Two Enhanced Differential Evolution Algorithm Variants for Constrained Engineering Design Problems

Pravesh Kumar
Department of Paper Technology,
IIT Roorkee, Saharanpur Campus

Millie Pant
Department of Paper Technology,
IIT Roorkee, Saharanpur Campus

V.P. Singh
Director,
Stallion Institute of Technology, Saharanpur

ABSTRACT
Many engineering design problems can be formulated as optimization problems with constraints. In this paper we have proposed two modified variants of differential evolution (DE) for solving constrained engineering design problems. Pareto-ranking method is used to handle constrained with proposed approaches. The proposed variants named EDE-1 and EDE-2 are tested on 4 engineering design optimization problems taken from literature. Simulation results prove the efficiency of proposed approaches.

Keywords
Differential evolution, Donor mutation, Engineering design optimization, Constraints handling.

1. INTRODUCTION
Many engineering design problem can be formulated as optimization problem [1]. These types of problems normally have mixed (e.g., continuous and discrete)design variables, nonlinear objective functions and nonlinear constraints, some of which may be active at the global optimum. Constraints are very important in engineering design problems, since they are normally imposed on the statement of the problems and sometimes are very hard to satisfy, which makes the search difficult and inefficient [2].

Differential Evolution (DE), a kind of genetic algorithm, was proposed by Storn and Price [3] in 1995. It has emerged as a simple and powerful algorithm for global optimization over continuous space. According to frequently reported experimental studies, DE has shown better performance than many other evolutionary algorithms (EAs) in terms of convergence speed and robustness over several benchmark functions and real-world problems [4] It has many attractive characteristics, such as compact structure, ease to use, good convergence and robustness [5]. DE is capable of handling non-differentiable, nonlinear, multi-modal objective functions and has been successfully demonstrated to a wide range of real life problems of science and engineering field such that engineering design, chemical engineering, mechanical engineering pattern recognition, and so on [5].

In order to improve the performance of DE, several versions of DE variants have been proposed by many researchers over the last few decades. Some of the modified variants are; Learning enhance DE (LeDE) [5], DE with Trigonometric Mutation (TDE) [6], DE with simplex crossover local search (DEahcSPX) [7], Cauchy mutation DE (CDE) [8], Mixed mutation strategy based DE [9] Fuzzy adaptive DE (FADE) [10], DE with self-adaptive control parameter (jDE) [11], Opposition based DE(ODE) [12], Self adaptive DE (SaDE) [13], adaptive DE with optional external archive (JADE) [14], , Modified DE (MDE) [15] , DE with random localization (DERL)[16], DE with global and local neighborhood (DEGL) [17] and so on.

A recent literature survey of DE variants is given in [17]-[19]

In basic DE, the base vector is either randomly selected (DE/rand/bin) or is selected 'greedily'. In this paper we have proposed two new mutation schemes for DE, named EDE-1 and EDE-2. Both schemes aim at efficiently generating the base vector in the mutation phase of DE. The only difference to DE and both proposed algorithms at base vector in mutation operation.

Here we would like to mention that we have already successfully applied EDE-1 and EDE-2 on unconstrained benchmark problems in [20]. Encouraged by its performance, in the present study, we have extended EDE-1 and EDE-2 for solving constrained engineering design problems to check their efficiency and robustness on real world application problems.

The rest of the paper is structured as follows; in section 2 we give the introduction of basic DE. The description of proposed modified DE variants named EDE-1 and EDE-2 are given in section 3. In section 4 engineering problems are given. Experimental settings and numerical results are discussed in section 5 and finally paper is concluded in section 6.

2. DIFFERENTIAL EVOLUTION ALGORITHM
Simple DE (SDE) is a stochastic, population-based direct search method for optimizing real-valued functions of continuous variables. The whole structure of DE is similar to the Genetic Algorithm (GA)**,** and the main difference between standard GA and DE is mutation operation. The mutation is a main operation of DE, and it revises each individual's value according to the difference vector of the population. The algorithm uses mutation operation as a search mechanism and selection operation to direct the search toward the prospective regions in the search space.

The working of DE is as follows: First, all individuals are initialized with uniformly distributed random numbers and evaluated using the fitness function provided. let$P=\{X_{i,G}$, i=1, 2, ..,$NP\}$ be the population at any generation G. Here NP denotes the population size and each X_i is a D-dimensional vector i.e. $X_i=\{x_{1,i}, x_{2,i},...x_{D,i}\}$. For simple DE (DE/rand/1/bin) [1] the mutation, crossover and selection operator defined as follows

Mutation: For each target vector $X_{i,G}$, the mutant vector $V_{i,G}=\{v_{1,i,G}, v_{2,i,G},,...v_{D,i,G},\}$ is defined as

$$V_{i,G} = X_{r_1,G} + F*(X_{r_2,G} - X_{r_3,G}) \quad (1)$$

where $r_1, r_2, r_3 \in \{1,2,...., NP\}$ 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.

Crossover: Crossover is introduced to increase the diversity of perturbed parameter vectors $V_{i,G}=\{v_{1,i,G}, v_{2,i,G},...v_{D,i,G}\}$.let $U_{i,G} = (u_{1,i,G}, . . . , u_{D,i,G})$ be the trail vector then $U_{i,G}$ is defined as;

$$u_{j,i} = \begin{cases} v_{j,i,G} & if \quad rand_j \le Cr \vee j = k \\ x_{j,i,G} & otherwise \end{cases} \quad (2)$$

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 $C_r \in [0, 1]$.

Selection: The final step in the DE algorithm is the selection process. Each individual of the temporary (trial) population is compared with its target vector in the current population. The one with the lower objective function value survives the tournament selection and go to the next generation.

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

3. PROPOSED ALGORITHMS[20]

In this section In this Section, we describe the proposed EDE-1 and EDE-2. In our proposed algorithms, we used two new mutation strategies based on Donor mutation [8] and then selected the mutation strategy stochastically either from the basic DE or from the newly proposed strategy. For this purpose first we fix a probability (*Pr*) and then generate a uniform random number (R) between 0 and 1. If the value of *R* is less than *Pr* then select a new mutation strategy otherwise select basic mutation strategy (as per eq. 1).

The two new mutation strategies (say) M1 and M2 are defined as below;

$$M1: m_{i,G} = (\mu_1 x_{r1,G} + \mu_2 x_{r2,G} + \mu_3 x_{r3,G}) + F(x_{r2,G} - x_{r3,G}) \quad (4)$$

Here μ_i $i=1, 2$are uniform random number between 0 and 1 and μ_3=1-(μ_1+ μ_2) satisfies the condition (μ_1+ μ_2+ μ_3=1).

The other strategy is defined as:

$$M2: m_{i,G} = (\lambda_1/\lambda)x_{r1,G} + (\lambda_2/\lambda)x_{r2,G} + (\lambda_3/\lambda)x_{r3,G} + F(x_{r2,G} - x_{r3,G})$$

(5)

Where λ_i i=1, 2, 3 are uniform random number between 0 and 1.and $\lambda = \sum_{i=1}^{3} \lambda_i$

3.1 Pseudo code of proposed algorithms

```
Begin
Generate uniformly distribution random population P={X1,G, X2,G,..., XNP,G}.
  Xi,G= Xlower +(Xupper –Xlower)*rand(0,1), where i =1, 2,..,NP
Evaluate f(Xi,G)
While (Termination criteria is met )
{
  For i=1:NP
    {
      Select three vectors Xr1,G, Xr2,Gand Xr3,Gdifferent from Pwhere r1≠r2≠r3≠i
        If(R<Pr) /* R=rand(0,1) and Pr= probability */
        {
Perform mutation operation as defined by Equation-4 (EDE-1) or Equation-5 (EDE-2)
        }
      Else
      {
        Perform mutation operation as defined by Equation-1
      }
        Perform crossover operation as defined by Equation-2
        Evaluate f(Ui,G+1)
        Select fittest vector from Xi,Gand Ui,G+1to the population of next
        generation by using Equation-3
      }
  Generate new population Q= {X1,G+1, X2,G+1,..., XNP,G+1}
 } /* end while loop*/
END
```

4. ENGINNERING DESIGN PROBLEM

To validate proposed EDE-1 and EDE-2 algorithms, four engineering design problem are taken from literature [2];

- *E01-Welded Beam Engineering Design problem*
- *E02-Pressure Vessel Design Optimization Problem*
- *E03-Speed Reducer Design Optimization Problem*
- *E04-Tension/Compression Spring Design Optimization Problem*

5. SIMULATION RESULTS AND COMPARISONS

5.1 Experimental Settings

The following settings are taken in the present study after consulting various literature:

- Population size (*NP*) is taken as 100 [11], [12], [14], [20]
- Control parameters, scaling factor *F* is taken as 0.5 and crossover rate*Cr* is fixed at 0.9[11], [14]
- Over all acceleration rate AR, which is taken for the purpose of comparison is defined as [12]:

$$AR = \frac{NFE_{Others} - NFE_{EDE}}{NFE_{others}}\%$$

- In every case, a run is terminated. $|f_{\max} - f_{\min}| \leq 10^{-04}$ is reached where f_{max} and f_{min} are respectively maximum and minimum fitness value [12] or when the maximum number of function evaluation (NFE=10^6) was obtained [8].
- All algorithms are implemented in Dev-C++ and the experiments are conducted on a computer with 2.00 GHz Intel (R) core (TM) 2 duo CPU and 2- GB of RAM [20]

5.2 Results and Discussion

Solutions of engineering problems are given in Table 1 - Table-4. Each solution is taken as the average of 50 runs by each algorithm. We comparison the algorithms in the term of number of function evaluation (NFE) and in term of CPU time.

In Table -1 solution of E01 is given. From the Table we can see that all three algorithms DE,EDE-1 and EDE-2 gives the exact solution of E01 but DE take 17200 NFE to reach the solution while total NFE taken by EDE-1 and EDE-2 are 10580 and 9470 respectively. Hence the acceleration rate of EDE-1 with respect to DE is 38.35 while acceleration rate of EDE-2 with respect to DE is 44.94. Also DE take average CPU time by DE is 0.2 sec while CPU time by EDE-1 is 0.1 sec and 0.1 sec also by EDE-2.

Similarly we can see results the for the other engineering problem from Table-2, Table3 and Table-4 and analysis the efficiency of MDE-1 and MDE-2.

The good performance of the proposed algorithms in terms of convergence can also be observed from Fig 1.

Table 1.Solution of E01

Solution	DE	EDE-1	EDE-2
x_1	0.2058	0.2058	0.2058
x_2	3.4684	3.4683	3.4683
x_3	9.0367	9.0366	9.0368
x_4	0.2057	0.2057	0.2057
f(x)	1.72515	1.72512	1.72515
NFE	17200	10580	*9470*
AR(%)	--	38.35	*44.94*
CPU Time(sec)	0.2	0.1	0.1

Table 2. Solution of E02

Solution	DE	EDE-1	EDE-2
x_1	0.8125	0.8125	0.8125
x_2	0.4375	0.4375	0.4375
x_3	42.1069	42.1085	42.1085
x_4	176.65	176.63	176.63
f(x)	6060.91	6059.93	6059.93
NFE	24700	16340	*14900*
AR(%)	--	33.84	*39.67*
CPU Time (sec)	0.1	0.04	0.02

Table 3. Solution of E03

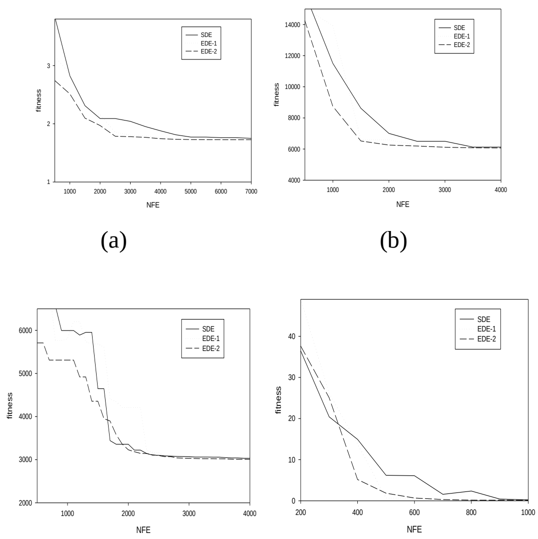
Solution	DE	EDE-1	EDE-2
x_1	3.4999	3.4999	3.4999
x_2	0.7	0.7	0.7
x_3	17.0	17.0	17.0
x_4	7.3	7.3	7.3
x_5	7.8	7.8	7.8
x_6	3.35021	3.35021	3.35021
x_7	5.28661	5.28661	5.28661
f(x)	2996.31	2996.31	2996.31
NFE	22080	13650	*12740*
AR(%)		38.17	*42.30*
CPU Time (sec)	0.2	0.1	0.05

Table 4. Solution of E04

Solution	DE	EDE-1	EDE-2
x_1	0.05169	0.05169	0.05169
x_2	0.3568	0.3567	0.3567
x_3	11.28	11.2870	11.2870
f(x)	0.012661	0.012664	0.012665
NFE	3300	2660	*2380*
AR(%)	--	19.39	*27.87*
CPU Time (sec)	0.1	0.1	*0.1*

Table 5. Comparisons of Proposed EDE-1and EDE-2 with other evolutionary algorithms in term of average fitness value

Algorithms	Problems			
	E01	**E02**	**E03**	**E04**
CPSO	1.72802	6061.0777	NA	0.012674
SicPSO	1.72485	6,059.7143	2,996.3481	0.012665
CoPSO	1.72485	6,059.7143	2,996.3481	0.012665
MBFOA	2.386	6060.460	NA	0.012665
He et al	2.381	6059.7143	NA	0.012671
Coello	1.74830	6288.7445	NA	0.012704
Coello& Montes	1.72822	6059.9463	NA	0.012681
EDE-1	1.72512	6059.93	2996.31	0.012664
EDE-2	1.72512	6059.93	2996.31	0.012665



(a) (b) (c) (d)

Fig 1: Convergence graphs of E01, E02 E03 and E04

5.3 Comparison with other algorithms

Comparison of proposed EDE-1 and EDE-2 with other algorithms CPSO [1], SicPSO [2], CoPSO[21] and MBFOA [22], He et al [23], Coello [24] andCoelloandMontes [25] are given in Table-5. From Table it can see that proposed EDE-1 and EDE-2 gives the similar solution as other evolutionary algorithms.

6. CONCLUSIONS

In the present study, two modified versions of DE, named EDE1 and EDE2 are proposed and validated on a set of 4 engineering design problems. All the problems are non linear in nature and the numerical results are compared with basic DE and also with other methods previously used for solving these problems. It was observed that the proposed variants are quite competent for solving such problems.

7. REFERENCES

[1] He, Q. and Wang, L. 2007. An effective co-evolutionary particle swarm optimization for constrained engineering design problems.Engineering Applications of Artificial Intelligence, pp. 89-99.

[2] Esquivel, S. C. and Cagnina, L.C. 2008. Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica 32, 319–326.

[3] Storn, R. and Price, K. 1995. Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous. Spaces. Berkeley, CA, Tech. Rep. TR-95-012.

[4] Vesterstrom, J. and Thomsen, R. 2004. A comparative study of differential evolution, particle swarm optimization. and evolutionary algorithms on numerical benchmark problems**.** Congress on Evolutionary Computation, 980-987.

[5] Cai, Y., Wang, J. and Yin, J. 2011. Learning enhanced differential evolution for numerical optimization. Springer-Verlag, Soft Computing . doi:10.1007/s00500-011-0744-x

[6] Fan, H. and Lampinen J. 2003. A trigonometric mutation operation to differentia evolution. Journal of Global Optimization. 27, 105-129.

[7] Noman, N. and Iba, H. 2008. Accelerating differential evolution using an adaptive local Search. IEEE Transaction of Evolutionary Computing. 12(1), 107–125.

[8] Ali, M. and Pant, M. 2010. Improving the performance of differential evolution algorithm using cauchy mutation. Soft Computing. doi:10.1007/s00500-010-0655-2

[9] Pant, M., Ali, M. and Abraham, A. 2009. Mixed mutation strategy embedded differential evolution. IEEE Congress on Evolutionary Computation, 1240-1246.

[10] Liu, J. and Lampinen, J. 2005. A fuzzy adaptive differential evolution algorithm. Soft Computing Fusion Found Methodol Appl. 9(6), 448–462.

[11] Brest, J., Greiner, S., Boskovic, B., Mernik, M. and Zumer, V. 2006.Self adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Transaction of Evolutionary Computing. 10(6), 646–657.

[12] Rahnamayan, S., Tizhoosh, H. and Salama, M. 2008. Opposition based differential evolution. IEEE Transaction of Evolutinary Computing. 12(1), 64–79.

[13] Qin, A. K., Huang, V.L.and Suganthan ,P.N. 2009. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transaction of Evolutionary Computing. 13 (2), 398–417.

[14] Zhang, J.and Sanderson, A. 2009. JADE: adaptive differential evolution with optional external archive. IEEE Transaction of Evolutionary Computing. 13(5), 945–958.

[15] Babu, B.V. and Angira, R. 2006. Modified differential evolution (MDE) for optimization of non-linear chemical processes. Computer and Chemical Engineering. 30, 989-1002.

[16] Kaelo, P. and Ali, M.M. 2006. A numerical study of some modified differential evolution algorithms. European Journal of Operational Research. 169, 1176-1184.

[17] Das, S., Abraham, A., Chakraborty, U. and Konar, A. 2009. Differential evolution using a neighborhood based mutation operator. IEEE Transaction of Evolutionary Computing. 13(3), 526–553 .

[18] Neri, F. and Tirronen, V. 2010. Recent advances in differential evolution: a survey and experimental analysis. ArtifIntell Rev. 33 (1–2), 61–106.

[19] Das, S. and Suganthan, P.N. 2011. Differential evolution: a survey of the state-of-the-art. IEEE Transaction of Evolutionary Computing. 15(1), 4-13.

[20] Kumar, P., Pant, M. and Abraham. A. 2011. Two enhanced differential evolution variants for solving global optimization problems. In Proceeding of Third World Congress on Nature and Biologically Inspired Computing (NABIC 2011) IEEE, pp. 208-213.

[21] Aguirre, A. H., Zavala, A. M., Diharce, E. V. and Rionda, S. B. 2007. COPSO: Constrained optimization via PSO algorithm. Technical report No. I-07-04/22-02-2007, Center for Research in Mathematics (CIMAT).

[22] Montes, E.M. and Ocana, B. H. Bacterial foraging for engineering design problems: preliminary results.

[23] He, S., Prempain, E. and Wu, Q.H. 2004. An improved particle swarm optimizer for mechanical design optimization problems. Engineering Optimization, 36(5):585–605.

[24] Coello, C.A.C. 2000. Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry 41,113–127.

[25] Coello, C.A.C. and Montes, E.M. 2002. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics 16, 193–203.