Improving Differential Evolution Algorithm by Synergizing Different Improvement Mechanisms

M. ALI and M. PANT
Indian Institute of Technology Roorkee, India
AND
A. ABRAHAM
Machine Intelligence Research (MIR) labs, USA

Abstract— Differential Evolution (DE) has emerged as a popular and efficient population based meta-heuristic for solving global optimization problems. Practical experiences however show that DE is vulnerable to problems like slow and/ or premature convergence. Several attempts have been made in literature to improve its performance. In this paper we propose a simple and modified DE framework, called MDE which is a fusion of three recent modifications in DE (1) opposition based learning (OBL) (2) tournament method for mutation and (3) single population structure of DE. These features have a specific role which helps in improving the performance of DE. While OBL helps in giving a good initial start to DE, the use of tournament best base vector in the mutation phase helps in preserving the diversity besides maintaining the convergence. Finally the single population structure helps in faster convergence. The fusion of the three produces a synergized effect which helps in balancing the two antagonist factors exploitation and exploration, without compromising with the solution quality or the rate of convergence. The proposed MDE is validated on a set of 25 standard benchmark problems with varying degrees of complexities, 7 nontraditional shifted benchmark functions proposed at the special session of CEC2008, and three engineering design problems. To justify the effect of synergy, MDE is compared with algorithms that use these schemes separately namely ODE, DERL and MDE1 and also with other DE variants; JADE, SaDE and jDE. Numerical results and statistical analysis show that the proposed MDE is better than or at least comparable with these algorithms.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control. Methods, and Search—*Heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization—*Global optimization*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Differential Evolution, Evolutionary Algorithms, Synergy

Authors' addresses: Musrrat Ali, Department of Paper Technology of Indian Institute of Technology, Roorkee, India. (Ph.No. +91-132-2714356; fax: +91-132-2714011; e-mail: musrrat.iitr@gmail.com). Millie Pant, Department of Paper Technology of Indian Institute of Technology, Roorkee, India.(e-mail: millidma@gmail.com). Ajith Abraham, Machine Intelligence Research Labs (MIR Labs), Scientific Network of Innovation and research excellence, Auburn, Washington, 98071-2259, USA (e-mail: ajith.abraham@ieee.org).

ACM Reference Format:

1. INTRODUCTION

Differential Evolution (DE), proposed by [Storn and Price, 1995] is relatively a new optimization technique in comparison to Evolutionary Algorithms (EAs) such as Genetic Algorithms [Goldberg, 1989], Evolutionary Strategy [Back et al., 1991], and Evolutionary Programming [Fogel 1994 4]. Within a short span of around fifteen years, DE has emerged as one of the simple and efficient technique for solving global optimization problems. It has been successfully applied to diverse domains of science and engineering, such as mechanical engineering design [Rogalsky et al, 1999], [Joshi and Sanderson 1999], signal processing [Das and Konar 2006], chemical engineering [Wang and Jang, 2000], [Lampinen 1999], machine intelligence, and pattern recognition [Omran et al., 2005], [Das et al., 2008] etc.

Practical experience, however, shows that DE is not completely flawless. As pointed out by [Lampinen and Zelinka, 2000] DE may occasionally stop proceeding towards the global optimum even though the population has not converged to a local optimum or any other point. Occasionally, even new individuals may enter the population, but the algorithm does not progress by finding any better solutions. This situation is usually referred to as *stagnation*. DE also suffers from the problem of premature convergence, where the population converges to some local optima of a multimodal objective function, losing its diversity. The probability of stagnation depends on how many different potential trial solutions are available and also on their capability to enter into the population of the subsequent generations [Lampinen and Zelinka, 2000]. Further, like other evolutionary computing algorithms, the performance of DE deteriorates with the growth of the dimensionality of the search space as well.

Several instances are available in literature which aims at improving the performance of DE. A brief review of some of the modifications suggested on the basic structure of DE is given in Section III.

The modifications show that even a slight variation in the basic structure of DE helps a lot in improving its performance. Our objective in this study is to observe the combined effect of some of these variations. We have concentrated on three aspects of DE namely: initial population, mutation and DE structure which is based on two populations; current population and advance population.

(i) Generation of initial population is a crucial task in a population based search technique. In case no a priori information about the solution is available, random initialization is the most popular method of generating the initial population. Maaranen et al. [Maaranen et al., 2004] introduced quasi random sequences for population initialization in Genetic Algorithms. Their results showed that though there is an improvement in the quality of solution, there is no noticeable change in the convergence rate while using quasi random numbers to generate the initial population. Moreover, from the programming point of view, the generation of quasi random numbers is quite difficult. An interesting method to generate the initial population was suggested by [Rahnamayan et al., 2008], where they used o*pposition based learning* (OBL) [Tizhoosh, 2005] to generate the initial population. This method not only improves the quality of solution but also it helps in faster convergence. Further, it is very easy to program. The basic idea behind OBL is the simultaneous consideration of an estimate and its corresponding opposite estimate to achieve a better approximation for the current candidate solution. Mathematically, it has been proven in [Rahnamayan et al., 2008] that the opposite numbers are more likely to be closer to the optimal solution than purely random ones. We

have considered ODE [Rahnamayan et al., 2008], in which this scheme is used, as one of the parent algorithm.

(ii) The second aspect which is the focus of this study is mutation. It is the most important operator of DE. The mutation operation utilizes the vector differences between the current population members for determining both the degree and direction of perturbation applied to the individual subject of this operation. Besides the basic mutation schemes of DE some other variants of mutation are also available in literature (please see Section III). In the present study we have considered a tournament best process [Kaelo and Ali, 2006] to select the base vector for the process of mutation. The corresponding DERL algorithm [Kaelo and Ali, 2006] is taken as the second parent algorithm. The rationale of using tournament best process is to prevent the search from becoming a purely random search or a purely greedy search.

(iii) The third aspect is the general structure of DE which maintains two populations; current population and an advance population. In the present study we have considered a single population structure of DE [Babu and Angira, 2006, Thompson, 2004]. For the purpose of comparison we have used the algorithm given in [18] as a parent algorithm for MDE. In [Babu and Angira, 2006], the algorithm is named as MDE. In order to avoid confusion we shall refer to it as MDE1 in the present study.

We can say that MDE is motivated by the law of synergy which states that a combined effort is always better than the individualistic effort.

As already mentioned there are several modified versions of DE available in literature. The idea of the present study is to build a DE framework which is simple to understand and easy to apply, therefore we selected three simple but efficient modifications which have reportedly given good performance over the other contemporary optimization algorithms.

Here we would like to mention that a preliminary version of this work has already been presented in a conference [Ali et al., 2009]. However, in the present study we present its elaborated version. Here we provide a comprehensive set of experimental verifications of the proposed MDE. Specifically, we have investigated the convergence speed and robustness, effect of dimensionality and population size, effect of jumping on the proposed MDE and its comparison with other algorithms. The numerical experiments are conducted on a comprehensive set of 25 standard benchmark problems, 7 nontraditional shifted functions and three real life problems.

In order to investigate the effect of fusion, the proposed MDE is compared with DE and with its parent algorithms ODE, DERL and MDE1. We have discussed the improvements made by the parent algorithms over DE, individually and the improvement made when they are fused together in MDE.

Further, MDE compared with some of the other latest modifications of DE namely jDE, JADE and SaDE. The comparison of algorithms is done using the standard performance measures like error, number of function evaluations (NFE) etc. The performance of the algorithms is also analyzed statistically using various tests like Wilcoxon test, Bonferrani Dunn test etc.

The remainder of the paper is structured as follows. Section II describes the basic Differential Evolution. In Section III we give a brief review of the work done in the past to improve the performance of basic DE. In Section IV we explain the proposed MDE algorithm. Performance metrics and experimental settings are given in Section V. Problems used in the present study are listed in Section VI. Section VII provides results

and discussions for MDE, DE, ODE, DERL and MDE1. A brief description of other state of the art algorithms (jDE, JADE and SaDE) used in the present study and their comparison with MDE is given in Section VIII. Finally the conclusions based on the present study are drawn in Section IX.

2. A BRIEF INTRODUCTION TO DIFFERENTIAL EVOLUTION

Like all other population based search algorithms, DE starts with a population S 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 which may be defined as follows:

Mutation: Once the initialization is complete, DE enters the mutation phase. In this phase a *donor* vector is created corresponding to each member or *target* vector $X_{i,G}$ in the current generation. The method of creating donor vector differentiates one DE scheme from another. The most often used mutation strategies implemented in the DE codes are listed below.

DE/rand/1: $V_{i,G} = X_{r_1,G} + F*(X_{r_2,G} - X_{r_3,G})$

DE/rand/2: $V_{i,G} = X_{r_1,G} + F*(X_{r_2,G} - X_{r_3,G}) + F*(X_{r_4,G} - X_{r_5,G})$

DE/best/1: $V_{i,G} = X_{best,G} + F*(X_{r_1,G} - X_{r_2,G})$

DE/best/2: $V_{i,G} = X_{best,G} + F*(X_{r_1,G} - X_{r_2,G}) + F*(X_{r_3,G} - X_{r_4,G})$

DE/rand-to-best/1: $V_{i,G} = X_{r_1,G} + F*(X_{best,G} - X_{r_2,G}) + F*(X_{r_3,G} - X_{r_4,G})$

The indices r_1, r_2, r_3, r_4 and r_5 are mutually exclusive integers randomly chosen from the range *[1, NP]* and all are different from the base index *i.* these indices are randomly generated once for each vector. The scaling factor F is a positive control parameter and is used for scaling the difference vectors. $X_{best,G}$ is the individual having the best fitness function value in the population at generation G.

Frequently referred strategies implemented in the public-domain DE codes for producing the donor vectors are also available online at:
http://www.icsi.berkeley.edu/~storn/code.html.

Crossover: once the donor vector is generated in the mutation phase, the crossover phase of DE is activated. The crossover operation of DE helps in increasing the potential diversity of the DE population. The DE family of algorithms may use two types of crossover schemes; *exponential (exp)* and *binomial (bin)*. During the crossover operation, the donor vector exchanges its components with the target vector $X_{i,G}$ to form a *trial* vector $U_{i,G+1} = (u_{1,i,G+1}, \ldots, u_{n,i,G+1})$. In the present study we shall follow the binomial scheme. According to this scheme, the trial vectors are generated as follows:

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

Where, $j = 1 \ldots n$, $k \in \{1, \ldots, n\}$ is a random parameter's index, chosen once for each i. Cr is a positive control parameter set by the user.

A general DE scheme may be defined as DE/X/Y/Z, where DE denotes the Differential Evolution algorithm; X represents a string denoting the vector to be perturbed; Y indicates the number of difference vectors considered for perturbation of X and Z stands for the type of crossover being used.

Throughout the present study we shall follow *DE/rand/1/bin* version of DE which is perhaps the most frequently used version and shall refer to it as basic version.

Selection: The final phase of the DE algorithm is that of selection, which determines whether the target or the trial vector generated in the mutation and crossover phases will survive to the next generation. The population for the next generation is selected from the individual in current population and its corresponding trial vector according to the following rule:

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

Thus, each individual of the advance (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 its counterpart in the current generation.

The computational steps of basic DE are as follows:

Step 1: Randomly generate a population set S of *NP* vectors, each of dimension n as follows: $x_{i,j} = x_{min,j} + rand(0, 1)(x_{max,j}-x_{min,j})$, where $x_{min,j}$ and $x_{max,j}$ are lower and upper bounds for j^{th} component respectively, *rand(0,1)* is a uniform random number between 0 and 1.

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

Step 3: Select three points from population and generate perturbed individual V_i using equation (1).

Step 4: Recombine each of the target vector X_i with perturbed individual generated in step 3 to generate a trial vector U_i using equation (2).

Step 5: Check whether each variable of the trial vector is within the specified range. If yes, then go to step 6 otherwise bring 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 $u_{i,j} > x_{max,j}$, and go to step 6.

Step6: Calculate the objective function value for trial vector U_i.

Step 7: Choose better of the two (function value at target and trial point) using equation (3) for next generation.

Step 8: Check whether convergence criterion is met if yes then stop; otherwise go to step 3.

3. A BRIEF REVIEW OF PREVIOUS WORK

Several attempts have been made to improve the ultimate performance of DE. These variations may broadly be classified as (1) investigating optimum choice of DE control parameters (2)its hybridization with other search techniques (3) development/ modification in the mutation/ crossover/ selection operators of DE and (4)other variations. In this section we give a brief review of some of the modifications suggested in the structure of DE.

DE has three main control parameters namely population size, crossover rate *Cr* and scaling factor *F*. A number of investigations have been carried out to determine the optimum settings of these parameters. Storn and Price [1] indicated that a reasonable population size could be between 5*n* and 10*n*, where n denotes the dimensionality of the problem. They also recommended that a good initial choice of *F* can be 0.5. Gamperle et

al. in [Gamperle et al., 2002] suggested that the population size should be between 3*n* and 8*n*, scaling factor *F* should be 0.6 and the crossover rate *Cr* should be in the range of [0.3, 0.9] for best results. [Ronkkonen et al., 2005] suggested using *F* values between [0.4, 0.95] with *F*=0.9 being a good initial choice. They further pointed out that the *Cr* values should lie in [0, 0.2] when the function is separable while it should lie in [0.9, 1] when the function's parameters are dependent. However, a drawback in their analysis is that in case of real life problems, it is very difficult to examine in advance the true nature of the function. Thus we can see that there is no concrete proof/ discussion available in literature for the selection of parameters. The researchers rely either on fine-tuning of parameters for a particular problem or consider self-adaptation techniques to avoid manual tuning of the parameters of DE. Liu and Lampinen introduced Fuzzy Adaptive Differential Evolution (FADE) [Liu and Lampinen, 2005] using fuzzy logic controllers, Qin et al. proposed a Self-adaptive DE (SaDE) [Qin et al., 2009] algorithm. In this algorithm, the trial vector generation strategies and their associated parameters are gradually self-adapted by learning from their previous experiences of generating promising solutions. [Zaharie 2003] proposed a parameter adaptation strategy for DE (ADE) based on the idea of controlling the population diversity, and implemented a multi-population approach. Later, [Zaharie and Petcu, 2004] designed an adaptive Pareto DE algorithm for multi-objective optimization and analyzed its parallel implementation. [Abbass, 2002] self-adapted the crossover rate *Cr* for multi-objective optimization problems, by encoding the value of *Cr* into each individual and simultaneously evolving it with other search variables. The scaling factor *F* was generated for each variable from a Gaussian distribution *N* (0, 1). [Omran et al., 2005] proposed an algorithm called SDE in which they introduced a self-adaptive scaling factor parameter *F* and generated the value of *Cr* for each individual from a normal distribution *N* (0.5, 0.15). Recently, [Brest et al., 2007] proposed jDE algorithm using adaptive *F* and *Cr*. Although, most of the self adaptive versions of DE, involve adaption of *Cr* and *F*, work has also been done on the adaption of the population size. [Teng et al., 2009] proposed DE with Self Adapting Populations for DE in DESAP.

Other class of modification in DE involves its hybridization with some other techniques. [Yang et al., 2008] proposed hybridization of DE with Neighborhood Search (NS) and called their algorithm, NSDE. In this algorithm mutation is performed by adding a normally distributed random value to each target-vector component. Later, [Yang et al., 2008] used Self-adaptive NSDE in the cooperative coevolution framework for optimizing large scale non-separable problems (up to 1000 dimensions). [Hendtlass, 2001] hybridized DE with Particle Swarm Optimization (PSO). He used the DE perturbation approach to adapt particle positions. Particles' positions are updated only if their offspring have better fitness.

At the specified intervals, the swarm serves as the population for DE algorithm, and the DE is executed for a number of generations. After execution of DE, the evolved population is further optimized using PSO. [Zhang and Xie, 2003] and [Talbi and Batchoue, 2004] used the DE operator to provide mutations in PSO. [Kannan et al., 2004] applied DE to each particle of the swarm for a number of iterations, and replaced the particle with the best individual obtained from the DE process. [Omran et al., 2008] proposed a hybrid version of Bare Bones PSO and DE called BBDE. In their approach, they combined the concept of barebones PSO with self adaptive DE strategies. [Zhang et al., 2009] proposed a DE-PSO algorithm in which a random moving strategy is proposed

to enhance the algorithm's exploration abilities and modified DE operators are used to enhance each particle's local tuning ability. [Wu and Gu, 2009] proposed Particle Swarm Optimization with prior crossover differential evolution (PSOPDE). [Caponio et al., 2009] proposed a hybridization of DE with three metaheuristics viz. PSO and two local search methods. A Free Search DE (FSDE) was proposed by [Omran and Engelbrecht, 2009]. In their algorithm they hybridized DE with a newly developed 'Free Search Algorithm' and Opposition Based Learning. Hybridization of Nelder Mead algorithm with DE for solving constrained optimization problems was suggested in [Andriana and Coello Coello 2009].

Besides optimum choice for parameters and hybridization of DE, some other modifications in DE include development of new mutation schemes. [Fan and Lampinen, 2003] proposed a Trigonometric Mutation Operator (TMO). In TMO, the donor to be perturbed is taken to be the centre point of the hypergeometric triangle. Parent Centric and Laplace Mutation were suggested by [Pant et al. 2009] and [Pant et al. 2009] respectively. The parent centric mutation operator is inspired by the PCX operator given by Deb et al. in [Deb, 2005], while in Laplace mutation the scaling factor *F* was replaced by a random number following Laplace distribution and an absolute weighted difference between the vectors was used. [Pant et al. 2009] suggested a mixed strategy DE (MSDE) in which two mutation strategies were used in a competitive game environment. More recently a new mutation operator based on wavelet theory was suggested by [Lai et al., 2009]. A crossover based local search method for DE was proposed in [Noman and Iba 2008, 2009].

Some other interesting modifications in DE include the use of opposition based learning (OBL) for generating the initial population by [Rahnamayan et al. 2008]. [Rahnamayan and Wang, 2008] also applied it for solving large scale optimization problems. [Yang et al. 2009] developed an adaptive coevolutionary DE. They applied their algorithm, called JACC-G, for solving large scale global optimization problems. [Brest et al., 2009] performed dynamic optimization using DE, [Ting and Huang, 2009] varied the number of difference vectors in DE, [Epitropakis et al. 2009] suggested evolutionary adaption of the control parameters of differential evolution. [Tasgetiren et al. 2009] included of variable parameter search in DE. Some variants and applications of DE can also be found in [Montgomery, 2009], [Wang et al., 2009], [Peng et al., 2009], [Chakraborty, 2008].

4. STRATEGIES USED IN THE PROPOSED MDE ALGORITHM

In this section we describe the strategies / concepts used in the proposed MDE algorithm which are: opposition based initial population, random localization and one population DE framework. To make the proposed algorithm self explanatory, first we will describe the three schemes briefly.

A. Opposition based initial population

It is based on the concept of opposite numbers. We can say that if $x \in [l, u]$ is a real number, then its opposite number x' is defined as

$$x' = l + u - x \tag{4}$$

This definition can be extended for higher dimensions also as suggested in [14]. If $X = (x_1, x_2, \dots, x_n)$ is a point in n-dimensional space, where $x_1, x_2, \dots, x_n \in R$ and $x_i \in$

$[l_i, u_i]\ \forall\ i \in \{1,2,\dots,n\}$, then the opposite point $X' = (x_1', x_2', \dots, x_n')$ is completely defined by its components

$$x'_i = l_i + u_i - x_i \quad (5)$$

Now, by employing the opposite point definition, the opposition- based initial population can be generated in three simple steps:

- Generate a point $X = (x_1, x_2, \dots, x_n)$ and its opposite $X' = (x_1', x_2', \dots, x_n')$ in an n-dimensional search space (i.e., a candidate solution).
- Evaluate the fitness of both points $f(X)$ and $f(X')$
- If $f(X') \leq f(X)$ (for minimization problem), then replace X with X'; otherwise, continue with X.

Thus, we see that the point and its opposite point are evaluated simultaneously in order to continue with the fitter one.

B. Randomized Localization

According to this rule, three distinct points X_{r1}, X_{r2} and X_{r3} are selected randomly from the population corresponding to target point X_i. A tournament is then held among the three points and the region around the best point is explored. That is to say if X_{r1} is the point having the best fitness function value then the region around it is searched with the hope of getting a better solution. For the sake of convenience we will denote the tournament best point as (say) X_{tb}. Assuming that $X_{tb} = X_{r1}$, the mutation equation (1) becomes:

$$V_{i,G+1} = \boldsymbol{X_{tb,G}} + F \times (X_{r2,G} - X_{r3,G})$$

This variation gradually transforms itself into search intensification feature for rapid convergence when the points in S form a cluster around the global minima.

In order to see the effect of tournament best method for mutation, we shall first discus in brief two common strategies of DE; DE/best/1/bin and DE/rand/1/bin. In DE/best/1/bin the base vector is always selected as the one having the best fitness function value. We can see that here the probability of selecting the best vector as the base vector is always 1. This strategy may provide a fast convergence in the initial stages. However, as the search procedure progresses it may lead to the loss of diversity in the population due to its greedy nature resulting in premature convergence. On the other hand, the strategy DE/rand/1/bin is completely random in nature. Here all the points for mutation are randomly selected and the best point of the point of the population may or may not be included in them. This strategy, due to its random nature helps in preserving the diversity but may lead to a slower convergence. Now, if we look at the tournament best method we see that although the three points for mutation are randomly selected, the base vector is always chosen as the one having the best fitness. This makes it neither purely greedy nor purely random in nature, but provides a localized effect which helps in exploring the different regions of the search space around the potential candidates.

Making use of hypergeometric distribution, we can say that the probability of getting the best vector among the three chosen points for mutation is $\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}$, where M is the number of best points in the population. Initially, it is very much likely that there is one best point but as the evaluation process proceeds the number of best points keeps on increasing.

In case of strategy, *DE/rand/1/bin* the probability that the best point of the population is among the three chosen points for mutation is $\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}$ and the probability that the best point is also selected as the base vector is $\frac{1}{3} * \left[\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}\right]$ When

we apply the tournament best strategy the probability selecting the best point from the three chosen will be 1. Thus the probability that the selected base vector is the best solution of the population becomes $1 * \left[\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}\right]$.

Thus we see that the probability of selecting the best point of the population as base vector, for tournament best strategy lies between the probabilities of DE/rand/1/bin and DE/best/1/bin.

$\frac{1}{3} * \left[\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}\right] < \left[\binom{M}{1} \times \binom{NP-M}{3-1} \div \binom{NP}{3}\right] < 1$. This helps in maintaining the exploration and exploitation capabilities of the proposed MDE ensuring fast convergence and balanced diversity.

C. Concept of single population

It was suggested in [Babu and Angira, 2006, Thompson, 2004]. They have discussed in their work that in the basic structure of DE, two populations (current and advance) are considered simultaneously in all the iterations which results in the consumption of extra memory and CPU time leading to higher number of function evaluations. On the other hand in a single population DE, only one population is maintained and the individuals are updated as and when a better solution is found. Also, the newly found better solutions can take part in mutation and crossover operation in the current generation itself as opposed to basic DE (where another population is maintained and the better solutions take part in mutation and crossover operations in next generation). Updating the single population continuously enhances the convergence speed leading to lesser number of function evaluations as compared to basic DE.

Based on the above modifications we will now discuss the computational steps of MDE which are same as that of basic DE given in Section II and differ from it only in the following steps:

1. initialization
2. mutation
3. populations structure

A point to point comparison of working two algorithms DE and MDE is given in Table I.

5. PERFORMANCE METRICES AND EXPERIMENTAL SETUP

In order to authenticate the viability of the proposed MDE algorithm we conducted a series of experiments following various criteria to test its efficiency, robustness and reliability. These criteria have been widely used to analyze the performance of an algorithm.

Performance Metrics I

- Number of function evaluations (NFE)
- Average error = known global optimum – value to reach VTR (desired accuracy)
- Percentage Acceleration rate (AR) = ratio of the NFE of the algorithm to be compared and the NFE of the algorithm to which we want to compare [Rahnamayan et al., 2008]. Thus the %AR of MDE in comparison to DE will be:

$$\%\text{AR} = \left(1 - \frac{(NFE)MDE}{(NFE)DE}\right) * 100$$

- Average AR = $\frac{1}{N}\sum_{i=1}^{N} AR_i$
- Success rate (SR) = $\frac{No.\ of\ times\ VTR\ obtained}{Total\ no.of\ trials}$
- Average SR = $\frac{1}{N}\sum_{i=1}^{N} SR_i$

Where *N* denotes the number of problems.

Performance Metrics II

The proposed algorithm is also analyzed statistically using various tests like Wilcoxon test, Friedmann test and Bonferrani Dunn test etc [Garcia et al., 2009]. Using these tests, we performed *multiple-problem analysis*, a comparison of algorithms over more than one problem simultaneously. In the multiple-problem analysis, due to the dissimilarities in the results obtained and the small size of the sample to be analyzed, a parametric test (paired t-test) may reach erroneous conclusions so we have analyzed the results by both, parametric and non parametric tests.

All the tests used here obtain the associated *p*-value, which represents the dissimilarity of the sample of results. Hence, a low *p*-value points out a critical difference. In this study, we have considered a level of significance α = 0.05 and 0.1. A *p*-value greater than α indicates that there is no significant difference between the algorithms.

Experimental Settings – after conducting several experiments and referring to various literatures, we took the following settings for all the experiments unless otherwise mentioned.

- Population Size (*NP*) = 100 for traditional benchmark problems and 500 for nontraditional problems [Zhang and Sanderson, 2009], [Rahnamayan and Wang, 2008].
- Scaling/ amplitude Factor *F* = 0.5 [Rahnamayan et al., 2008].
- Crossover Rate *Cr* = 0.9 [Rahnamayan et al., 2008].
- Maximum NFE = 10000**n*, where *n* is the dimension of the problem [Noman and Iba, 2008].
- VTR = 10^{-8} for all the test problems except noisy function (f_7) for which it is set as 10^{-2} [Zhang and Sanderson, 2009].

Software used for statistical analysis – we used the following softwares for analyzing the proposed algorithm.

- SPSS
- MATLAB

PC configuration – All the algorithms have been executed on dual core processor with 1GB RAM. The programming language used is DEV C++. The random numbers are generated using inbuilt *rand* () function with same seed for every algorithm.

In order to have a fair comparison for all the experiments, the parameter settings are kept same and the reported values are the average of the results of 50 independent runs.

6. PROBLEMS USED IN THE PRESENT STUDY

In the present study we used three types of problems given below in order to investigate the effectiveness of the proposed MDE algorithm.

A. Traditional Benchmark problems

First of all we scrutinized the performance of the proposed MDE on a test suite of twenty five standard benchmark problems taken from [Rahnamayan, 2008], [Zhang and Sanderson, 2009], with varying degrees of complexities and having box constraints. The test set includes fixed, lower dimension problems as well as scalable problems for which the dimension can be increased to increase the complexity of the problem. The problem set though small act as a good launch pad to investigate the effectiveness of an optimization algorithm. Mathematical models of the functions along with the true optimum value are given in Table II (A).

B. Real life problems

The effectiveness of an algorithm can be justified, if it is able to solve the real life problems with equal ease with which it solved the test problems. Therefore, besides considering the benchmark functions we have also taken three real life application problems which are; transistor modeling problem, frequency modulation sound parameter identification problem and spread spectrum radar poly- phase code design problem from [Price, 1983] and [Das et al, 2009]. Mathematical model of real life problems are given below.

TABLE I
POINT WISE COMPARISON OF WORKING OF MDE AND DE ALGORITHMS

Operation	DE	MDE
Initialization	Construct an initial population S of *NP* individuals, dimension of each vector being n, using the following rule: $x_{i,j}= x_{min,j} + rand(0, 1)(x_{max,j}-x_{min,j})$, Where $x_{min,j}$ and $x_{max,j}$ are lower and upper bound for j^{th} component respectively and rand(0,1) is a uniform random number between 0 and 1.	Randomly construct a population P of *NP* individuals, dimension of each vector being n, using the following rule: $x_{i,j}= x_{min,j} + rand(0, 1)(x_{max,j}-x_{min,j})$, Where $x_{min,j}$ and $x_{max,j}$ are lower and upper bound for j^{th} component respectively and rand(0,1) is a uniform random number between 0 and 1. Construct another population *OP* of *NP* individuals using the following rule: $y_{i,j} = x_{min,j} + x_{max,j} - p_{i,j}$ Where $p_{i,j}$ are the points of population P. Construct initial population S taking *NP* best individuals from union of P and OP.
Mutation	Select randomly three distinct individuals X_{r1}, X_{r2} and X_{r3} from population S and perform mutation using formula: $V_i = \boldsymbol{X_{r1}} + F\times(X_{r2} - X_{r3})$ Where individual X_{r1} is randomly chosen (i.e. it may be any one from the three individuals).	Select randomly three distinct individuals X_{r1}, X_{r2} and X_{r3} from population S and perform mutation using formula: $V_i = \boldsymbol{X_{tb}} + F\times(X_{r2} - X_{r3})$ Where individual X_{tb} (=X_{r1}) is the individual having the best fitness value among the three individuals.
Crossover:	Perform crossover according to equation (2).	Perform crossover according to equation (2).
Selection	Perform selection of candidates for the next generation using equation (3).	Perform selection of candidates for the next generation using equation (3).
Structure	Maintains two populations; operations performed in current population and data stored in advance population.	All DE operations are performed on a Single population.

TABLE II (A)
NUMERICAL BENCHMARK FUNCTIONS WITH A VARYING NUMBER OF DIMENSIONS (n). FUNCTIONS SINE AND COSINE TAKE ARGUMENTS IN RADIANS. THE MATRIX A USED IN FUNCTION f_{14}, THE VECTORS a AND b USED IN f_{15} AND THE MATRIX a AND VECTOR c USED IN $f_{21} - f_{23}$ ARE DEFINED IN THE APPENDIX. f_{min} DENOTES THE MINIMUM VALUE OF THE FUNCTION.

Test functions	n	Range	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	[-100,100]	0
$f_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	30	[-100,100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
$f_6(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	30	[-100,100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[-128,128]	0
$f_8(x) = \sum_{i=1}^{n} -x_i sin\sqrt{\lvert x_i \rvert}$	30	[-500,500]	-12569.5
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12, 5.12]	0
$f_{10} = -20exp\left(-0.2 \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - exp\left(\sum_{i=1}^{n} \frac{1}{n} cos 2\pi x_i \right) + 20 + e$	30	[-32, 32]	0
$f_{11} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600,600]	0
$f_{12} = \frac{\pi}{n}\left\{ 10sin^2(\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(xi - a)^m, x_i > a \\ 0, -a \le x_i \le a_i \\ k(-x_i - a)^m, x_i < -a \end{cases}$	30	[-50, 50]	0

$f_{13} = 0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2[1+sin^2(3\pi x_{i+1})] + (x_n-1)[1+sin^2(2\pi x_1)\right\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	30	[-50,50]	0
$f_{14} = \left[\frac{1}{500} + \sum_{1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$	2	[-65.536, 65.536]	0.998004
$f_{15} = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_1x_3+x_4}\right]^2$	4	[-5,5]	0.0003075
$f_{16} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]	-1.0316285
$f_{17} = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1-\frac{1}{8\pi}\right)cosx_1 + 10$	2	[-5,10]x[0,15]	0.397887
$f_{18} = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)] \times[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	[-2,2]	3
$f_{19} = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{4} a_{ij}(x_j-p_{ij})^2\right]$	3	[0,1]	-3.86278
$f_{20} = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2\right]$	6	[0,10]	-3.32237
$f_{21} = -\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T + c_i]^{-1}$	4	[0,10]	-10.1532
$f_{22} = -\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T + c_i]^{-1}$	4	[0,10]	-10.4029
$f_{23} = -\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T + c_i]^{-1}$	4	[0,10]	-10.5364
$f_{24} = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$	30	[-5, 10]	0
$f_{25} = -\cos(x_1)\cos(x_2)exp[-(x_1-\pi)^2-(x_1-\pi)^2]$	2	[-10, 10]	-1

Frequency modulation sound parameter identification. [Das et al., 2009]

Frequency-modulated (FM) sound synthesis plays an important role in several modern music systems. Here we consider a system that can automatically generate sounds similar to the target sounds. It consists of an FM synthesizer, a DE optimizer, and a feature extractor. The DE algorithm initializes a set of parameters and the FM synthesizer generates the corresponding sounds. In the feature extraction step, the dissimilarities of features between the target sound and synthesized sound are used to compute the fitness value. The process continues until synthesized sounds become very similar to the target. The specific instance considered in this paper involves determination of six real parameters $X = \{a_1, w_1, a_2, w_2, a_3, w_3\}$ of the FM sound wave given by $y(t) = a_1 \times \sin\left(w_1 \times t \times \theta + a_2 \times \sin\left(w_2 \times t \times \theta + a_3 \times \sin\left(w_3 \times t \times \theta\right)\right)\right)$ for approximating it to

the sound wave given as $y_0(t) = 1.0 \times \sin\left(5.0 \times t \times \theta + 1.5 \times \sin\left(4.8 \times t \times \theta + 2.0 \times \sin\left(4.9 \times t \times \theta\right)\right)\right)$ where $\theta = \pi/100$. The parameters are defined in the range [-6.4, 6.35]. The fitness function is defined as minimizing the sum of square error between the evolved data and the model data as follows:

$$f(a_1, w_1, a_2, w_2, a_3, w_3) = \sum_{t=0}^{100} (y(t) - y_0(t))^2$$

It is a highly complex multimodal problem with a strong interrelation among the variables. The optimum value of the problem is zero.

The spread spectrum radar poly- phase code design problem. [Das et al., 2009]

A famous problem of optimal design arises in the field of spread spectrum radar poly-phase codes. Such a problem is very well suited for validating a global optimization algorithm like DE. A formal definition of the problem can be given as: $\min f(X) = \max\{f_1(X), ..., f_{2m}(X)\}$

Where

$$X = \{(x_1, ..., x_n) \in R^n \mid 0 \le x_j \le 2\pi, j = 1, ..., n\}$$

and m=2n-1,

With

$$f_{2i-1}(X) = \sum_{j=i}^{n} \cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k\right), i = 1, 2, ..., n$$

$$f_{2i}(X) = .5 + \sum_{j=i+1}^{n} \cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k\right), i = 1, 2, ..., n-1$$

$$f_{m+i}(X) = -f_i(X), i = 1, 2, ..., m$$

Here the objective is to minimize the module of the biggest among the samples of the so-called autocorrelation function which is related to the complex envelope of the compressed radar pulse at the optimal receiver output, while the variables represent symmetrized phase differences.

The objective function of this problem for the dimension n=2 is illustrated in Fig. 1. The problem belongs to the class of continuous min–max global optimization problems. They are characterized by the fact that the objective function is piecewise smooth.

Transistor Modeling [Price, 1983]

The mathematical model of the transistor design is given by,

Minimize $f(x) = \gamma^2 + \sum_{k=1}^{4} (\alpha_k{}^2 + \beta_k{}^2)$

Where $\alpha_k = (1 - x_1 x_2) x_3 \{\exp[x_5(g_{1k} - g_{3k} x_7 \times 10^{-3} - g_{5k} x_8 \times 10^{-3})] - 1\} g_{5k} + g_{4k} x_2$

$\beta_k = (1 - x_1 x_2) x_4 \{\exp[x_6(g_{1k} - g_{2k} - g_{3k} x_7 \times 10^{-3} + g_{4k} x_9 \times 10^{-3})] - 1\} g_{5k} x_1 + g_{4k}$.

$\gamma = x_1 x_3 - x_2 x_4$

Subject to: $x_i \ge 0$ and the numerical constants g_{ik} are given by the matrix.

$$\begin{bmatrix} 0.485 & 0.752 & 0.869 & 0.982 \\ 0.369 & 1.254 & 0.703 & 1.455 \\ 5.2095 & 10.0677 & 22.9274 & 20.2153 \\ 23.3037 & 101.779 & 111.461 & 191.267 \\ 28.5132 & 111.8467 & 134.3884 & 211.4823 \end{bmatrix}$$

This objective function provides a least-sum-of-squares approach to the solution of a set of nine simultaneous nonlinear equations, which arise in the context of transistor modeling.

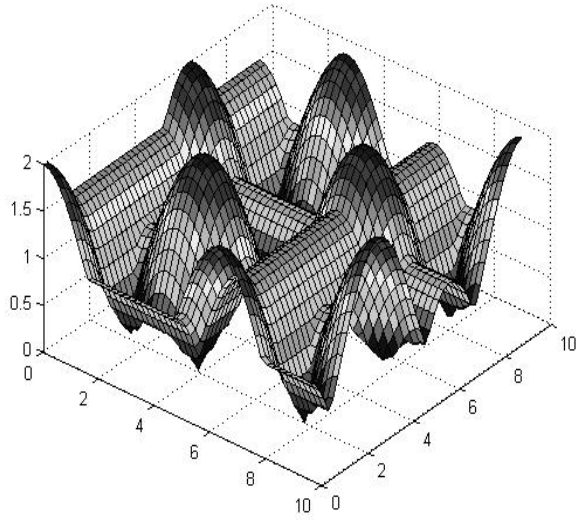


Figure 1: Objective function of spread spectrum radar poly- phase code design problem for n=2.

C. Nontraditional Benchmark Problems of CEC 2008

We validated the efficiency of proposed MDE on a selected set of recently proposed benchmark test suite for *CEC 2008 special session and competition on large scale global optimization* [Tang et al., 2007]. This test suite was specially designed to test the efficiency and robustness of a global optimization algorithm like DE. We considered seven problems from this test suite and tested them for dimension 500. It includes the two unimodal (F_1 and F_2) and five multimodal (F_3- F_7) functions among which four are non-separable (F_2, F_3, F_5, F_7) and three separable (F_1, F_4, F_6). Name of the functions and their properties are listed in Table II (B).

TABLE II (B)
SELECTED BENCHMARK PROBLEMS PROPOSED IN CEC2008 [65]. ALL PROBLEMS ARE EXECUTED FOR DIMENSION 500.

Fun	Name	Properties	Search Space
F1	Shifted Sphere	Unimodal, Separable , scalable	[−100, 100]
F2	Shifted Schwefel's 2.21	Unimodal, Non-separable	[−100, 100]
F3	Shifted Rosenbrock's	Multi-modal, Non-separable. A narrow valley from local optimum to global optimum.	[−100, 100]

F4	Shifted Rastrigin's	Multi-modal, Separable Huge number of local optima	[−5, 5]
F5	Shifted Griewank's	Multi-modal, Non-separable	[−600, 600]
F6	Shifted Ackley's	Multi-modal, Separable	[−32, 32]
F7	FastFractal DoubleDip	Multi-modal, Non-separable	[-1, 1]

7. RESULTS AND DISCUSSIONS

A. Comparison of MDE and its parent algorithms with DE

The proposed MDE algorithm is a fusion of three other algorithms MDE1, ODE and DERL. Therefore first of all we compared the performance of all these algorithms with the basic DE in terms of the various performance criteria mentioned in the previous section. In Table III (A) we have recorded the performance of all the four algorithms in terms of error and standard deviation. From this table we can see that out of 25 test cases MDE outperformed the other three algorithms in 10 cases in terms of both error and standard deviation. In 13 cases all the algorithms gave similar results while in the remaining two cases MDE1 and DERL gave the best performance.

TABLE III (A)
COMPARISON OF PROPOSED MDE WITH DE, MDE1, ODE AND DERL FOR 25 STANDARD BENCHMARK PROBLEMS IN TERMS OF ERROR AND STANDARD DEVIATION (Std.). THE BEST RESULTS OBTAINED ARE HIGHLIGHTED IN BOLDFACE. EACH ALGORITHM IS RUN FOR MAXIMUM NFE = 10000*n, n is DIMENSION OF THE PROBLEM

Fun	n	Error (Std.)				
		DE	MDE	MDE1	ODE	DERL
f_1	30	5.24848e-32 (2.50302e-32)	**1.77622e-76** **6.41905e-78**	3.43423e-48 8.43943e-50	6.34393e-41 2.40300e-43	3.15789e-63 8.92424e-65
f_2	30	7.64876e-16 (5.96538e-16)	**9.76428e-38** **6.74634e-38**	4.23421e-25 3.45932e-26	3.44593e-21 3.49532e-25	2.10125e-31 7.48388e-34
f_3	30	1.77664e-30 (1.04434e-30)	**1.07703e-75** **4.44153e-77**	3.45543e-38 1.38309e-40	7.43932e-34 4.30094e-35	1.56214e-62 3.40335e-60
f_4	30	2.57862e-04 (1.51977e-07)	**1.22089e-09** **6.95936e-09**	9.23275e-04 2.48839e-05	4.49593e-04 6.43490e-04	6.54522e-04 5.30902e-04
f_5	30	1.74229e-01 (1.43011e+00)	**1.14867e-25** **1.52704e-26**	2.34439e-02 1.24943e+00	7.32341e-01 1.82344e+01	4.13605e-14 8.49383e-17
f_6	30	0 0	0 0	0 0	0 0	0 0
f_7	30	7.08548e-03 (6.67423e-03)	**1.91822e-03** **2.51635e-03**	3.43094e-03 3.40023e-03	5.39234e-03 4.30893e-03	3.32073e-03 3.49588e-03
f_8	30	6.81866e+01 7.72931e+01	2.02660e+00 **1.18452e+01**	**1.34393e-02** 1.29922e+01	1.10366e+01 **1.39020e+01**	1.42086e+02 1.44324e+01
f_9	30	1.49594e+02 1.70130e+02	**4.92040e+01** **1.49244e+01**	1.30031e+02 2.30439e+01	1.13584e+02 3.00283e+02	1.24723e+02 9.43885e+01
f_{10}	30	3.69735e-15 0	3.69735e-15 0	3.69735e-15 0	3.69735e-15 0	3.69735e-15 0
f_{11}	30	0 0	0 0	0 0	0 0	0 0
f_{12}	30	1.35360e-19 0	**1.35360e-31** 0	4.49594e-21 0	1.04493e-19 0	3.54594e-23 0
f_{13}	30	1.29115e-19 0	**1.29115e-29** 0	3.45943e-20 0	1.03113e-19 0	5.43222e-22 0
f_{14}	2	0 0	0 0	0 0	0 0	0 0
f_{15}	4	0 4.84870e-20	0 **3.57137e-20**	0 4.59043e-20	0 4.59043e-20	0 4.59043e-20
f_{16}	2	0 2.22045e-16	0 2.22045e-16	0 2.22045e-16	0 2.22045e-16	0 2.22045e-16
f_{17}	2	0 0	0 0	0 0	0 0	0 0
f_{18}	2	0 4.44089e-16	0 4.44089e-16	0 4.44089e-16	0 4.44089e-16	0 4.44089e-16
f_{19}	3	3.00012e-06 4.44089e-16	3.00012e-06 4.44089e-16	3.00011e-06 4.44089e-16	3.00011e-06 4.44089e-16	3.00011e-06 4.44089e-16
f_{20}	6	4.75550e-02 5.82455e-02	3.56650e-02 5.44837e-02	3.75550e-02 5.44837e-02	3.8675e-02 5.44837e-02	**3.2675e-02** 5.44837e-02
f_{21}	4	0 1.58882e-15	0 1.58882e-15	0 1.58882e-15	0 1.58882e-15	0 1.58882e-15
f_{22}	4	0 1.48621e-15	0 1.68520e-15	0 1.68520e-15	0 1.68520e-15	0 1.68520e-15
f_{23}	4	0 1.77636e-15	0 1.77636e-15	0 1.77636e-15	0 1.77636e-15	0 1.77636e-15
f_{24}	30	1.95165e-32 2.06563e-32	**1.47748e-76** **5.93376e-77**	5.32332e-42 8.34993e-43	3.32494e-38 7.39920e-41	1.12489e-64 5.94992e-67
f_{25}	2	0 0	0 0	0 0	0 0	0 0

However, on the basis of the number of function evaluations (NFE) taken by all the algorithms for which the results are given in Table III(B) we see that MDE gave a superior performance in comparison to other algorithms in 24 out of 25 cases. DE, ODE and MDE1 were not able to solve the fifth function f_5, while none of the algorithms were able to reach the desired accuracy of 10^{-8} for the function f_9 and were therefore terminated when the maximum NFE (= 10*n) was reached. On an average the NFE taken by the proposed MDE algorithm for solving 25 problems is only 40318.7. While the average NFE taken by DE, ODE, DERL and MDE1 are 74840.4, 72770.1, 44803.7 and 70861.6 respectively. The performance graphs of few selected functions are illustrated in Figure 2. These graphs are drawn according to the fixed accuracy and not according to fixed NFE.

The faster convergence rate of the proposed MDE is also justified with the help of the acceleration rate (AR). We calculated the %AR of MDE, ODE, DERL and MDE1 against DE and recorded the results in Table III(C). From this Table we can clearly see that for 11 test problems the % AR of MDE in comparison to DE is more than 50%. For 7 test problems the % AR is more than 40% and for 4 test problems it is more than 30%. In case of f_9, AR is not recorded because none of the algorithms were able to meet the desired accuracy criteria. In case of f_5, AR is not recorded because DE was not able to solve it successfully. On an average the AR of MDE vs. DE is 46.12%. When we compare ODE against DE we see that the % AR is less than 10% for all the test problems with ODE performing worse than DE for 4 test cases. On an average the AR for ODE vs. DE is 1.15%. The performance of DERL is closest to the performance of MDE with DERL giving % AR of more than 45% in 9 test cases and more than 50% in 1 case. In 9 cases DERL gave an AR of more than 30% and in 4 cases the % AR is more than 20%. In case of MDE1 and DE, the %AR of MDE1 is less than 10% for 10 test cases, while in 2 cases the performance of MDE1 is worse than DE. For 10 cases the % AR for MDE1 is more than 20%.

The successful performance of all the algorithms is summarized in Table III (D). Here we see that on an average, the success rate of the proposed MDE algorithm is 94%, while for DE, ODE, DERL and MDE1, the average success rate is 88, 88, 90 and 87 % respectively. DE and ODE were not able to reach the desired accuracy for function f_5 and none of the algorithm was able to meet the desired accuracy criteria for function f_9.

TABLE III (B)
COMPARISON OF PROPOSED MDE WITH DE, MDE1, ODE AND DERL FOR 25 STANDARD BENCHMARK PROBLEMS IN TERMS OF NFE. THE BEST RESULTS OBTAINED ARE HIGHLIGHTED IN BOLDFACE. ACCURACY IS SET AS 10^{-08} FOR ALL FUNCTION EXCEPT f_7 WHERE IT IS 10^{-02}. MAXIMUM NFE IS SET AS 10000*n, n REPRESENTS THE DIMENSION OF THE PROBLEM. '- -' REPRESENTS THAT THE ALGORITHM WAS NOT ABLE TO ACHIEVE THE DESIRED ACCURACY. Ave. REPRESENTS AVERAGE

Fun	n	NFE				
		DE	MDE	ODE	DERL	MDE1
f_1	30	104310	**45980**	101040	56700	94700
f_2	30	173850	**77830**	165570	93890	160240
f_3	30	110700	**48600**	102400	59700	101400
f_4	30	274150	258886	263140	**245250**	297600
f_5	30	--	**190600**	--	257100	--
f_6	30	31890	**14850**	30030	17080	28770
f_7	30	131640	**70680**	130680	80660	137370
f_8	30	226850	**101067**	222033	108800	210986
f_9	30	--	--	--	--	--
f_{10}	30	163020	**72800**	162310	87430	149200
f_{11}	30	108930	**48077**	106300	58430	99600
f_{12}	30	95400	**43340**	94460	50910	85600

f_{13}	30	104310	**46680**	104060	55110	91100
f_{14}	2	5220	**3330**	5260	3640	5360
f_{15}	4	11220	**6050**	11800	7780	9750
f_{16}	2	5720	**3330**	5690	4020	4810
f_{17}	2	6930	**4790**	7050	4970	6750
f_{18}	2	4470	**2850**	4460	3200	3930
f_{19}	3	5010	**2870**	4950	3410	4390
f_{20}	6	14400	**7050**	13560	8825	13100
f_{21}	4	11990	**6640**	11920	7570	10350
f_{22}	4	11290	**6220**	11260	7430	9380
f_{23}	4	11330	**6190**	11090	7440	10090
f_{24}	30	104540	**46580**	100300	55050	91500
f_{25}	2	4160	**2640**	4350	3190	3840
Ave		74840.4	**40318.7**	72770.1	44803.1	70861.6

TABLE III (C)
COMPARISON OF PROPOSED MDE, MDE1, ODE AND DERL FOR 25 STANDARD BENCHMARK PROBLEMS AGAINST DE IN TERMS OF AR. THE BEST RESULTS OBTAINED ARE HIGHLIGHTED IN BOLDFACE. ACCURACY IS SET AS 10^{-08} FOR ALL FUNCTION EXCEPT f_7 WHERE IT IS 10^{-02}. MAXIMUM NFE IS SET AS 10000*n, n REPRESENTS THE DIMENSION OF THE PROBLEM. '- -' IN f_5 AND f_9 INDICATES THAT AR CANNOT BE CALCULATED FOR THEM. Ave. REPRESENTS AVERAGE

Fun	n	Acceleration Rate (AR)			
		MDE vs. DE	ODE vs. DE	DERL vs. DE	MDE1 vs. DE
f_1	30	55.92	3.13	45.64	9.21
f_2	30	55.24	4.76	45.99	7.83
f_3	30	56.1	7.50	46.07	8.40
f_4	30	5.57	4.01	10.54	-8.55
f_5	30	--	--	--	--
f_6	30	53.44	5.83	46.44	9.78
f_7	30	46.31	0.73	38.73	-4.35
f_8	30	55.45	2.12	52.04	6.99
f_9	30	--	--	--	--
f_{10}	30	55.35	0.44	46.37	8.48
f_{11}	30	55.87	2.41	46.36	8.57
f_{12}	30	54.58	0.99	46.64	10.27
f_{13}	30	55.25	0.24	47.17	12.66
f_{14}	2	36.21	-0.77	30.27	-2.68
f_{15}	4	46.08	-5.17	30.66	13.10
f_{16}	2	41.79	0.52	29.72	15.91
f_{17}	2	30.89	-1.73	28.28	2.60
f_{18}	2	36.25	0.22	28.41	12.08
f_{19}	3	42.72	1.20	31.94	12.38
f_{20}	6	51.05	5.83	38.72	9.03
f_{21}	4	44.63	0.58	36.86	13.68
f_{22}	4	44.91	0.27	34.19	16.92
f_{23}	4	45.37	2.12	34.33	10.94
f_{24}	30	55.45	4.06	47.34	12.47
f_{25}	2	36.54	-4.57	23.31	7.69
Ave		**46.12**	1.51	37.65	8.41

Analysis of results – Although on the basis of error and standard deviation no concrete conclusion can be drawn on the performance of the proposed MDE algorithm but if we look at other performance criteria we can clearly observe the efficient performance of MDE. The improvement in NFE taken by ODE, MDE1 and DERL in comparison to DE

is 2.8%, 5.3% and 40.1% respectively, whereas for MDE this improvement is 46.1% justifying the effect of synergy. A similar trend can be observed in terms of % AR and SR. On an average the % AR for MDE is 46.12% while for ODE, DERL and MDE1 it is 1.5, 37.65 and 8.5% respectively in comparison to DE. Further MDE emerges as the most successful algorithm with an average SR of 94%.

TABLE III (D)
COMPARISON OF PROPOSED MDE WITH DE, MDE1, ODE AND DERL FOR 25 STANDARD BENCHMARK PROBLEMS IN TERMS OF SR.THE BEST RESULTS OBTAINED ARE HIGHLIGHTED IN BOLDFACE. ACCURACY IS SET AS 10^{-08} FOR ALL FUNCTION EXCEPT f_7 WHERE IT IS 10^{-02}. MAXIMUM NFE IS SET AS 10000*n, n REPRESENTS THE DIMENSION OF THE PROBLEM. AVERAGE SR IS RECORDED IN THE LAST ROW.

Fun	n	Success Rate (SR)				
		DE	MDE	ODE	DERL	MDE1
f_1	30	1	1	1	1	1
f_2	30	1	1	1	1	1
f_3	30	1	1	1	1	1
f_4	30	0.36	**0.75**	0.52	0.44	0.52
f_5	30	0	1	0	1	0
f_6	30	1	1	1	1	1
f_7	30	1	1	1	1	1
f_8	30	**0.9**	0.88	0.88	0.72	0.76
f_9	30	0	0	0	0	0
f_{10}	30	1	1	1	1	1
f_{11}	30	1	1	1	1	1
f_{12}	30	1	1	1	1	1
f_{13}	30	1	1	1	1	1
f_{14}	2	1	1	1	1	1
f_{15}	4	1	1	1	1	1
f_{16}	2	1	1	1	1	1
f_{17}	2	1	1	1	1	1
f_{18}	2	1	1	1	1	1
f_{19}	3	1	1	1	1	1
f_{20}	6	0.84	0.78	0.62	0.44	0.48
f_{21}	4	1	1	1	1	1
f_{22}	4	1	1	1	1	1
f_{23}	4	1	1	1	1	1
f_{24}	30	1	1	1	1	1
f_{25}	2	1	1	1	1	1
Ave		0.88	**0.94**	0.88	0.90	0.87

TABLE III (E)
RESULTS OF FRIEDMAN TEST BASED ON ERROR

N	Friedman value	df	*p*-value
25	30.384	4	<0.001

df – Degrees of Freedom N - Total No of functions

TABLE III (F)
RANKING OBTAINED THROUGH FRIEDMAN'S TEST AND CRITICAL DIFFERENCE (CD) CALCULATED THROUGH BONNFERRONI-DUNN'S PROCEDURE

Algorithm	**Mean Rank**
DE	3.82
MDE	**2.18**
MDE1	3.00
ODE	3.32
DERL	2.68
CD for α = 0 .05	1.11714
CD for α = 0.10	1.002206

TABLE III (G)
RESULTS OF PAIRWISE COMPARISON BASED ON ERROR

MDE	paired t-test		Wilcoxon test				
Vs.	Stat	*p*-value	+ve	-ve	tie	Stat	*p*-value
DE	-1.419	0.196	12	0	13	-2.803	0.005
MDE1	-0.974	0.340	11	2	12	-1.852	0.044
ODE	-1.147	0.263	12	1	12	-2.691	0.007
DERL	-1.279	0.181	11	2	12	-1.852	0.064

TABLE III (H)
RESULTS OF FRIEDMAN TEST BASED ON NFE

N	Friedman value	df	*p*-value
25	85.849	4	<0.001

df – Degrees of Freedom N - Total No of functions

TABLE III (I)
RANKING OBTAINED THROUGH FRIEDMAN'S TEST AND CRITICAL DIFFERENCE (CD) CALCULATED THROUGH BONNFERRONI-DUNN'S PROCEDURE

Algorithm	**Mean Rank**
DE	4.60
MDE	**1.12**
MDE1	4.00
ODE	3.28
DERL	2.00
CD for α = 0 .05	1.11714
CD for α = 0.10	1.002206

TABLE III (J)
RESULTS OF PAIRWISE COMPARISON BASED OF NFE

MDE	paired t-test		Wilcoxon test				
Vs.	Stat	*p*-value	+ve	-ve	tie	Stat	*p*-value
DE	-4.584	0.000	24	0	1	-4.286	0.000
MDE1	-4.581	0.000	24	0	1	-4.286	0.000
ODE	-4.472	0.000	24	0	1	-4.286	0.000
DERL	-2.438	0.023	23	1	1	-3.686	0.002

B. Statistical Analysis

Error values included in Table III (A) allow us to carry out a rigorous statistical study in order to check whether the results of the algorithms are rather significant for considering them different in terms of quality on approximation of continuous functions. Our study will be focused on the algorithm that had the lowest average error rate in the comparison, MDE. We studied the behaviour of this algorithm with respect to the remaining ones, and

determined if the results it offered are better than the ones offered by the rest of algorithms, computing the *p*-value on each comparison. Table III (E) shows the result of applying Friedman's tests in order to see whether there are global differences in the results. Given that the *p*-value of Friedman test 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 is done to detect concrete differences among algorithms. First of all, we employed Bonferroni-Dunn's test to detect significant differences for the control algorithm MDE. Table III (F) summarizes the ranking obtained by Friedman's test and the critical difference (CD) of Bonferroni-Dunn's procedure. In Figure 3(a), Bonferroni-Dunn's graphic illustrates difference among rankings obtained for each algorithm. In this, we draw a horizontal cut line which represents the threshold for the best performing algorithm, the one with the lowest ranking bar, in order to consider it better than other algorithms. A cut line is drawn for each level of significance considered in the study at height equal to the sum of the ranking of the control algorithm and the corresponding Critical Difference computed by the Bonferroni-Dunn method. The bars which exceed this line are associated to an algorithm with worse performance than the control algorithm. The application of Bonferroni-Dunn's test informs us of the following significant differences with MDE as control algorithm:

• *MDE* is better than *DE* and *ODE* with $\alpha = 0.05$ and $\alpha = 0.10$ (2/4 algorithms).

Until now, we used procedures for performing multiple comparisons in order to check the behaviour of the algorithms. We then compared *MDE* with the rest of the algorithms pair wise using Wilcoxon and paired t-test. The corresponding results are given in Table III (G). It displays the statistics, *p*-value and number of +ve ranks (where control algorithm performed better than comparing algorithm), -ve ranks (where control algorithm performed worse than comparing algorithm) and tie (both algorithms performed equivalently).

From this Table we see that in case of DE, for 12 problems MDE performed better than it, while for 13 cases both the algorithms performed similarly. In case of MDE1, for 11 test cases MDE performed better than it while for 2 cases MDE1 performed better than MDE. In the remaining two cases both algorithms performed equivalently. MDE outperformed ODE in 12 cases and ODE outperformed MDE in 1 case. In the remaining 12 cases both algorithms gave a similar performance. MDE performed better than DERL in 11 cases, in 12 cases there was a tie i.e. both algorithms performed equivalently while in 2 cases DERL outperformed MDE. In an interesting observation we see that according to t-test there is no significant difference between MDE and other algorithms but according to Wilcoxon test we see that although there is no difference between MDE and DERL, but there is a significant difference between MDE and other algorithms.

Following the procedure given above we did a similar analysis for NFE given in Table III (B). The statistical results based on it are summarized in Tables III (H)-III (J). From these Tables and from the graphical illustration given in Figure 3(b), we see that in an overall comparison MDE and DERL are at par with each other while the remaining algorithms perform worse than MDE. However, when we perform a pair wise comparison for which the results are given in Table III (J), we see that there is a significant difference between MDE and other algorithms.

The performance of MDE and other algorithms in terms of NFE is also shown with the help of box-plot given in Figure 3(c).

Analysis of results **–** on the basis of error once again we cannot make a concrete judgment about the performance of MDE. But if we observe NFE we can see that MDE performs better than other algorithms.

C. Influence of dimensionality

In order to investigate the effect of dimensionality on the performance the proposed MDE algortihm we considered the scalable problems and varied their dimension as n/2 (=15) and 2n (=60). The corresponassociatedding results in terms of average NFE and Success Rate are reported in Table IV. For problems of dimension 15, MDE outperfromed DE by a significant difference for all the functions except f_9, for which neither of the algorithms were able to reach the desired accuracy and were therefore terminated when the maximum NFE was reached. The average NFE taken by DE for 15 dimensions problem is 61315 which is almost twice the average NFE taken by MDE which is 30588.75. further the average success rate for DE is 0.80 only whereas for MDE the average success rate is 0.90.

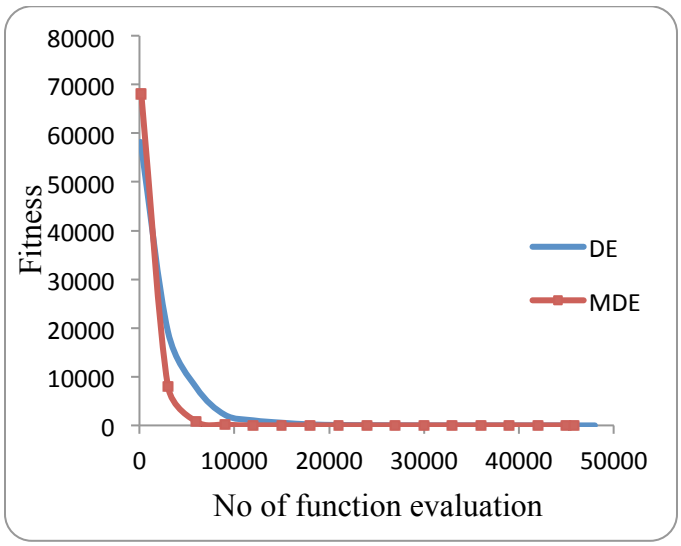
When we increased the dimension to 60, the performance of DE further deteriorated in comparison to MDE. DE was not able to solve problems f_4 and f_5 for dimension 60 under the given parameter settings. Once again neither of the algorithms were able to solve function f_9. The average NFE taken by DE comes out to be 259429.1 while the NFE taken by MDE comes out to be 162934.5, which is half the NFE taken by DE. The average success rate of DE and MDE are 0.74 and 0.83 respectively.

Analyisis of results **–** for smaller dimension (15) as well as for larger dimension (60) we see that in terms of NFE, MDE shows an improvement of around 50% for problems of dimension both 15 and 60. The SR of MDE is 10% better than the SR of DE for problems of dimension 15 and is 9% better than DE for problems of dimension 60.

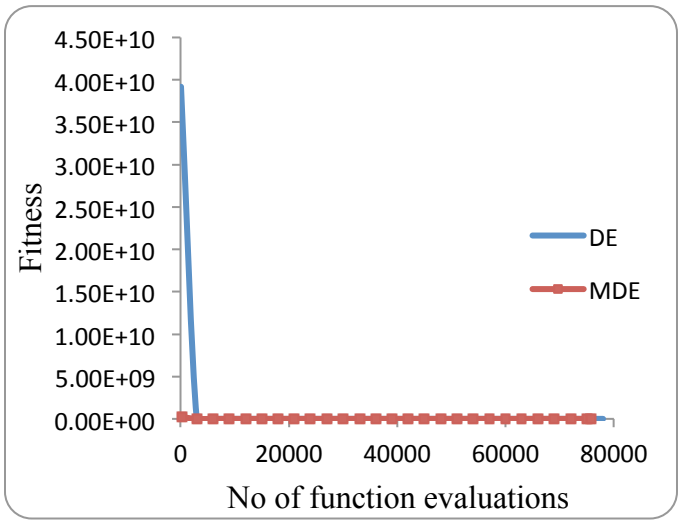
D. Influence of varying the population size (NP)

In order to observe the effect of varying population size on the proposed MDE algortihm, we considered two different population sizes *NP/2* (=50) and *2NP* (=200) and recorded the NFE and SR for DE and MDE algorithms. The corresponding results are given in Table V. For *NP*=100, the average NFE for DE and MDE was 74840.43 and 40318.7 respectively, which reduced to 29008.14 and 16580.91 respectively when the population size was reduced to *NP*=50. However the success rate also reduced from 0.88 to 0.85 for DE and from 0.94 to 0.86 for MDE. Likewise when we increased the population size to 200, the success rate incresed to 0.90 and 0.95 at the cost of higher NFE, which shot up to 174273.3 (for DE) and 101195.3 (for MDE).

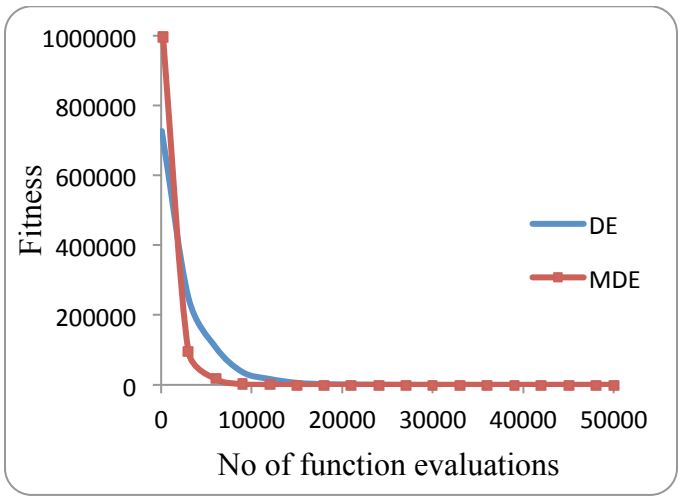
Anaysis of Results – for smaller population size (NP=50) both DE and MDE performed reasonably well in terms of NFE with MDE being almost 50% faster than DE. However the success rate deteriorated by 3% and 8% for DE and MDE repsectively in comparison to the success rate for population size 100. For larger population (NP=200), the SR improve by 2% and 1% respectively for DE and MDE at the cost of more than 50% increase in NFE for both DE and MDE. This is an expected outcome as most of the population based search techniques are sensitive to the population size. This also shows that NP=100 is quite effecient for solving problems up to dimension 30.


80000
70000
60000
50000
40000
30000
20000
10000
0
Fitness
0
10000
20000
30000
40000
50000
No of function evaluation
DE
MDE

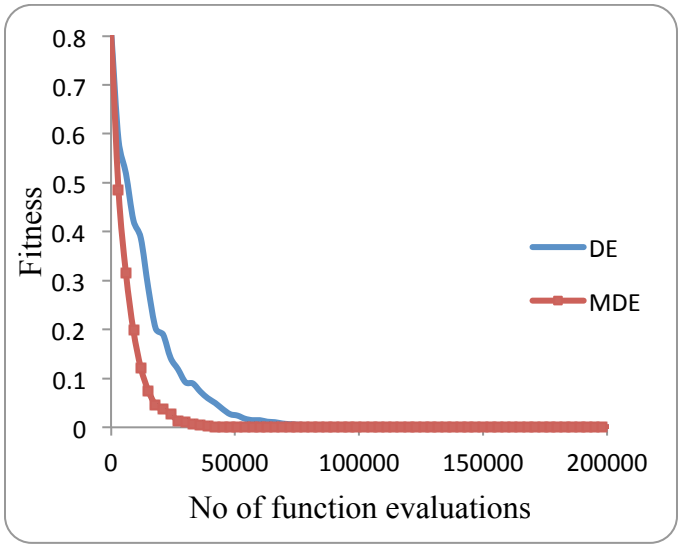
(a)


4.50E+10
4.00E+10
3.50E+10
3.00E+10
2.50E+10
2.00E+10
1.50E+10
1.00E+10
5.00E+09
0.00E+00
Fitness
0
20000
40000
60000
80000
No of function evaluations
DE
MDE

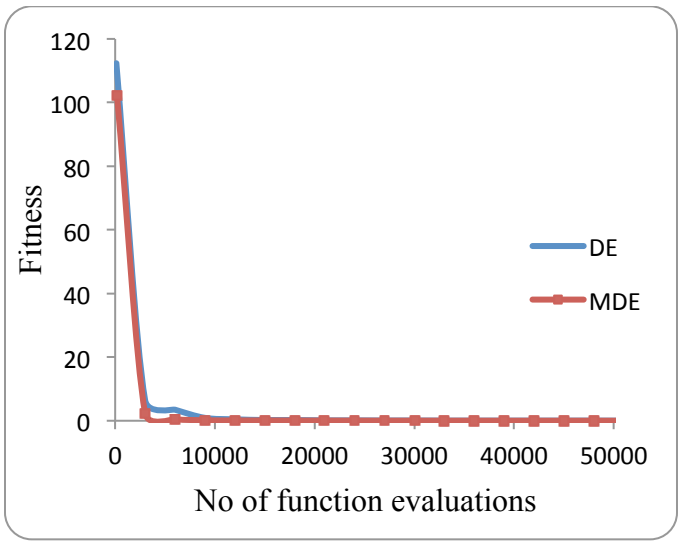
(b)


1000000
800000
600000
400000
200000
0
Fitness
0
10000
20000
30000
40000
50000
No of function evaluations
DE
MDE

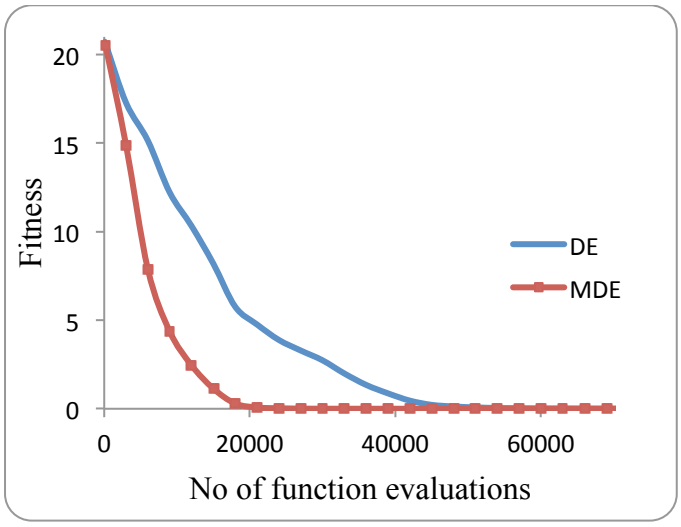
(c)


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Fitness
0
50000
100000
150000
200000
No of function evaluations
DE
MDE

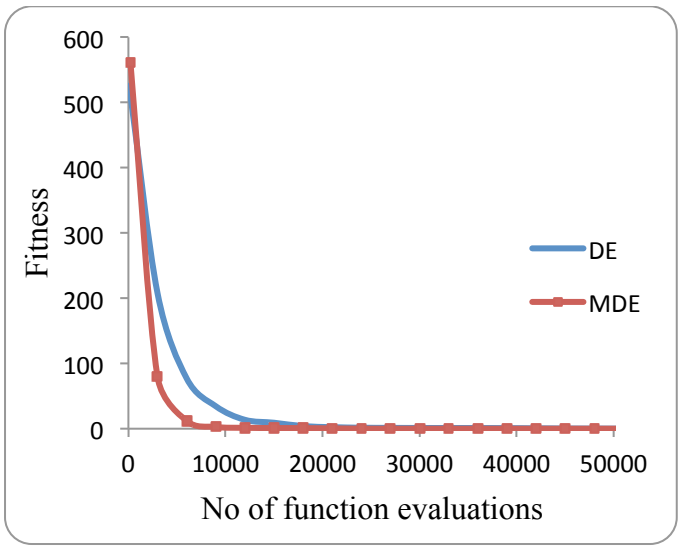
(d)


120
100
80
60
40
20
0
Fitness
0
10000
20000
30000
40000
50000
No of function evaluations
DE
MDE

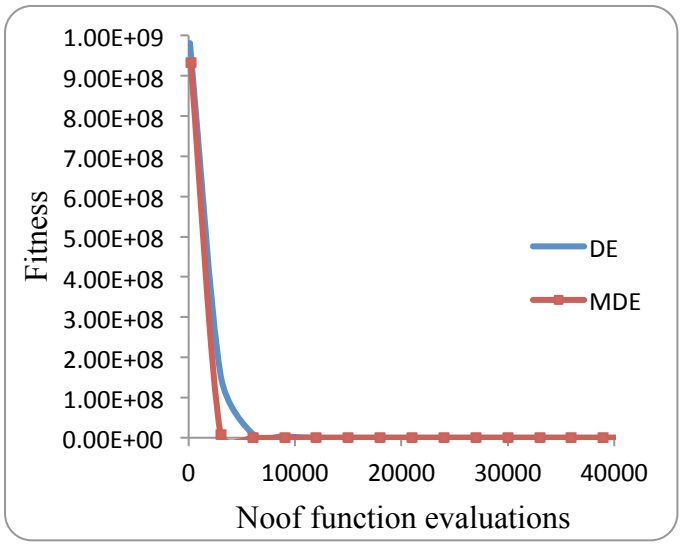
(e)


20
15
10
5
0
Fitness
0
20000
40000
60000
No of function evaluations
DE
MDE

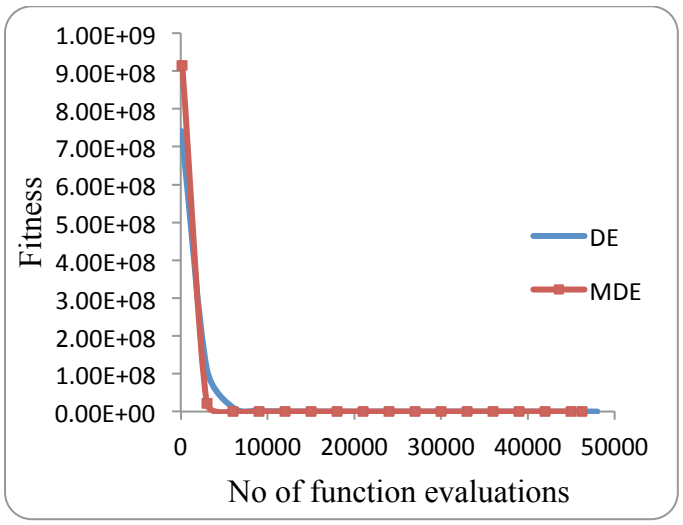
(f)


600
500
400
300
200
100
0
Fitness
0
10000
20000
30000
40000
50000
No of function evaluations
DE
MDE

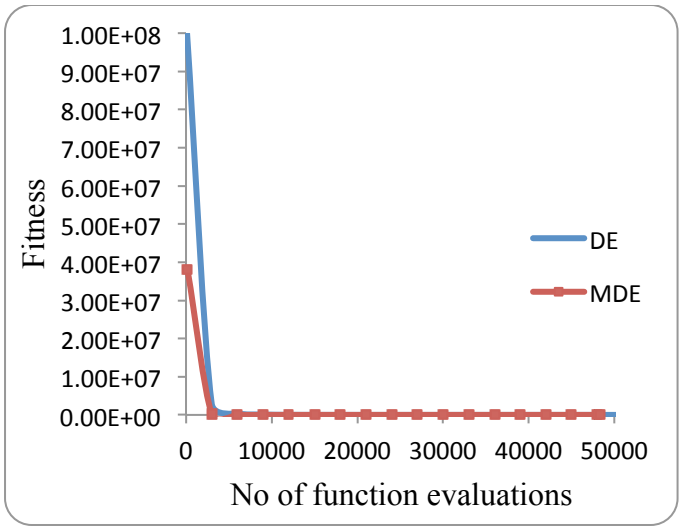
(g)


1.00E+09
9.00E+08
8.00E+08
7.00E+08
6.00E+08
5.00E+08
4.00E+08
3.00E+08
2.00E+08
1.00E+08
0.00E+00
Fitness
0
10000
20000
30000
40000
Noof function evaluations
DE
MDE

(h)


1.00E+09
9.00E+08
8.00E+08
7.00E+08
6.00E+08
5.00E+08
4.00E+08
3.00E+08
2.00E+08
1.00E+08
0.00E+00
Fitness
0
10000
20000
30000
40000
50000
No of function evaluations
DE
MDE

(i)



(j)

Figure 2 Sample graphs (best solution versus NFE) for performance comparison between DE and MDE. (a) – (j) represents functions $f_1, f_2, f_3, f_4, f_7, f_{10}, f_{11}, f_{12}, f_{13}$, *and* f_{24}

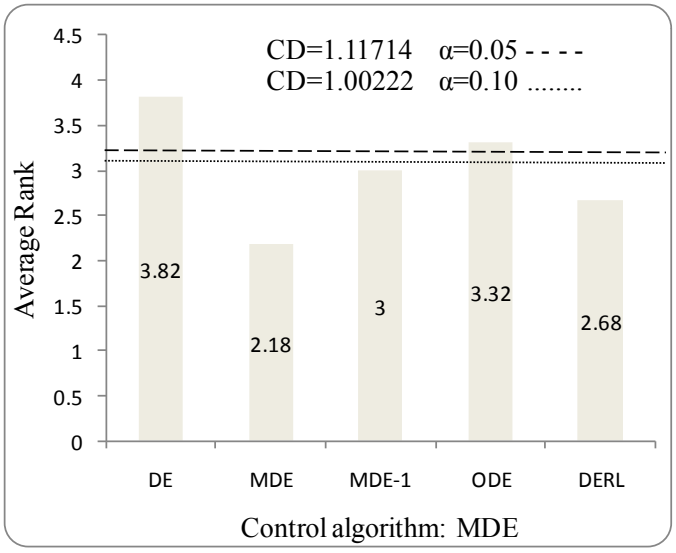


Figure 3(a) Bonferroni-Dunn's graphic corresponding to the error

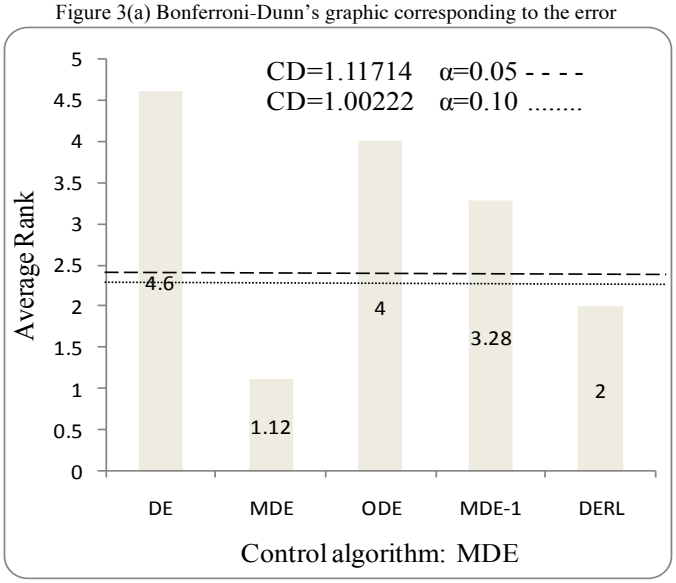


Figure 3(b) Bonferroni-Dunn's graphic corresponding to NFE

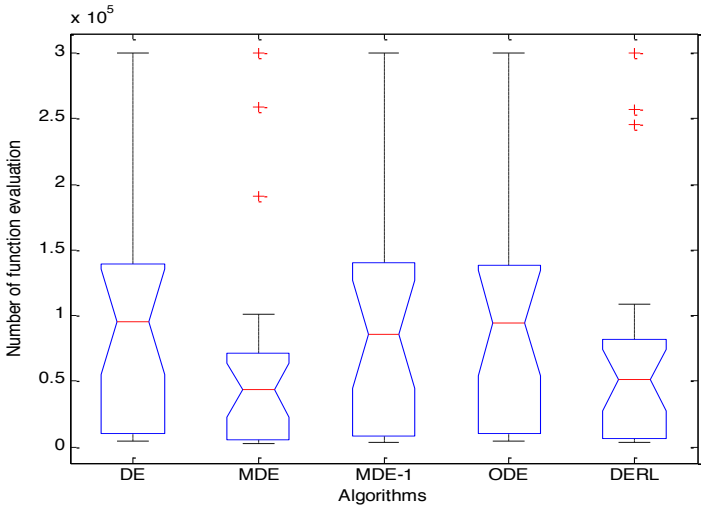


Figure 3(c) Box-plot corresponding to the average NFE

TABLE IV
INFLUENCE OF DIMENSIONALITY – COMPARISON OF MDE WITH DE IN TERMS OF NFE AND SR FOR SCALABLE PROBLEMS. THE DIMENSIONS ARE TAKEN AS n/2(=15) AND 2n (=60). THE BEST RESULTS OBTAINED ARE HIGHLIGHTED IN BOLDFACE. WHEN NO RESULT IS HIGHLIGHTED IT INDICATES THAT ALL THE ALGORITHMS HAVE SAME RESULTS. ACCURACY IS SET AS 10^{-08} FOR ALL FUNCTION EXCEPT NOISY f_7 FUNCTION WHERE IT IS 10^{-02}. MAXIMUM NFE IS SET AS 10000*n. Ave REPRESENTS AVERAGE AND '- -' REPRESENTS THAT THE ALGORITHM WAS NOT ABLE TO ACHIEVE THE DESIRED ACCURACY.

Fun	NFE				Success Rate (SR)			
	n=15		n=60		n=15		n=60	
	DE	MDE	DE	MDE	DE	MDE	DE	MDE
f_1	49050	**23250**	192400	**85800**	1	1	1	1
f_2	80440	**39350**	305120	**134890**	1	1	1	1
f_3	52900	**24100**	215700	**96800**	1	1	1	1
f_4	123210	**60400**	--	**528000**	0.65	1	0	0.52
f_5	--	**62550**	--	**546000**	0	1	0	0.87
f_6	14310	**7040**	57000	**26990**	1	1	1	1
f_7	43740	**29580**	451600	**432000**	1	1	1	1
f_8	57960	**40155**	594000	**516000**	0.92	0.95	0.9	0.71
f_9	--	--	--	--	0	0	0	0
f_{10}	77990	**37750**	288800	**125900**	1	1	1	1
f_{11}	98700	**39200**	184600	**85500**	0.57	0.65	0.47	0.5
f_{12}	44640	**21450**	165000	**90400**	1	1	1	1
f_{13}	47460	**23240**	191400	**102000**	1	1	1	1
f_{24}	45380	**21550**	208100	**96000**	1	1	1	1
Ave	61315	**30588.75**	259429.1	162934.5	0.80	**0.90**	0.74	**0.83**

TABLE V
INFLUENCE OF VARYING POPULATION SIZES – COMPARISON OF MDE WITH DE IN TERMS OF NFE AND SR FOR ALL THE 25 PROBLEMS. THE POPUALTION SIZES ARE TAKEN AS NP/2 (=50) AND 2NP (=200). THE BEST RESULTS OBTAINED ARE HIGHLIGHTED IN BOLDFACE. ACCURACY IS SET AS 10^{-08} FOR ALL FUNCTION f_7 WHERE IT IS 10^{-02}. MAXIMUM NFE IS SET AS 1000000. Ave. REPRESENTS AVERAGE AND '- -' REPRESENTS THAT THE ALGORITHM WAS NOT ABLE TO ACHIEVE THE DESIRED ACCURACY

Fun	n	NFE				Success Rate (SR)			
		NP=50		NP=200		NP =50		NP =200	
		DE	MDE	DE	MDE	DE	MDE	DE	MDE
f_1	30	40310	**19770**	286200	**128800**	1	1	1	1
f_2	30	61460	**29170**	397000	**223940**	1	1	1	1
f_3	30	42350	**21445**	352000	**138000**	1	1	1	1
f_4	30	--	--	897000	**447920**	0	0	0.65	0.89
f_5	30	--	**363000**	--	**385200**	0	1	0	1
f_6	30	12320	**6165**	87000	**39800**	1	1	1	1
f_7	30	70455	**65195**	260000	**245600**	1	1	1	1
f_8	30	96650	**75362**	335855	**296244**	0.65	0.52	0.93	0.96
f_9	30	--	--	--	--	0	0	0	0
f_{10}	30	62650	**32450**	385282	**202600**	1	1	1	1
f_{11}	30	41450	**20350**	297000	**133120**	0.87	0.70	1	1
f_{12}	30	42177	**20438**	208321	**118560**	1	0.9	1	1
f_{13}	30	65218	**27255**	186323	**123600**	1	0.9	1	1
f_{14}	2	2572	**1690**	6521	**5820**	1	1	1	1
f_{15}	4	6721	**3450**	18723	**11440**	1	1	1	1
f_{16}	2	2577	**1540**	8723	**6740**	1	1	1	1
f_{17}	2	3572	**2000**	13272	**9660**	1	1	1	1
f_{18}	2	2466	**1405**	9832	**5320**	1	1	1	1
f_{19}	3	2943	**1485**	8734	**5560**	1	1	1	1
f_{20}	6	6823	**3350**	19838	**14688**	0.72	0.43	0.89	0.93
f_{21}	4	6282	**3050**	18923	**12900**	1	1	1	1
f_{22}	4	7132	**2970**	20223	**12120**	1	1	1	1
f_{23}	4	6980	**3005**	19890	**11920**	1	1	1	1

f_{24}	30	52287	**21840**	165328	**128000**	1	1	1	1
f_{25}	2	2784	**1395**	6299	**5140**	1	1	1	1
Ave		29008.14	**16580.91**	174273.3	**101195.3**	0.85	**0.86**	0.90	**0.95**

TABLE VI
EFFECT OF JUMPING ON PROPOSED MDE ALGORITHM WITH JUMPING RATES AS 0.1 AND 0.3. THE RESULTS ARE TABULATED FOR NUMBER OF FUNCTION EVALUATIONS (NFE) AND SUCCESS RATE (SR). Ave. REPRESENTS AVERAGE AND '- -' REPRESENTS THAT THE ALGORITHM WAS NOT ABLE TO ACHIEVE THE DESIRED ACCURACY

Fun	n	NFE				SR			
		DE	MDE	MDEj0.3	MDEj0.1	DE	MDE	MDEj0.3	MDEj0.1
f_1	30	104310	45980	48870	48270	1	1	1	1
f_2	30	173850	77830	80960	78540	1	1	1	1
f_3	30	110700	48600	53400	50700	1	1	1	1
f_4	30	274150	258886	267000	261000	0.36	**0.75**	0.32	0.46
f_5	30	--	190600	248300	212000	0	1	1	1
f_6	30	31890	14850	15540	14670	1	1	1	1
f_7	30	131640	70680	82000	75420	1	1	1	1
f_8	30	226850	101067	--	--	0.9	0.88	0	0
f_9	30	--	--	--	--	0	0	0	0
f_{10}	30	163020	72800	76670	73200	1	1	1	1
f_{11}	30	108930	48077	52383	49444	1	1	0.62	0.84
f_{12}	30	95400	43340	43811	43280	1	1	0.88	1
f_{13}	30	104310	46680	48744	46920	1	1	0.92	1
f_{14}	2	5220	3330	3265	3290	1	1	1	1
f_{15}	4	11220	6050	6234	6540	1	1	1	1
f_{16}	2	5720	3330	3243	3510	1	1	1	1
f_{17}	2	6930	4790	4840	4520	1	1	1	1
f_{18}	2	4470	2850	2983	3070	1	1	1	1
f_{19}	3	5010	2870	3045	3560	1	1	1	1
f_{20}	6	14400	7050	7143	7620	0.84	0.78	0.44	0.52
f_{21}	4	11990	6640	6538	6930	1	1	1	1
f_{22}	4	11290	6220	6458	6510	1	1	1	1
f_{23}	4	11330	6190	6245	6350	1	1	1	1
f_{24}	30	104540	46580	48300	47700	1	1	1	1
f_{25}	2	4160	2640	6560	2960	1	1	1	1
Ave		67930.9	**37557.4**	39737.8	38363.8	0.88	**0.94**	0.85	0.87

TABLE VII
COMPARISON OF PROPOSED MDE ALGORITHM WITH DE AND ODE [50] ON 7 NONTRADITIONAL SHIFTED FUNCTIONS IN TERMS OF ERROR (BEST MEDIAN, WORST AND MEAN) AND STANDARD DEVIATION (Std). DIMENSION (n) OF ALL THE PROBLEMS IS TAKEN AS 500. MAXIMUM NFE IS SET AS 5000*n

Problem	n	Error value	DE	ODE [50]	MDE
F_1	500	Best	2, 636.54	15.66	**3.48**
		Median	3, 181.45	36.61	**5.32**
		Worst	4, 328.80	292.65	**7.57**
		Mean	3, 266.24	80.17	**4.86**
		Std	409.68	79.24	**4.34**
F_2	500	Best	79.74	**3.60**	19.82
		Median	82.39	**4.86**	11.88
		Worst	85.92	**11.91**	12.26
		Mean	82.93	**5.78**	11.87
		Std	2.09	2.37	**1.93**
F_3	500	Best	76, 615, 772.08	**39, 718.90**	727, 996.00
		Median	119, 733, 049.20	**137, 279.03**	731, 546.21
		Worst	169, 316, 779.50	**407, 661.64**	732, 763.93
		Mean	123, 184, 755.70	**154, 306.34**	730, 473.25
		Std	29, 956, 737.58	**114, 000.53**	116,325.43

F_4	500	Best	5, 209.99	2, 543.51	**1, 155.15**
		Median	5, 324.57	4, 279.56	**3, 243.87**
		Worst	5, 388.24	6, 003.94	**4, 478.90**
		Mean	5, 332.59	4, 216.34	**4, 212.76**
		Std	43.82	1, 017.94	**58.60**
F_5	500	Best	24.29	1.25	**0.31**
		Median	24.71	1.55	**0.87**
		Worst	27.59	2.13	**0.96**
		Mean	25.16	1.75	**0.56**
		Std	1.10	0.37	**0.05**
F_6	500	Best	4.66	2.49	**1.18**
		Median	4.97	4.12	**1.47**
		Worst	5.15	6.73	**1.56**
		Mean	4.94	4.51	**1.25**
		Std	0.17	1.44	**0.07**
F_7	500	Best	-3683.07	-3957.85	**-3992.76**
		Median	-3575.13	-3834.07	**-3836.65**
		Worst	-3565.73	-3830.36	**-3833.21**
		Mean	-3593.75	-3851.82	**-3863.59**
		Std	32.74	38.80	**29.31**

E. Effect of jumping on the proposed MDE algorithm

In [14], the authors proposed the concept of jumping in their algorithm ODE. We applied the same concept on the proposed MDE algorithm, taking jumping rates as 0.1 and 0.3 and recorded the results in terms of NFE and SR in Table VI. From this Table it can be seen that by applying the concept of jumping, the modified MDE algorithms (MDEj0.3 and MDEj0.1) were not able to solve function f_8 besides f_9. Also, the average NFE increased and the SR deteriorated (0.85 for MDEj0.3 and 0.872 for MDEj0.1).

Analysis of result: The idea of jumping is not beneficial for the MDE algorithm. The success rate for MDE which is 0.94 came down to 0.85 when jumping rate was kept as 0.3. This shows a deterioration of around 10%. When jumping was reduced to 0.1, there was an improvement in the average success rate (0.87) but still it was 7% lesser than the SR of MDE algorithm. Going by these results we can say that the idea of jumping is not favorable for MDE algorithm.

F. Numerical results for nontraditional benchmark problems

The performance of MDE is also validated on a set of 7 nontraditional benchmark functions and the corresponding numerical results are reported in Table VII in terms of best, median, worst and mean error and standard deviation**.** From these results we see that MDE outperforms basic DE for all the test problems in terms of error and standard deviation by a significant difference. In comparison to ODE [50], we see that MDE outperformed it in 5 out of 7 cases in terms of error as well as standard deviation. In the remaining two cases ODE [Rahnamayan and Wang, 2008] performed better than MDE.

Analysis of Results – MDE performed better than DE for all the test cases with an improvement of up to 99% in the best function value for F1, F3 and F5 and an improvement up to 75% for F2, F4 and F6. For the last function F7, the improvement is around 8%. In case of ODE, for the 5 function in which MDE gave a better performance, the improvement in F1 and F5 is more than 75%. For F4 and F6, the improvement is more than 50% and for F7, the improvement is around 1%. These results show the efficiency of MDE for solving large scale problems.

G. Numerical results of real life problems

The numerical results of three real life problems are recorded in Tables VIII (A), VIII (B) and VIII (C). In Table VIII (A), MDE is compared with DE and DEGL for frequency modulation problem in terms of average fitness function value and standard deviation (Std.). It can be clearly observed from the Table that MDE outperforms both DE and DEGL by a significant difference. Result for transistor modeling problem is given in Table VIII (B). Here MDE is compared with DE in terms objective function value which is clearly better for MDE. In Table VIII (C) results for spread spectrum radar poly phase code design problem are given in terms of average fitness function value and standard deviation. Here MDE is compared with DE and DEGL. The numerical results taken for dimensions 19 and 20 show that for 19 variables problem, MDE outperformed DE and DEGL in terms of average fitness function value and for 20 dimensions problem MDE performed better than both the other algorithms in terms of average fitness function value and standard deviation.

Analysis of results **–** from these results we can say that the proposed MDE is competent for solving the real life problems.

TABLE VIII (A)
AVERAGE AND STANDARD DEVIATION (IN PARENTHESES) OF THE BEST-OF-RUN SOLUTIONS FOR 50 RUNS ON THE FREQUENCY
MODULATOR SYNTHESIS PROBLEM. EACH ALGORITHM WAS RUN FOR 10^5 NFEs

DE	MDE	DEGL[64]
4.70081e-04 (3.4345e-05)	**1.24148e-28 (7.3288e-31)**	4.81520e-09 (6.2639e-09)

TABLE VIII(B)
AVERAGE AND PARAMETER VALUES OF THE BEST-OF-RUN SOLUTIONS FOR 50 RUNS OVER THE TRANSISTOR MODELING PROBLEM EACH ALGORITHM WAS RUN UP TO 5×10^5 NFEs

	DE	MDE
x1	0.901340	0.901337
x2	0.891164	0.891043
x3	3.87857	3.87943
x4	3.94653	3.94663
x5	5.32623	5.32509
x6	10.6267	10.6171
x7	0.0	0.0
x8	1.08924	1.08832
x9	0.705675	0.706734
f(X)	0.0937829	**0.0643636**

Table VIII(C)
AVERAGE AND STANDARD DEVIATION (IN PARENTHESES) OF THE BEST-OF-RUN SOLUTIONS FOR 50 RUNS OVER THE SPREAD
SPECTRUM RADAR POLY-PHASE CODE DESIGN PROBLEM (NUMBER OF DIMENSIONS ARE n=19 AND n=20). FOR ALL CASES
EACH ALGORITHM WAS RUN UP TO 5×10^5 NFEs

Dim	DE	MDE	DEGL[64]
19	3.80121e-01 (2.3434e-02)	**2.50000e-01 (3.0993e-03)**	7.44390e-01 (5.8400e-04)
20	4.57939e-01 (4.3874e-03)	**2.50483e-01 (1.3290e-04)**	8.03040e-01 (2.7300e-03)

8. STATE OF THE ART DE ALGORITHMS USED FOR COMPARISON

In this section we give a brief description of other state of the art DE algorithms used in this paper. These are recently proposed algorithms and have reportedly given good performance on a set of various benchmark problems.

SaDE - SaDE [Qin et al., 2009] was proposed by Qin and Suganthan to simultaneously implement two mutation strategies "DE/rand/1" and "DE/current-to-best/1." It adapts the probability of generating offspring by either strategy based on their success ratios in the past 50 generations. It is believed that this adaptation procedure can gradually evolve the most suitable mutation strategy at different learning stages for the problem under consideration. In SaDE, the mutation factors are independently generated at each generation according to a normal distribution with mean 0.5, standard deviation 0.3, and truncated to the interval (0, 2]. To speed up the convergence of SaDE, the authors further applied a local search procedure (quasi-Newton method) to some good

individuals after 200 generations. SaDE has been applied to both constrained and unconstrained problems.

jDE - Brest *et al*. [Brest et al., 2007], [Brest et al., 2006] proposed a new adaptive DE, jDE, based on the classic DE/rand/1/bin. Similar to other schemes, jDE fixes the population size during the optimization while adapting the control parameters *F and Cr* associated with each individual. It is believed that better parameter values tend to generate individuals which are more likely to survive and thus these values should be propagated to the next generation. Experimental results suggest that jDE performs remarkably better than the classic DE/rand/1/bin, and many other adaptive and non adaptive algorithms.

JADE – was proposed by Zhang and Sanderson [Zhang and Sanderson, 2009]. They implemented a new mutation strategy named "DE/current-to-*p*best" with an optional external archive and updated control parameters in an adaptive manner. Their strategy is a generalization of the classic "DE/current-to-best," while the optional archive operation utilizes historical data to provide information of progress direction. The parameter adaptation automatically updates the control parameters to appropriate values and avoids a user's prior knowledge of the relationship between the parameter settings and the characteristics of optimization problems. In JADE, the crossover probability *Cr* and scaling factor *F* are generated independently for each individual using normal and Cauchy distribution. They validated their algorithm on a set of 20 benchmark problems and compared it with other adaptive and non adaptive algorithms.

A. Comparison of MDE with other state of the art algorithms

The proposed MDE is compared with three other state of the art DE algorithms given in the previous section on the basis of average fitness, standard deviation (Std.), number of function evaluations and success rate (SR). Here we fixed the number of generations as given in Table IX (A). The remaining parameters are kept same as discussed in the earlier section V. From Table IX (A) which gives the results on the basis of fitness and standard deviation we see that JADE performed better than MDE and other algorithms in 5 cases, while MDE gave the best performance in 8 cases. In the remaining cases all the algorithms performed in a similar manner. On the basis of NFE the results are given in Table IX (B). From this Table we see that JADE took lesser NFE than other algorithms in 8 cases, while MDE outperformed other algorithms in 5 cases. On an average JADE took 38012 NFE for solving 25 test problems while MDE took 46580 NFE which is slightly worse than JADE. However, in comparison to jDE and SaDE which took on an average 82012 and 71365 NFE respectively, the performance of MDE is quite good.

The success rate of JADE comes out to be 97% while for MDE the success rate comes out to be 96% for SaDE and jDE, the success rates are 94% each.

We also compared the algorithms statistically on the basis of NFE for which the results are given in Tables IX(C)-IX(E). Once again we followed the same procedure given in Section VII-B. An overall comparison of algorithms is given in Table IX (C) and IX(D). Table IX(C) shows that there is a significant difference between the algorithms. From Table IX (D) we see that MDE and JADE are at par with each other while the remaining two algorithms jDE and SaDE do not perform as well as MDE. This is illustrated graphically in Figure 4(a). Pairwise comparison of MDE with JADE, jDE and SaDE is summarized in Table IX(E). From this Table we see that, though the paired t-test shows that there is no significant difference between MDE and other algorithms, Wilcoxon test shows that there is a significant difference between MDE and jDE and SaDE, while there is no difference between MDE and JADE.
This result can also be verified from Figure 4(b) which gives the box plot of algorithms on the basis of NFE.

Analysis of results –On the basis of fitness we cannot make a concrete judgment on the working of MDE. On the basis of success rate we see that JADE performs marginally better (1%) than MDE. On the basis of NFE, we see that MDE on an average took more

NFE than JADE but its performance was significantly better than jDE and SaDE. However, statistically we see that MDE and JADE are at par with each other on the basis of NFE.

TABLE IX (A)
COMPARISON OF MDE WITH jDE, JADE, AND SaDE IN TERMS OF FITNESS FUNCTION VALUE. Std. REPRESENTS THE STANDARD DEVIATION AND n REPRESENTS THE DIMENSION OF THE PROBLEMS. FOR ALL CASES EACH ALGORITHM IS RUN UP TO MXIMUM # OF GENERATIONS.

Fun	n	#Gen	Fitness (Std.)			
			jDE	JADE	SaDE	MDE
f_1	30	1500	2.34343e-28 1.92383e-28	**1.73443e-60** **7.34344e-60**	3.54533e-20 5.79432e-20	2.74298e-36 3.94901e-36
f_2	30	2000	3.09343e-23 8.38772e-24	**2.38353e-25** **8.47876e-25**	1.02398e-14 1.83421e-15	1.10654e-24 8.28990e-25
f_3	30	5000	3.39041e-14 3.82921e-14	**4.44584e-61** **1.32743e-60**	9.04322e-37 3.44302e-37	**4.81002e-131** **0**
f_4	30	5000	0 0	**8.43245e-24** **4.20037e-23**	6.49202e-11 1.63430e-10	5.92984e-11 8.36854e-10
f_5	30	20000	**0** **0**	8.94840e-02 5.97326e-01	2.34993e-01 2.33498e-01	**0** **0**
f_6	30	1500	0 0	0 0	0 0	0 0
f_7	30	3000	2.31545e-03 7.38443e-04	8.54564e-04 2.32534e-04	3.58832e-03 1.62992e-03	**2.05093e-04** **1.04551e-03**
f_8	30	9000	-12569.5 8.00132e-12	**-12569.5** **0**	-12569.5 8.43901e-08	-12569.5 1.09766e-10
f_9	30	5000	0 0	0 0	0 0	8.95493e+00 1.59359e+01
f_{10}	30	1500	7.09431e-15 1.72928e-15	5.65784e-15 0	7.38286e-14 3.48321e-14	**4.05954e-15** **0**
f_{11}	30	2000	0 0	0 0	0 0	0 0
f_{12}	30	1500	5.93708e-30 2.32384e-30	1.06754e-32 3.43503e-48	2.43748e-19 0	**1.35993e-35** **0**
f_{13}	30	1500	6.90221e-29 3.84204e-29	4.65656e-32 4.14394e-48	2.83043e-19 0	**1.29390e-32** **0**
f_{14}	2	100	0.998004 1.90023e-16	0.998004 0.998004	0.998004 1.32943e-16	0.998004 **1.21077e-16**
f_{15}	4	4000	4.29044e-04 3.28494e-04	6.78786e-05 3.07102e-04	8.43984e-04 4.54989e-08	**3.07102e-05** **9.71256e-09**
f_{16}	2	100	-1.03163 8.43843e-12	-1.03163 -1.03163	-1.03163 1.48430e-16	-1.03163 **2.22875e-16**
f_{17}	2	100	0.397887 4.43492e-08	0.397887 0.397887	0.397887 0	0.397887 **0**
f_{18}	2	100	3.0 1.98237e-15	3.0 3.0	3.0 2.43493e-16	3.0 **1.61278e-16**
f_{19}	3	100	-3.8623 9.32384e-15	-3.8626 7.76755e-14	-3.8623 7.34399e-15	-3.8623 **4.44089e-16**
f_{20}	6	100	-3.2863 6.34934e-06	-3.2986 5.75433e-05	-3.3182 6.34348e-03	**-3.2807** 2.39493e-03
f_{21}	4	100	-10.1532 3.34321e-06	-10.1532 4.88743e-13	-10.1532 4.23484e-15	-10.1532 **1.77532e-15**
f_{22}	4	100	-10.4029 5.25234e-07	-10.4029 8.57584e-13	-10.4029 2.43438e-15	-10.4029 **1.25879e-15**
f_{23}	4	100	-10.5364 5.02913e-06	-10.5364 8.76765e-11	-10.5364 7.43483e-14	-10.5364 **1.94865e-15**
f_{24}	30	10000	3.49941e-51 7.34301e-53	6.67607e-61 8.57008e-63	7.38393e-58 3.45843e-60	**1.47748e-76** **5.93376e-77**
f_{25}	2	100	-1 0	-1 0	-1 0	-1 0

TABLE IX (B)
COMPARISON OF MDE WITH jDE, JADE, AND SaDE IN TERMS OF FUNCTION EVALUATION. MAXIMUM NFE IS TAKEN AS 10000*n. WHERE n DENOTE THE DIMENSION OF THE PROBLEM. Ave. REPRESENTS AVERAGE AND '- -' REPRESENTS THAT THE ALGORITHM WAS NOT ABLE TO ACHIEVE THE DESIRED ACCURACY 10^{-08} FOR ALL FUNCTIONS EXCEPT f_7 FOR WHICH IT IS 10^{-02}.

Fun	n	NFE				SR			
		JADE	MDE	jDE	SaDE	JADE	MDE	jDE	SaDE
f_1	30	**29900**	45980	60100	73490	1	1	1	1
f_2	30	**52550**	77830	83220	118932	1	1	1	1
f_3	30	94840	**48600**	339399	181673	1	1	1	1
f_4	30	**170890**	258886	300650	290380	0.92	**1**	0.8	0.86
f_5	30	**151000**	190600	575990	278890	0.9	**1**	0.6	0.29
f_6	30	**11560**	14850	24860	28410	1	1	1	1
f_7	30	**30000**	70680	98000	128764	1	1	1	1
f_8	30	130480	**101067**	88940	121830	1	1	0.68	0.81
f_9	30	131000	--	118630	170765	1	0	1	1
f_{10}	30	**45610**	72800	90620	119090	1	1	1	1
f_{11}	30	**34000**	48077	64270	80688	1	1	1	1
f_{12}	30	**26950**	43340	54310	72346	1	1	1	1
f_{13}	30	**30988**	46680	61287	73432	1	1	1	1
f_{14}	2	3455	**3330**	3578	3672	1	1	1	1
f_{15}	4	6532	**6050**	6648	6438	1	1	1	1
f_{16}	2	3310	3330	3298	3320	1	1	1	1
f_{17}	2	**4520**	4790	4872	4810	1	1	1	1
f_{18}	2	3080	**2850**	3389	3010	1	1	1	1
f_{19}	3	2990	**2870**	3154	3080	1	1	1	1
f_{20}	6	**7000**	7050	7410	7832	0.47	**0.95**	0.41	0.44
f_{21}	4	6745	**6640**	6829	6770	1	1	1	1
f_{22}	4	6389	**6220**	6194	6395	1	1	1	1
f_{23}	4	6090	6190	6530	6672	1	1	1	1
f_{24}	30	50380	**46580**	71290	89372	1	1	1	1
f_{25}	2	3050	**2640**	3456	3480	1	1	1	1
Ave		**38012.9**	46580.4	82012.2	71365.7	0.97	0.96	0.94	0.94

Table IX (C)
RESULTS OF FRIEDMAN TEST BASED ON NFE

N	Friedman value	df	*p*-value
25	34.776	3	<0.001

df – Degrees of Freedom N - Total No of functions

Table IX (D)
RANKING OBTAINED THROUGH FRIEDMAN'S TEST AND CRITICAL DIFFERENCE (CD) CALCULATED THROUGH BONNFERRONI-DUNN'S PROCEDURE

Algorithm	**Mean Rank**
JADE	**1.68**
MDE	1.84
jDE	3.00
SaDE	3.84
CD for α=0.05	0.874165
CD for α=0. 01	0.777036

Table IX (E)
RESULTS OF PAIRWISE COMPARISON BASED OF NFE

Algo.	paired t-test			Wilcoxon test			
	Stat	Sig.	+ve	-ve	tie	Stat	Sig.
JADE	1.852	0.076	10	15	0	-1.493	0.135
jDE	-1.304	0.205	21	4	0	-3.269	0.001
SaDE	-2.092	0.047	23	2	0	-3.700	0.000

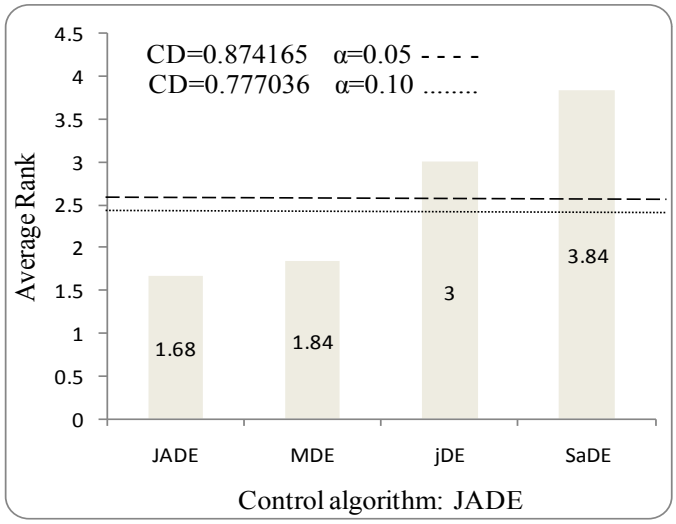


Figure 4(a) Bonferroni-Dunn's graphic corresponding to NFE

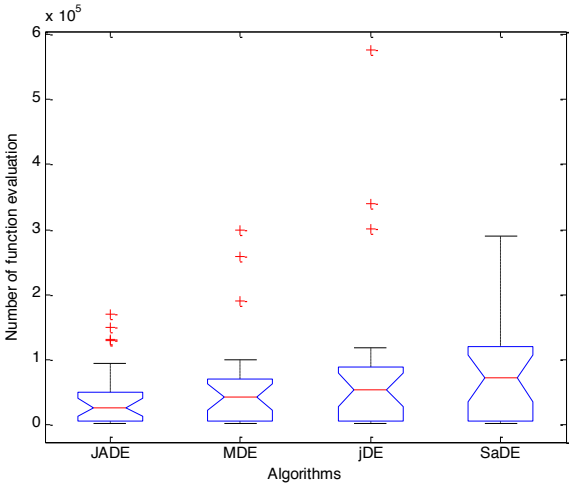


Figure 4(b) Box-plot corresponding to the average NFE

9. CONCLUSIONS

In the present study we presented a simple and efficient variant of DE, called Modified Differential Evolution (MDE). The proposed MDE is a fusion of three schemes; opposition based learning for generating the initial population, tournament best method for mutation and one population DE structure. The parent algorithms (ODE, DERL and MDE1) using these schemes individually have reportedly given very good performance.

While the OBL helps in proving an efficient start to the DE algorithm, the use of tournament best base vector induces a localized effect in the search procedure. Both these features help in enhancing the exploratory and exploitation capabilities of the DE algorithm which in turn helps in preventing premature convergence. The third feature which is the use of a single population DE structure (in contrast to the two set structure used in basic DE) helps in faster convergence.

As expected, these schemes when combined together produce a synergized effect which was better than any of the scheme used separately.

The performance of the proposed MDE algorithm is investigated on a set of traditional benchmark problems, nontraditional benchmark problems and real life problems. Its performance is compared with DE and its parent algorithms ODE, MDE1 and DERL.

Numerical results show that on the basis of error all the algorithms performed more or less in a similar manner. However, on the basis of NFE, %AR and SR we can clearly see that the combined effect of ODE, DERL and MDE1 in MDE makes it superior not only to DE but also to its parent algorithms.

These results are also validated with the help of statistical analysis using an overall and pairwise comparison of algorithms.

MDE is further compared with JADE, SaDE and jDE. Although these algorithms are adaptive in nature and their comparison with MDE may not be completely justified but these are some of the recent variants of DE and have given good performance in comparison to both adaptive and nonadaptive algorithms. Numerical results using standard performance measures showed that JADE performed better than MDE in terms of NFE, though the performance of MDE was much better than jDE and SaDE. Statistical analysis however showed that JADE and MDE are at par with each other.

The objective of this study is not to defeat DE or any of its variants but is to present an algorithm which is simple to understand and easy to apply by fusing together some of the efficient schemes available in literature. However, claiming that MDE will outperform every other variant of DE for every optimization problem, with any degree of complexity, does not sound justified. There are several other variants which may be successfully combined to produce an algorithm which is better than the proposed MDE. Even the performance of many existing versions of DE can be improved further by judicious tuning of parameters alone.

The only case where MDE was not able to perform successfully was function f_9. This indicates that some further fine tuning is needed in MDE so that it can solve all types of problems.

At this stage the conclusion that can be drawn from the present study is that the proposed MDE version can serve as an attractive alternative for a wide range of optimization problems. The paper can be extended in several directions. Fine tuning of parameters for MDE can be replaced with some suitable adaptive technique. Effects of adding some local search technique can also be observed.

REFERENCES

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

[2] D.E. Goldberg1989. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley.

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

[4] L.Fogel 1994. Evolutionary programming in perspective: the top-down view. *Computational Intellingece: Imitating Life*. Edited by J. M. Zurada. R. J. Marks and C. J. Robinson IEEE Press, Piscataway. 135-146.

[5] T. Rogalsky, R. W. Derksen, and S. Kocabiyik 1999. Differential evolution in aerodynamic optimization. in *Proc. 46th Annu. Conf. Can. Aeronautics Space Inst*. 29–36.

[6] R. Joshi and A. C. Sanderson 1999. Minimal representation multi-sensor fusion using differential evolution. *IEEE Trans. Syst. Man. Cybern. Part A*, 29, 1. 63–76.

[7] S. Das and A. Konar 2006. Design of two dimensional IIR filters with modern search heuristics: A comparative study. *Int. J. Comput. Intell. Applicat.* 6, 3, 329–355,

[8] F. S.Wang and H. J. Jang 2000. Parameter estimation of a bio-reaction model by hybrid differential evolution, in *Proc. IEEE Congr. Evol. Comput. 2000*, 1. Piscataway, NJ: IEEE Press 410–417.

[9] J. Lampinen. 1999. A bibliography of differential evolution algorithm. Lappeenranta University of Technology. Department of Information Technology, Laboratory of Information Processing, Tech. Report [Online]. Available: http://www.lut.fi/jlampine/debiblio.htm

[10] M. Omran, A. P. Engelbrecht, and A. Salman 2005. Differential evolution methods for unsupervised image classification, in *Proc. 7th Congr. Evol. Comput. (CEC-2005)*, 2. Piscataway, NJ: IEEE Press, 966–973.

[11] S. Das, A. Abraham, and A. Konar 2008. Adaptive clustering using improved differential evolution algorithm, *IEEE Trans. Syst., Man, Cybern. A*, 38, 1, 218–237.

[12] J. Lampinen and I. Zelinka 2000. On stagnation of the differential evolution algorithm, *in Proc. of MENDEL 2000, 6th International Mendel Conference on Soft Computing*, 76 – 83, , Brno, Czech Republic.

[13] H. Maaranen, K. Miettinen, M.M. Makela 2004. A Quasi-Random Initial Population for Genetic Algorithms, *Computers and Mathematics with Applications*, 47 (12), 1885–1895,

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

[15] H. R. Tizhoosh 2005, Opposition-based learning: A new scheme for machine intelligence, *in Proc. Int. Conf. Comput. Intell. Modeling Control and automation CIMCA2005,* 695-701, Austria.

[16] S. Rahnamayan, H.R. Tizhoosh, M.M.A.Salama 2008. Opposition versus randomness in soft computing techniques, *Applied soft computing,* 8, 2, 906-918.

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

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

[19] Musrrat Ali, Millie Pant and Ajith Abraham, 2009. A Modified Differential Evolution Algorithm and Its Application to Engineering Problems, *in proc. International conference of soft computing and pattern recognition (SoCPaR-2009)*, 196-201.

[20] R. Gamperle, S. D. Muller, and A. Koumoutsakos, 2002. A Parameter study for differential evolution*, in WSEAS NNA-FSFS-EC 2002, Interlaken, Switzerland*, 11-15.

[21] J. Ronkkonen, S. Kukkonen and K. V. Price, 2005. Real parameter optimization with differential evolution, *in Proc. of IEEE Congress on Evolutionary Computation (CEC-2005)*, 1, 506 – 513.

[22] J. Liu and J. Lampinen2005. A fuzzy adaptive differential evolution algorithm, *Soft computing*, 9, 6, 448-462.

[23] K. Qin, V. L. Huang, and P. N. Suganthan, 2009. Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computations*, 13, 2, 398-417

[24] D. Zaharie, 2003. 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, 41-46

[25] D. Zaharie and D. Petcu, 2004. Adaptive pareto differential evolution and its parallelization, *in Proc. of 5th International Conference on Parallel Processing and Applied Mathematics*, Czestochowa, Poland, 3019, 261 – 268

[26] H. Abbass, 2002. The self-adaptive pareto differential evolution algorithm, *in Proc. of the 2002 Congress on Evolutionary Computation*, 831-836.

[27] M. Omran, A. Salman, and A. P. Engelbrecht 2005. Self-adaptive differential evolution, *in proc. computational intelligence and security, Lecture Notes In Artificial Intelligence,* 3801, 192-199,

[28] Janez Brest, Borko Boskovi, Saso Greiner, Viljem Zumer, Mirjam Sepesy Mau cec. 2007. Performance comparison of self-adaptive and adaptive differential evolution algorithms, *Soft Computing*, 11, 7, 617-129.

[29] Nga Sing Teng, Jason Teo, Mohd. Hanafi A. Hijazi, 2009. Self-adaptive population sizing for a tune-free differential evolution, *Soft Computing*, 13, 7, 709-724.

[30] Z. Yang, K. Tang and X. Yao, 2008. Self-adaptive differential evolution with neighborhood search, *in Proc. IEEE Congress on Evolutionary Computation (CEC-2008)*, Hong Kong, 1110-1116

[31] Z. Yang, K. Tang and X. Yao, 2008. Large Scale Evolutionary Optimization Using Cooperative Co evolution, *Information Sciences*, 178, 15, 2985-2999

[32] T.Hendtlass, 2001. A combined swarm differential evolution algorithm for optimization problems, *In proceedings of the Fourteenth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Lecture Notes in Computer Science*, 2070. Springer-Verlag, 11–18.

[33] W.J.Zhang and X.F Xie, 2003. DEPSO: Hybrid particle swarm with differential evolution operator, *In proc. Of IEEE International Conference on Sys tems, Man, and Cybernetics*, 4, 3816–3821.

[34] H. Talbi and M. Batouche, 2004. Hybrid particle swarm with differential evolution for multimodal image registration. In: Proceedings of the IEEE international conference on industrial technology, 3, 1567–1573.

[35] S. Kannan, S. Slochanal, P. Subbaraj, N. Padhy, 2004. Application of particle swarm optimization technique and its variants to generation expansion planning, *Electric Power Systems Research,* 70, 3, 203–210.

[36] M.G.H. Omran, A.P.Engelbrecht and A.Salman 2008. Bare bones differential evolution, *European journal of operational research*, doi:10.1016/j.ejor.2008.02.035.

[37] C.Zhang, J. Ning, S. Lu, D.Ouyang and T.Ding, 2009. A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization, *Operations Research Letters* 37, 117 – 122.

[38] W. Xu, and X. Gu, 2009. A hybrid particle swarm optimization approach with prior crossover differential evolution. *In proc. of GEC09* 671 – 677.

[39] Andrea Caponio, Ferrante Neri and Ville Tirronen, 2009. Superfit control adaption in memetic differential evolution frameworks, *Soft Computing*, 13, 811-831.

[40] Mohammad G.H. Omran and Andries P Engelbrecht, 2009. Free Search Differential Evolution, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 110-117.

[41] Adriana Menchaca-Mendez and Carlos A. Coello Coello, 2009. A new proposal to hybridize the Nelder Mead Differential Evolution Algorithm for Constrained Optimization, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 2598-2605.

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

[43] M.Pant, M.Ali and V.P.Singh, 2009. Parent centric differential evolution algorithm for global optimization, *Opsearch,* 46, 2, 153-168.

[44] M.Pant, R.Thangaraj, A.Abraham and C.Grosan, 2005. Differential Evolution with Laplace Mutation Operator, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 2841-2849.

[45] K. Deb 2005. A population-based algorithm-generator for real-parameter optimization, *soft Comput J*, 9, 236–253.
[46] M.Pant, M.Ali and A.Abraham, 2009. Mixed Strategy Embedded Differential Evolution, *in proc. of IEEE Congress on Evolutionary Computation*, Norway, 1240-1246.
[47] Lai, JCY, Leung FHF and Ling SH 2009. A new Differential Evolution Algorithm with Wavelet Theory based Mutation operation. IEEE Congress on Evolutionary Computation, Norway, 1116-1122.
[48] N. Noman and H. Iba 2005. Enhancing differential evolution performance with local search for high dimensional function optimization, *in Proc. of the 2005 Conference on Genetic and Evolutionary Computation*, 967–974.
[49] N. Noman and H. Iba, 2008. Accelerating Differential Evolution Using an Adaptive Local Search, *IEEE Transactions on Evolutionary Computation*, 12, 1, 107 – 125.
[50] S. Rahnamayan and G.G.Wang, 2008. Solving large scale optimization problems by opposition based differential evolution (ODE), *WSEAS transaction on computers*, 7 (10), 1792-1804.
[51] Zhenyu Yang, Jingqiao Zhang, Ke Tang, Xin Yao and Arthur C. Sanderson, 2009. An adaptive Coevolutionary Differential Evolution Algorithm for large Scale optimization, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 102 – 108.
[52] Janez Brest, Ales Zamuda, Boroko Boskovic, Mirjam Sepesy Maucec and Viljem Zumer, 2009. Dynamic Optimization using Differential Evolution, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 415-421.
[53] Chuan Kang Ting and Chih-Hui Huang, 2009. Varying numbers of Difference Vectors in Differential Evolution, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 1351-1358.
[54] M.G Epitropakis, V.P. Plagianakos and M.N Vrahatis, 2009. Evolutionary Adaption of the Differential Evolution Control Parameters, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 1359-1366.
[55] Faith Tasgetiren, M., Quan-Ke Pan, P.N. Suganthan, and Yun Chia Liang, 2009. A Differential Evolution Algorithm with Variable Parameter Search for Real Parameter Continuous Function Optimization, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 1247-1254.
[56] James Montgomery, 2009. Differential Evolution: Difference Vectors and Movement in Solution Space, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 2833-2840.
[57] Yu Wang, Bin Li and Xuexiao Lai 2009. Variance Priority based Cooperative Co-evolution Differential Evolution for large scale Global Optimization, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 1232-1239.
[58] Fei Peng, Ke Tang, Guoliang Chen and Xin Yao, 2009. Multistart JADE with Knowledge Transfer for Numerical Optimization, *in proc. Of IEEE Congress on Evolutionary Computation*, Norway, 1889-2895
[59] U. K. Chakraborty (Ed.), 2008. *Advances in Differential Evolution*, Springer-Verlag, Heidelberg.
[60] R. Thomsen, 2004. Multimodal optimization using crowding-based differential evolution, *in proc. Of 2004 congress on evolutionary computation CEC2004*, 2, 1382-1389.
[61] S. García, D. Molina, M. Lozano and F. Herrera, 2009. 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 Parameter Optimization, *Journal of Heuristics,* 15, 617-644.
[62] Jingqiao Zhang and A.C. Sanderson, 2009. JADE: Adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation,* 13(5), 945–958.
[63] W.L.Price, global optimization by controlled random search, 1983. *Journal of optimization theory and applications*, 40, 3, 333-348.
[64] S.Das, A.Abraham, U.K.Chakraborty and A.Konar, 2009. Differential evolution using a neighborhood based mutation operator, *IEEE Transaction on evolutionary computation*, 13, 2, 526-553.
[65] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, Z. Yang, 2007. *Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization*, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, http://nical.ustc.edu.cn/cec08ss.php.
[66] J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, 2006. Self Adapting Control parameters in Differential Evolution: A comparative study of numerical benchmark problems, IEEE Transactions on evolutionary computation, 10, 6, 646 – 657

Appendix:

f_{14}:

$$a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & -32 & -16 & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 & 32 & 32 \end{pmatrix}$$

f_{15}:

i	a_i	b_i^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

f_{19}:

i	c_i	a_{ij}			p_{ij}		
		$j=1$	2	3	$j=1$	2	3
1	1	3	10	30	0.3689	0.1170	0.2673
2	1.2	.1	10	35	0.4699	0.4387	0.7470
3	3	3	10	30	0.1091	0.8732	0.5547
4	3.2	.1	10	35	0.3815	0.5743	0.8828

f_{21}, f_{22} and f_{23}:

i	a_{ij} j=1,…,4				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

f_{20}:

i	c_i	a_{ij} j=1,…,6						p_{ij} j=1,…,6					
1	1	10	3	17	3.5	1.7	8	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	1.2	.05	10	17	0.5	8	14	0.2329	0.4135	0.8307	0.3736	0.1004	0.999
3	3	3	3.5	1.7	10	17	8	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	3.2	17	8	.05	10	0.1	14	0.4047	0.8828	0.8732	0.5743	0.1091	0.038