Applied Mathematics and Computation 216 (2010) 532–544

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

New mutation schemes for differential evolution algorithm and their application to the optimization of directional over-current relay settings

Radha Thangaraj [a,*], Millie Pant [a], Ajith Abraham [b]

[a] Department of Paper Technology, Indian Institute of Technology Roorkee, Saharanpur 247001, India
[b] Q2S, Norwegian University of Science and Technology, Norway

ARTICLE INFO

Keywords:
Constrained optimization
Differential evolution
Laplace distribution
Mutation
Over-current relay coordination

ABSTRACT

Differential evolution is a novel evolutionary approach capable of handling non-differentiable, nonlinear and multimodal objective functions. It has been consistently ranked as one of the best search algorithm for solving global optimization problems in several case studies. In the present study we propose five new mutation schemes for the basic DE algorithm. The corresponding versions are termed as MDE1, MDE2, MDE3, MDE4 and MDE5. These new schemes make use of the absolute weighted difference between the two points and instead of using a fixed scaling factor F, use a scaling factor following the Laplace distribution. The performance of the proposed schemes is validated empirically on a suit of ten benchmark problems having box constraints. Numerical analysis of results shows that the proposed schemes improves the convergence rate of the DE algorithm and also maintains the quality of solution. Efficiency of the proposed schemes is further validated by applying it to a real life electrical engineering problem dealing with the optimization of directional over-current relay settings. It is a highly constrained nonlinear optimization problem. A constraint handling mechanism based on repair methods is used for handling the constraints. Once again the simulation results show the compatibility of the proposed schemes for solving the real life problem.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

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

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

EAs have been applied to a wide range of functions and real life problems [3–6]. Some common EAs are genetic algorithms (GA), evolutionary programming (EP), differential evolution (DE), etc. Besides these algorithms, there are some other population based methods like particle swarm optimization (PSO) and Ant colony optimization (ACO) based on social behavior shown by different species like birds, ants bees, etc.

* Corresponding author.
E-mail address: t.radha@ieee.org (R. Thangaraj).

0096-3003/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2010.01.071

In the present research paper, we have concentrated our work on DE, which is comparatively a newer addition to the class of population based search techniques. It was developed by Storn and Price [7] in 1995. DE is a novel evolutionary approach capable of handling non-differentiable, nonlinear and multimodal objective functions. DE has been designed as a stochastic parallel direct search method, which utilizes concepts borrowed from the broad class of EAs. It typically requires few, easily chosen control parameters. Experimental results have shown that performance of DE is better than many other well known EAs [8,9]. While DE shares similarities with other EAs, it differs significantly in the sense that in DE, distance and direction information is used to guide the search process [10].

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

The present study proposes five new mutation schemes for DE algorithm. Here we would like to mention that a preliminary version of this work appears in a conference proceeding [22]. However in [22], only one mutation scheme is proposed. Also in [22], only unconstrained optimization problems are tested whereas in the present study, we have considered a constrained real life optimization problem as well. All the mutation schemes proposed in the present study are based on the absolute weighted difference between the two points and use Laplace distributed random number as amplifying factor.

The structure of the paper is as follows: in Section 2, we briefly explain the differential evolution algorithm, in Section 3; we have defined and explained the proposed schemes for MDE algorithm. Section 4 deals with experimental settings, Section 5 gives the benchmark problems. In Section 6, the algorithms used for comparison are given and the numerical results of benchmark problems are analyzed in Section 7. Section 8 deals with the performance of MDE algorithms for constrained real life problem. Finally the paper concludes with Section 9.

2. Differential evolution

DE shares a common terminology of selection, crossover and mutation operators with GA however it is the application of these operators that make DE different from GA; while, in GA crossover plays a significant role, it is the mutation operator which affects the working of DE [11]. A general DE variant may be denoted as DE/*X*/*Y*/*Z*, where *X* denotes the vector to be mutated, *Y* specifies the number of difference vectors used and *Z* specifies the crossover scheme which may be binomial or exponential. For the more details the interested reader may please refer to [23].

The working of DE is as follows: First, all individuals are initialized with uniformly distributed random numbers and evaluated using the fitness function provided. Then the following are executed until a stopping criterion is satisfied.

2.1. Mutation

For a D-dimensional search space, for each target vector $X_{i,g}$ at the generation g, its associated mutant vector is generated via certain mutation strategy. The most often used mutation strategies implemented in the DE codes are listed below.

$$\text{DE/rand/1}: V_{i,g} = X_{r_1,g} + F^*(X_{r_2,g} - X_{r_3,g}) \tag{1}$$
$$\text{DE/rand/2}: V_{i,g} = X_{r_1,g} + F^*(X_{r_2,g} - X_{r_3,g}) + F^*(X_{r_4,g} - X_{r_5,g}) \tag{2}$$
$$\text{DE/best/1}: V_{i,g} = X_{best,g} + F^*(X_{r_1,g} - X_{r_2,g}) \tag{3}$$
$$\text{DE/best/2}: V_{i,g} = X_{best,g} + F^*(X_{r_1,g} - X_{r_2,g}) + F^*(X_{r_3,g} - X_{r_4,g}) \tag{4}$$
$$\text{DE/rand} - \text{to} - \text{best/1}: V_{i,g} = X_{r_1,g} + F^*(X_{best,g} - X_{r_2,g}) + F^*(X_{r_3,g} - X_{r_4,g}) \tag{5}$$

where $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, \ldots, NP\}$ are randomly chosen integers, different from each other and also different from the running index *i*. *F*(>0) is a scaling factor which controls the amplification of the difference vector. $X_{best,g}$ is the best individual vector with the best fitness value in the population at generation *g*.

2.2. Crossover

Once the mutation phase is over, crossover is performed between the target vector and the mutated vector to generate a trial point for the next generation. Crossover is introduced to increase the diversity of the population [8].

The mutated individual, $V_{i,G+1} = (v_{1,i,G+1}, \dots, v_{D,i,G+1})$, and the current population member, $X_{i,G} = (x_{1,i,G}, \dots, x_{D,i,G})$, are then subject to the crossover operation, that finally generates the population of candidates, or "trial" vectors, $U_{i,G+1} = (u_{1,i,G+1}, \dots, u_{D,i,G+1})$, as follows:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } \text{rand}_j \leqslant C_r \vee j = k \\ x_{j,i,G} & \text{otherwise} \end{cases} \tag{6}$$

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

2.3. Selection

The final step in the DE algorithm is the selection process. Each individual of the temporary (trial) population is compared with its counterpart in the current population. The one with the lower objective function value survives the tournament selection and go to the next generation. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. A notable point in DE's selection scheme is that a trial vector is not compared against all the individuals in the current generation, but only against one individual, its counterpart, in the current generation. The population for the next generation is thus selected from the individuals in current population and its corresponding trial vector according to the following rule:

$$X_{i,G+1} = \begin{cases} U_{i,G+1} & \text{if } f(U_{i,G+1}) \leqslant f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \tag{7}$$

In the present study we shall be following the scheme DE/rand/1/bin version which is apparently the most commonly used version and shall refer to it as basic DE or classical DE.

3. Proposed scheme for modified DE algorithm

The structural difference between the proposed MDE schemes and the basic DE lies in the mutation phase only. These schemes are based on the absolute weighted difference between the vectors to generate a mutant vector. The amplification factor, F, is replaced by a random variable (say L) following Laplace distribution. Based on these modifications five schemes namely MDE1, MDE2, MDE3, MDE4 and MDE5 are proposed. The first scheme, MDE1, uses only two vectors to generate a mutant vector. The second scheme, MDE2, is like target to best scheme of basic DE where the vector having the best fitness function value is used as a base vector. In MDE3, which is the third scheme two vectors are generated and the one having the better fitness function value is accepted as a mutant vector. In the fourth scheme the original mutation scheme as given by Eq. (1) and the MDE1 scheme are applied stochastically according to the user defined parameter P_{MDE}. Uniformly distributed random numbers between 0 and 1 are generated. If the random number is greater than the parameter P_{MDE}, then MDE1 is applied to generate the mutant vector otherwise the mutant vector is generated using Eq. (1). In the fifth case the mutant vector is generated by adding a random vector to the amplified distance between the best vector and another randomly generated vector. Mathematical definitions of the proposed schemes are given in Table 1.

As mentioned earlier the amplifying factor in all the cases is a random variable **L** following Laplace distribution. The probability density function (pdf) of Laplace distribution is similar to that of normal distribution however, the normal distribution is expressed in terms of squared difference from the mean, Laplace density is expressed in terms of absolute difference from the mean. The density function of Laplace distribution is given as:

Table 1
Proposed schemes of MDE algorithm.

Scheme	Definition	Number of points used for the generation of mutant vector
*MDE*1	$v_{i,g+1} = x_{r_1,g} + L^*\lvert x_{r_1,g} - x_{r_2,g}\rvert$	2, both points are selected randomly
*MDE*2	$v_{i,g+1} = x_{best,g} + L^*\lvert x_{r_1,g} - x_{r_2,g}\rvert$	3, one best point and the other two are randomly selected
*MDE*3	$v'_{i,g+1} = x_{r_1,g} + L^*\lvert x_{r_1,g} - x_{r_2,g}\rvert$	2, both points are randomly selected
	$v''_{i,g+1} = x_{r_2,g} + L^*\lvert x_{r_1,g} - x_{r_2,g}\rvert$	
	If $(f(v'_{i,g+1}) < f(v''_{i,g+1}))$ then	
	$v_{i,g+1} = v'_{i,g+1}$	
	Else $v_{i,g+1} = v''_{i,g+1}$	
*MDE*4	If (U(0,1) > P_{MDE}) then	MDE1 and Eq. (1) are applied stochastically
	$v_{i,g+1} = x_{r_1,g} + L^*\lvert x_{r_1,g} - x_{r_2,g}\rvert$	
	Else	
	$v_{i,g+1} = x_{r_1,g} + F^*(x_{r_2,g} - x_{r_3,g})$	
*MDE*5	$v_{i,g+1} = x_{r_1,g} + L^*\lvert x_{best,g} - x_{r_2,g}\rvert$	3, one is the best point and the other two are randomly selected.

$$f(x/\theta) = \frac{1}{2\mu} \exp\left(\frac{-|x-\theta|}{\mu}\right), \quad -\infty \leqslant x \leqslant \infty \tag{8}$$

Its distribution function is given by:

$$= \frac{1}{2\mu} \begin{cases} \exp\left(-\frac{x-\theta}{\mu}\right) & \text{if} \quad x \leqslant \theta \\ 1 - \exp\left(-\frac{\theta - x}{\mu}\right) & \text{if} \quad x > 0 \end{cases} \tag{9}$$

$\mu > 0$ is the scale parameter.

From Table 1, it can be seen that the newly generated mutant vector will lie in the vicinity of the base vector. However its nearness or distance from base vector will be controlled by **L**. For smaller values of μ, the mutant vector is likely to be produced near the initially chosen vector, whereas for larger values of μ, the mutant vector is more likely to be produced at a distance from the chosen vector. This behavior makes the algorithm self-adaptive in nature, which in turn helps in preserving the diversity of the population by exploring the search space more effectively.

4. Experimental settings

Setting of control parameters or fine tuning of parameters is a crucial task in EA and is mainly done empirically to select the best value of parameters. The main parameters of DE are population size, crossover rate Cr and scaling factor F. The population size is taken as 50 for all the test problems. However, this is a heuristic choice and may be increased, depending on the complexity of the problem. The other parameters, crossover rate Cr and scaling factor F, for classical DE, are fixed at 0.2 and 0.9, respectively. For MDE schemes we did a sensitivity analysis for various crossover rates varying it from 0.1 to 0.9 for all the test problems and observed that the crossover rate of 0.2 is most suitable for all the schemes used for solving the test suit taken in the present study. The value of additional parameter P_{MDE} in MDE4 scheme is taken as 0.2. As mentioned in the previous section, the scaling factor for all MDE schemes is a random variable **L** which follows Laplace distribution.

In order to make a fair comparison of DE and MDE algorithms, we fixed the same seed for random number generation so that the initial population is same for all the algorithms. For each algorithm, the maximum number of iterations allowed was set to 5000 and the error goal was set as 1*e−04. The numerical results are recorded for 30 runs for each algorithm. The algorithms are programmed using Developer C++ and are executed on a Pentium IV PC.

5. Benchmark problems

For the present study we considered a test bed of 10 benchmark problems given in Table 2. Though this test bed is rather narrow, we have tried to include problems having different characteristics. Except for the last two functions; f_9 and f_{10}, all the problems are solved for dimension 30. In this section we describe briefly the properties of these functions.

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

6. Algorithms used for comparison

Besides using the basic DE we have also used two recent versions of DE namely opposition based DE (ODE) and differential evolution with preferential crossover (DEPC), for comparison with the proposed schemes.

Table 2
Numerical benchmark problems.

Function	Function definition	Range	Min. Value
Rastringin function	$f_1(x) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	[−5.12,5.12]	0
Spherical function	$f_2(x) = \sum_{i=1}^{n} x_i^2$	[−5.12,5.12]	0
Griewank function	$f_3(x) = \frac{1}{4000}\sum_{i=0}^{n-1} x_i^2 + \sum_{i=0}^{n-1}\cos\left(\frac{x_i}{\sqrt{i+1}}\right) + 1$	[−600,600]	0
Rosenbrock function	$f_4(x) = \sum_{i=0}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	[−30,30]	0
Noisy function	$f_5(x) = \left(\sum_{i=0}^{n-1}(i+1)x_i^4\right) + rand[0,1]$	[−1.28,1.28]	0
Schwefel function	$f_6(x) = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	[−500,500]	−12569.5
Ackley Function	$f_7(x) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	[−32,32]	0
Michalewicz function	$f_8(x) = -\sum_{i=1}^{n}\sin(x_i)(\sin(i\frac{x_i^2}{\pi}))^{2m}, \quad m = 10$	$[-\pi, \pi]$	–
Himmelblau function	$f_9(x) = (x_2 + x_1^2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + x_1$	[−5,5]	−3.78396
Shubert function	$f_{10}(x) = \sum_{j=1}^{5} j\cos((j+1)x_1 + j)\sum_{j=1}^{5} j\cos((j+1)x_2 + j)$	[−10,10]	−186.7309

Table 3
Parameter Settings of MDE schemes.

Algorithm	Population size	F	CR	Max. generations	Error goal	Max. run
MDE schemes vs. DE						
DE	50	0.9	0.2	5000	1*e−04	30
MDE schemes	50	Laplace distributed	0.2	5000	1*e−04	30
MDE schemes vs. DEPC						
DEPC	10*n	$[-1, -0.4] \cup [0.4, 1]$	0.5	–	1*e−04	100
MDE schemes	10*n	Laplace distributed	0.5	–	1*e−04	100
MDE schemes vs. ODE						
ODE	100	0.5	0.9	10000	1*e−08	50
MDE schemes	100	Laplace distributed	0.9	10000	1*e−08	50

n – dimension of the problem.

Differential evolution with preferential crossover was suggested by Ali in 2007 [24]. In this work he suggested three changes in the basic DE structure. The DEPC algorithm uses F_i as a random variable in $[-1, -0.4] \cup [0.4, 1]$ for each target point. Secondly DEPC used two population sets S_1 and S_2 containing N points. The function of the auxiliary set S_2 in DEPC is to keep record of the trial points that are discarded in DE. Potential trial points in S_2 are then used for further explorations. Finally DEPC uses a new crossover rule, namely the preferential crossover, which always generates feasible trial points. Ali tested his algorithm on comprehensive set of benchmark problems and showed that DEPC outperforms the basic DE in most of the test cases. The second algorithm that we have used for comparison is opposition based DE, suggested by Rahnamayan et al. [25]. They suggested a novel method of generating the population based on opposition based-learning. They made use of two population sets one containing the randomly generated points and the other containing the points opposite to that of the initial points. Finally the two populations were merged and the best n points were taken to form the initial population. They also introduced the concept of generation jumping to further improve the performance of ODE. The parameter settings of MDE schemes for comparison of DE, DEPC and ODE are given in Table 3.

7. Numerical results and comparison

7.1. Performance measures

In order to compare the proposed MDE schemes with basic DE and other modified versions of DE we considered various performance metrics like average fitness function value and standard deviation (STD) to check the efficiency and reliability of the algorithm. To compare the convergence speed of algorithms we considered the average number of function evaluations (NFE). Smaller number of function evaluations indicates faster convergence. The speed of the algorithm is also measured by recording the total CPU time and the average CPU time taken by the algorithm to meet the stopping criteria. Besides this we have measured the success rate (SR) and average success rate (ASR). A run is considered to be a success if the value obtained at the end of the algorithm is within one percent of the desired accuracy. The definitions of performance measures used in Tables 5–7 are given as:

$$\text{Average NFE} = \frac{\sum_{i=1}^{n}\text{NFE}(f_i)}{n}$$

$$\text{Improvement (\%) in terms of NFE} = \frac{\text{Total NFE (basic DE algorithm)} - \text{Total NFE (Algorithm to be compared)}}{\text{Total NFE (basic DE algorithm)}} = 100$$

$$\text{Acceleration rate (AR)} = \frac{\text{Total NFE for basic DE}}{\text{Total NFE for algorithm to be compared}}$$

$$\text{Average CPU time} = \frac{\sum_{i=1}^{n}\text{Time}(f_i)}{n}$$

$$\text{Improvement (\%) in terms of CPU Time} = \frac{\text{Total time (basic DE algorithm)} - \text{Total time (Algorithm to be compared)}}{\text{Total NFE (basic DE algorithm)}} * 100$$

$$\text{Average SR} = \frac{\sum_{i=1}^{n}\text{SR}(f_i)}{n}$$

7.2. Performance analysis I: Comparison of MDE schemes with basic DE

Performance comparisons of MDE schemes with basic DE are given in Tables 4–7. From Table 4 which gives the average fitness function value, we can see that all the MDE schemes performed better than the basic DE for all the test problems. Particularly in case of f_1 (Rastringin function) and f_4 (Rosenbrock function), there is a significant improvement in the performance of DE using the proposed MDE3, MDE4 and MDE5 schemes. In case of f_1, there is an improvement of 97% in the function value while using MDE5 scheme. Similarly for f_4, the use of MDE3 scheme improves the function value up to 99%. For other functions also, the proposed schemes outperform the basic DE algorithm.

The superior performance of proposed schemes is more evident from Tables 5–7 which give the convergence speed, average CPU time and success rate of the proposed schemes and the basic DE. From these tables, it can be seen that there is more than 50% improvement in the convergence speed with the implementation of MDE1, MDE4 and MDE5 schemes. MDE3 scheme improves the performance by 44%. Under the present parameter settings, MDE2 scheme did not show much improvement as the improvement in convergence rate is only 0.33%. The concept of acceleration rate (AR), which again is a criterion of measuring the convergence speed of an algorithm is taken from [25]. When AR is greater than 1, then it means that the proposed algorithm is better than the basic algorithm. For all the proposed MDE schemes, the AR is greater than 1.

When we observe the CPU time given in Table 6, we can see that the average time taken by all the proposed MDE schemes to solve the given test problems is less than the time taken by DE algorithm. With MDE1 scheme, the improvement is 64% and with MDE3, MDE4 and MDE5 schemes the improvement in time is more than 50%. However with MDE2 scheme, this improvement is only 5%. If we talk about the success rate, which is given in Table 7, we can see that on an average the proposed MDE1, MDE3 MDE4 and MDE5 gives more than 80% success while MDE2 gives more than 65% success for all the test problems considered in this study. The performance curves of MDE vs. DE for all benchmark problems are shown in Fig. 1a–j.

7.3. Performance analysis II: Comparison of MDE schemes with other algorithms

While comparing the performance of proposed MDE schemes with DEPC and ODE, we changed the parameter settings of MDE schemes same as that of the algorithms to which they were compared. This was done to give an equal opportunity to all

Table 4
Average fitness function value and (standard deviation) obtained by basic DE and proposed schemes for 30 runs.

Fun.	DE	MDE1	MDE2	MDE3	MDE4	MDE5
f_1	29.9076 (1.34989)	5.87024 (2.10827)	27.7223 (7.18165)	4.97478 (1.33962)	2.78592 (0.974859)	**0.895465 (0.696468)**
f_2	6.87e−05 (9.13e−06)	**3.99e−06 (1.18e−06)**	9.45e−06 (3.70e−06)	5.41e−06 1.54e−06)	4.14e−05 (1.40e−05)	5.09e−06 (1.18e−06)
f_3	7.70e−05 (8.63e−06)	4.83e−06 (2.22e−06)	2.06491 (0.790521)	**4.08e−11 3.53e−09)**	4.82e−05 (1.17e−05)	0.017624 (0.052857)
f_4	26.3194 (1.4247)	8.98702 98.01641)	17.2028 (4.56154)	1.35307 (3.10551)	**0.334056 (8.00e−05)**	4.79998 (3.29972)
f_5	0.0177813 (0.0042194)	0.0039471 (0.00110212)	0.0761519 (0.055258)	0.0039252 (0.0007672)	**0.0031820 (0.0006822)**	0.003726 (0.000837)
f_6	−12474.7 (4.73753)	−12534 (3.58375)	−11618.2 (3.5559)	−12545.8 (2.10634)	**−12569.5 (0.00000)**	**−12569.5 (1.31e−06)**
f_7	0.0001830 (2.077e−05)	6.84e−05 (0.0001639)	**1.13e−06** (0.739362)	1.25e−05 (0.13524)	0.0001516 (2.38e−05)	1.55e−05 **(2.09e−06)**
f_8	−27.095 (0.32179)	−28.6223 (0.215474)	−27.2475 (1.29499)	−28.8925 (0.201602)	−29.1373 (0.181723)	**−29.5502 (0.028403)**
f_9	−3.28972 (0.388473)	−3.49703 (0.470752)	−3.31278 (0.012865)	**−3.78396 (0.00000)**	−3.39549 (0.475781)	−3.29837 (0.485592)
f_{10}	−186.731 (1.11e−07)	−186.731 (1.77e−08)	−186.731 **(4.01e−09)**	−186.731 (8.79e−09)	−186.731 (2.38e−07)	−186.731 (1.94e−08)

Table 5
MDE vs. DE (number of function evaluations (NFE)).

Function	DE	MDE1	MDE2	MDE3	MDE4	MDE5
f_1	250050	**34585**	250050	86375	37440	37935
f_2	57000	18935	19455	45020	**16540**	18570
f_3	175570	27005	26715	78305	**24715**	29165
f_4	250050	**178750**	197189	192980	216285	242105
f_5	250050	250050	250050	750050	250050	250050
f_6	122525	28290	31735	75425	**28025**	31460
f_7	100655	32170	**19030**	82415	28290	32450
f_8	250050	53580	**25475**	141005	53655	74385
f_9	5470	4755	5155	13490	4155	**4070**
f_{10}	18120	3950	**1610**	9545	4675	8315
$\sum$	1479540	632070	826464	1474610	663830	728505
Average NFE	147954	63207	82646.4	147461	66383	72850.5
Improvement (%) of NFE		57.27929	44.14048	0.333212	55.13268	50.76139
AR		2.340785	1.790205	1.003343	2.228794	2.030926

Table 6
CPU Time (in seconds) taken by basic de algorithm and proposed MDE schemes.

Function	DE	MDE1	MDE2	MDE3	MDE4	MDE5
f_1	42.8	**5.4**	37.3	11.8	5.7	5.9
f_2	8.6	2.8	2.9	5.7	**2.5**	2.8
f_3	28.9	4.1	4.1	11.3	**3.6**	4.3
f_4	106.9	**64.9**	66.7	146.4	88.2	87.1
f_5	37.9	**37.0**	37.1	97.4	40.2	37.8
f_6	2.3	**0.5**	0.6	1.2	0.7	0.6
f_7	13.9	6.1	3.9	15.7	**4.6**	5.2
f_8	129.1	11.9	**9.6**	61.4	13.0	18.1
f_9	0.1	0.1	0.4	0.3	0.1	0.1
f_{10}	0.1	0.01	0.01	0.1	0.01	0.1
$\sum$	370.6	132.81	162.61	351.3	158.61	162
Average Time	37.06	13.281	16.261	35.13	15.861	16.2
Improvement (%) of Time		64.16352	56.1225	5.207771	57.20183	56.2871

Table 7
MDE vs. DE (success rate (SR) (%)).

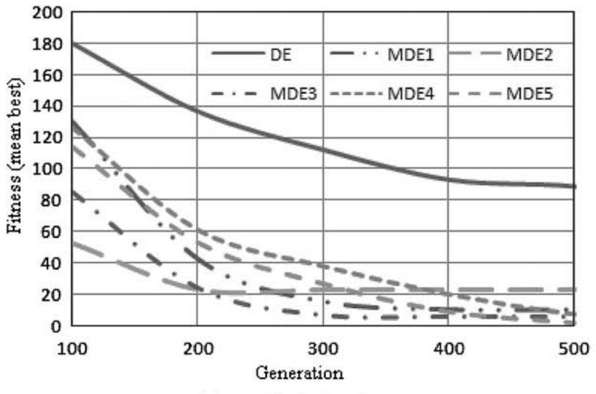
Function	DE	MDE1	MDE2	MDE3	MDE4	MDE5
f_1	–	100	–	100	100	100
f_2	100	100	100	100	100	100
f_3	100	100	100	100	100	100
f_4	–	70	30	**90**	70	10
f_5	–	–	–	–	–	–
f_6	100	100	70	100	100	100
f_7	100	100	100	100	100	100
f_8	–	100	100	100	100	100
f_9	100	100	100	100	100	100
f_{10}	70	90	70	**100**	**100**	**100**
Average SR	57	86	67	89	87	81

the algorithms. In these comparisons we have not recorded the average CPU time because it was not mentioned in the literature. The remaining performance metrics are kept same as mentioned in Section 7.2. Performance analysis of MDE schemes with ODE is given in Table 8. From this Table we can see that under the changed parameter settings, except for MDE2 scheme for function f_1 (Rastringin function) where it failed to give any result, the performance of the remaining MDE schemes is either better or at par with ODE in terms of NFE. In terms of reliability, the SR for ODE is 85% while MDE3 and MDE5 gave an average of 100% success for all the test problems that were considered. The SR of MDE1 and MDE4 was more than 95%. However, the SR of MDE2 scheme was only 75%.

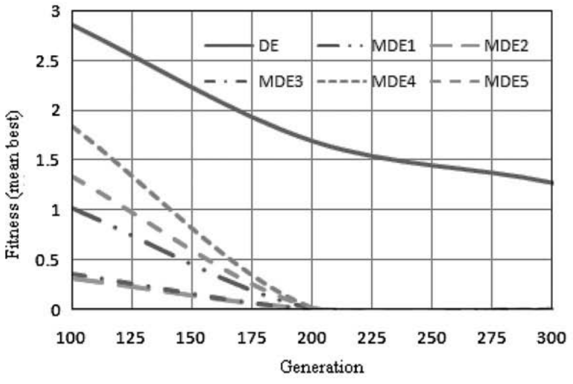
In Table 9, the performance comparison of proposed MDE schemes is given with DEPC algorithm. Once again, we changed the parameter settings of MDE schemes according to DEPC [24]. Here, we observed an interesting thing that MDE2 scheme which was giving the worst performance in previous cases started performing very well under the changed parameter settings. It gave the best results in terms of NFE for function f_1 for which it failed in previous cases. The other schemes performed more or less in a stable manner giving good results (giving an average success rate of 90%) which were again either better or at par with the DEPC algorithm. The success rate for DEPC algorithm was however 98% but this is quite expected because the parameter settings were in favor of DEPC.

8. Application of proposed MDE algorithms: Optimization of directional over-current relay settings

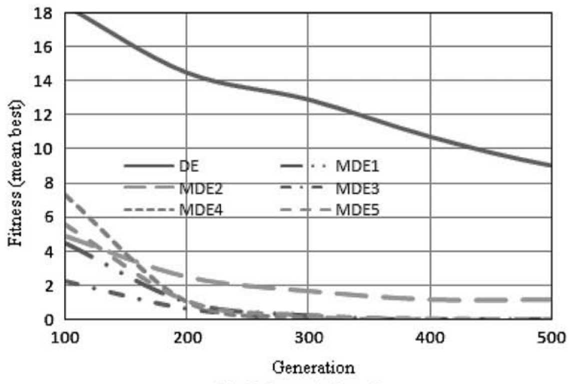
An optimization algorithm is said to be successful only if it is capable of solving real life problems, which may or may not be assisted with constraints, along with the benchmark problems. Therefore in order to check the efficiency of the proposed MDE algorithms, we tested them on an engineering design problem, optimization of directional over-current relay (DOCR) settings [26], which is an important problem in electrical engineering. The problem is modeled as a nonlinear constrained optimization problem in which the two settings namely time delay setting (TDS) and plug setting (PS) of each relay are considered as decision variables; the sum of the operating times of all the primary relays, which are expected to operate in order



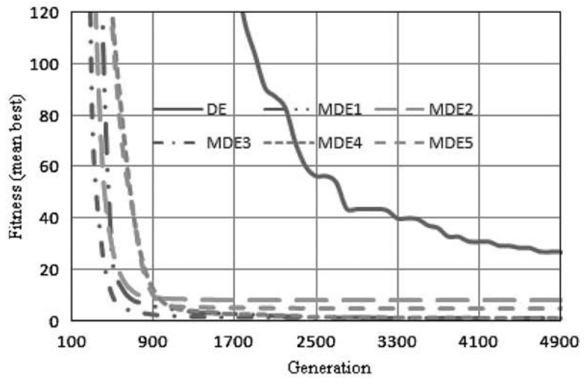
(a) Rastringin function



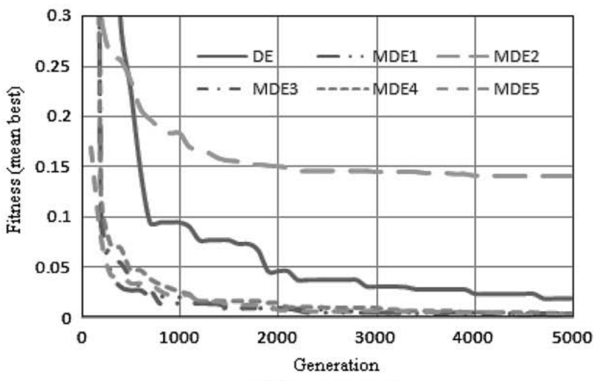
(b) Sphere function



(c) Griewank function

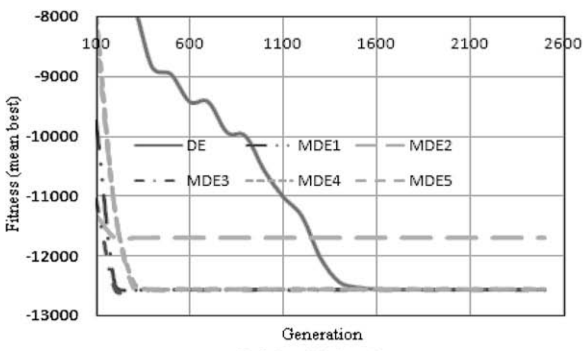


(d) Rosenbrock function

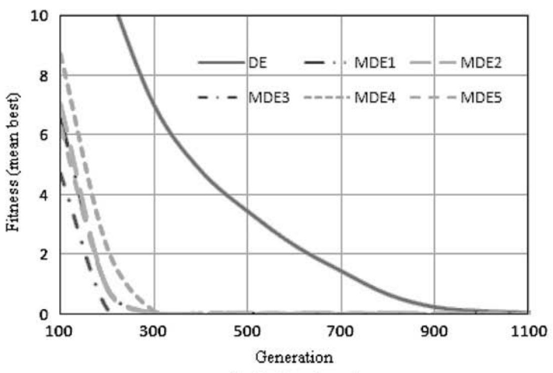


(e) Noisy function

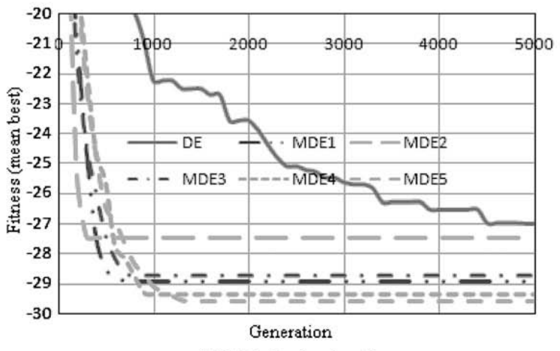
Fig. 1. (a)–(j) Performance curves of DE, MDE1, MDE2, MDE3, MDE4 and MDE5 for the given benchmark problems.



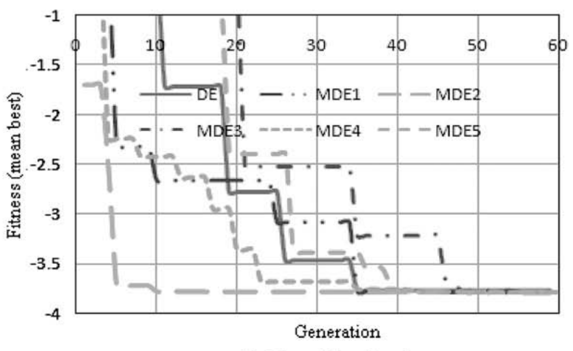
(f) Schwefel function



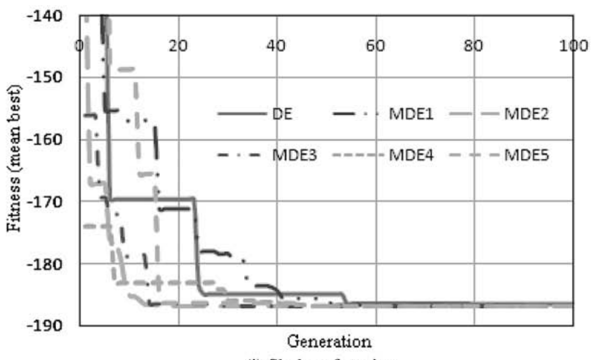
(g) Ackley function



(h) Michalewicz function



(i) Himmelblau function



(j) Shubert function

Fig. 1 (*continued*)

to clear the faults of their corresponding zones, is considered as objective function. Two models are considered namely IEEE 3-bus model and IEEE 4-bus model.

8.1. General model of the problem

The optimal coordination problem of DOCRs using optimization technique consists of minimizing an objective function (performance function) subject to certain coordination criteria and limits on problem variables. The relay, which is supposed to operate first to clear the fault, is called primary relay. A fault close to relay is known as the close-in fault for the relay and a fault at the other end of the line is known as a far-bus fault for this relay. Conventionally, objective function in coordination studies is constituted as the summation of operating times of all primary relays, responding to clear all close-in and far-bus faults. The objective function is as follows:

$$\text{Minimize } OBJ = \sum_{i=1}^{N_{cl}} T^{i}_{pri\_cl\_in} + \sum_{j=1}^{N_{far}} T^{j}_{pri\_far\_bus} \tag{10}$$

Table 8
Comparison results of MDE vs. ODE [25] (NFE, success rate (%)).

Fun	Dim	ODE		MDE1		MDE2		MDE3		MDE4		MDE5	
		NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR
f_1	10	70389	76	52866	94	–	–	70090	100	30050	100	**29793**	100
f_2	30	47716	100	46893	100	51500	100	115630	100	**43440**	100	50133	100
f_3	30	69342	96	**61146**	100	66424	100	150400	100	65860	100	65933	100
f_7	30	98296	100	**88453**	100	98460	100	219700	100	96530	100	95026	100
f_8	10	213330	56	174446	86	183256	76	**9790**	100	146487	88	15100	100

Table 9
Comparison results of MDE vs. DEPC [24] (NFE, success rate (%)).

Fun	Dim	DEPC		MDE1		MDE2		MDE3		MDE4		MDE5	
		NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR	NFE	SR
f_1	10	26927	100	25800	100	**11180**	100	60340	100	24510	100	24140	100
f_3	10	47963	100	36866	100	**18400**	100	92050	100	30490	100	36770	100
f_4	10	512165	100	**209787**	100	930950	40	1191700	70	484033	80	815130	40
f_6	10	24046	100	19120	100	**10940**	100	51550	100	21800	100	20100	100
f_7	10	29825	100	24020	100	**12690**	100	68980	100	25320	100	24930	100
f_{10}	2	1955	89	1333	100	**630**	100	3566	100	1644	100	1918	100

where, N_{cl} is number of relays responding for close-in fault; N_{far} is number of relays responding for far-bus fault; $T_{pri\_cl\_in}$ is primary relay operating time for close-in fault; $T_{pri\_far\_bus}$ is primary relay operating time for far-bus fault. The constraints are as follows:

Bounds on variables TDSs:

$$\mathrm{TDS}^i_{\min} \leqslant \mathrm{TDS}^i \leqslant \mathrm{TDS}^i_{\max}, \text{ where } i \text{ varies from 1 to } N_{cl}.$$

where $\mathrm{TDS}^i_{\min}$ is lower limit and $\mathrm{TDS}^i_{\max}$ is upper limit of TDS^i. These limits are 0.05 and 1.1, respectively.

Bounds on variables PSs

$$\mathrm{PS}^j_{\min} \leqslant \mathrm{PS}^j \leqslant \mathrm{PS}^j_{\max}, \text{ where } j \text{ varies from 1 to } Ncl.$$

where $\mathrm{PS}^j_{\min}$ is lower limit and $\mathrm{PS}^j_{\max}$ is upper limit of PS^j. These are 1.25 and 1.50, respectively.

Limits on primary operation times:

This constraint imposes constraint on each term of objective function to lie between 0.05 and 1.0.

Selectivity constraints for all relay pairs:

$$T_{\text{backup}} - T_{\text{primary}} - \mathrm{CTI} \geqslant 0$$

where T_{backup} is operating time of backup relay, T_{primary} is operating time of primary relay and CTI is coordinating time interval.

8.2. Model I: IEEE 3-bus model

For the coordination problem of IEEE 3-bus model, value of each of N_{cl} and N_{far} is 6 (equal to number of relays or twice the lines). Accordingly, there are 12 decision variables (two for each relay) in this problem i.e. TDS^1 to TDS^6 and PS^1 to PS^6.

Objective function (*OBJ*) to be minimized as given by:

$$OBJ = \sum_{i=1}^{6} T^i_{pri\_cl\_in} + \sum_{j=1}^{6} T^j_{pri\_far\_bus} \tag{11}$$

where

$$T^i_{pri\_cl\_in} = \frac{0.14 \times TDS^i}{\left(\frac{a^i}{PS^i \times b^i}\right)^{0.02} - 1} \tag{12}$$

$$T^i_{pri\_far\_bus} = \frac{0.14 \times TDS^j}{\left(\frac{c^i}{PS^j \times d^i}\right)^{0.02} - 1} \tag{13}$$

The values of constants a^i, b^i, c^i and d^i are given in the Table 10.

Table 10
Values of constants a^i, b^i, c^i and d^i for Model I.

$T^i_{pri\_cl\_in}$			$T^i_{pri\_far\_bus}$		
TDS^i	a^i	b^i	TDS^j	c^i	d^i
TDS^1	9.46	2.06	TDS^2	100.63	2.06
TDS^2	26.91	2.06	TDS^1	14.08	2.06
TDS^3	8.81	2.23	TDS^4	136.23	2.23
TDS^4	37.68	2.23	TDS^3	12.07	2.23
TDS^5	17.93	0.8	TDS^6	19.2	0.8
TDS^6	14.35	0.8	TDS^5	25.9	0.8

Constraints for the model are:
Bounds on variables TDSs:

$$\mathrm{TDS}^i_{\min} \leqslant \mathrm{TDS}^i \leqslant \mathrm{TDS}^i_{\max}, \quad \text{where, } i \text{ varies from 1 to 6 } (N_{cl})$$

Bounds on variables PSs:

$$\mathrm{PS}^j_{\min} \leqslant \mathrm{PS}^j \leqslant \mathrm{PS}^j_{\max}, \quad \text{where, } j \text{ varies from 1 to 6 } (N_{cl})$$

Limits on primary operation times:
This constraint imposes constraint on each term of objective function to lie between 0.05 and 1.0.
Selectivity constraints are:

$$T^i_{backup} - T^i_{primary} - \mathrm{CTI} \geqslant 0 \tag{14}$$

T_{backup} is operating time of backup relay and $T_{primary}$ is operating time of primary relay. Value of *CTI* is 0.3. Here,

$$T^i_{backup} = \frac{0.14 \times TDS^p}{\left(\frac{e^i}{PS^p \times f^i}\right)^{0.02} - 1} \tag{15}$$

$$T^i_{primary} = \frac{0.14 \times TDS^q}{\left(\frac{g^i}{PS^q \times h^i}\right)^{0.02} - 1} \tag{16}$$

Table 11
Values of constants e^i, f^i, g^i and h^i for Model I.

T^i_{backup}			$T^i_{primary}$		
p	e^i	f^i	q	g^i	h^i
5	14.08	0.8	1	14.08	2.06
6	12.07	0.8	3	12.07	2.23
4	25.9	2.23	5	25.9	0.8
2	14.35	0.8	6	14.35	2.06
5	9.46	0.8	1	9.46	2.06
6	8.81	0.8	3	8.81	2.23
2	19.2	2.06	6	19.2	0.8
4	17.93	2.23	5	17.93	0.8

Table 12
Values of constants a^i, b^i, c^i and d^i for Model II.

$T^i_{pri\_cl\_in}$			$T^i_{pri\_far\_bus}$		
TDS^i	a^i	b^i	TDS^j	c^i	d^i
TDS^1	20.32	0.48	TDS^2	23.75	0.48
TDS^2	88.85	0.48	TDS^1	12.48	0.48
TDS^3	13.61	1.1789	TDS^4	31.92	1.1789
TDS^4	116.81	1.1789	TDS^3	10.38	1.1789
TDS^5	116.7	1.5259	TDS^6	12.07	1.5259
TDS^6	16.67	1.5259	TDS^5	31.92	1.5259
TDS^7	71.7	1.2018	TDS^8	11	1.2018
TDS^8	19.27	1.2018	TDS^7	18.91	1.2018

Table 13
Values of constants e^i, f^i, g^i and h^i for Model II.

T^i_{backup}			$T^i_{primary}$		
p	e^i	f^i	q	g^i	h^i
5	20.32	1.5259	1	20.32	0.48
5	12.48	1.5259	1	12.48	0.48
7	13.61	1.2018	3	13.61	1.1789
7	10.38	1.2018	3	10.38	1.1789
1	1.16	0.48	4	116.81	1.1789
2	12.07	0.48	6	12.07	1.1789
2	16.67	0.48	6	16.67	1.5259
4	11	1.1789	8	11	1.2018
4	19.27	1.1789	8	19.27	1.2018

Table 14
Comparison results of IEEE 3-bus, 4-bus and 6-bus models: interms of objective function value (OBJ) and NFE.

Algorithm	IEEE 3-bus model		IEEE 4-bus model	
	OBJ	NFE	OBJ	NFE
DE	4.84218	78360	3.67744	95400
MDE1	4.80699	72350	3.66945	43400
MDE2	4.78728	73350	3.67349	67200
MDE3	4.78227	97550	3.66925	99700
MDE4	**4.78067**	69270	**3.66749**	55100
MDE5	4.78068	**38250**	3.66941	**35330**

Table 15
Improvement(%) of modified DE algorithms in comaprison with DE interms of objective function values.

Algorithm	IEEE 3-bus	IEEE 4-bus
MDE1	0.726739	0.217271
MDE2	1.133787	0.107412
MDE3	1.237253	0.222709
MDE4	1.270296	0.270569
MDE5	1.270089	0.218358

The values of constants e^i, f^i, g^i and h^i are given in the Table 11.

8.3. Model II: IEEE 4-bus model

For coordination problem of IEEE 4-bus model, value of each of N_{cl} and N_{far} is 8 (equal to number of relays or twice the lines). Accordingly, there are 16 decision variables (two for each relay) in this problem i.e. TDS^1 to TDS^8 and PS^1 to PS^8. The value of for this model is 0.3. The number of selectivity constraints is 9.

The objective function and constraints for this model will be of same form as in the case of Model I problem (with $N_{cl} = 8$) described in Section 8.2. The values of constants a^i, b^i, c^i, d^i and e^i, f^i, g^i, h^i for Model II are given in Tables 12 and 13, respectively.

8.4. Results and discussion

Parameter settings for all the real life problems are kept same as that of the test functions. Constraint handling approach based on repair methods [27] is used for handling constraints. The best solution obtained by DE and modified DE algorithms of IEEE 3-bus model and IEEE 4-bus model in terms of objective function value and number of function evaluations are given in Table 14. From the numerical results, we can see that MDE4 gave better result than the other algorithms in terms of objective function value for both the models. On the other hand, if we compare the NFE, then the performance of MDE5 is better than all other compared algorithms. Also, from the numerical results of Table 14 we can see that all the modified versions of DE outperform the basic DE algorithm by a significant difference. In Table 15, we have given the improvement (%) of modified DE algorithms in comparison with basic DE.

9. Conclusions

In the present study we proposed five new mutation schemes for the basic DE algorithm. The two main differences between the basic DE mutation operation and the proposed schemes are (i) use of absolute difference between the two points rather than simple vector difference between the points and (ii) use of Laplace distribution for scaling factor instead of having a predefined value. The performance of the proposed schemes is validated on a set of 10 test problems and the numerical results are compared with basic DE and two other versions of DE. The numerical results show that the proposed schemes help in improving the convergence rate up to 50% in comparison to the basic DE and at the same time maintain a good SR as well. Also it was observed that out of the five proposed schemes MDE2 was most sensitive to the parameter settings as its performance changed drastically when the parameter settings were changed. However the remaining four schemes performed more or less in a stable manner giving good performance even when the parameter settings were changed according to the algorithms to which they were being compared (i.e. DEPC and ODE). The efficiency of MDE algorithms were further tested by applying them on a real life problem, optimization of directional over-current relay (DOCR) settings, which is an important problem in electrical engineering. This problem was modeled as a nonlinear constrained optimization problem; the constraints were dealt with the constraint handling mechanism based on repair methods. Numerical results of real as well as test problems show the robustness and efficiency of proposed MDE algorithms.

References

[1] T. Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Inst. Phys. and Oxford Univ. Press, New York, 1997.
[2] J. Zhang, J. Xu, Q. Zhou, A new differential evolution for constrained optimization problems, in: Proceedings of the Sixth Int. Conf. on Intelligent Systems, Design and Applications, 2006, pp. 1018–1023.
[3] M. Pant, R. Thangaraj, A. Abraham, Optimization of a kraft pulping system: using particle swarm optimization and differential evolution, in: Proceedings of Second Asia Int. Conf. on Modeling and Simulation, Malaysia, IEEE Computer Society Press, USA, 2008, pp. 637–641.
[4] A. Abbasy, S.H. Hosseini, A novel multi-agent evolutionary programming algorithm for economic dispatch problems with non-smooth cost functions, in: Proceedings of IEEE Power Engineering Society General Meeting, 2007, pp. 1–7.
[5] M. Pant, R. Thangaraj, V.P. Singh, Efficiency optimization of electric motors: a comparative study of stochastic algorithms, World Journal of Modeling and Simulation 4 (2008) 140–148.
[6] E. Cao, M. Lai, An improved differential evolution algorithm for the vehicle routing problem with simultaneous delivery and pick-up service, Proceedings of Third International Conference on Natural Computation 3 (2007) 436–440.
[7] R. Storn, K. Price, Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report, International Computer Science Institute, Berkley, 1995.
[8] R. Storn, K. Price, Differential evolution – a simple and efficient Heuristic for global optimization over continuous spaces, Journal Global Optimization 11 (1997) 341–359.
[9] R. Stom, System design by constraint adaptation and differential evolution, IEEE Transactions on Evolutionary Computation 3 (1999) 22–34.
[10] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, John Wiley and Sons Ltd., 2005.
[11] D. Karaboga, S. Okdem, A simple and global optimization algorithm for engineering problems: differential evolution algorithm, Turk J. Elec. Engin. 12 (2004) 53–60.
[12] J. Lampinen, I. Zelinka, On stagnation of the differential evolution algorithm, in: Pavel Ošmera (Ed.), Proceedings of MENDEL 2000, Sixth International Mendel Conference on Soft Computing, Brno, Czech Republic, June 7–9, 2000, pp. 76–83.
[13] D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, in: D. Matousek, P. Osmera (Eds.), Proceedings of MENDEL 2003, Ninth International Conference on Soft Computing, Brno, Czech Republic, June 2003, pp. 41–46.
[14] H. Abbass, The self-adaptive Pareto differential evolution algorithm, in: Proceedings of the 2002 Congress on Evolutionary Computation, 2002, pp. 831–836.
[15] M. Omran, A. Salman, A.P. Engelbrecht, Self-adaptive differential evolution, computational intelligence and security, PT 1, in: Proceedings Lecture Notes in Artificial Intelligence 3801, 2005, pp. 192–199.
[16] J. Brest, S. Greiner, B. Boškovic, M. Mernik, V. Žumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation 10 (2006) 646–657.
[17] S. Das, A. Konar, U.K. Chakraborty, Two improved differential evolution schemes for faster global search, in: ACM-SIGEVO Proceedings of GECCO, Washington, DC, June 2005, pp. 991–998.
[18] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution, IEEE Transactions on Evolutionary Computation 12 (2008) 64–79.
[19] Z. Yang, J. He, X. Yao, Making a difference to differential evolution, in: Z. Michalewicz, P. Siarry (Eds.), Advances in Metaheuristics for Hard Optimization, Springer, 2007, pp. 415–432.
[20] N. Noman, H. Iba, Enhancing differential evolution performance with local search for high dimensional function optimization, in: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, June 2005, pp. 967–974.
[21] U.K. Chakraborty (Ed.), Advances in Differential Evolution, Springer-Verlag, Heidelberg, 2008.
[22] M. Pant, R. Thangaraj, A. Abraham, C. Grosan, Differential evolution with laplace mutation operator, in: Proceedings of IEEE Congress on Evolutionary Computation, 2009, pp. 2841–2849.
[23] <http://www.icsi.Berkley.edu/~storn/code.html>.
[24] M.M. Ali, Differential evolution with preferential crossover, European Journal of Operational Research 181 (2007) 1137–1147.
[25] S. Rahnamayan, H.R. Tizhoosh, M.A. Salama, Opposition-based differential evolution, IEEE Transactions on Evolutionary Computation 12 (2008) 64–79.
[26] K. Deep, D. Birla, R.P. Maheshwari, H.O. Gupta, M. Takur, A population based heuristic algorithm for optimal relay operating time, World Journal of Modeling and Simulation 3 (2006) 167–176.
[27] M. Pant, R. Thangaraj, V.P. Singh, Optimization of mechanical design problems using improved differential evolution algorithm, International Journal of Recent Trends in Engineering 1 (2009) 21–25.