Advances in Mechanical Engineering and its Applications (AMEA)
Vol. 2, No. 3, 2012, ISSN 2167-6380

www.worldsciencepublisher.org

Differential Evolution with Interpolation Based Mutation Operators for Engineering Design Optimization

[1]Pravesh Kumar, [1]Millie Pant, [2]VP Singh

[1]Indian Institute of Technology, Roorkee, India

[2] Stallion College for Engineering and Technology, Saharanpur, India

Email: praveshtomariitr@gmail.com; millidma@gmail.com; singhvp3@gmail.com

Abstract – Differential evolution (DE) has attracted much attention recently as an effective approach for solving numerical optimization problems which arise in many science and engineering fields. This paper presents two modified mutation schemes of differential evolution algorithm for engineering design optimization problem with constraints. These modified mutation scheme are based on interpolation rules, first scheme is based on Inverse Quadratic Interpolation called IQI-DE and second scheme is based on sequential parabolic interpolation called SPI-DE. The proposed variants are tested on 4 engineering design optimization problems, taken from literature. The results show that the proposed modifications improve the performance of basic DE.

Keywords – Differential Evolution; Mutation; Engineering Design Optimization Problem

1. Introduction

The general engineering optimization problem can be defines as follows:

$$\begin{aligned} &Minimizae\ f(X),\\ &subject\ to, g_i(X) \le 0,\ i = 1,2,\ldots,p\\ &and\ h_i(X) = 0, i = p+1,\ldots,m\\ &where\ \ x_i^L \le x_i \le x_i^U, i = 1,2,\ldots,D \end{aligned}$$

where $X = (x_1, x_2, \ldots, x_D) \in R$ is D-dimensional vector of decision space, $f(X)$ is objective function, $g_i(X) \le 0$ are p-inequality constraints, $h_i(X) = 0$ are m-p equality constraints. The function f, g_i, and h_i are linear or non-linear real valued functions. The values x_i^L, x_i^U are lower and upper bounded of x_i respectively.

The numerical optimization problems may occur in almost every field of science and engineering, and in many other real life optimization problems. A literature of several evolutionary algorithms for solving real life problems can be found in [2]-[7].

Differential Evolution (DE) is an evolutionary algorithm, proposed by Storn and Price in 1997 [1] for solving continuous global optimization problems. DE is a simple and efficient, stochastic; population set based methods for global optimization over continuous space. It is capable of handling non-differentiable, nonlinear and multi-modal objective functions and has been successfully demonstrated on a wide range of real life problems such as aircraft landing system [8], engineering design, chemical engineering, mechanical engineering pattern recognition, and so on [9].

Several variants of DE are available in literature, which aim at improving its performance. Some of the modified variants are; Learning enhance DE (LeDE) [9], Cauchy mutation DE (CDE) [10], Modified DE (MDE) [11], DE with self adaptive control parameter (jDE) [12], DE with Trigonometric Mutation (TDE) [13], DE with global and local neighborhood (DEGL) [14], DE with random localization (DERL) [15], Fuzzy adaptive DE (FADE) [16], DE with simplex crossover local search (DEahcSPX) [17], Mixed mutation strategy based DE [18], Self adaptive DE (SaDE) [19], Opposition based DE(ODE) [20], adaptive DE with optional external archive (JADE) [21], and so on.

Most of the above variants have been applied to global optimization problems, the reason being that most of the real life problems can be formulated as optimization problems [22]. 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. The presence of constraints usually increases the complexity of the problems However, it is not possible to avoid or ignore the constraints as they are very important in engineering design problems. They are normally imposed on the statement of the problems and sometimes are very

hard to satisfy, which makes the search difficult and inefficient [23]. A detail literature review of implementation of different evolutionary algorithms on engineering problems can be found in [23]- [30].

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 interpolation based ideas to generate base vector for mutation operation. The first idea is based on Inverse Quadratic Interpolation (IQI) and the second is based on Sequential Parabolic Interpolation (SPI). The corresponding DE variants are named as IQI-DE and SPI-DE. Both variants 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. The significance of selecting efficient base vector is discussed later in the paper.

Here we would like to mention that we have already successfully applied IQI-DE and SPI-DE on unconstrained benchmark problems in [31]. Encouraged by its performance, in the present study, we have extended IQI-DE and SPI-DE 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 simple DE. The description of proposed modified DE variants named IQI-DE and SPI-DE are given in section 3. In section 4 engineering problems are given. Experimental settings and numerical results are discussed in section 5 and section 6 respectively and finally paper is concluded in section 7.

2. Simple Differential Evolution (SDE)

DE is a stochastic, population-based direct search method for optimizing real-valued functions of continuous variables.

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 will be executed until maximum number of generation has been reached or an optimum solution is found. For simple DE (DE/rand/1/bin) the mutation, crossover and selection operator defined as follows:

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 strategy implemented in the DE is given by Equation.(1)

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

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. Here, $X_{r1,G}$ is base vector and F (>0) is a scaling factor which controls the amplification of the difference vector $(X_{r_2,G} - X_{r_3,G})$.

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.

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

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & if \quad rand_j \le Cr \vee j = k \\ x_{j,i,G} & otherwise \end{cases} \quad (2)$$

where $j, k \in \{1,\ldots, D\}$ 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. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation.

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

3. Proposed Algorithms

Mutation is the most important operator of DE. it is based on the distance and magnitude of vectors and it helps in directing the population vectors towards the optimum solution. In basic DE, we see that the base vector is usually selected randomly vector of the population. This base vector plays an important role in generating the mutant vector. A random base vector provides diversity to the population but may slow down the convergence rate. On the other hand when base vector is selected as the best solution vector of the population, the nature of the search process becomes greedy this makes it faster but may also lead to a premature convergence. In the present study we suggest two novel methods of generating the base vector; Inverse quadratic interpolation and sequential parabolic interpolation. These are defined below;

3.1. Inverse Quadratic Interpolation (IQI) based Base Vector

Inverse quadratic interpolation uses three prior points to fit an inverse quadratic function (x as a quadratic function of y) whose value at y=0 is taken as the next estimate of the root x. If the three points are (a, $f(a)$), (b, $f(b)$), (c, $f(c)$) then the interpolation formula is given as;

$$x = \frac{(y-f(a))(y-f(b))c}{(f(c)-f(a))(f(c)-f(b))} + \frac{(y-f(b))(y-f(c))a}{(f(a)-f(b))(f(a)-f(c))} + \frac{(y-f(c))(y-f(a))b}{(f(b)-f(c))(f(b)-f(a))} \quad (4)$$

Setting y=0 gives a result for the next root estimate.

The working of IQI-DE is as follows;

i. Select 3 mutually different random vectors X_{r1}, X_{r2}, and X_{r3} from the population and then find the best among these (say X_{tb}).

ii. Now take a=X_{r1}, b=X_{r2} and c=X_{r3}. and find a new vector (say X_Q) using by Equation (4).

iii. If any of ($f(a)$~$f(b)$), ($f(b)$~$f(c)$) and ($f(c)$~$f(a)$) are zero then we take these values as ($f(a)$+$f(b)$), ($f(b)$+$f(c)$) and ($f(c)$+$f(a)$).

iv. Now, this new vector X_Q will be the best vector from all X_{r1}, X_{r2}, and X_{r3} because it is next root estimation of function using of X_{r1}, X_{r2}, and X_{r3} as the initial solutions.

3.2. Sequential Parabolic Interpolation (SPI) based Base Vector

Like inverse quadratic interpolation SPI uses the three points (a, $f(a)$), (b, $f(b)$), (c, $f(c)$) to find the next estimation solution. The formulation of SPI is given as below;

$$x = a + \frac{1}{2}\frac{(a-b)^2(f(a)-f(c))-(a-c)^2(f(a)-f(b))}{(a-b)(f(a)-f(c))-(a-c)(f(a)-f(b))} \quad (5)$$

Now take a=X_{tb} (let it is X_{r1}), b=X_{r2} and c=X_{r2}. and find a new vector (say X_Q) using by Equation (5). All other steps of SPI-DE are same as defined in IQI-DE.

Graphical description of SPI and IQI is given in Figure 1.

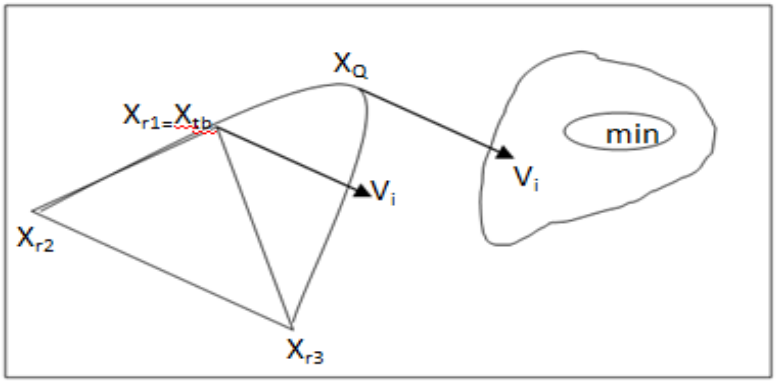


Figure 1. Differentiate between perturbed vector generated by simple DE and generated by interpolation based DE

3.3. Proposed Variants: IQI-DE and SPI-DE

The computational steps of the proposed variants are same as that of basic DE except in the selection of base vector. Further, in order to provide more diversity to the algorithm and to maintain a balance between exploration and exploitation we fixed a probability (pr) between X_{tb} and X_Q to be selected as a base vector.

Pseudo Code of Proposed Variants

```
1   BEGIN
2   Generate uniformly distributed random population
    P_G = {X_i,G, i=1,2,...NP}.
    X_i,G = X_lower + (X_upper – X_lower)*rand(0,1), where i =1,
    2,..,NP
3   Evaluate f(X_i,G)
4   while (Termination criteria is not met )
5   {
6     for i=1:NP
7       {
8         Select three vectors X_r1,G, X_r2,G and X_r3,G
          which are different from each other and
          also different from X_i,G
9         Find best vector X_tb,G among these X_r1,G,
          X_r2,G and X_r3,G
10        Put a= X_r1,G, b= X_r2,G and c= X_r3,G in
          Equation(4) and Equation (5) and
          find new vector, X_Q,G
11          if (rand (0,1)< pr )
```

	$X_{r1,G} = X_{Q,G}$ /* *for IQI using Equation (4) and for SPI using Equation(5)*/*
12	*Else*
13	$X_{r1,G} = X_{tb,G}$
14	*End if*
15	Perform mutation operation as defined by Equation (1)
16	Perform crossover operation as defined by Equation (2)
17	Evaluate $f(U_{i,G+1})$
18	Select fittest vector from $X_{i,G}$ and $U_{i,G+1}$ to the population of next Generation by using Equation-3
19	}/* End for loop*/
20	Generate new population $P_{G+1} = \{X_{i,G+1}, i=1,2,...NP\}$
21	} /* End while loop*/
22	END

3.4. Constraint Handling

For the constraint problems various methods have been suggested in literature. A survey of different methods for constraint handling can be found in [25], [26], [27] and [29]. In this paper, Pareto-Ranking method is used for handling the constraints [32].

4. Engineering Design Problems

To validate proposed IQI-DE and SPI-DE algorithms, four engineering design problem have been taken from [23];

4.1. E01: Welded Beam Engineering Design Problem

The problem is to minimize the fabrication cost of a welded beam design subject to some constraints such as shear stress τ, bending stress in the beam σ, buckling load on the bar P_c, end deflection on the beam δ and side constraints. Figure 2 shows the welded beam structure which consists of a beam A and the weld required to hold it to member B

The optimization model is given as below;

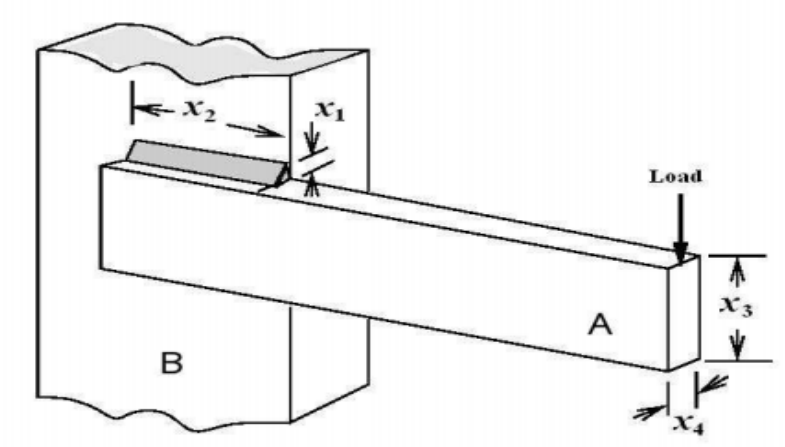


Figure 2. Welded Beam Design

$$\min f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$$

s.t.

$$g_1(x) = \tau(x) - 13600 \le 0$$

$$g_2(x) = \rho(x) - 30000 \le 0$$

$$g_3(x) = x_1 - x_4 \le 0$$

$$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5.0 \le 0$$

$$g_5(x) = 0.125 - x_1 \le 0$$

$$g_6(x) = \delta(x) - 0.25 \le 0$$

$$g_7(x) = 6000 - P_c(x) \le 0$$

where

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$$

$$\tau' = 6000/\sqrt{2}x_1x_2, \tau'' = MR/J$$

$$M = 6000(14 + (x_2/2)), R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2},$$

$$J = 2\left(x_1x_2\sqrt{2}\left[\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2\right]\right)$$

$$\sigma(x) = \frac{504000}{x_4x_3^2}, \delta(x) = \frac{65856000}{(30\times10^6)x_4x_3^3}$$

$$P_c = \frac{4.013(30\times10^6)\sqrt{\frac{x_3^2x_4^6}{36}}}{196}\left(1 - \frac{0.79056x_3}{28}\right)$$

with

$$0.1 \le x_1, x_4 \le 2.0 \text{ and } 0.1 \le x_2, x_3 \le 10.0$$

best solution : $X^* = (0.205730, 3.470489, 9.036624, 0.205729)$,

$$f(X^*) = 1.724852$$

4.2. E02: Pressure Vessel Design Optimization Problem

A compressed air storage tank with a working pressure of 3,000 psi and a minimum volume of 750 ft 3. A cylindrical vessel is capped at both ends by hemispherical heads (Figure. 3). Using rolled steel plate, the shell is made in two halves that are joined by teo longitudinal welds to form a cylinder.

The objective of the problem is to minimize the total cost of material forming and welding of a cylindrical vessel. The design variables are: thickness x_1, thickness of the head x_2, the inner radius x_3, and the length of the cylindrical section of the vessel x_4. The variables x_1 and x_2 are discrete values which are integer multiples of 0.0625 inch.

The formal statement is:

$$\min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

s.t.

$$g_1(x) = -x_1 + 0.0193x_3 \le 0$$

$$g_2(x) = -x_2 + 0.00954x_3 \le 0$$

$$g_3(x) = -\pi x_3^2x_{34}^2 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0$$

$$g_4(x) = -x_4 - 240 \le 0$$

with

$$1\times0.0625 \le x_1, x_2 \le 99\times0.0625 \text{ and } 10 \le x_3, x_4 \le 200$$

best solution : $X^* = (0.8125, 3.4375, 42.0984, 176.6365)$,

$$f(X^*) = 6059.714335$$

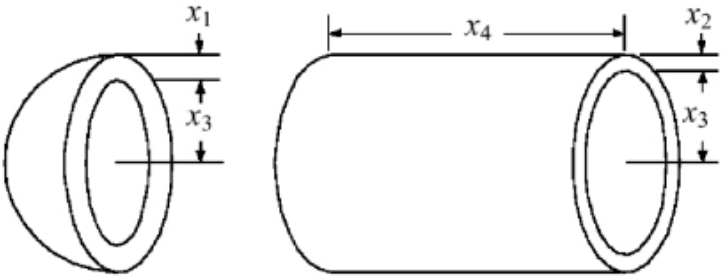


Figure 3. Pressure Vessel Design

4.3. E03: Speed Reducer Design Optimization Problem

The design of the speed reducer is shown in Figure. 4, with the face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of the first shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6, and diameter of the first shaft x_7 (all variables continuous except x_3 that is integer). The weight of the speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shaft.

The problem is:

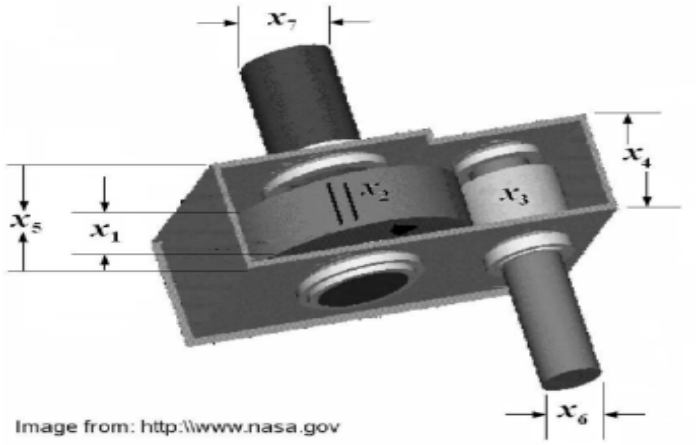


Figure 4. Speed Reducer Design

$$\min\ f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934)$$
$$-1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

s.t.

$$g_1(x) = \frac{397.5}{x_1x_2^2x_3} - 1 \le 0.$$

$$g_2(x) = \frac{27}{x_1x_2^2x_3^2} - 1 \le 0$$

$$g_3(x) = \frac{1.932x_4^3}{x_2x_3x_6^4} - 1 \le 0$$

$$g_4(x) = \frac{1.932x_5^3}{x_2x_3x_7^4} - 1 \le 0$$

$$g_5(x) = \frac{1.0}{110x_6^3}\sqrt{\left(\frac{445.0x_4}{x_2x_3}\right)^2 + 16.9\times10^6} - 1 \le 0$$

.

$$g_6(x) = \frac{1.0}{85x_7^3}\sqrt{\left(\frac{445.0x_4}{x_2x_3}\right)^2 + 157.5\times10^6} - 1 \le 0$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \le 0$$

$$g_8(x) = \frac{5x_2}{x_1} - 1 \le 0$$

$$g_9(x) = \frac{x_1}{12x_2} - 1 \le 0$$

$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

with :

$2.6 \le x_1 \le 3.6,\ 0.7 \le x_2 \le 0.8,\ 17 \le x_3 \le 28,\ 7.3 \le x_4, x_5 \le 8.3,$

$2.9 \le x_6 \le 3.9,\ 5.0 \le x_7 \le 5.5$

best solution :

$X^* = (3.50, 0.7, 17, 7.30, 7.80, 3.350214, 5.286683)$

and $f(X^*) = 2996.348165$

4.4. E04: Tension/Compression Spring Design Optimization Problem

This problem minimizes the weight of a tension/compression spring, subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables (Figure-5). There are three design variables: the wire diameter x_1, the mean coil diameter x_2, and the number of active coils x_3. The mathematical formulation of this problem is:

$$\min\ f(x) = (x_3 + 2)x_2x_1^2$$

s.t.

$$g_1(x) = 1 - \frac{x_2^3x_3}{7178x_1^4} \le 0$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566x_2x_1^3 - x_1^4} + \frac{1}{5108x_1^2} - 1 \le 0$$

$$g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0$$

$$g_3(x) = \frac{x_1 + x_2}{1.5} - 1 \le 0$$

with :

$0.05 \le x_1 \le 2.0,\ 0.25 \le x_2 \le 1.3,\ 2.0 \le x_3 \le 15.0$

best solution :

$X^* = (0.051690, 0.356750, 11.287126)$

and $f(X^*) = 0.012665$

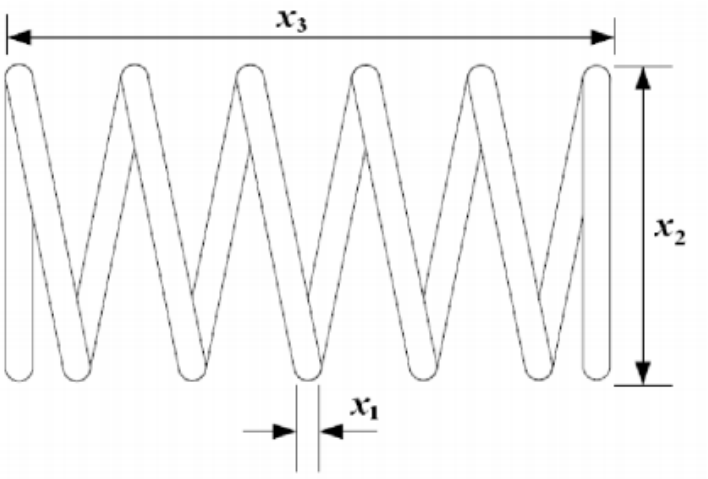


Figure 5. Tension/Compression Spring Design

6. Experimental Settings:

The following settings are taken in the present study:

- Population size (*NP*) is taken as 100 [9] [20] [21].
- Control parameters, crossover rate and scaling factor F are both fixed at 0.5 [18].
- Probability rate (*Pr*) is taken as 0.1 [31].
- Over all acceleration rate AR, which is taken for the purpose of comparison is defined as[20]:

$$AR = \frac{NFE_A - NFE_B}{NFE_A}\%$$

where *A* and *B* are different algorithms.

- In every case, a run was terminated when $|f_{\max} - f_{\min}| \le 10^{-04}$ was reached where f_{max} and f_{min} are respectively maximum and minimum fitness value [20] or when the maximum number of function evaluation (NFE=10^6) was obtained [10].
- Total Runs=50 [20], [21].
- 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.

6. Simulated Results and Analysis:

6.1. Comparison with simple DE (SDE)

Solution of engineering problems is 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-1 we can see that the comparison for E01. Here it is clear that SDE take 35480 NFE to reach the solution while total NFE taken by SPI-DE and IQI-DE are 32170 and 30370 respectively. Hence the acceleration rate of SPI-DE with respect to SDE is 9.32% while acceleration rate of IQI-DE with respect to SDE is 14.40%.

Similarly we can see results for the other engineering problems from Table-2, Table-3 and Table-4 and analysis the efficiency of SPI-DE and IQI-DE. For *E02* SPI-DE and IQI-DE are 36.75% and 37.19% respectively faster than SDE. In case of E03, SPI-DE and IQI-DE are 55.53% and 55.91% respectively faster than SDE. So we can see that for E01, E02 and E03, IQI-DE gives faster performance than SPI-DE and SDE but in case of E04, the acceleration rate of IQI-DE is 54.42% with respect to SDE while acceleration rate of SPI-DE is 57.25% with respect to SDE. Hence in case of SPI-DE gives better performance than IQI-DE.

So we can see that both SPI-DE and IQI-DE gives fast convergent speed rather than SDE and shows their superiority over SDE.

Table 1. Solution of E01 and comparisions in term of average nfe of 50 runs.

Solution	SDE	SPI-DE	IQI-DE
x_1	0.2058	0.2058	0.2058
x_2	3.4684	3.4683	3.4683
x_3	9.0367	9.0366	9.0366
x_4	0.2057	0.2057	0.2057
Mean f(x)	1.72512	1.724851	1.724851
CPU time	0.5 sec	**0.3 sec**	**0.2 sec**
Mean NFE	35480	**32170**	**30370**
AR(%)	--	**9.32**	**14.40**

Table 2. Solution of E02 and comparisions in term of average nfe of 50 runs.

Solution	SDE	SPI-DE	IQI-DE
$\mathbf{x_1}$	0.8125	0.8125	0.8125
$\mathbf{x_2}$	0.4375	0.4375	0.4375
$\mathbf{x_3}$	42.1085	42.1085	42.1085
$\mathbf{x_4}$	176.65	176.63	176.63
Mean f(x)	6060.93	6059.7143	6059.7143
CPU time	0.8 sec	**0.5 sec**	**0.5 sec**
Mean NFE	77840	**49220**	**48890**
AR(%)	--	**36.76**	**37.19**

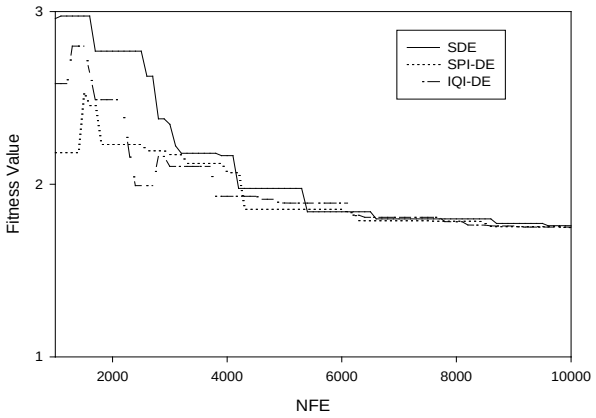
Table 3. Solution of E03 and comparisions in term of average nfe of 50 runs.

Solution	SDE	SPI-DE	IQI-DE
$\mathbf{x_1}$	3.4999	3.4999	3.4999
$\mathbf{x_2}$	0.7	0.7	0.7
$\mathbf{x_3}$	17.0	17.0	17.0
$\mathbf{x_4}$	7.3	7.3	7.3
$\mathbf{x_5}$	7.8	7.8	7.8
$\mathbf{x_6}$	3.35021	3.35021	3.35021
$\mathbf{x_7}$	5.28668	5.28668	5.28668
Mean f(x)	2996.31	2996.31	2996.31
CPU time	0.5 sec	**0.1 sec**	**0.1 sec**
Mean NFE	34480	**15330**	**15200**
AR(%)	--	**55.53**	**55.91**

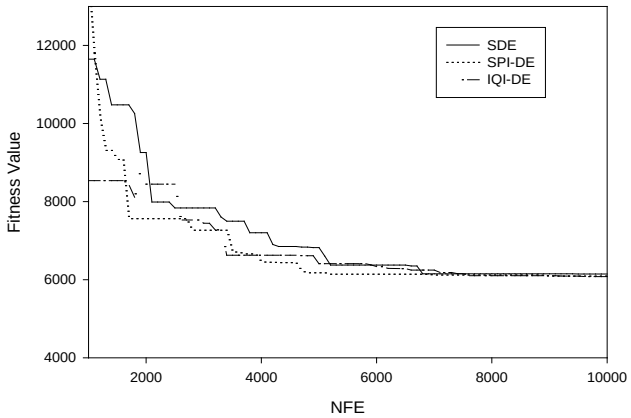
Table 4. Solution of E04 and comparisions in term of average nfe of 50 runs.

Solution	SDE	SPI-DE	IQI-DE
$\mathbf{x_1}$	0.05169	0.05169	0.05169
$\mathbf{x_2}$	0.3568	0.3567	0.3567
$\mathbf{x_3}$	11.2914	11.2871	11.2871
Mean f(x)	0.012675	0.012665	0.012665
CPU time	0.1 sec	**0.05** sec	**0.06 sec**
Mean NFE	7790	**3330**	**3550**
AR(%)	--	**57.25**	**54.42**

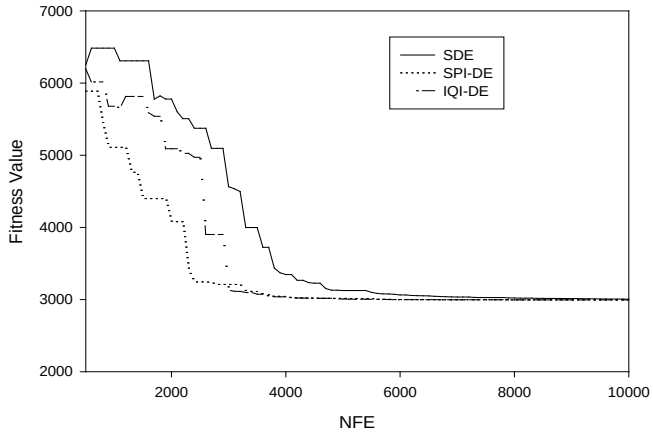
In Figure 6, the convergence graphs for engineering problems are given in term of NFE and fitness values. Here we would like to mention that these graphs are based on the best results in 50 runs by each algorithm.



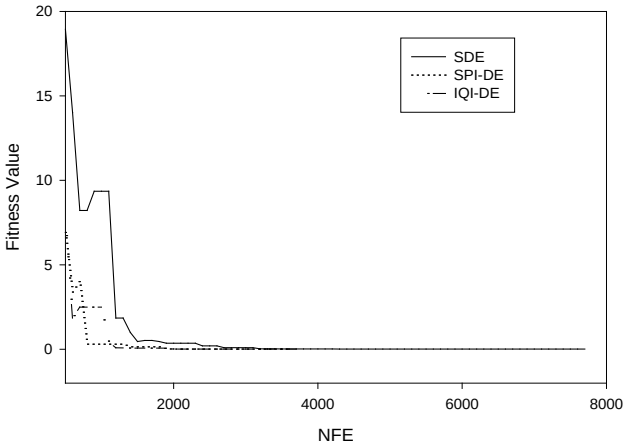
(a) E01



(b) E02



(c) E03



(d) E04

Figure 6. Convergence graphs for (a) E-01 (b) E-02 (c) E-03, (d) E-04

6.2. Comparison with Other Algorithms

Comparison of proposed SPI-DE and IQI-DE with other algorithms, CPSO [22], SicPSO [23], CoPSO [24], Coello [25], Coello and Montes [26] , He et al [28], and MBFOA [30], are given in Table-5. These results are given in term of average fitness which is taken from different literature. From Table-5 it can be see that every algorithm is able to solve the given set of problems but our proposed SPI-DE and IQI-DE gives the better solution in the comparison of other evolutionary algorithms.

Table 5. Comparisons of Proposed SPI-DE and IQI-DE with other evolutionary algorithms in terms of average fitness value

Algorithms	***Problems***			
	E01	***E02***	***E03***	***E04***
CoPSO	1.72485	**6,059.7143**	2,996.3481	**0.012665**
Coello	1.74830	6288.7445	NA	0.012704
Coello & Montes	1.72822	6059.9463	NA	0.012681 0
SicPSO	1.72485	**6,059.7143**	2,996.3481	**0.012665**
CPSO	1.72802	6061.0777	NA	0.012674
He et al	2.381	6059.7143	NA	0.012671
MBFOA	2.386	6060.460	NA	**0.012665**
SPI-DE	**1.724851**	**6059.7143**	**2996.31**	**0.012665**
IQI-DE	**1.724851**	**6059.7143**	**2996.31**	**0.012665**

7. Conclusions

In the present study, two modified methods are suggested for selecting the base vector. Both methods are based on interpolation: inverse quadratic interpolation and sequential parabolic interpolation. The corresponding DE variants are named IQI-DE and SPI-DE. Basically the idea behind these variants; IQI and SPI is to exploit the local information of the search domain for generating the base vector. The proposed IQI-DE and SPI-DE are validated on a set of 4 engineering design problems. All the problems are non linear and constrained in nature and the numerical results are compared with basic DE and also with other methods previously used for solving these problems. Form the results; it was observed that the proposed variants are quite competent for solving such problems and give the superior performance in the comparison of SDE and some other enhance version of evolutionary algorithms.

Acknowledgements

The author's would like to thank the editor's of this special issue: Tarun Kumar Sharma and the unknown reviewers/ referees for giving their valuable suggestions, which helped us in improving the shape of the paper.

References

[1] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11:4 (1997) 341–359.

[2] T. K. Sharma, M. Pant, V.P. Singh, Adaptive Bee Colony in an Artificial Bee Colony for Solving Engineering Design Problems, Advances in Computational Mathematics and its Applications, 1:4(2012) 213-221.

[3] Y. Ali, S. Iman, Optimal tuning of TCSC controller using particle swarm optimization, Advances in Electrical Engineering Systems, 1:1 (2012) 24-29.

[4] Y. Zhang, L. Wu, Rigid Image Registration by PSOSQP Algorithm, Advances in Digital Multimedia, 1:1(2012) 4-8.

[5] Z. Zhang, Pattern recognition by PSOSQP and rule based system, Advances in Electrical Engineering Systems, 1:1 (2012) 30-34

[6] T. K. Sharma, M. Pant, V.P. Singh, Improved local search in artificial bee colony using golden section search, Journal of Engineering, 1:1 (2012) 14-19.

[7] S. Wang, L. Wu, An improved PSO for bankruptcy prediction, Advances in Computational Mathematics and its Applications, 1:1 (2012) 1-6.

[8] A. Ghoreishi, A. Ahmadivand, State feedback design aircraft landing system with using differential evolution algorithm, Advances in Computer Science and its Applications, 1:1 (2012) 16-20.

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

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

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

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

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

[14] S. Das, A. Abraham, U. Chakraborty, A. Konar, Differential evolution using a neighborhood based mutation operator, IEEE Transaction of Evolutionary Computing, 13:3 (2009) 526–553.
[15] P. Kaelo, M.M. Ali, A numerical study of some modified differential evolution algorithms, European Journal of Operational Research, 169 (2006) 1176-1184.
[16] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Computing Fusion Found Method Appl., 9:6 (2005) 448–462.
[17] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Transaction of Evolutionary Computing, 12:1 (2008) 107–125.
[18] M. Pant, M. Ali, A. Abraham, Mixed mutation strategy embedded differential evolution, IEEE Congress on Evolutionary Computation, 2009, pp. 1240-1246.
[19] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transaction of Evolutionary Computing, 13:2, (2009) 398–417.
[20] S. Rahnamayan, H. Tizhoosh, M. Salama, Opposition based differential evolution, IEEE Transaction of Evolutionary Computing, 12:1 (2008) 64–79.
[21] J. Zhang, A. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Transaction of Evolutionary Computing, 13:5 (2009) 945–958.
[22] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Engineering Applications of Artificial Intelligence, 2007, pp. 89-99.
[23] L.C. Cagninal, S.C. Esquive, Solving engineering optimization problems with the simple constrained particle swarm optimizer, Informatica, 32 (2008) 319–326.
[24] A.H. Aguirre, A.M. Zavala, E.V. Diharce, S.B. Rionda, COPSO: Constrained optimization via PSO algorithm, Technical Report No. I-07-04/22-02-2007, Center for Research in Mathematics (CIMAT), 2007.
[25] C.A.C. Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Computers in Industry, 41 (2000) 113–127.
[26] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Advanced Engineering Informatics, 16 (2002) 193–203.
[27] C.A.C. Coello, Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art, Computer Methods in Applied Mechanics and Engineering, 191:11-12 (2002), 1245–1287.
[28] S. He, E. Prempain, Q.H. Wu, An improved particle swarm optimizer for mechanical design optimization problems, Engineering Optimization, 36:5 (2004) 585–605.
[29] L. Jouni, A constraint handling approach for differential evolution algorithm,. in: Proceeding IEEE Congress on Evolutionary Computation (CEC 2002), 2002, pp. 1468-1473.
[30] E.M. Montes, B.H. Ocaña, Modified bacterial foraging optimization for engineering design, In: Dagli Cihan H, et al., editors. Proceedings of the Artificial neural networks in engineering conference (ANNIE'2009), ASME Press series, Intelligent engineering systems through artificial neural networks, 19, 2009, pp. 357–364.
[31] P. Kumar, M. Pant, V.P. Singh, Modified mutation operators for differential evolution,. in: Proceeding International Conference of Soft Computing for Problem Solving (SOCPROS-2011), Springer, 2011, pp 579-588.
[32] T. Ray, T. Kang, S.K. Chye, An evolutionary algorithm for constraint optimization, In Whitley, D., Goldberg, D, Cantu-Paz, E., Spector, L.,Parmee, I., Beyer, H.G., eds.: Proceeding of the Genetic and Evolutionary Computation Conference (GECCO 2000) 2000, pp. 771-777.

Vitae

Mr. Pravesh Kumar was born in Baraut, India. He obtained Msc (mathematics) degree in 2006 in Mathematics department from CCS University Meerut, India. After that he received ME (software engineering) degree in 2009 in Computer Science and Engineering department from Thapar University Patiala, India. He worked as a research scholar in the Department of Applied Science and Engineering, IIT Roorkee, India. His research interest includes Global optimization, Evolutionary algorithms such as PSO, and Differential evolution, Soft computing etc. He has been published and presented more than 15 research paper in various international journals and conferences.

Dr Millie Pant is working as an Assistant Professor in Department of Applied Science and Engineering, Indian Institute of Technology (IIT), Roorkee, India since 2007. Her research interest includes evolutionary and swarm intelligence algorithms and their applications in various complex engineering design problems.

.