dimension 20; (c): performance curves of DE vs. MSDE for function f_{PAT}, dimension 5; (d): performance curves of DE vs. MSDE for function f_{CV}; (e): performance curves of DE vs. MSDE for function f_{H3}; (f): performance curves of DE vs. MSDE for function f_{ACK}

Although we have applied the concept of mixed strategy on DE/rand/1/bin version, the discussion in Section 8 shows that all the versions of DE will perform more or less in a similar manner on application of mixed strategy. This shows that mixed mutation strategy is beneficial in comparison to single strategy. One apparent drawback of proposed MSDE is that for noisy functions like f_3 it takes more time than the basic DE, although the numbers of function evaluations are same.

We would like to maintain that the work is still in the stage of infancy and we are working on it to further improve its performance. In this paper, we have taken only two strategies we intend to work with more strategies in future. The concept of mixed strategy can be applied to population generation and crossover rates also.

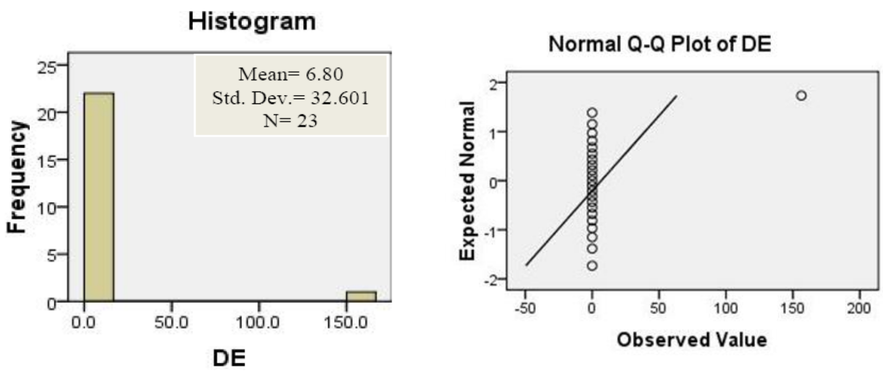

FIGURE 5. DE algorithm: Histogram and Q-Q graphics

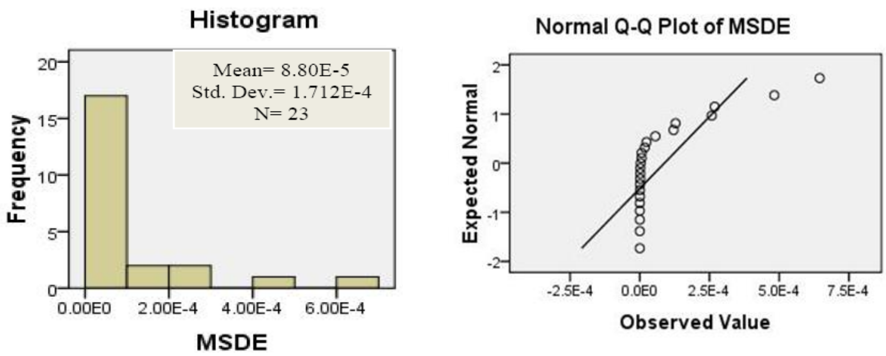

FIGURE 6. MSDE algorithm: Histogram and Q-Q graphics

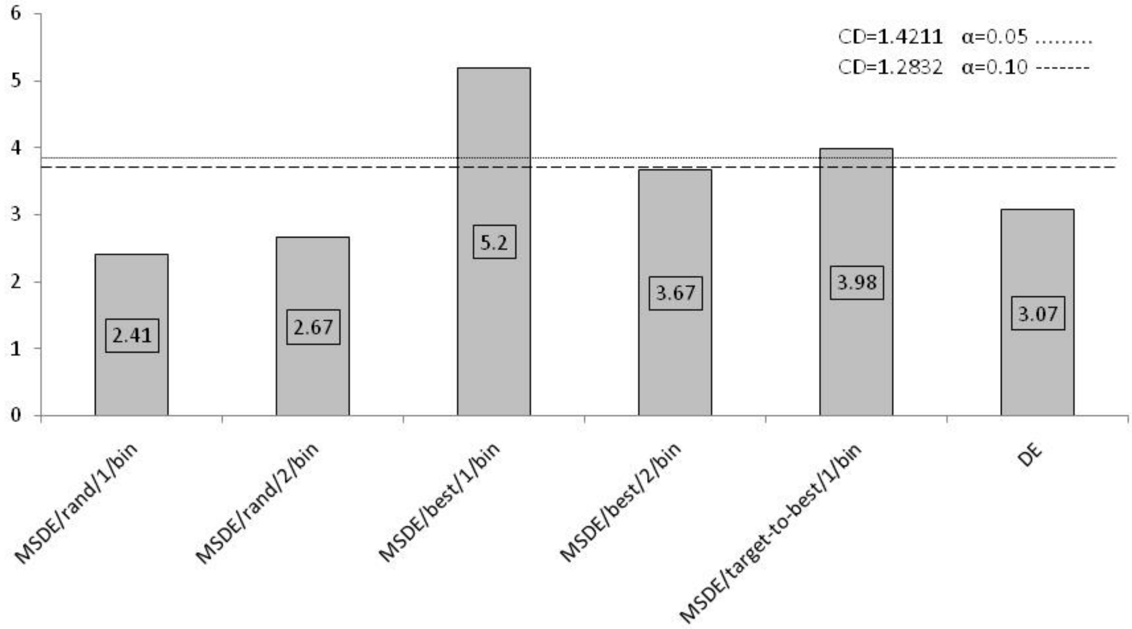

FIGURE 7. Bonferroni-Dunn's graphic corresponding to the results for family of DE algorithms

REFERENCES

[1] Y. Wang, Fuzzy clustering analysis by using genetic algorithm, *ICIC Express Letters*, vol.2, no.4, pp.331-337, 2008.

[2] C. Liu, New evolutionary algorithm for multi-objective constrained optimization, *ICIC Express Letters*, vol.2, no.4, pp.339-344, 2008.

[3] C.-A. Liu and Y. Wang, A new dynamic multi-objective optimization evolutionary algorithm, *International Journal of Innovative Computing, Information and Control*, vol.4, no.8, pp.2087-2096, 2008.

[4] Y.-P. Guo, X.-B. Cao and J. Zhang, Constraint handling based multiobjective evolutionary algorithm for aircraft landing scheduling, *International Journal of Innovative Computing, Information and Control*, vol.5, no.8, pp.2229-2238, 2009.

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

[6] T. Back, F. Hoffmeister and H. Schwefel, A survey of evolution strategies, *Proc. of the 4th International Conference on Genetic Algorithms and Their Applications*, pp.2-9, 1991.

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

[8] J. Kennedy and R. C. Eberhart, Particle swarm optimization, *IEEE Int. Conf. on Neural Networks*, Perth, Australia, pp.1942-1948, 1995.

[9] 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.

[10] T. Rogalsky, S. Kocabiyik and R. Dirkson, Differential evolution in aerodynamic optimization, *Canadian Aeronautics and Space Journal*, vol.46, no.4, pp.183-190, 2000.

[11] G. Stumberger, D. Dolinar, U. Pahner and K. Hameyer, Optimization of radial active magnetic bearings using the finite element technique and differential evolution algorithm, *IEEE Transactions on Magnetics*, vol.36, no.4, pp.1009-1013, 2000.

[12] S. Doyle, D. Corcoran and J. Connell, Automated mirror design using an evolution strategy, *Optical Engineering*, vol.38, no.2, pp.323-333, 1999.

[13] F. S. Wang and J. W. Sheu, Multi-objective parameter estimation problems of fermentation processes using high ethanol tolerance yeast, *Chemical Engineering Science*, vol.55, no.18, pp.3685-3695, 2000.

Table 9. Mean of fitness of functions in 30 runs

Fun.	Dim.	Mean fitness					
		MSDE/rand/1	MSDE/rand /2	MSDE/best/1	MSDE/best/ 2	MSDE/tar-to-best/1	DE
f_{SWF}	10	-4189.83	-4189.83	-4189.83	-4189.83	-4189.83	-4189.83
	20	-8379.66	-8379.66	-8142.78	-8379.66	-8379.66	-8379.66
	50	-20949.1	-20949.1	-18955.4	-20238.5	-19172.6	-20949.1
f_{ACK}	10	9.36269e-007	**7.35553e-007**	9.04351e-007	1.23575e-006	1.31125e-006	4.79698e-006
	20	**2.81864e-006**	2.99136e-006	4.87134e-006	3.20752e-006	4.58544e-006	1.05519e-005
	50	6.69271e-006	**5.80025e-006**	9.17246e-006	7.42857e-006	1.09127e-005	2.48575e-005
f_{CV}	4	5.20984e-010	6.07e-010	3.58381e-010	2.76867e-010	**1.20257e-011**	5.69121e-008
f_{QU}	10	2.15633e-05	**1.01884e-005**	2.25509e-005	4.89016e-005	4.57038e-005	0.000175087
	20	0.000102196	0.00011877	7.11829e-005	0.000119381	**7.10113e-005**	0.000783961
	50	0.00043024	0.000441801	**0.000403578**	0.000577908	0.000513707	0.00692452
f_{PAT}	5	0.000000	0.00000	0.00000	0.00000	0.00000	0.000514828
f_{RG}	10	3.70693e-008	1.74434e-008	1.2059e-007	7.49299e-008	**0.00000**	1.62983e-006
	20	5.15673e-007	3.03167e-007	9.56508e-007	4.4907e-007	**0.00000**	3.30759e-006
	50	1.7742e-006	1.51082e-006	3.57033e-006	2.0836e-006	**0.00000**	156.347
f_{RB}	10	5.57968e-005	0.000241223	0.000436838	0.163859	**3.91871e-012**	1.81887e-006
f_{GW}	10	5.82793e-008	1.8993e-008	1.01552e-007	6.05627e-008	**0.00000**	1.28814e-006
	20	4.2308e-007	4.17229e-007	8.99982e-007	5.97633e-007	**0.00000**	3.8824e-006
	50	1.89749e-006	1.35678e-006	3.1648e-006	1.74817e-006	**3.25261e-020**	9.66551e-006
f_{PN1}	10	1.8547e-007	1.40694e-007	2.25764e-007	2.16883e-007	**4.09094e-019**	9.38479e-007
	20	1.55568e-006	1.17675e-006	1.37731e-006	9.53749e-007	**4.89669e-017**	3.80868e-006
	50	6.37824e-006	7.21941e-006	5.88418e-006	6.12007e-006	**5.73471e-013**	9.30713e-006
f_{CB6}	2	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163
f_{H3}	3	-3.8623	-3.8623	-3.8623	-3.8623	-3.8623	-3.8623

[14] S. Das, A. Abraham and A. Konar, Adaptive clustering using improved differential evolution algorithm, *IEEE Transactions on Systems, Man and Cybernetics – Part A*, 2007.
[15] T. Masters and W. Land, A new training algorithm for the general regression neural network, *Proc. of Computational Cybernetics and Simulation*, vol.3, pp.1990-1994, 1997.
[16] M. Omran, A. Engelbrecht and A. Salman, Differential evolution methods for unsupervised image classification, *Proc. of the IEEE Congress on Evolutionary Computation*, vol.2, pp.966-973, 2005.
[17] R. Storn, Differential evolution design for an IIR-filter with requirements for magnitude and group delay, *Technical Report TR-95-026*, International Computer Science Institute, Berkeley, CA 1995.
[18] B. Babu and R. Angira, Optimization of non-linear functions using evolutionary computation, *Proc. of the 12th ISME International Conference on Mechanical Engineering*, India, pp.153-157, 2001.
[19] R. Angira and B. Babu, Evolutionary computation for global optimization of non-linear chemical engineering processes, *Proc. of International Symposium on Process Systems Engineering and Control*, Mumbai, pp.87-91, 2003.
[20] H. Abbass, A memetic pareto evolutionary approach to artificial neural networks, *Lecture Notes in Artificial Intelligence*, vol.2256, pp.1-12, 2002.

Table 10. Mean of function evaluation in 30 runs

Fun.	Dim.	NFE					
		MSDE/rand/1	MSDE/rand /2	MSDE/best/1	MSDE/best/ 2	MSDE/tar- to-best/1	DE
f_{SWF}	10	19690	18760	**11566**	16875	81600	26710
	20	40450	44370	25900	**29300**	86787	56090
	50	**141520**	233300	54100	95800	1e+006	190700
f_{ACK}	10	14100	14640	**12180**	14120	15830	30910
	20	21100	21090	**17990**	20650	22010	57460
	50	36220	36870	**28880**	32880	33320	153390
f_{CV}	4	58410	53120	**34690**	41280	48790	80730
f_{QU}	10	1e+006	1e+006	1e+006	1e+006	1e+006	1e+006
	20	1e+006	1e+006	1e+006	1e+006	1e+006	1e+006
	50	1e+006	1e+006	1e+006	1e+006	1e+006	1e+006
f_{PAT}	5	27200	**15560**	26070	17640	51730	1e+006
f_{RG}	10	14060	13170	**11470**	13500	27540	52850
	20	23160	20550	**17580**	21150	40210	345100
	50	36580	34710	**28640**	33520	66490	1e+006
f_{RB}	10	888730	1e+006	1e+006	1e+006	1e+006	**141340**
f_{GW}	10	13930	13680	**13630**	13950	32350	66760
	20	16660	16790	**15340**	16520	45790	53830
	50	28420	28570	**22600**	26180	66980	121050
f_{PN1}	10	8070	8340	**7590**	8180	24630	20600
	20	13610	13370	**12120**	13320	37930	45250
	50	28880	28800	**23070**	26200	81330	213080
f_{CB6}	2	2580	2830	**2450**	2670	3760	7460
f_{H3}	3	3070	3230	**2750**	3080	3990	5270

[21] J. Vesterstroem and R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, *Proc. Congr. Evol. Comput.*, vol.2, pp.1980-1987, 2004.
[22] J. Andre, P. Siarry and T. Dognon, An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization, *Advance in Engineering Software*, vol.32, pp.49-60, 2001.
[23] O. Hrstka and A. Kucerová, Improvement of real coded genetic algorithm based on differential operators preventing premature convergence, *Advance in Engineering Software*, vol.35, pp.237-246, 2004.
[24] J. Lampinen and I. Zelinka, On stagnation of the differential evolution algorithm, *Proc. of the 6th International Mendel Conference on Soft Computing*, pp.76-83, 2000.
[25] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, *Proc. of the 9th International Conference on Soft Computing*, pp.41-46, 2003.
[26] H. Abbass, The self-adaptive pareto differential evolution algorithm, *Proc. of the 2002 Congress on Evolutionary Computation*, pp.831-836, 2002.
[27] M. Omran, A. Salman and A. P. Engelbrecht, Self-adaptive differential evolution, computational intelligence and security, *Lecture Notes in Artificial Intelligence*, vol.3801, pp.192-199, 2005.

TABLE 11. Average of error of functions in 30 runs

Fun.	Dim.	Average Error					
		MSDE/rand/1	MSDE/rand /2	MSDE/best/1	MSDE/best/ 2	MSDE/tar-to-best/1	DE
f_{SWF}	10	1.00000e-02	1.00000e-02	1.00000e-02	1.00000e-02	1.00000e-02	1.00000e-02
	20	2.00000e-02	2.00000e-02	2.36878e+02	2.00000e-02	2.00000e-02	2.00000e-02
	50	4.50000e-01	4.50000e-01	1.99374e+03	7.10645e+02	1.77654e+03	4.50000e-01
f_{ACK}	10	9.36269e-07	7.35553e-07	9.04351e-07	1.23575e-06	1.31125e-06	4.79698e-06
	20	2.81864e-06	2.99136e-06	4.87134e-06	3.20752e-06	4.58544e-06	1.05519e-05
	50	6.69271e-06	5.80025e-06	9.17246e-06	7.42857e-06	1.09127e-05	2.48575e-05
f_{CV}	4	5.20984e-10	6.07000e-10	3.58381e-10	2.76867e-10	1.20257e-11	5.69121e-08
f_{QU}	10	2.15633e-05	1.01884e-05	2.25509e-05	4.89016e-05	4.57038e-05	1.75087e-04
	20	1.02196e-04	1.18770e-04	7.11829e-05	1.19381e-04	7.10113e-05	7.83961e-04
	50	4.30240e-04	4.41801e-04	4.03578e-04	5.77908e-04	5.13707e-04	6.92452e-03
f_{PAT}	5	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	5.14828e-04
f_{RG}	10	3.70693e-08	1.74434e-08	1.20590e-07	7.49299e-08	0.00000e+00	1.62983e-06
	20	5.15673e-07	3.03167e-07	9.56508e-07	4.49070e-07	0.00000e+00	3.30759e-06
	50	1.77420e-06	1.51082e-06	3.57033e-06	2.08360e-06	0.00000e+00	1.56347e+02
f_{RB}	10	5.57968e-05	2.41223e-04	4.36838e-04	1.63859e-01	3.91871e-12	1.81887e-06
f_{GW}	10	5.82793e-08	1.89930e-08	1.01552e-07	6.05627e-08	0.00000e+00	1.28814e-06
	20	4.23080e-07	4.17229e-07	8.99982e-07	5.97633e-07	0.00000e+00	3.88240e-06
	50	1.89749e-06	1.35678e-06	3.16480e-06	1.74817e-06	3.25261e-20	9.66551e-06
f_{PNI}	10	1.85470e-07	1.40694e-07	2.25764e-07	2.16883e-07	4.09094e-19	9.38479e-07
	20	1.55568e-06	1.17675e-06	1.37731e-06	9.53749e-07	4.89669e-17	3.80868e-06
	50	6.37824e-06	7.21941e-06	5.88418e-06	6.12007e-06	5.73471e-13	9.30713e-06
f_{CB6}	2	3.00000e-05	3.00000e-05	3.00000e-05	3.00000e-05	3.00000e-05	3.00000e-05
f_{H3}	3	4.00000e-04	4.00000e-04	4.00000e-04	4.00000e-04	4.00000e-04	4.00000e-04

TABLE 12. Normality test over multiple problem analysis

	Kolmogorov-Smirnov			Shapiro-Wilk		
Algo	Statistic	df	Sig.	Statistic	df	Sig.
DE	.539	23	.000	.215	23	.000
MSDE	.342	23	.000	.595	23	.000

TABLE 13. Results of the Friedman test (A=0.05)

N(Total No of fun)	Friedman value	df	p-value
23	41.829	5	0.000

TABLE 14. Ranking obtained through Friedman's test and critical difference of Bonferroni-Dunn's procedure

Algorithm	Mean Rank
MSDE/rand/1/bin	2.41
MSDE/rand/2/bin	2.67
MSDE/best/1/bin	5.20
MSDE/best/2/bin	3.67
MSDE/target-to-best/1/bin	3.98
DE	3.07
Crit.Diff. $\alpha = 0.05$	**1.4211**
Crit.Diff. $\alpha = 0.10$	**1.2832**

TABLE 15. Average and standard deviation for all real life problems. For $F1$ maximum NFE is 10^5 and for $F2$ maximum NFE is $5 * 10^6$

Dim.	Mean of fitness and standard deviation for frequency modulation sound parameter problem.				
	DE	MSDE	DEGL/SAW [55]	SADE [28]	NSDE [31]
6	2.25431e-01 1.65639e-03	6.27847e-017 1.39884e-016	4.8152e-09 6.2639e-08	7.8354e-02 5.8254e-03	9.4559e-03 6.924e-01
Dim.	Mean of fitness and standard deviation for spread spectrum radar poly phase code design problem.				
	DE	MSDE	DEGL/SAW [55]	SADE [28]	NSDE [31]
19	7.74390e-01 1.86246e-02	6.53674e-01 1.91778e-02	7.4439e-01 5.84e-04	7.5932e-01 3.88e-05	7.6094e+01 4.72e-03
20	8.12229e-01 7.38374e-02	2.58407e-01 1.89746e-02	8.0304e-01 2.73e-03	8.3453e-01 6.53e-04	8.4283e-01 3.44e-02

[28] 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, no.6, pp.646-657, 2006.

[29] S. Das, A. Konar and U. Khakraborty, Two improved differential evolution schemes for faster global search, *ACM-SIGEVO Proc. of GECCO*, Washington D.C., pp.991-998, 2005.

[30] S. Rahnamayan, H. R. Tizhoosh and M. M. ASalama, Opposition-based differential evolution, *IEEE Transactions on Evolutionary Computation*, vol.12, no.1, pp.64-79, 2008.

[31] Z. Yang, J. He and X. Yao, Making a difference to differential evolution, *Advances in Metaheuristics for Hard Optimization*, pp.415-432, 2007.

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

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

[34] H.-Y. Fan and J. Lampinen, A trigonometric mutation operation to differential evolution, *Journal of Global Optimization*, vol.27, pp.105-129, 2003.

[35] M. Pant, M. Ali and V. P. Singh, Parent centric differential evolution algorithm for global optimization problems, *Opsearch Springer*, vol.46, no.2, pp.153-168, 2009.

[36] M. G. H. Omran, A. P. Engelbrecht and A. Salman, Bare bones differential evolution, *European Journal of Operational Research*, 2008.

[37] P. K. Bergey and C. Ragsdale, Modified differential evolution: A greedy random strategy for genetic recombination, *Omega the International Journal of Management Science*, vol.33, pp.255-265, 2005.

[38] X. Zhang, Q. Lu, S. Wen, M. Wu and X. Wang, A modified differential evolution for constrained optimization, *ICIC Express Letters*, vol.2, no.2, pp.181-186, 2008.

[39] U. K. Chakraborty (ed.), *Advances in Differential Evolution*, Springer-Verlag, Heidelberg, 2008.
[40] J. W. Weibull, *Evolutionary Game Theory*, MIT Press, Cambridge, MA, 1995.
[41] H. Dong, J. He, H. Huang and W. Hou, Evolutionary programming using a mixed mutation strategy, *Information Science*, vol.177, pp.312-327, 2007.
[42] M. Pant, M. Ali and A. Abraham, Mixed mutation strategy embedded differential evolution, *IEEE Congress on Evolutionary Computation*, pp.1240-1246, 2009.
[43] K. Price, An introduction to DE, in *New Ideas in Optimization*, D. Corne, D. Marco and F. Glover (eds.), London, UK, McGraw-Hill, 1999.
[44] M. Pant, R. Thangaraj and A. Abraham, A new PSO algorithm incorporating reproduction operator for solving global optimization problems, *The 7th International Conference on Hybrid Intelligent Systems*, Kaiserslautern, Germany, pp.144-149, 2007.
[45] C. Mohan and K. Shanker, A controlled random search technique for global optimization using quadratic approximation, *Asia-Pacific Journal of Operational Research*, vol.11, pp.93-101, 1994.
[46] K. Deep and K. N. Das, Quadratic approximation based hybrid genetic algorithm for function optimization, *Applied Mathematics and Computation*, vol.203, pp.86-98, 2008.
[47] N. Noman and H. Iba, Accelerating differential evolution using an adaptive local search, *IEEE Transactions on Evolutionary Computation*, vol.12, no.1, pp.107-125, 2008.
[48] P. Kaelo and M. M. Ali, Numerical study of some modified differential evolution algorithms, *European Journal of Operational Research*, vol.169, pp.1176-1184, 2006.
[49] S. Garcia, D. Molina, M. Lozano and F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC' 2005 special session on real parametric optimization, *Journal of Heuristics*, 2008.
[50] T. Micceri, The unicorn, the normal curve and other improbable creatures, *Psychol Bull*, vol.105, no.1, pp.156-166, 1989.
[51] W. J. Conover, *Practical Nonparametric Statistics*, 2nd Edition, Wiley, New York, 1980.
[52] J. Demsar, Statistical comparison of classifiers over multiple data sets, *Journal of Machine Learning Research*, vol.7, pp.1-30, 2006.
[53] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 4th Edition, Chapman and Hall, Boca Raton, 2007.
[54] Y. Tenne And S. W. Armfield, A framework for memetic optimization using variable global and local surrogate models, *Soft Computing*, vol.13, pp.781-793, 2009.
[55] S. Das, A. Abraham, U. K. Chakraborty and A. Konar, Differential evolution using a neighborhood based mutation operator, *IEEE Transaction on Evolutionary Computation*, vol.13, no.3, pp.526-553, 2009.