Soft Comput (2011) 15:991–1007
DOI 10.1007/s00500-010-0655-2

ORIGINAL PAPER

Improving the performance of differential evolution algorithm using Cauchy mutation

Musrrat Ali · Millie Pant

Published online: 28 September 2010

Abstract Differential evolution (DE) is a powerful yet simple evolutionary algorithm for optimization of real-valued, multimodal functions. DE is generally considered as a reliable, accurate and robust optimization technique. However, the algorithm suffers from premature convergence and/or slow convergence rate resulting in poor solution quality and/or larger number of function evaluation resulting in large CPU time for optimizing the computationally expensive objective functions. Therefore, an attempt to speed up DE is considered necessary. This research introduces a modified differential evolution (MDE) that enhances the convergence rate without compromising with the solution quality. The proposed MDE algorithm maintains a *failure_counter* (*FC*) to keep a tab on the performance of the algorithm by scanning or monitoring the individuals. Finally, the individuals that fail to show any improvement in the function value for a successive number of generations are subject to Cauchy mutation with the hope of pulling them out of a local attractor which may be the cause of their deteriorating performance. The performance of proposed MDE is investigated on a comprehensive set of 15 standard benchmark problems with varying degrees of complexities and 7 nontraditional problems suggested in the special session of CEC2008. Numerical results and statistical analysis show that the proposed modifications help in locating the global optimal solution in lesser numbers of function evaluation in comparison with basic DE and several other contemporary optimization algorithms.

Keywords Differential evolution · Cauchy mutation · Global optimization

1 Introduction

Differential evolution (DE) was proposed by Storn and Price (1995). It soon became a popular tool for solving global optimization problems because of several attractive features like having fewer control parameters, ease in programming, efficiency, etc. DE is similar to genetic algorithms (GA) in the sense that it uses same evolutionary operators like mutation, crossover, and selection for guiding the population towards the optimum solution. Nevertheless, it is the application of these operators that makes DE different from GA. The main difference between GA and DE is that in GAs, mutation is the result of small perturbations to the genes of an individual while in DE mutation is the result of arithmetic combinations of individuals. DE has been successfully applied to solve a wide range of real-life application problems like image classification, IIR filter design etc. (Shih and Edupuganti 2009; Omran et al. 2005a; Storn 1995) and has reportedly outperformed several other optimization techniques (Vesterstroem and Thomsen 2004; Andre et al. 2001; Hrstka and Kucerová 2004).

Despite having quite a few positive features, it has been observed that DE sometimes does not perform as good as the expectations. It was pointed out by Lampinen and Zelinka (2000), that DE may occasionally stop proceeding towards the global optimum even though the population has not converged to a local optimum or any other point.

M. Ali (✉) · M. Pant
Department of Paper Technology, Indian Institute of Technology Roorkee, Roorkee 247667, India
e-mail: musrrat.iitr@gmail.com

M. Pant
e-mail: millifpt@iitr.ernet.in

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 many other evolutionary algorithms (EA), the performance of basic DE deteriorates with the increase in dimensionality of the objective function. It is therefore necessary to assist DE (or for that matter any EA, in its basic form) with suitable mechanism which will help in improving its performance. Several modifications have been suggested in literature to enhance the structure of basic DE. A state-of-the-art review of some of the earlier work done related to DE is given in Sect. 3.

In the present study we propose two modifications in the basic scheme of DE. The first modification is the introduction of the concept of *failure_counter* (*FC*). The work of *FC* is to scan the performance of individuals in every generation and to keep an account of the number of times an individual fails to show any improvement in the fitness function value. The presences of such individuals which do not show any progress in successive generations hamper the working of DE algorithm. In order to avoid such a situation, we suggest the second modification which is the application of Cauchy mutation. Use of Cauchy mutation perturbs such individuals and forces them to move to some other location thereby providing them an opportunity to improve their performance. These two changes when combined together help the DE individuals in escaping the local attractor and also help in faster convergence.

The usefulness of these two modifications is empirically investigated. For this purpose we have considered a set of 15 standard benchmark problems that are generally used for validating the performance of an optimization algorithm. Also have considered seven nontraditional shifted functions proposed in the special session of CEC2008. We conducted several experiments to verify the robustness and efficiency of the proposed algorithm. Specifically we investigated (1) the convergence speed and robustness; (2) the effect of dimensionality; (3) effect of Cauchy mutation and settings of *failure_counter*; (4) comparison of modified differential evolution (MDE) with basic DE, BBDE, DEahcSPX, SACDE, SFMDE, GA, PSO, ES, and CLPSO. Furthermore, the performance of MDE is analyzed statistically using tests like Wilcoxon test, paired *t* test, Bonferroni Dunn test, etc.

The remaining of the paper is organised as follows: in Sect. 2, we give a brief description of DE. In Sect. 3, literature review related to the earlier works done in DE is given. In Sect. 4, we describe the proposed MDE version. In Sect. 5, experimental settings and numerical results are given. Finally, the conclusions based on the present study are drawn in Sect. 6. Benchmark problems taken for study are given in the "Appendix".

2 Basic DE

Storn and Price proposed ten variants of the DE-algorithm (Price 1999). Throughout the present study we shall follow the version *DE/rand/1/bin*, which is apparently the most commonly used version, and we shall refer to it as basic version.

This particular scheme is briefly described as follows:

DE starts with a population of *NP* candidate solutions which may be represented as $X_{i,G}$, $i = 1, \ldots, NP$, where i index denotes the ith individual of the population and G denotes the generation to which the population belongs. The working of DE depends on the manipulation and efficiency of three main operators mutation, reproduction, and selection, which are briefly described in this section.

2.1 Mutation

The mutation operation of DE applies the vector differentials between the existing population members for determining both the degree and direction of perturbation applied to the individual subject of the mutation operation. The mutation process at each generation begins by randomly selecting three individuals in the population. The ith perturbed individual, $V_{i,G+1}$, is then generated based on the three chosen individuals as follows:

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

where $i = 1, \ldots, NP$, $r_1, r_2, r_3 \in \{1, \ldots, NP\}$ are randomly selected and satisfy: $r_1 \neq r_2 \neq r_3 \neq i$, $F \in [0, 1]$, where F is the control parameter proposed by Storn and Price (1995).

2.2 Crossover

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

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

where $j = 1, \ldots, n$, $k \in \{1, \ldots, n\}$ is a random parameter's index, chosen once for each i, and the crossover rate, $Cr \in [0, 1]$, the other control parameter of DE, is set by the user.

2.3 Selection

The population for the next generation is selected from the individual in current population and its corresponding trial vector according to the following rule:

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

Thus, each individual of the temporary (trial) population is compared with its counterpart in the current population. The one with the lower objective function value will survive from the tournament selection to the population of the next generation. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. In DE trial vector is not compared against all the individuals in the current generation, but only against one individual, its counterpart, in the current generation.

3 A brief literature review of previous work related to DE

The simplicity and effectiveness of DE has fascinated the researchers working in the field of evolutionary algorithms, numerical optimization, etc., over the past few years. Several attempts have been made to improve its ultimate performance. Investigations have been made to determine the optimum settings of control parameters of DE viz. population size, crossover rate *Cr,* and scaling factor *F*. Storn and Price indicated that a reasonable population size could be between 5*D* and 10*D*, where *D* denotes the dimensionality of the problem. They also recommended that a good initial choice of *F* can be 0.5. Gamperle et al. (2002) suggested a population size between 3*D* and 8*D*, a scaling factor $F = 0.6$, and the crossover rate *Cr* in the range of [0.3, 0.9] for best results. Ronkkonen et al. (2005) on the other hand claimed that *F* should vary in the range of [0.4, 0.95] while *Cr* should be provided with a value according to the nature of the function.

Thus, it can be seen that there are contradictory views regarding the parameter settings and no setting can be claimed as optimum. Consequently, 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 (2005) introduced fuzzy adaptive differential evolution (FADE) using fuzzy logic controllers. Qin et al. (2009) proposed a self-adaptive DE (SaDE) algorithm, in which both 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 multipopulation approach. Later, Zaharie and Petcu (2003) designed an adaptive Pareto DE algorithm for multiobjective optimization and analyzed its parallel implementation. Abbass (2002) self-adapted the crossover rate *Cr* for multiobjective 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. (2005b) 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 SADE 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 a self-adaptive DE.

Other class of modification in DE involves hybridization of DE with some other technique. Yang et al. (2008a) proposed a hybridization of DE with the neighborhood search (NS) and called their algorithm "NSDE" in which mutation is performed by adding a normally distributed random value to each target-vector component. Recently, Yang et al. (2008b) used a self-adaptive NSDE in the cooperative coevolution framework that is capable of optimizing large-scale nonseparable problems (up to 1,000 dimensions). Hendtlass (2001) used the DE perturbation approach to adapt particle positions. Particle positions are updated only if their offspring have better fitness. Zhang and Xie (2003) and Talbi and Batouche (2004) used the DE operator to provide mutations. Kannan et al. (2004) applied DE to each particle 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. Xu 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 namely PSO, Rosenbrock Algorithm, and Nelder-Mead Algorithm.

Some other modifications in DE include development of new mutation operators for DE (Fan and Lampinen 2003; Pant et al. 2009), use of opposition-based learning for generating the initial population (Rahnamayan et al. 2008), crossover-based local search method for DE (Noman and Iba 2005, 2008).

4 Modified differential evolution

Differential evolution algorithm described in Sect. 2 typically converges at a rapid pace in the initial stages of the search procedure and then gradually slows down as it approaches global optimum. The performance of DE can easily be viewed by observing the fitness function value. If there is an improvement in fitness in successive generations, it then signifies that the new trial point generated by DE is better than target vector. Consequently, the new trial point replaces the target vector in next generation. However, in case there is no improvement in fitness of an individual, then it is an indication that the particles are clustered together in a region and their vector difference (second term of Eq. 1) is either zero or very insignificant to allow any improvement of function value. Also, it may be pointed out here that the region in which the particles are clustered around may or may not be a local attractor basin. In such a case it is necessary to introduce a perturbation in the population which will help the individuals to move to a new location.

The focus of this research is to introduce a mechanism which will not only keep a track of the progress of individuals but will also help the individuals in escaping the local basin by allowing them to jump to a new region.

In order to keep a record of the success of individuals, in the present study, we introduce a concept of '*failure_counte*' (*FC*). The work of *FC* is to monitor the working of individuals in terms of fitness function value for a specified number of generations. If there is no improvement in fitness, then *FC* is increased by unity in each generation. This process is repeated until we achieve user-defined value of *maximum failure counter* (*MFC*). Once *MFC* is attained, it is an indication that perturbation in the population is needed which will allow the individual to jump to a new position. In order to achieve this, we employed Cauchy mutation (Stacey et al. 2003) for which the probability density function (PDF) is given by the following equation:

$$f(x;x_0,\gamma) = \frac{1}{\pi\gamma\left[1+\left(\frac{x-x_0}{\gamma}\right)^2\right]} = \frac{1}{\pi}\left[\frac{\gamma}{(x-x_0)^2+\gamma^2}\right] \tag{4}$$

where x_0 is the location parameter, specifying the location of the peak of the distribution, and γ is the scale parameter which specifies the half-width at half-maximum. The graph of PDF for different values of x_0 and γ is shown in Fig. 1. From this figure, we can see that for large values of γ we get a fat tail curve, whereas for smaller values of γ, the shape of the curve changes towards a sharper peak. In the present study we have taken γ as 0.1, which will produce a very sharp peak, resulting in a small area around the mean.

A new trial vector $U_{i,G+1} = (u_{1,i,G+1}, \ldots, u_{n,i,G+1})$ by MDE is generated as follows:

$$u_{j,i.G+1} = \begin{cases} X_{j,\text{best},G} + \mathrm{C}(\gamma,0) & \text{if } rand(0,1) \le 0.9 \\ x_{j,i.G} & \text{otherwise} \end{cases} \tag{5}$$

where C (γ, *0*) stands for random number generated by Cauchy probability distribution with scale parameter γ and centred at origin. After generation of a new point, selection process, similar to that of basic DE is used.

This modification enables the algorithm to get a better trade off between the convergence rate and robustness. Thus, it is possible to increase the convergence rate of the differential evolution algorithm and thereby obtain an acceptable solution with a lower number of objective function evaluations. Such an improvement can be advantageous in many real-world problems where the evaluation of a candidate solution is a computationally expensive operation and consequently finding the global optimum or a good suboptimal solution with the original differential evolution algorithm is too time-consuming or even impossible within the time available.

As mentioned earlier, the basic structure of MDE is same as DE except for the two major differences: first, it is the use of a failure counter to avoid a possible stagnation of

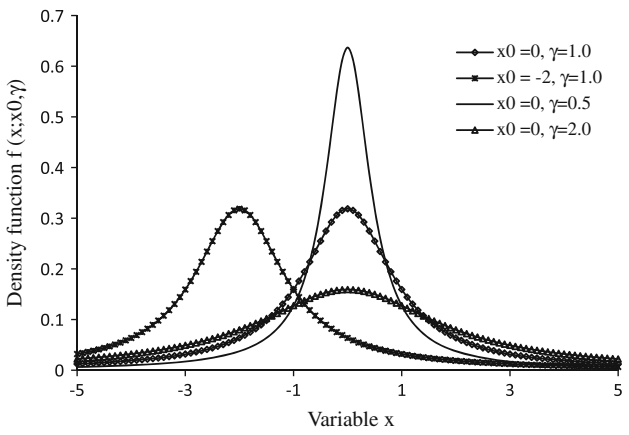

Fig. 1 Probability density function of Cauchy distribution for different values of γ and x_0

performance of an individual and second, the use of two mutation operators depending on the failure counter; basic DE mutation, and Cauchy mutation. The working procedure of the algorithm is outlined below:

Step 1: Generate randomly NP vectors, each of n dimensions, $x_{i,j} = x_{\min,j} + rand(0, 1)(x_{\max,j} - x_{\min,j})$, where $x_{\min,j}$ and $x_{\max,j}$ are lower and upper bound for jth component, respectively, *rand(0, 1)* uniform random number between 0 and 1. Calculate the objective function value $f_i = f(X_i)$ for all X_i. Set the values of control parameters F, Cr, and MFC. Set *failure_counter[i] = 0.*
Step 2: Set $i = 0$.
Step 3: $i = i + 1$.
Step 4: If *failure_counter[i]* < MFC, then go to step 5; otherwise, go to step 7.
Step 5: Mutation: Select three points from population and generate perturbed individual V_i using Eq. 1.
Step 6: Crossover: Recombine the each target vector X_i with perturbed individual generated in step 5 to generate trial vector U_i using Eq. 2 and go to step 8.
Step 7: Generate a trial vector using Eq. 5 and go to step 8.
Step 8: Selection: Calculate the objective function value for vector U_i. Choose better of the two (function value at target and trial point) using Eq. 3 for next generation if trial vector is selected for next generation, then *failure_counter[i] = 0;* otherwise, increase the value of *failure_counter[i]* by one.
Step 9: If $i < NP$, then go to step 3; otherwise, go to step 10.
Step 10: Check whether the termination criterion met. If yes then stop; otherwise, go to step 2.

4.1 Application of Cauchy mutation

Gaussian and Cauchy are two popular distributions used for mutating the particles in an evolutionary algorithm. Studies have shown that while applying Gaussian distribution, large mutation steps are very unlikely. This may reduce the convergence speed of a search algorithm and may also increase its risk getting caught in a local minimum. Cauchy distribution, on the other hand, is a more slowly decaying distribution in comparison with Gaussian and allows even the large mutation step sizes. Cauchy distribution has reportedly outperformed Gaussian distribution in several instances available in literature (Lan and Lan 2008; Rudolph 1997; Yao et al. 1999; Birru et al. 1999; Wang et al. 2006; Coelho and Krohling 2003).

5 Parameter settings and numerical results

5.1 Parameter settings

The three main parameters of basic DE are population size (*NP*), crossover rate *Cr,* and amplitude factor or scaling factor *F*. With DE, the lower limit for *NP* is 4 since the mutation process requires at least three other chromosomes for each parent. As a general rule, an effective *NP* is between $3 \times n$ and $10 \times n$, but can often be reduced or increased depending on the complexity of the problem. For the present study we performed various experiments with the population size as well as with the crossover rate and mutation probability rate and observed that for problems up to dimension 30 a population size of $10 \times n$ is sufficient. Values of scaling factor, *F,* outside the range of 0.4–1.2 are rarely effective; therefore, $F = 0.5$ may be considered a good initial choice. In general, *Cr* should be sufficiently large to speed up the convergence, so in this study we have taken $Cr = 0.5$. The parameter settings used in the present study are summarized in Table 1. These settings are used throughout the paper unless otherwise mentioned.

All the algorithms are executed on a *PIV PC*, using *DEV C++,* thirty times for each problem. Random numbers are generated using the inbuilt random number generator *rand () function* available in DEV C++. In order to have a fair comparison, the parameter settings are kept the same for all algorithms over all benchmark functions during the simulations.

For MDE the additional variable MFC is assigned a value five, i.e., if the fitness of an individual is not improved even after five generations then Cauchy mutation is applied with the hope of getting a better value. The scale parameter γ in Cauchy distribution is taken as 0.1. The nonparametric statistical tests are conducted using the software package SPSS.

5.2 Performance measures

In order to validate the performance of the proposed MDE algorithm we conducted a series of experiments following

Table 1 Parameter settings for DE and MDE

Parameter	Value
Population size *NP*[a]	$10 \times n$
Scaling factor *F*	0.5
Crossover probability *Cr*	0.5
Maximum failure counter MFC	5.0
Scale parameter γ	0.1

[a] For non traditional problems *NP* is taken as 500

various criteria to test its efficiency, robustness, and reliability. These criteria have been widely used to analyze the performance of an algorithm and are listed in the following subsections.

5.2.1 Function evaluation

It is one of the most common metric used for evaluating the performance of an algorithm. The number of function evaluations (NFEs) required for reaching an error value less than ε (provided that the maximum limit of number of function evaluation is $10{,}000 \times n$ NFEs) is recorded in different runs and then average and standard deviation of the number of evaluations are calculated. In this study ε is set to 10^{-04} for all the functions while it is kept as 10^{-06} when MDE is compared with DEahcSPX.

On the basis of NFE we also calculated the improvement in average NFE for each problem.

5.2.2 Error

It is another common criterion used for measuring the reliability of the algorithm. The minimum function error value that an algorithm can find, using a predefined maximum NFE, is recorded in each run and then average and standard deviation of the error values are calculated. The maximum NFE is fixed at 200,000 when DE and MDE are compared on traditional benchmark problems while it is set as 2.5×10^6 in case of nontraditional functions. To compare MDE with other algorithms maximum NFE is set according to relevant literatures.

The function error value, used for performance measures defined in 5.2.1 and 5.2.2, for an obtained solution x is defined as $|f(x) - f(x^*)| = \epsilon$ where x^* is the global optimum of the function.

5.2.3 CPU time

The average CPU time taken by the algorithm before some stopping criteria is met.

5.2.4 Statistical analysis

In order to further analyze the performance of the proposed MDE algorithm, we compared it statistically with other algorithms. For this purpose, we used nonparametric tests to check if there is any significant difference (at 5% level of significance) between the two algorithms. This analysis is done for multiple problems.

For the comparison of two algorithms, the most commonly used statistical criterion is paired t test. However, the application of paired t test requires the fulfilment of the following three conditions:

(i) The data in the set should be independent
(ii) The sample should be normally distributed
(iii) Variance of the samples should be equal

If the above three conditions are not satisfied, then the paired t-test may give a wrong conclusion. In such a situation a nonparametric test (like that of Wilcoxon test) is more suitable because it does not require any specific condition for its application. The normality of the sample can be checked by applying the Kolmogorov–Smirnov and Shapiro–Wilk tests. If the p value obtained by these tests is <0.05 at 5% level of significance, then the sample is not normally distributed. To check the equality of variance of two samples, Leneve's test may be applied, and once again, if the p value is <0.05, then the variance is not equal at 5% level of significance.

A detailed study on the use of nonparametric tests is given in García et al. (2009a, b), where the authors performed these tests on a set of benchmark problems taken from CEC2005 to analyze the performance of genetics-based machine learning.

5.2.5 Convergence graphs

The performance of MDE is also illustrated graphically with the help of convergence graphs (or performance curves) where fitness function values are plotted against the NFE corresponding to minimum error in all runs.

5.3 Analyses of parameter maximum failure counter

MDE has an additional parameter called maximum failure counter or MFC for which the value is given by the user. We conducted a series of experiments to determine an appropriate value for MFC, and it was observed that very high and very low values for MFC are rarely effective. For example, if MFC is taken as 1, then it is more or less like applying mutation after every iteration. On the other hand, keeping MFC very high will naturally result in higher NFE. In the present study, considering the brevity of space we recorded the NFE obtained for three values of MFC viz. 3, 5, 10 to achieve an accuracy of 10^{-4} for the 15 benchmark problems taken in the present study. The corresponding results are summarized in Table 2.

From the table it can be seen that except for f_{RG}, for which none of the cases were able to achieve the desired accuracy, the values of MFC can be set in the range of 5–10.

5.4 Numerical results

In this section we give the numerical results obtained by MDE for solving the problems listed in the Appendix and

Table 2 Number of function evaluations (NFC) attained to achieve an accuracy of 10^{-4} for different values of maximum failure counter (MFC)

Function	Dim.	NFE			
		DE	MDE		
			MFC = 3	MFC = 5	MFC = 10
f_{EP}	2	833	620	670	672
f_{FX}	2	977	580	482	628
f_{CB6}	2	1,020	960	712	986
f_{GP}	2	970	950	988	948
f_{H3}	3	1,170	1,290	1,284	1,260
f_{SP}	30	206,400	157,080	153,570	135,060
f_{ACK}	30	259,410	261,870	232,740	231,900
f_{SWF}	30	366,570	140,100	130,800	135,120
f_{GW}	30	224,910	142,860	133,440	140,250
f_{LM2}	30	182,700	143,580	133,770	134,370
f_{ST}	30	124,800	62,160	58,800	62,970
f_{RB}	30	3,066,300	4,160,700	3,603,000	3,834,530
f_{RG}	30	–	–	–	–
$f_{SWF2.22}$	30	249,960	276,840	245,070	232,020
f_{RHE}	30	232,200	210,900	200,400	202,800

compare it with other algorithms using the evaluation criteria given in Sect. 5.2.

5.4.1 Comparison of MDE with basic DE

5.4.1.1 Performance criteria 1: comparison of absolute error for fixed number of function evaluation We fixed the total NFE as 200,000, to record the average best fitness function value, absolute error, and standard deviation. The corresponding results are given in Table 3. From this Table it can be clearly observed that average of fitness function values are much closer to theoretical optima for all 30 dimension problems except f_{RB}, while for lower dimension problem both the algorithms give same results.

From Table 4, which gives the statistical analysis, we see that the p values obtained by Kolmogorov–Smirnov test, Shapiro–Wilk test, and Leneve's are 0.00, 0.00, and 0.041, respectively, which shows that the samples are neither normally distributed nor they have an equal variance.

The application of Wilcoxon signed rank test gave the p value as 0.008 (<0.05). Thus statistically, it can be said that there is a significant difference at 5% level of significance between MDE and DE in terms of absolute error. Interestingly, the p value obtained by paired t test came out to be 0.290 (>0.05) showing that there is no significant difference between DE and MDE.

5.4.1.2 Comparison criteria 2: convergence speed and time We noted the NFEs and time taken by both the algorithms to achieve the desired accuracy. The corresponding results are given in Table 5, which also gives the percentage improvement in terms of NFE and average CPU time. From this Table we can see that in almost all the test cases, the proposed version converged faster than the original DE. Likewise, the time taken by MDE is also less in comparison with the basic DE. For Rastrigin function f_{RG}, neither MDE nor DE was able to achieve the desired accuracy and therefore its value is not recorded.

Performance curves (convergence graphs) of few selected functions are given in Fig. 2a–d. From these illustrations it is evident that the convergence of proposed algorithm is faster than basic DE.

The difference between the two algorithms can also be verified statistically from Table 6 with the help of Wilcoxon signed rank test which gave the p value as 0.026 (<0.05). This shows that there is a significant difference between the two algorithms at 5% level of significance in terms of average number of function evaluations.

5.4.1.3 Comparison criteria 3: effect of the dimension To investigate the effect of dimension on searching quality of MDE we took three multimodal test functions, namely Ackley, Griewank, and Levy and Montalvo2. All these functions have several local optima which increase with the increase in the dimension of the problem. We varied the dimension of these problems from 30 to 200 for DE and MDE and recorded the corresponding results in terms of average fitness function value in Table 7.

From the numerical results listed in Table 7, it can be easily seen that MDE is a clear winner. In case of Ackley Function (f_{ACK}), when the dimension was 30, MDE gave a fitness value of 1.70692e−04 while DE gave a value of 2.57992e−03. When the dimensions were increased to 50, 100, 150, and 200 we can see that the performance of basic DE deteriorates consistently with the increase in dimension while MDE shows a more stable performance. A similar behavior is observed for Griewank and Levy and Motalvo2 functions as well where MDE gave a much superior performance than basic DE. The better performance of the proposed MDE is also evident from Fig. 3a, which illustrates the varying curves of the mean objective value derived by each approach with respect to dimension and from Fig. 3b, which gives the percentage improvement of MDE with respect to DE for the three functions.

5.4.1.4 Comparison criteria 4: nontraditional benchmark problems We validated the efficiency of proposed MDE on a selected set of recently proposed benchmark test suit for *CEC 2008 special session and competition on large-scale global optimization* (Tang et al. 2007). This test suite

Table 3 Comparison of DE and MDE in terms of average fitness function value, absolute error, and standard deviation for 30 runs

Fun.	Dim	Fitness		Error		Standard deviation	
		DE	MDE	DE	MDE	DE	MDE
f_{EP}	2	−9.99999e−01	−9.99999e−01	1.34241e−06	1.34241e−06	6.98197e−07	1.25952e−06
f_{FX}	2	9.98004e−01	9.98004e−01	1.83231e−03	1.83231e−03	3.62360e−10	1.79905e−10
f_{CB6}	2	−1.03163e−00	−1.03163e−00	1.53829e−06	1.53829e−06	8.17617e−07	8.16449e−07
f_{GP}	2	3.00000e−00	3.00000e−00	0.00000e−00	0.00000e−00	1.07046e−06	6.80806e−07
f_{H3}	3	−3.86230e−00	−3.86230e−00	4.82231e−04	4.82231e−04	1.46942e−06	7.19190e−07
f_{SP}	30	8.24162e−05	9.25282e−07	8.24162e−05	9.25282e−07	9.76165e−06	1.94921e−07
f_{ACK}	30	2.57992e−03	5.52000e−04	2.57992e−03	5.52000e−04	1.70692e−04	8.7203e−05
f_{SWF}	30	5.96987e+02	6.81986e−04	5.96987e+02	6.81986e−04	5.10682e+02	6.81986e−04
f_{GW}	30	4.89673e−04	1.82288e−07	4.89673e−04	1.82288e−07	9.31764e−05	6.79847e−08
f_{LM2}	30	9.31764e−05	1.57783e−07	9.31764e−05	1.57783e−07	1.51904e−06	2.41843e−08
f_{ST}	30	0.00000e−00	0.00000e−00	0.00000e−00	0.00000e−00	0.00000e−00	0.00000e−00
f_{RB}	30	3.02422e+01	2.44160e+01	3.02422e+01	2.44160e+01	9.70109e−01	1.56378e−01
f_{RG}	30	1.37732e+02	8.64681e+01	1.37732e+02	8.64681e+01	7.40387e−00	1.27605e+01
$f_{SWF2.22}$	30	1.91824e−03	1.00041e−03	1.91824e−03	1.00041e−03	1.91824e−03	8.94059e−05
f_{RHE}	30	9.85092e−04	4.53821e−05	9.85092e−04	4.53821e−05	1.02531e−04	2.50376e−05

Table 4 Statistical analysis of DE and MDE on the basis of parametric and nonparametric tests

Kolmogorov–smirnov		Shapiro–wilk		Levene	Paired *t* test	Wilcoxon signed rank test
DE	MDE	DE	MDE			
0.000	0.000	0.000	0.000	0.041	0.290	0.008

Table 5 Comparison of average number of function evaluation (NFE) and time in second for all algorithms

Function	Dimension	NFE		% Improvement in NFE	Time (s)		% Improvement in time
		DE	MDE		DE	MDE	
f_{EP}	2	833	670	19.567	0.10	0.06	40
f_{FX}	2	977	482	50.665	0.11	0.07	36.36364
f_{CB6}	2	1,020	712	30.196	0.11	0.06	45.45455
f_{GP}	2	970	988	–	0.11	0.12	−9.09091
f_{H3}	3	1,170	1,284	–	0.10	0.11	0
f_{SP}	30	206,400	153,570	25.596	15.00	10.30	31.33333
f_{ACK}	30	259,410	232,740	10.281	18.90	16.20	14.28571
f_{SWF}	30	366,570	130,800	64.318	5.20	1.80	65.38462
f_{GW}	30	224,910	133,440	40.669	17.10	10.00	41.52047
f_{LM2}	30	182,700	133,770	26.782	25.20	17.20	31.74603
f_{ST}	30	124,800	58,800	52.884	8.90	4.00	55.05618
f_{RB}	30	3,066,300	3,603,000	–	525.00	621.00	−18.2857
f_{RG}	30	–	–	–	–	–	–
$f_{SWF2.22}$	30	249,960	245,070	1.956	2.400	1.50	37.5
f_{RHE}	30	232,200	200,400	13.695	229.00	193.00	15.72052

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 nonseparable (F_2, F_3, F_5, F_7) and three separable (F_1, F_4, F_6). Here, the comparison is made on the basis of solution accuracy. In order to get a clear picture on

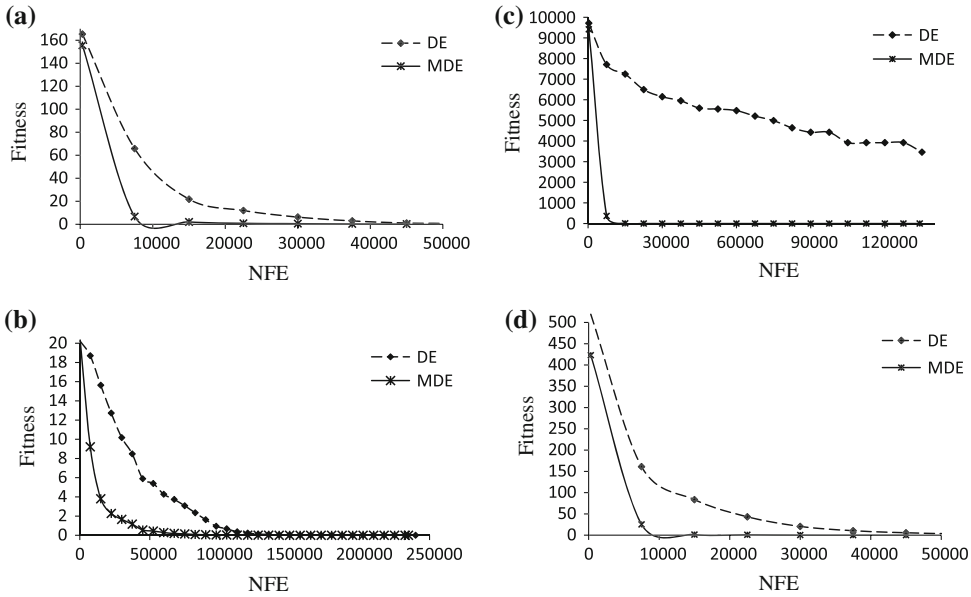

Fig. 2 **a** Performance curves of DE versus MDE for function f_{SP}. **b** Performance curves of DE versus MDE for function f_{ACK}. **c** Performance curves of DE versus MDE for function f_{SWF}. **d** Performance curves of DE versus MDE for function f_{GW}

Table 6 Statistical analysis of DE and MDE on the basis of parametric and nonparametric tests

Kolmogorov–smirnov		Shapiro–wilk		Levene	Paired *t* test	Wilcoxon signed rank test
DE	MDE	DE	MDE			
0.000	0.000	0.000	0.000	0.951	0.972	0.026

Table 7 Comparison of average fitness function value for both the algorithms for high dimension, multimodal Ackley, Griewank and Levy and Montalvo2 function

Dim.	Algorithm.	Function name		
		Ackley	Griewank	Levy and Montalvo2
30	DE	2.57992e−03	4.89673e−04	9.31764e−05
	MDE	1.70692e−04	9.31764e−05	1.51904e−06
	% Improvement	93.383	80.97	98.30
50	DE	4.78024e−00	3.40085e−00	1.88359e+01
	MDE	4.25099e−01	3.52779e−02	8.05453e−02
	% Improvement	91.11	99.00	99.57
100	DE	1.85250e+01	6.13357e+02	3.02926e+03
	MDE	4.02639e-00	8.56930e−01	6.42696e−00
	% Improvement	78.30	99.90	99.78
150	DE	1.99304e+01	1.66332e+03	7.23171e+03
	MDE	5.34925e−00	1.22264e−00	3.35907e+01
	% Improvement	73.20	99.90	99.53
200	DE	2.02761e+01	2.94865e+03	1.26987e+04
	MDE	7.10065e−00	4.79949e−00	1.48855e+02
	% Improvement	65.00	99.80	98.82

algorithm's efficiency, the best, median, worse, mean, and standard deviation of fitness value are computed with respect to 30 runs per function. Name of the functions and their properties are listed in Table 8 and the corresponding results are listed in Table 9. Here, MDE is compared with basic DE and ODE (Rahnamayan and Wang 2008).

For the nontraditional shifted functions, we have used the settings same as given in Rahnamayan and Wang (2008). Parameter scaling factors F and crossover rate Cr are taken as 0.5 and 0.9, respectively. Numerical results show that except for functions F_2 and F_3, in the remaining five functions MDE performed better than both DE and

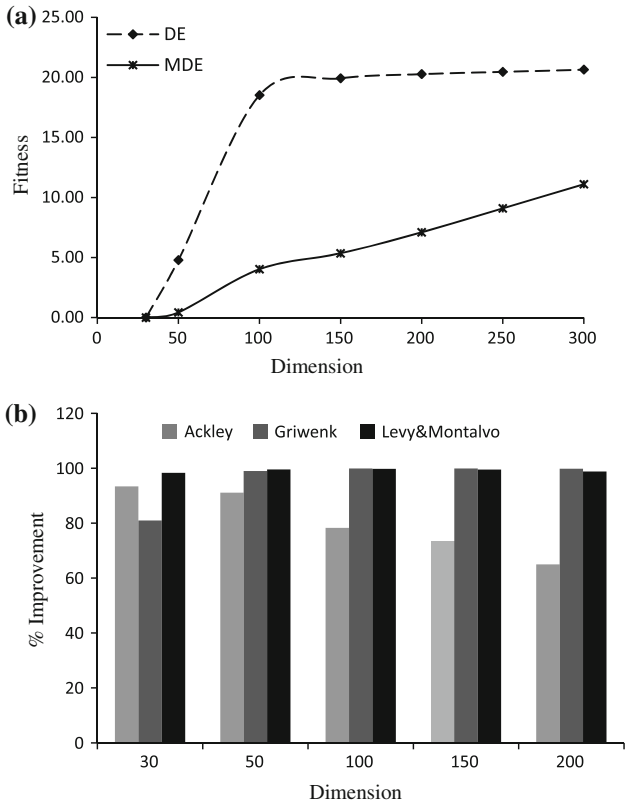

Fig. 3 **a** Performance curves of DE versus MDE for function f_{ACK}. **b** Percentage improvement in average fitness function value while using MDE for function Ackley (f_{ACK}), Griewank (f_{GR}) and Levy and Montalvo2 (f_{LM2}) functions

ODE. Although, for F_2 and F_3, ODE (Rahnamayan and Wang 2008) gave the best performance, the performance of MDE was much better than the performance of basic DE.

5.4.2 Comparison of MDE with other DE algorithms

5.4.2.1 MDE versus BBDE In this section, the performance of MDE is compared with bare bones differential evolution (BBDE) (Omran et al. 2008). It is a recent modified, hybrid version of DE and bare bones PSO and has reportedly given good results on selected benchmark problems. We changed the parameter settings of our algorithm according to parameter settings of BBDE (Omran et al. 2008) and tested the two algorithms for the 15 benchmark problems taken in the present study. This was done to give an advantage to the BBDE algorithm and also to test our algorithm under the changes parameter settings. The corresponding results are shown in Table 10.

From this table we see that in terms of error both the algorithms performed more or less in a similar manner with both outperforming each other in 4 out of 15 cases with marginal difference. In the remaining 11 problems both algorithms gave a similar performance. This shows that both the algorithms are at par with each other in terms of error. This conclusion is further verified with the help of statistical analysis which we give in Table 11. From this Table also we can see that according to the nonparametric Wilcoxon test there is no significant difference between the two algorithms at 5% level of significance.

When we compare the algorithms in terms of NFE, we can see from Table 10 that MDE outperformed the BBDE in 11 out of 15 test problems while BBDE performed better that MDE in the remaining 4 problems. The superior performance of MDE in terms of NFE is further validated in Table 11 where it is statistically compared with BBDE. From this Table we can see that according to the nonparametric Wilcoxon test there is significant difference between the two algorithms in terms of average NFE at 5% level of significance.

5.4.2.2 MDE versus DEahcSPX In this section we compare the performance of MDE with local search self-adaptive differential evolution (DEahcSPX) (Noman and Iba 2008). It is a hybrid version of DE and simple hill-climbing mechanism. Once again, to give an advantage to DEahcSPX, we changed the parameter settings of our algorithm according to parameter settings of DEahcSPX as given in (Noman and Iba 2008) and compared the algorithms for the 15 benchmark problems considered in the

Table 8 Selected benchmark problems proposed in CEC2008 (Tang et al. 2007)

Function	Name	Properties	Search space
F_1	Shifted Sphere	Unimodal, separable, scalable	[−100, 100]
F_2	Shifted Schwefel's 2.21	Unimodal, nonseparable	[−100, 100]
F_3	Shifted Rosenbrock's	Multimodal, nonseparable. A narrow valley from local optimum to global optimum	[−100, 100]
F_4	Shifted Rastrigin's	Multimodal, separable huge number of local optima	[−5, 5]
F_5	Shifted Griewank's	Multimodal, nonseparable	[−600, 600]
F_6	Shifted Ackley's	Multi-modal, separable	[−32, 32]
F_7	FastFractal DoubleDip	Multi-modal, nonseparable	[−1, 1]

All problems are executed for dimension 500

Table 9 Comparison of proposed MDE algorithm with DE and ODE (Rahnamayan and Wang 2008) on seven nontraditional shifted functions in terms of fitness (best, median, worst and mean) and standard deviation (std)

Problem	Error value	DE	ODE (Rahnamayan and Wang 2008)	MDE
F_1	Best	2,636.54	15.66	10.48
	Median	3,181.45	36.61	19.32
	Worst	4,328.80	292.65	23.43
	Mean	3,266.24	80.17	19.54
	Std	409.68	79.24	37.32
F_2	Best	79.74	3.60	20.42
	Median	82.39	4.86	12.32
	Worst	85.92	11.91	34.69
	Mean	82.93	5.78	27.00
	Std	2.09	2.37	8.50
F_3	Best	76,615,772.08	39,718.90	807,641.56
	Median	119,733,049.20	137,279.03	582,743.33
	Worst	169,316,779.50	407,661.64	673,494.23
	Mean	123,184,755.70	154,306.34	467,232.85
	Std	29,956,737.58	114,000.53	124,845.43
F_4	Best	5,209.99	2,543.51	1,287.74
	Median	5,324.57	4,279.56	4,133.49
	Worst	5,388.24	6,003.94	5,238.32
	Mean	5,332.59	4,216.34	4,141.32
	Std	43.82	1,017.94	61.32
F_5	Best	24.29	1.25	1.10
	Median	24.71	1.55	1.36
	Worst	27.59	2.13	1.56
	Mean	25.16	1.75	1.52
	Std	1.10	0.37	0.26
F_6	Best	4.66	2.49	2.52
	Median	4.97	4.12	3.98
	Worst	5.15	6.73	4.13
	Mean	4.94	4.51	4.02
	Std	0.17	1.44	0.14
F_7	Best	−3,683.07	−3,957.85	−3,961.29
	Median	−3,575.13	−3,834.07	−3,832.43
	Worst	−3,565.73	−3,830.36	−3,830.24
	Mean	−3,593.75	−3,851.82	−3,856.47
	Std	32.74	38.80	31.34

Dimension (n) of all the problems is taken as 500. Maximum NFE is set as $5{,}000 \times n$

present study. The numerical results of these problems in terms of error, standard deviation, and average NFE to obtain an accuracy of 10^{-6} are stored in Table 12.

Once again from this Table it is difficult to conclude on the behaviour of the two algorithms in terms of error as both the algorithms perform similar to each other. Statistical analysis given in Table 13 further shows that, on basis of Wilcoxon test, there is no significant difference between the two algorithms at 5% level of significance.

On the other hand, when we analyze the NFE obtained by the two algorithms to achieve an accuracy of 10^{-6}, we can easily see from Table 12 that MDE takes lesser numbers of function evaluations in 10 out of 15 cases while DEahcSPX performs better than MDE in 2 problems only. In the remaining three problems both the algorithms took same numbers of NFE. The better performance of MDE can also be verified from Table 13 which shows that according to Wilcoxon test there is a significant difference between the two algorithms at 5% level of significance.

5.4.3 *Comparison of MDE with classical algorithms and some state-of-the-art versions of DE*

In this section we compare the performance of MDE with some other contemporary algorithms available in literature which are genetic algorithms (GA), particle swarm

Table 10 Comparison of average error value, standard deviation, and NFE

Function	Error		Std		NFE to achieve an accuracy 10^{-4}	
	MDE	BBDE	MDE	BBDE	MDE	BBDE
f_{EP}	1.12001e−05	1.12001e−05	3.45456e−14	6.44212e−14	578	589
f_{FX}	4.20201e−06	4.20201e−06	8.39921e−10	4.44902e−09	434	545
f_{CB6}	1.51121e−06	1.51121e−06	3.99220e−10	9.38872e−11	756	721
f_{GP}	0.00000e+00	0.00000e+00	4.82912e−18	9.98832e−14	976	1,024
f_{H3}	4.82021e−04	4.82021e−04	7.49823e−09	3.38921e−10	1,198	1,287
f_{SP}	4.14031e−12	0.00000e−00	1.96762e−12	0.00000e−00	21,900	20,342
f_{ACK}	1.42885e−06	0.00000e−00	3.68491e−07	0.00000e−00	33,100	33,564
f_{SWF}	6.81827e−04	1.51688e+03	5.45697e−13	1.17889e+03	25,783	100,000
f_{GW}	1.31940e−12	6.57000e−04	8.45358e−13	2.58300e−03	21,850	52,843
f_{LM2}	4.66334e−06	2.45324e−05	5.43353e−12	4.45323e−09	21,700	28,434
f_{ST}	0.00000e−00	0.00000e−00	0.00000e−00	0.00000e−00	11,600	11,893
f_{RB}	2.55786e+01	4.78571e+01	6.57419e−01	3.18354e+01	100,000	100,000
f_{RG}	1.14928e+02	3.75512e+01	1.96286e+01	1.52549e+01	100,000	100,000
$f_{SWF2.22}$	2.46775e−06	0.00000e−00	7.51392e−07	0.00000e−00	36,450	37,442
f_{RHE}	1.60152e−10	5.64674e+01	1.47536e−10	3.89752e+01	29,200	93,434

Table 11 Statistical comparison of average error value and number of function evaluations (NFE)

Comparison criteria	Kolmogorov–smirnov		Shapiro–wilk		Levene	Paired *t* test	Wilcoxon signed rank test
	MDE	BBDE	MDE	BBDE			
Error	0.000	0.000	0.000	0.000	0.060	0.335	0.314
NFE	0.000	0.000	0.000	0.000	0.027	0.084	0.009

Table 12 Comparison of MDE with DEahcSPX in terms of average error, standard deviation, and NFE

Fun	Error		Std		NFE to achieve an accuracy 10^{-6}	
	MDE	DEahcSPX	MDE	DEahcSPX	MDE	DEahcSPX
f_{EP}	1.12001e−05	1.12001e−05	1.23485e−17	3.45210e−16	720	867
f_{FX}	4.20201e−06	4.20201e−06	7.39021e−10	8.52803e−11	540	520
f_{CB6}	1.51121e−06	1.51121e−06	5.34873e−16	7.24857e−17	870	934
f_{GP}	0.00000e+00	0.00000e+00	7.32091e−21	3.97321e−21	1,020	1,170
f_{H3}	4.82021e−04	4.82021e−04	8.43921e−13	7.49721e−12	1,140	1,245
f_{SP}	6.97359e−29	9.50323e−31	5.61454e−31	3.09332e−30	123,945	87,027
f_{ACK}	4.20859e−09	1.78785e−14	2.46374e−09	1.00981e−32	73,425	129,211
f_{SWF}	6.81827e−04	2.89323e+02	1.63709e−12	5.48334e+02	300,000	300,000
f_{GW}	3.25261e−19	4.94325e−03	1.08420e−19	4.79443e−04	95,250	121,579
f_{LM2}	3.57702e−21	8.49321e−18	3.39534e−15	7.39232e−15	100,815	123,843
f_{ST}	0.00000e−00	0.00000e−00	7.33490e−28	3.38218e−30	5,265	60,309
f_{RB}	1.11792e+01	1.87932e+00	1.84403e+00	5.00932e+01	300,000	299,913
f_{RG}	6.96571e+00	3.98219e+01	2.38987e−02	4.68245e+00	300,000	300,000
$f_{SWF2.22}$	7.73096e−10	8.34821e−09	8.23401e−15	3.88013e−13	211,905	218,301
f_{RHE}	2.54821e−12	7.42495e−12	2.94001e−22	7.39922e−24	150,780	169,368

optimization (PSO), evolutionary strategies (ES), CLPSO, and also two other advanced versions of DE: SACDE (Caponio et al. 2009) and SFMDE (Caponio et al. 2009). These algorithms are tested for all the 15 test problems considered in the present study. Here, we have fixed the dimensions of the test problems as 10 and recorded the

Table 13 Statistical comparison of average error value and number of function evaluations (NFE)

Comparison criteria	Kolmogorov–smirnov		Shapiro–wilk		Levene	Paired *t* test	Wilcoxon signed rank test
	MDE	DEahcSPX	MDE	DEahcSPX			
Error	0.000	0.000	0.000	0.000	0.037	0.289	0.214
NFE	0.057	0.180	0.008	0.025	0.027	0.123	0.033

Table 14 MDE versus some other state versions of DE and other contemporary algorithms in terms of average error

Function	MDE	SACDE	SFMDE	GA	PSO	ES	CLPSO
f_{EP}	1.939e−08	1.939e−04	1.831e−05	9.721e−04	1.939e−04	9.721e−04	1.939e−04
f_{FX}	4.211e−06	4.211e−06	4.211e−06	4.211e−06	4.211e−06	4.211e−06	4.211e−06
f_{CB6}	1.512e−06	1.512e−06	1.512e−06	1.512e−06	1.512e−06	1.512e−06	1.512e−06
f_{GP}	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
f_{H3}	4.820e−06	4.820e−04	4.820e−04	4.820e−04	4.820e−04	4.820e−04	4.820e−04
f_{SP}	1.293e−39	3.341e−24	9.642e−23	7.432e−18	5.345e−21	4.587e−19	8.432e−20
f_{ACK}	1.210e−15	3.642e−15	9.882e−15	3.593e−01	2.044e−02	3.412e−01	3.267e−02
f_{SWF}	1.710e−01	3.710e−01	5.710e−01	1.703e+03	1.045e+03	1.739e+03	5.183e+01
f_{GW}	0.000e+00	5.014e−04	3.449e−08	3.646e+00	9.573e−01	3.556e+00	5.742e−01
f_{LM2}	1.291e−19	9.321e−14	8.423e−16	9.342e−09	2.432e−13	4.453e−10	5.432e−14
f_{ST}	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
f_{RB}	6.282e−08	9.593e−08	8.453e−08	8.323e−04	9.321e−05	6.345e−04	3.535e−05
f_{RG}	0.000e+00	0.000e+00	0.000e+00	3.156e+01	3.006e+00	3.239e+01	7.063e−01
$f_{SWF2.22}$	7.964e−22	2.445e−16	8.543e−18	5.984e−08	7.920e−14	4.633e−09	7.533e−15
f_{RHE}	7.078e−38	6.543e−29	5.342e−27	9.432e−18	5.343e−21	8.643e−17	6.643e−24

Table 15 Results of the Friedman test ($\alpha = 0.05$)

N	Friedman value	*df*	*p* value
15	53.538	6	0.000

df degrees of freedom

N total no of function

Table 16 Ranking obtained through Friedman's test and critical difference of Bonnferroni–Dunn's procedure

Algorithm	Mean rank
MDE	2.37
SACDE	3.03
SFMDE	2.80
GA	5.77
PSO	5.43
ES	5.57
CLPSO	4.43
Crit. diff. $\alpha = 0.05$	2.080882
Crit. diff. $\alpha = 0.10$	1.888413

average fitness function values for maximum NFE of 10^5. The parameter settings for all the algorithms are kept the same as that of (Caponio et al. 2009). The results in terms of average error are stored in Table 14. Once again from this Table we do not get a clear picture of the performance of the algorithms. Therefore, we analyzed them statistically.

Since the algorithms to be compared are more than two, we cannot apply the paired *t* test or the Wilcoxon test as we did in the previous sections. Here, we have analyzed the seven algorithms with the help of nonparametric Friedman test and Bonferroni–Dunn test for which the results are given in Tables 15 and 16, respectively.

Values in Table 14 allow us to carry out a rigorous statistical study. Our study is focused on the algorithm that had the lowest average error rate in comparison which in the present case is MDE. We studied the behavior of this algorithm with respect to remaining ones and determined if the results it offered are better than the ones offered by the rest of algorithms, computing the *p*-values for each comparison. Table 15, shows the result of applying Friedman's test in order to see whether there are global differences in the results. Given that the *p* values of Friedman are lower than the level of significance considered $\alpha = 0.05$, there are significant differences among the observed results. Attending to these results, a post-hoc statistical analysis could help us to detect concrete differences among algorithms. For this, we will employ Bonferroni–Dunn's test to detect significant differences for the control algorithm

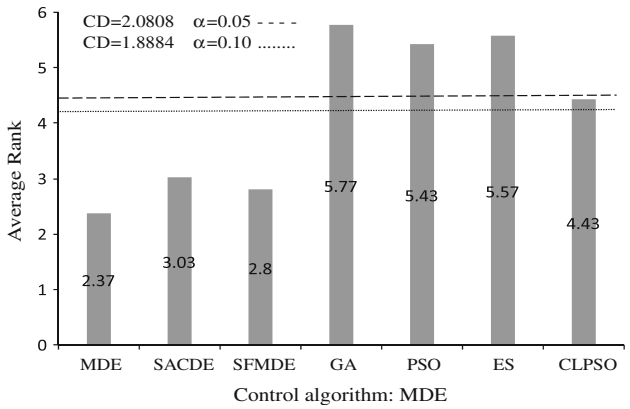

Fig. 4 Bonferroni–Dunn's graphic corresponding to the error

(MDE). Table 16 summarizes the ranking obtained by Friedman's test and the critical difference of Bonferroni–Dunn's procedure. The critical difference obtained is then added to the rank of control algorithm and compared with other algorithms. In the present case this value is coming out to be 4.450882 and 4.258413 at $\alpha = 0.05$ and $\alpha = 0.10$ level of significance. Algorithms having rank greater than these values perform worse in comparison with control algorithm. Therefore from these criteria, we can see that MDE is at par with SACDE, SFMDE, and CLPSO at 0.05 level of significance and is better than GA, PSO, and ES at this level, while at 0.1 level of significance MDE, SACDE, and SFMDE gave more or less a similar performance which is better than CLPSO, GA, PSO, and ES. The corresponding results are illustrated in Fig. 4.

6 Discussions and conclusions

The proposed MDE algorithm presented in this paper introduces a mechanism to avoid the stagnation of population and to improve the performance of the basic structure of DE. MDE introduces a concept of '*failure_counter*' (*FC*), which keeps a record of the progress of the algorithm. This is done by keeping a track of the successful performance of the individuals of the DE population. When the individuals do not show any improvement in the objective function value, an indication of the stagnation of population, they are subject to Cauchy mutation which helps them in moving to a new position, which is hopefully better than the previous position. Performance of MDE is validated on a set of 15 standard benchmark problems and seven nontraditional shifted functions.

MDE is compared with basic DE as well as with some of the advanced versions of DE like BBDE, DEahcSPX, SACDE, SFMDE and classical algorithms like GA, PSO, ES, and CLPSO. The results are analyzed using the common performance measures like average fitness function value, average error, average numbers of function evaluations (NFE), average CPU time etc. Besides these measures we also compared the algorithms statistically using the nonparametric tests. The following observations were made on the basis of numerical results:

- In comparison with basic DE, it was observed that for lower dimension problems both MDE and DE algorithms perform reasonably well for multimodal functions. However, when we increased the dimension there was a decline in the performance of DE while MDE performed remarkably well for dimensions as high as 200 for multimodal functions. This is particularly advantageous in case of real-life problems which may have several variables.
- MDE took lesser NFE in comparison with DE, indicating faster convergence. Statistically also it was observed that there is a significant difference between MDE and DE at 5% level of significance in terms of average NFE.
- Performance of MDE on nontraditional shifted functions shows its competence for solving high-dimensional complex optimization problems.
- Statistical comparison of MDE with BBDE and DEahcSPX showed that although the algorithms are at par with each other in terms of average fitness function value, there is a significant difference between the algorithms at 5% level of significance in terms of NFE, with MDE performing better than BBDE and DEahcSPX.
- Comparison of MDE with SACDE, SFMDE, GA, PSO, ES, and CLPSO was done for dimension 10. Statistical analysis showed that MDE, SACDE, SFMDE, and CLPSO performed at par with each other and as expected were better than GA, PSO, and CLPSO at 5% level of significance, while at 10% level of significance only MDE, SACDE, and SFMDE performed at par with each other and were better than the remaining algorithms.
- Although MDE gave a good performance in most of the test problems, there are a few cases in which MDE did not perform very well. Rosenbrock (f_{RB}) and Rastrigin (f_{RG}) are two functions where it did not give good results for dimension 30 and required further investigations. However, for lower dimension (=10) MDE gave good performance for both the functions. In case of nontraditional shifted functions, MDE did not perform as good as the competing algorithm ODE for two functions F_2 and F_3, but was significantly better than the basic DE. This shows that some further fine tuning of parameters are needed in MDE so that it is able to solve all types of problems.

The above observations show that the proposed MDE has a faster convergence mechanism without compromising with the quality of solution and is either better or at par with the other algorithms taken for consideration. We may say that MDE is a good choice for multimodal, high-dimensional optimization problems. Moreover, the modifications proposed in the present work are simple and general enough to be applied to any version of DE or any other population-based search technique. In future we plan to analyze theoretically the behaviour of Cauchy mutation used in the present study. We are also looking into the possibility of using Cauchy mutation adaptively. Further, this work can be extended for constrained problems as well.

Acknowledgments The authors would like to thank the unknown referees whose comments and suggestions helped in improving the shape of the paper. The authors would also like to acknowledge the financial support provided by the MHRD, India, and DST, India, respectively.

Appendix

Benchmark problems:

1. Easom function (EP):

$$f_{EP}(x) = -\cos(x_1)\cos(x_2)\exp\left[-(x_1-\pi)^2-(x_2-\pi)^2\right]$$
$$\text{With} -10 \le x_i \le 10,\ \min f_{\text{EP}}(\pi,\pi) = -1$$

   It is a multimodal, nonseparable function having several local optima.

2. Foxhole function (FX):

$$f_{\text{FX}}(x) = \left(\frac{1}{500} + \sum_{i=1}^{25} \frac{1}{i + \sum_{j=1}^{2} (x_j - a_{ij})^6}\right)^{-1}$$
$$\text{With} -65.536 \le x_i \le 65.536,\ \min f_{\text{FX}}(-32,-32) = .998004$$
$$a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 \ldots & -32 & -16 & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 \ldots & 32 & 32 & 32 & 32 & 32 \end{pmatrix}$$

   It is a multimodal nonseparable function having several local optima. Many standard optimization algorithms get stuck in the first peak they find.

3. Six hump Camel back function (CB6):

$$f_{\text{CB6}}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$
$$\text{With} -5 \le x_i \le 5,$$
$$\min f_{\text{CB6}}(0.0898,\ -0.7126)/(-0.0898, 0.7126) = -1.0316285$$

   It is a multimodal separable function having two global optima and four local optima.

4. Goldstien problem (GP):

$$f_{\text{GP}}(x) = \left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times \left(15 - 32x_1 + 12x_1^2 + 42x_2 - 36x_1x_2 + 27x_2^2\right)\right]$$
$$\text{With} -2 \le x_i \le 2,\ \min f_{\text{GP}}(0,-1) = 3$$

   It is a multimodal, nonseparable function having one global minimum and four local minima.

5. Hartman 3 function (H3):

$$f_{\text{H3}}(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right] \quad \text{With } 0 \le x_i \le 1,$$
$$\min f_{\text{H3}}(0.114614,\ 0.555649,\ 0.852547) = -3.862782$$

I	c_i	a_{ij}			p_{ij}		
		$j=1$	2	3	$j=1$	2	3
1	1	3	10	30	0.3689	0.117	0.2673
2	1.2	0.1	10	35	0.4699	0.4387	0.747
3	3	3	10	30	0.1091	0.8732	0.5547
4	3.2	0.1	10	35	0.3815	0.5743	0.8828

   It is a multimodal nonseparable function having four local minima and one global minimum.

6. Sphere function:

$$f_{\text{SP}}(x) = \sum_{i=1}^{n} x_i^2,$$
$$\text{With} -100 \le x_i \le 100,\ \min f_{\text{SP}}(0,\ldots,0) = 0$$

   It is a simple, continuous unimodal, separable and highly convex function. It serves as test case for validating the convergence speed of an algorithm.

7. Ackley's function (ACK):

$$f_{\text{ACK}}(X) = -20 * \exp\left(-0.2\sqrt{1/n\sum_{i=1}^{n} x_i^2}\right) - \exp\left(1/n\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e,$$
$$\text{With } -30 \le x_i \le 30,\ \min f_{\text{ACK}}(0,\ldots,0) = 0$$

The presence of an exponential term in the Ackley's function covers its surface with numerous local minima. The complexity of this function is moderated. An algorithm that only uses the gradient steepest descent will be trapped in a local optima, but any search strategy that analyzes a wider region will be able to cross the valley among the optima and achieve better results.

8. Schwefel's problem (SWF):

$$f_{\mathrm{SWF}}(x) = 418.9829 \times n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right)$$
$$\text{With} -500 \le x_i \le 500,\ \min f_{\mathrm{SWF}}(s,\ldots,s) = 0$$

Where $s = 420.97$
It is a multimodal function with very deep sinusoidal interactions. It is generally considered to be difficult for optimization algorithms.

9. Griewank function (GW):

$$f_{\mathrm{GW}}(x) = \frac{1}{4,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$
$$\text{With} -600 \le x_i \le 600,$$
$$\min f_{\mathrm{GW}}(0,\ldots,0) = 0$$

The Griewank function is a highly multimodal, nonseparable function. It has many regularly distributed local minima. It tests both convergence speed and the ability to escape from a shallow local minimum.

10. Levy and Montalvo2 problem (LM2):

$$f_{\mathrm{LM2}}(x) = \sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)(1 + \sin^2(3\pi x_{i+1})) + (x_n - 1)(1 + \sin^2(2\pi x_n))$$

$$\text{With} -50 \le x_i \le 50,\ \min f_{\mathrm{LM2}}(1,\ldots,1) = 0$$
It is a multimodal, nonseparable function having several local optima.

11. Step function(ST):

$$f_{\mathrm{ST}}(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$$
$$\text{With} -100 \le x_i \le 100,$$
$$\min f_{\mathrm{ST}}(-0.5 \le x_i \le 0.5) = 0$$

12. Rosenbrock problem (RB):

$$f_{\mathrm{RB}}(x) = \sum_{i=1}^{n-1} [100(x_{i+1}^2 - x_i^2) + (1 - x_i^2)]$$
$$\text{With} -30 \le x_i \le 30,\ \min f_{\mathrm{RB}}(1,\ldots,1) = 0$$

Rosenbrock function is unimodal for smaller dimensions; however, as we increase the number of dimensions it ceases to be unimodal. Rosenbrock function is like Colville function for which the optimum lies inside a long, narrow, parabolic-shaped float valley. Like Colville function it helps in testing the ability of an algorithm to prevent premature convergence.

13. Rastrigin's function (RG):

$$f_{\mathrm{RG}}(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$$
$$\text{With} -5.12 \le x_i \le 5.12,\ \min f_{\mathrm{RG}}(0,\ldots,0) = 0$$

This function is highly multimodal, nonseparable function having large number of local optima

14. Schwefel's problem 2.22:

$$f_{\mathrm{SWF2.22}}(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$$
$$\text{With} -10 \le x_i \le 10,\ \min f_{\mathrm{SWF2.22}}(0,\ldots,0) = 0$$

15. Rotated hyper ellipsoid function (RHE):

$$f_{\mathrm{RHE}}(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$$
$$\text{With} -100 \le x_i \le 100,\ \min f_{\mathrm{RHE}}(0,\ldots,0) = 0$$

References

Abbass H (2002) The self-adaptive pareto differential evolution algorithm. In: Proceedings of the 2002 congress on evolutionary computation, pp 831–836

Andre J, Siarry P, Dognon T (2001) An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. Adv Eng Software 32:49–60

Birru HK, Chellapilla K, Rao SS (1999) Local search operators in fast evolutionary programming. Proc IEEE Int Conf Evol Comput 2:1506–1513

Brest J, Boskovic B, Greiner S, Zumer V, Maucec MS (2007) Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Comput 11(7):617–629

Caponio A, Neri F, Tirronen V (2009) Superfit control adaptation in memetic differential evolution frameworks. Soft Comput 13:811–831

Coelho LS, Krohling RA (2003) Predictive controller tuning using modified particle swarm optimization based on Cauchy and Gaussian distributions. In: Proceedings of the 8th on-line world conference on soft computing in industrial applications. WSC8

Fan H-Y, Lampinen J (2003) A trigonometric mutation operation to differential evolution. J Glob Optim 27:105–129

Gamperle R, Muller SD, Koumoutsakos A (2002) Parameter study for differential evolution. In: WSEAS NNA-FSFS-EC 2002. Interlaken, Switzerland

García S, Molina D, Lozano M, Herrera F (2009a) 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. J Heuristics 15:617–644

García S, Fernández A, Luengo J, Herrera f (2009b) A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput 13:10:959–977. doi:10.1007/s00500-008-0392-y

Hendtlass T (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. Springer, Berlin, vol 2070, pp 11–18

Hrstka O, Kucerová A (2004) Improvement of real coded genetic algorithm based on differential operators preventing premature convergence. Adv Eng Software 35:237–246

Kannan S, Slochanal S, Subbaraj P, Padhy N (2004) Application of particle swarm optimization technique and its variants to generation expansion planning. Electr Power Syst Res 70(3): 203–210

Lampinen J, Zelinka I (2000) On stagnation of the differential evolution algorithm. In: Ošmera P (ed) Proceedings of MENDEL 2000, 6th international mendel conference on soft computing. Brno, Czech Republic, pp 76–83

Lan K-T, Lan C-H (2008) Notes on the distinction of Gaussian and Cauchy mutations. In: Eighth international conference on intelligent systems design and applications, vol 1, pp 272–277

Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. Soft Comput Fusion Found Methodol Appl 9(6): 448–462

Noman N, Iba H (2005) Enhancing differential evolution performance with local search for high dimensional function optimization. In: Proceedings of the 2005 conference on genetic and evolutionary computation, pp 967–974

Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12(1):107–125

Omran M, Engelbrecht A, Salman A (2005a) Differential evolution methods for unsupervised image classification. Proc IEEE Congr Evol Comput 2:966–973

Omran M, Salman A, Engelbrecht AP (2005b) Self-adaptive differential evolution, computational intelligence and security, PT 1. In: Proceedings lecture notes in artificial intelligence, vol 3801, pp 192–199

Omran MGH, Engelbrecht AP, Salman A (2008) Bare bones differential evolution. Eur J Oper Res. doi:10.1016/j.ejor.2008.02.035

Pant M, Ali M, Singh VP (2009) Parent centric differential evolution algorithm for global optimization. Opsearch 46(2):153–168

Price K (1999) An introduction to DE. In: Corne D, Marco D, Glover F (eds) New ideas in optimization. McGraw-Hill, London (UK), pp 78–108

Qin K, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417

Rahnamayan S, Wang GG (2008) Solving large scale optimization problems by opposition based differential evolution (ODE). WSEAS Trans Comput 7(10):1792–1804

Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79

Ronkkonen J, Kukkonen S, Price KV (2005) Real parameter optimization with differential evolution. In: Proceedings of IEEE congress on evolutionary computation (CEC-2005). IEEE Press, vol 1, pp 506–513

Rudolph G (1997) Local convergence rates of simple evolutionary algorithms with Cauchy mutations. IEEE Trans Evol Comput 1(1):249–256

Shih FY, Edupuganti VG (2009) A differential evolution based algorithm for breaking the visual steganaliytic system. Soft Comput 13(4):345–353

Stacey A, Jancie M, Grundy I (2003) Particle swarm optimization with mutation. In: Proceeding of IEEE congress on evolutionary computation, pp 1425–1430

Storn R (1995) Differential evolution design for an IIR-filter with requirements for magnitude and group delay. Technical Report TR-95-026. International Computer Science Institute, Berkeley, CA

Storn R, Price K (1995) DE-a simple and efficient adaptive scheme for global optimization over continuous space. Technical Report TR-95-012, ICSI, March 1995. ftp://ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.ps.Z

Talbi H, Batouche M (2004) Hybrid particle swarm with differential evolution for multimodal image registration. Proc IEEE Int Conf Indust Technol 3:1567–1573

Tang K, Yao X, Suganthan PN, MacNish C, Chen YP, Chen CM, Yang Z (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

Teng NS, Teo J, Hijazi MHA (2009) Self-adaptive population sizing for a tune-free differential evolution. Soft Comput 13(7):709–724

Vesterstroem J, Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. Proc Congr Evol Comput 2:1980–1987

Wang H, Liu Y, Li C, Zeng S (2006) A hybrid particle swarm algorithm with Cauchy mutation. In: IEEE swarm intelligence symposium 2007 (SIS 2007), Honolulu, Hawaii, USA (in press)

Xu W, Gu X (2009) A hybrid particle swarm optimization approach with prior crossover differential evolution. In: Proceedings of GEC09, pp 671–677

Yang Z, Tang K, Yao X (2008a) Self-adaptive differential evolution with neighborhood search. In: Proceedings of IEEE congress on evolutionary computation (CEC-2008), Hong Kong, pp 1110–1116

Yang Z, Tang K, Yao X (2008b) Large scale evolutionary optimization using Cooperative Co evolution. Inf Sci 178(15):2985–2999

Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102

Zaharie D (2003) Control of population diversity and adaptation in differential evolution algorithms. In: Matousek D, Osmera P (eds) Proceedings of MENDEL 2003, 9th international conference on soft computing. Brno, Czech Republic, pp 41–46

Zaharie D, Petcu D (2003) Adaptive pareto differential evolution and its parallelization. In: Proceedings of 5th international conference on parallel processing and applied mathematics. Czestochowa, Poland, vol 3019, pp 261–268

Zhang WJ, Xie XF (2003) DEPSO, hybrid particle swarm with differential evolution operator. IEEE Int Conf Syst Man Cybern 4:3816–3821

Zhang C, Ning J, Lu S, Ouyang D, Ding T (2009) A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization. Oper Res Lett 37:117–122