

# Multiobjective Optimization Using Differential Evolution

B.V. Babu\*, J.H. Syed Mubeen and Pallavi G. Chakole

Department of Chemical Engineering  
Birla Institute of Technology & Science  
Pilani 333031 (Rajasthan) India

## Abstract

Real world engineering design problems are usually characterized by the presence of many conflicting objectives. The optimal solution to such problems results in a number of trade-off solutions, rather than a unique solution. The use of evolutionary algorithms for solving such problems has attracted much attention recently, as they can find multiple optimal solutions in one single simulation run due to their population based search approach. In this study some of the limitations of Non-dominated Sorting Genetic Algorithm (NSGA), a multiobjective evolutionary algorithm are addressed using Differential Evolution (DE), which is an improved version of Genetic Algorithm (GA). The solutions provided by Multiobjective Differential Evolution (MODE) for three standard test problems and an engineering problem are compared with those obtained using NSGA, and the proposed algorithm is found to give encouraging results.

*Keywords: Genetic Algorithm; Differential Evolution; Multiobjective optimization; Pareto optimal front*

## I. INTRODUCTION

Optimization refers to finding one or more feasible solutions, which correspond to extreme values of one or more objectives. A significant portion of research and application in the field of optimization considers a single objective, although most of the natural world problems involve the use of more than one objective which are conflicting in nature. When an optimization problem involves more than one objective function, the task of finding one or more optimum solutions is known as multi-objective optimization (MOOP) [1]. In these situations, the aim is to simultaneously optimize a group of conflicting objectives. Different solutions may produce trade-offs (conflicting scenarios) among different objectives. A solution that is extreme (in a better sense) with respect to one objective requires a compromise in other objectives [2]. In multiobjective optimization, there may not exist a solution that is best with respect to all objectives. Instead, there are equally

good solutions which are known as Pareto optimal solutions [1].

A Pareto optimal set of solution is such that when we go from any one point to another in the set, atleast one objective function improves and at least one other worsens [3]. Neither of the solutions dominate over each other. All the sets of decision variables on the Pareto front are equally good and is expected to provide flexibility for the decision maker. Normally, the decision about “what the best answer is” corresponds to the so-called decision maker [4].

Traditionally, there are several methods available in the literature for solving MOOP problems [1]. These methods follow a preference-based approach, where a relative preference vector is used to scalarize multiple-objectives. Since classical search and optimization methods use a point by point approach, where one solution in each iteration is modified to a different solution, the outcome of using a classical optimization method is a single optimized solution. However, Evolutionary Algorithms (EAs) can find multiple optimal solutions in one single simulation run due to their population-based search approach. Thus, EAs are ideally suited for multi-objective optimization problems. A detailed account of multi-objective optimization using evolutionary algorithms and some of the applications using genetic algorithms can be found in literature [1, 5, 6, 7].

Recently, Bhaskar et al. [8] have carried out the multiobjective optimization of reactors producing fiber grade polyethylene terephthalate (PET) using NSGA. They reported that the multiobjective optimization problem had a unique optimal solution (no Pareto set was obtained). Also they found that several minima in the decision variable space were present, where the values of the two objective functions were almost identical for all practical purposes [9]. This test problem triggered us to search for a robust MOOP algorithm. As DE is found to give better results than GA for single objective optimization problems [10, 11, 12, 13, 14], we tried to extend the application of DE to MOOP problems.

\* Corresponding Author: Assistant Dean-ESD & Head-Chemical Engineering Department, B.I.T.S., PILANI – 333031 (Rajasthan) INDIA.

Email: [bvbabu@bits-pilani.ac.in](mailto:bvbabu@bits-pilani.ac.in);

Phone: +91-01596-245073 Ext. 205 / 224;

Home Page: <http://discovery.bits-pilani.ac.in/discipline/chemical/BVb>;

Fax: +91-01596-244183;

The approach shows promising results when compared with the NSGA results. The paper is organized as follows: concepts about DE are explained in Section II followed by the proposed algorithm in Section III. Experiments are then presented in Section IV with discussion in Section V and conclusions are drawn in Section VI.

## II. DIFFERENTIAL EVOLUTION (DE)

Differential Evolution [15] is an improved version of GA [16] for faster optimization. Unlike simple GA that uses binary coding for representing problem parameters, DE is a simple yet powerful population based, direct search algorithm for globally optimizing functions with real valued parameters. Among the DE's advantages are its simple structure, ease of use, speed and robustness. Price & Storn [15] gave the working principle of DE with single strategy. Later on, they suggested ten different strategies of DE [17]. A strategy that works out to be the best for a given problem may not work well when applied for a different problem. Also, the strategy and key parameters to be adopted for a problem are to be determined by trial & error. The schematic diagram in Figure 1 provides a way to visualize the working principle of DE. Pseudocode for solving single objective optimization problems using DE is discussed in Section 1.1. The crucial idea behind DE is a scheme for generating trial parameter vectors. Basically, DE adds the weighted difference between two population vectors to a third vector. The key parameters of control in DE are:  $NP$ -the population size,  $CR$ -the crossover constant, and  $F$ -the weight applied to random differential (scaling factor). Price & Storn [17] have given some simple rules for choosing key parameters of DE for any given application. Normally,  $NP$  should be about 5 to 10 times the dimension (number of parameters in a vector) of the problem. As for  $F$ , it lies in the range 0.4 to 1.0. Initially  $F = 0.5$  can be tried then  $F$  and/or  $NP$  is increased if the population converges prematurely. As for  $CR$ , it lies in the range 0.1 to 1.0. Babu et al. [13] proposed a new concept called 'nested DE' to automate the choice of DE key parameters. In addition, some new strategies have been proposed and successfully applied to optimization of extraction process [18].

DE has been successfully applied in various fields. Some of the successful applications of DE include: digital filter design [19], batch fermentation process [20, 21], estimation of heat transfer parameters in trickle bed reactor [22], optimal design of heat exchangers [10, 11], synthesis & optimization of heat integrated distillation system [23], optimization of an alkylation reaction [24], optimization of non-linear functions [25], optimization of thermal cracker operation [12], global optimization of MINLP problems [14], optimization of water pumping system [26], optimization of biomass pyrolysis [27], etc. Many engineering applications using various evolutionary algorithms have been reported in literature [28, 29, 30].

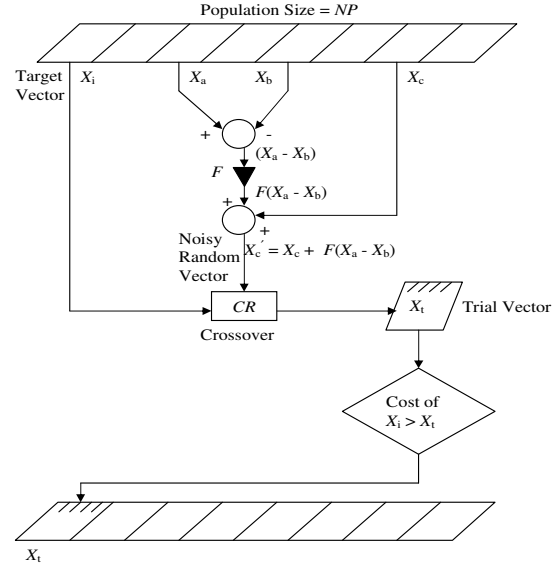


Figure 1 Schematic representation of DE algorithm

### A. Pseudocode for DE

The pseudo code of DE for single objective optimization is given below:

- Choose a seed for the random number generator.
- Initialize the values of  $D$ ,  $NP$ ,  $CR$ ,  $F$  and  $MAXGEN$  (maximum generation).
- Initialize all the vectors of the population randomly. The variables are normalized within the bounds.
 

```
for i = 1 to NP
  { for j = 1 to D
     $X_{ij} = \text{Lower bound} + \text{random number} * (\text{upper bound} - \text{lower bound})$ 
  }
```
- All the vectors generated should satisfy the constraints. Penalty function approach, i.e., penalizing the vector by giving it a large value, is followed only for those vectors, which do not satisfy the constraints.
- Evaluate the cost/profit of each vector. Cost/profit here is the value of the objective function to be minimized/maximized.
 

```
for i = 1 to NP
   $C_i = \text{funct}()$ 
```
- Perform mutation, crossover, selection and evaluation of the objective function for a specified number of generations.
 

```
While (gen < MAXGEN)
  { for i = 1 to NP
    {
```
- For each vector  $X_i$  (target vector), select three distinct vectors  $X_a$ ,  $X_b$  and  $X_c$  randomly from the current population other than the vector  $X_i$ 

```
do
  {  $r_1 = \text{random number} * NP$ 
     $r_2 = \text{random number} * NP$ 
     $r_3 = \text{random number} * NP$ 
  }
```
- while
 

```
( $r_1=i$ )OR( $r_2=i$ )OR( $r_3=i$ )OR( $r_1=r_2$ )OR( $r_2=r_3$ )OR( $r_1=r_3$ )
```

- Perform crossover for each target vector  $X_i$  with its noisy vector  $X_{n,i}$  and create a trial vector,  $X_{t,i}$ . The noisy vector is created by performing mutation.
- for binomial crossover
 

```

      {
        p = random number
        j_rand = int(rand[0,1]*D)+1
        for n = 1 to D
          { if (p < CR or n = j_rand)
              X_{n,i} = X_{a,i} + F ( X_{b,i} - X_{c,i} )
              X_{t,i} = X_{n,i}
            } else X_{t,i} = X_{i,j}
          }
      
```
- Again, the NP noisy random vectors that are generated should satisfy the constraints.
- Perform selection for each target vector,  $X_i$  by comparing its cost/profit with that of the trial vector,  $X_{t,i}$ ; for maximization problem whichever has the maximum profit will survive for the next generation.
 

```

      /* for i=1 to NP */
      C_{t,i} = funct()
      if (C_{t,i} > C_i )
        new X_i = X_{t,i}
      else new X_i = X_i }
      
```
- Print the results (after the stopping criteria is met).  
The stopping criteria may be of two kinds. One may be some convergence criterion that states that the error in the minimum or maximum between two previous generations should be less than some specified value (standard deviation may be used). The other may be an upper bound on the number of generations. The stopping criteria may be a combination of the two as well.

### III. MULTIOBJECTIVE DIFFERENTIAL EVOLUTION (MODE)

The pseudocode of the proposed algorithm for MOOP using DE is presented in Section 2.1. The DE algorithm for MOOP is similar to the one presented in Section 1.1 but with the following changes. In each generation, the dominated solutions are removed from the list and only the non-dominated solutions are allowed to undergo DE operations. The scaling factor  $F$  is generated from a random generator between 0 and 1. The offspring's are placed into the population if they dominate the main parent.

The algorithm works as follows: An initial population is generated at random. All dominated solutions are removed from the population. The remaining non-dominated solutions are retained for recombination. Three parents are selected at random. A child is generated from the three parents and is placed into the population if it dominates the first selected parent; otherwise a new selection process takes place.

#### A. Pseudocode for MODE

The pseudocode of MODE used in the present study is given below:

- Choose a seed for the random number generator.
- Initialize the values of  $D$ ,  $NP$ ,  $CR$  and  $MAXGEN$  (maximum generation).

- Initialize all the vectors of the population randomly. The variable are normalized within the bounds.
 

```

      for i = 1 to NP
        { for j = 1 to D
            X_{i,j} = Lower bound + random number *( upper bound - lower bound) }
      
```
- All the vectors generated should satisfy the constraints (if present). Penalty function approach, i.e., penalizing the vector by giving it a large value, is followed only for those vectors, which do not satisfy the constraints.
- Evaluate the functions of each vector.
 

```

      for i = 1 to NP
        C_{i,j} = funct_j()      j=1,..., no of objectives
      
```
- Remove all the dominated solutions using any one of the approaches proposed(Deb.,2001).
- Perform mutation, crossover, selection and evaluation of the objective function for non-dominated solutions for a specified number of generations.
 

```

      While (gen < MAXGEN)
      { for i = 1 to number of Non-dominated solutions
          {
            • For each vector  $X_i$  (target vector), select three distinct vectors  $X_a$ ,  $X_b$  and  $X_c$  randomly from the current population other than the vector  $X_i$ 
              do
                { r_1 = random number * NP
                    r_2 = random number * NP
                    r_3 = random number * NP
                  }
                while(r_1=i)OR(r_2=i)OR(r_3=i)OR(r_1=r_2)OR(r_2=r_3)OR(r_1=r_3)
            }
          }
      
```
- Perform crossover for each target vector  $X_i$  with its noisy vector  $X_{n,i}$  and create a trial vector,  $X_{t,i}$ . The noisy vector is created by performing mutation.
- for binomial crossover
 

```

      { p = random number
        j_rand = int(rand[0,1]*D)+1
        for n = 1 to D
          { if (p < CR or n = j_rand)
              X_{n,i} = X_{a,i} + F ( X_{b,i} - X_{c,i} )
              X_{t,i} = X_{n,i}
            } else X_{t,i} = X_{i,j}
          }
      
```
- Again, the NP noisy random vectors that are generated should satisfy the constraints.
- Perform selection for each target vector,  $X_i$  by comparing its function value with that of the trial vector,  $X_{t,i}$ . If  $X_{t,i}$  dominates completely  $X_i$  then replace  $X_i$  by  $X_{t,i}$ 

```

      if (C_{t,i} dominates C_i )
        new X_i = X_{t,i}
      else new X_i = X_i }
      /* return the set of nondominated solutions */
      }
      
```
- Print the results (after the stopping criteria is met).

The stopping criteria may be of two kinds: 1) There is no new solution added to the non-dominated front for a specified number of generations, or 2) Assign an upper bound on the number of generations. The stopping criteria may be a combination of the two as well. However, in this study, the second criteria is applied.

#### IV. EXPERIMENTAL RESULTS

The algorithm is tested on the following solved problems from [1]. All the test problems contain two objective functions and two variables. Test problem 3 and 4 are MOOP problems involving constraints. The initial population is randomly generated. The scaling factor  $F$  is randomly generated for each variable. The algorithm is written in standard C++.

##### A. Test Problem 1

$$\text{Minimize } f_1(x) = x_1 \quad (1)$$

$$\text{Minimize } f_2(x) = \frac{1+x_2}{x_1} \quad (2)$$

$$\text{subject to } 0.1 \leq x_1 \leq 1 \quad (3)$$

$$0 \leq x_2 \leq 5 \quad (4)$$

##### 1) Parameters Used for DE:

Number of Population points ( $NP$ ) = 100

Number of Iterations ( $Ng$ ) = 200

DE Key Parameters:

Scaling Factor ( $F$ ) = Using random generator

Cross-over Constant ( $CR$ ) = 0.15

2) **Simulation Results:** In Figure 2, all the non-dominated solutions obtained using MODE are plotted with those obtained using NSGA. As can be seen, the results obtained by both MODE and NSGA are exactly matching in terms of objective function values. Also both follow the same pareto optimal front. This simple minimization problem is selected to check the validity of the code developed. As the results are exactly matching, it is further extended to solve other complex MOOP problems.

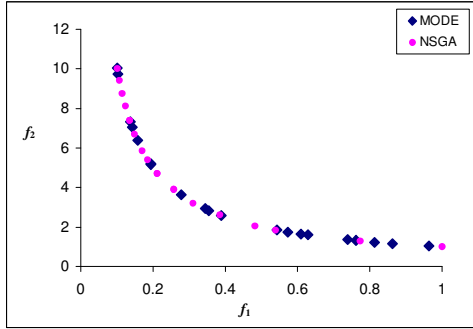


Figure 2 The performance of MODE compared with NSGA for test problem 1

##### B. Test Problem 2

$$\text{Maximize } f_1(x) = 1.1 - x_1 \quad (5)$$

$$\text{Maximize } f_2(x) = 60 - \frac{1+x_2}{x_1} \quad (6)$$

$$\text{subject to } 0.1 \leq x_1 \leq 1 \quad (7)$$

$$0 \leq x_2 \leq 5 \quad (8)$$

##### 1) Parameters Used for DE:

Number of Population points ( $NP$ ) = 100

DE Key Parameters:

Scaling Factor ( $F$ ) = Using random generator

Cross-over Constant ( $CR$ ) = 0.15

2) **Simulation Results:** In Figure 3, the non-dominated solutions that are obtained from DE are plotted with NSGA results for generations 1, 10 and 200. It is observed that NSGA captures multiple trade-off solutions for a few iterations. However, if continued for a larger number of iterations, a NSGA population tended to show poor spread of solutions ( $Ng = 200$ ). However, in DE once the Pareto optimal set is obtained, irrespective of the increase in generation number, the distribution of points on the Pareto front is the same as shown in Figure 4 ( $Ng = 1000$ ).

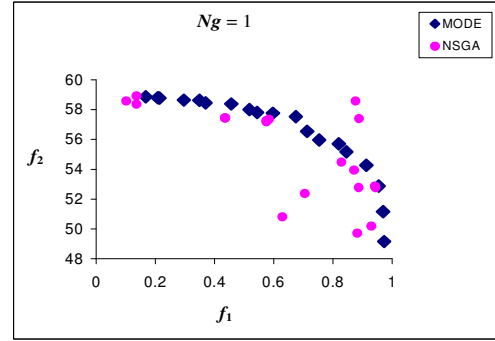


Figure 3(a)  $f_1$  vs.  $f_2$  values for different values of the generation number,  $Ng = 1$

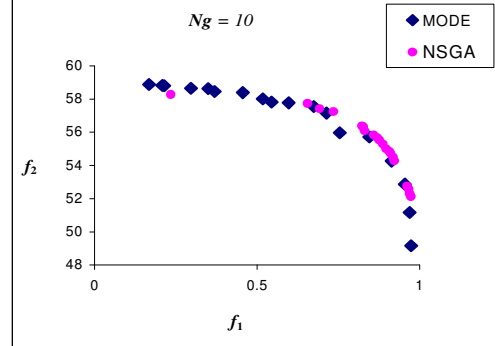


Figure 3(b)  $f_1$  vs.  $f_2$  values for different values of the generation number,  $Ng = 10$

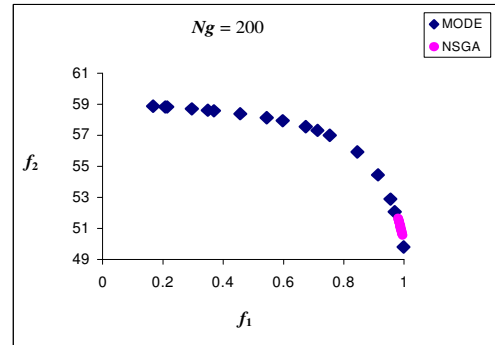


Figure 3(c)  $f_1$  vs.  $f_2$  values for different values of the generation number,  $Ng = 100$

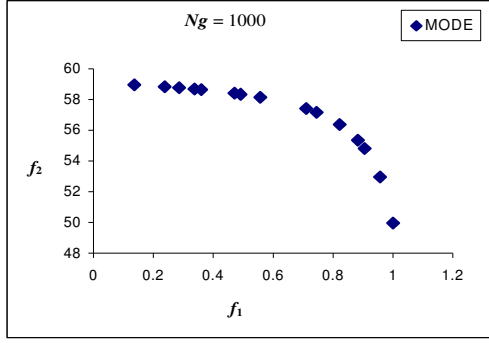


Figure 4  $f_1$  vs.  $f_2$  values for different values of the generation number,  $Ng = 1000$

### C. Test Problem 3

$$\text{Minimize } f_1(x) = x_1 \quad (9)$$

$$\text{Minimize } f_2(x) = \frac{1+x_2}{x_1} \quad (10)$$

$$\text{subject to } g_1(x) \equiv x_2 + 9x_1 \geq 6 \quad (11)$$

$$g_2(x) \equiv -x_2 + 9x_1 \geq 1 \quad (12)$$

$$0.1 \leq x_1 \leq 1 \quad (13)$$

$$0 \leq x_2 \leq 5 \quad (14)$$

#### 1) Parameters Used for DE:

Number of Population points ( $NP$ ) = 100

DE Key Parameters:

Scaling Factor ( $F$ ) = Using random generator

Cross-over Constant ( $CR$ ) = 0.15

2) **Simulation Results:** In this problem the constraints are handled using penalty function approach. This approach is generally applied to minimization problems; however, a maximization function can be handled by converting it into a minimization function using the duality principle [1]. The constraints (Eqs. 11 and 12) are normalized as shown in Eq. (15).

$$F(x) = R_1 \left[ \begin{array}{l} (6-9x_1-x_2) + \\ |(6-9x_1-x_2)| \end{array} \right] \quad (15)$$

$$+ R_2 \left[ \begin{array}{l} (1-9x_1+x_2) + \\ |(1-9x_1+x_2)| \end{array} \right]$$

where  $R_1, R_2$  are the penalty parameters. In this case both the penalty parameter values are taken to be equal ( $R_1 = R_2 = R$ ). Now, the objective function is reformulated as,

$$\text{Minimize } f_1(x) = x_1 + F(x) \quad (16)$$

$$\text{Minimize } f_2(x) = \frac{1+x_2}{x_1} + F(x) \quad (17)$$

The simulation results are carried out using different values of  $R$  for  $Ng$  equal to 200. The results are shown in Figure 5.

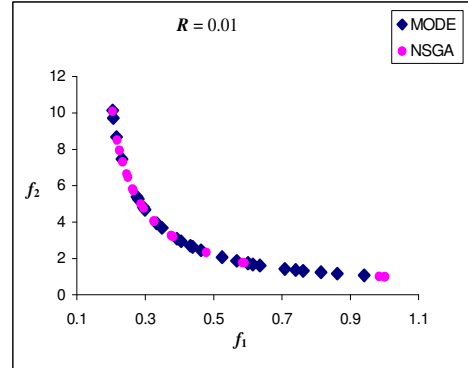


Figure 5(a)  $f_1$  vs.  $f_2$  values for penalty parameter,  $R = 0.01$  and  $Ng = 200$

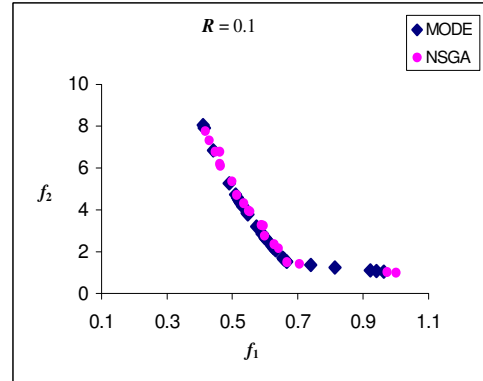


Figure 5(b)  $f_1$  vs.  $f_2$  values for penalty parameter,  $R = 0.1$  and  $Ng = 200$

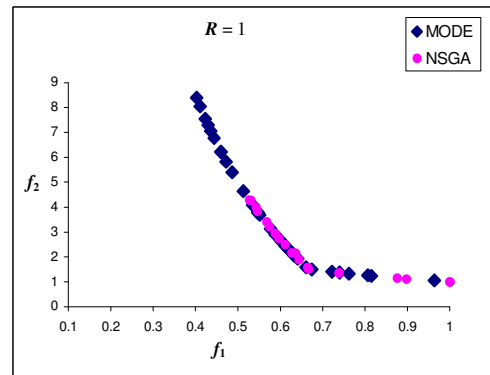


Figure 5(c)  $f_1$  vs.  $f_2$  values for penalty parameter,  $R = 1.0$  and  $Ng = 200$

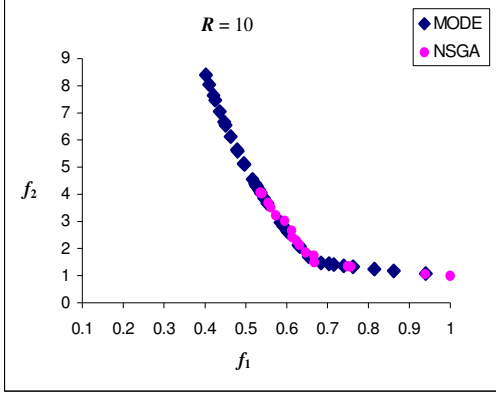


Figure 5(d)  $f_1$  vs.  $f_2$  values for penalty parameter,  $R = 10.0$  and  $Ng = 200$

When a small penalty parameter of 0.01 is used, the resulting penalized objective function forms a Pareto-optimal front different from the true Pareto-optimal front reported in [1]. This is true even when the number of generations is considerably increased as shown in Figure 6.

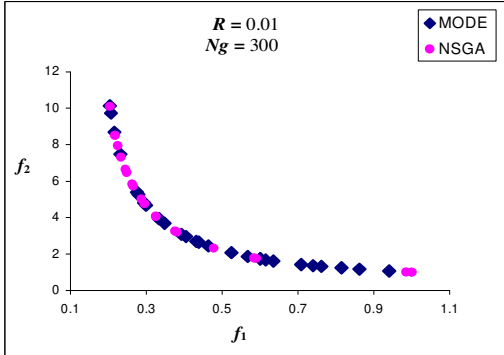


Figure 6(a)  $f_1$  vs.  $f_2$  values for generation number,  $Ng = 300$  and  $R = 0.01$

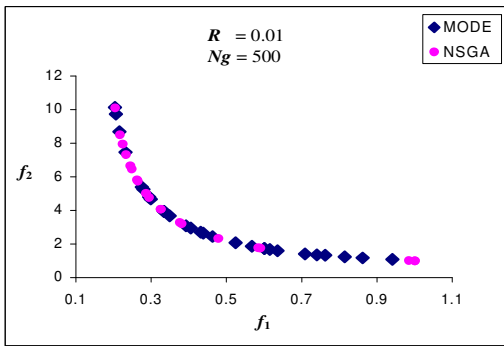


Figure 6(b)  $f_1$  vs.  $f_2$  values for generation number,  $Ng = 500$  and  $R = 0.01$

Though both DE and NSGA follows the same pattern, it is observed that MODE gives good spread of solutions than NSGA. When the penalty parameter is increased to 0.1 (Figure 5), the resulting Pareto-optimal front for the two penalized functions given in Eq.s (16) and (17) is

close to true Pareto-optimal solution reported in [1]. It is interesting to note that when the penalty parameter is increased to  $R = 1$  and  $R = 10$ , the spread of obtained solutions is not as good as that with  $R = 0.1$  in the case of NSGA. Thus when the penalty parameter is increased NSGA converges near a portion of the Pareto-optimal front. These results show, how important is the choice of penalty parameter to solve optimization problems involving constraints using NSGA. However in DE, the Pareto-optimal front is not so sensitive as NSGA. Due to its robustness, DE could find the true Pareto-optimal front even if the penalty parameter given is as high as  $R = 10000$ , as shown in Figure 7.

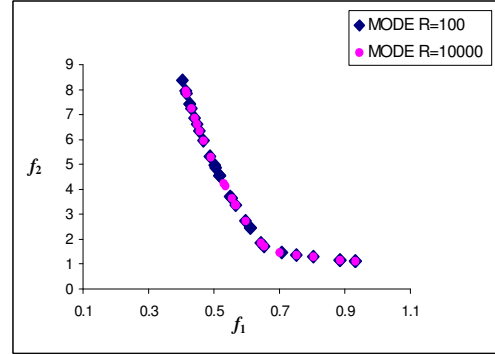


Figure 7 MODE for very high values of penalty parameter,  $R$

#### D. Test Problem 4

Application of MODE is further tested on an engineering problem. This is a cantilever design problem (Figure 8) with two decision variables, i.e., diameter ( $d$ ) and length ( $l$ ). The beam has to carry an end load  $P$ . In this problem the two conflicting objectives are minimization of the weight  $f_1$  and minimization of end deflection  $f_2$ . The first objective will resort to an optimum solution having the smaller dimensions of  $d$  and  $l$ , so that the overall weight of the beam is minimum. Since the dimensions are small, the beam will not be adequately rigid and the end deflection of the beam will be large. On the other hand, if the beam is minimized for end deflection, the dimensions of the beam are expected to be large, thereby making the weight of the beam large. In this problem two constraints are considered: 1) the developed maximum stress  $\sigma_{\max}$  is less than the allowable strength  $S_y$ , and 2) the end deflection  $\delta$  is smaller than a specified limit of  $\delta_{\max}$ . Considering all the above aspects, the optimization problem is formulated as follows:

$$\text{Minimize } f_1(d,l) = \rho \frac{\pi d^2 l}{4} \quad (18)$$

$$\text{Minimize } f_2(d,l) = \delta = \frac{64Pl^3}{3E\pi d^4} \quad (19)$$

$$\text{subject to } \sigma_{\max} \leq S_y \quad (20)$$

$$\delta \leq \delta_{\max} \quad (21)$$

$$\sigma_{\max} = \frac{32Pl}{\pi d^3} \quad (22)$$

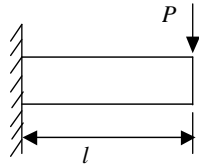
$$10 \leq d \leq 50 \text{ mm} \quad (23)$$

$$200 \leq l \leq 1000 \text{ mm} \quad (24)$$

The following parameter values are used:

$$\rho = 7800 \text{ kg/m}^3 \quad P = 1 \text{ kN} \quad E = 207 \text{ GPa}$$

$$S_y = 300 \text{ MPa} \quad \delta_{\max} = 5 \text{ mm}$$



**Figure 8** A schematic representation of a cantilever beam

### 1) Parameters Used for DE:

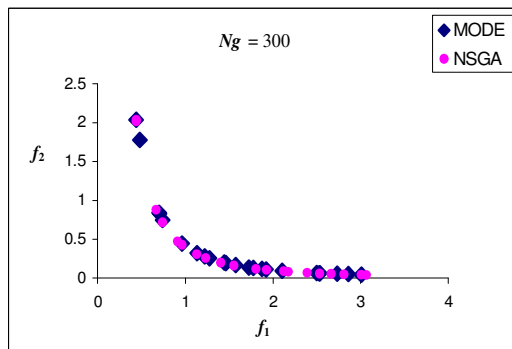
Number of Population points ( $NP$ ) = 100

DE Key Parameters:

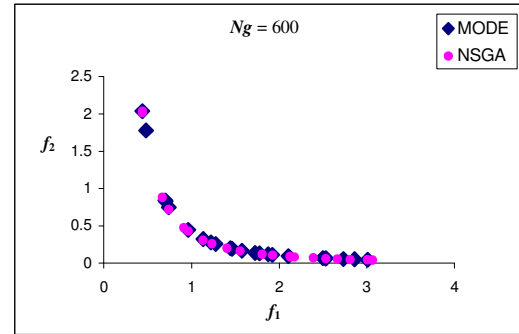
Scaling Factor ( $F$ ) = Using Random generator

Cross-over Constant ( $CR$ ) = 0.15

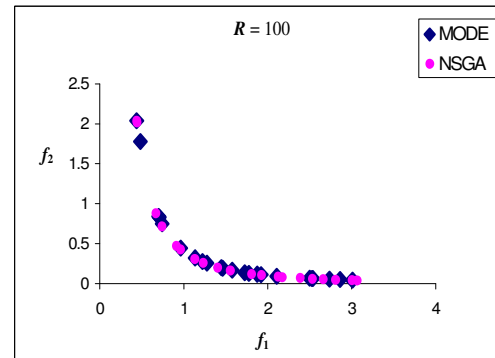
**2) Simulation Results:** Figures 9 and 10 show many solutions trading-off differently between the two objectives. It is found that the results obtained using MODE and NSGA are following the same front reported in the literature [1]. An interesting feature to be noted is that, irrespective of the increase in the number of generations ( $Ng = 300$  and  $Ng = 600$ ) for fixed value of  $R$  (Fig. 9), the pareto optimal front remains the same. The same is also seen with increase in penalty parameter values ( $R = 100$  and  $R = 1000$ ) for fixed number of generations (Fig. 10). This observation is different from that observed using NSGA in problems 2 and 3. This shows that NSGA is problem specific, while DE is robust to all the problems considered. Thus the robustness of MODE is also tested by applying to an engineering problem.



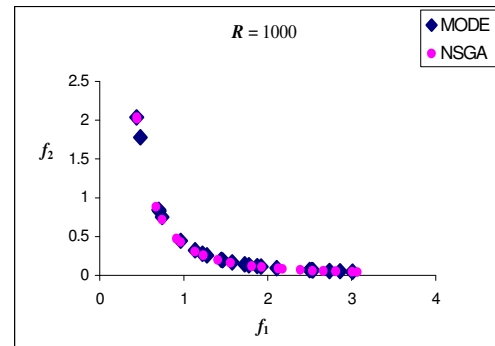
**Figure 9(a)**  $f_1$  vs.  $f_2$  values for generation number,  $Ng = 300$  and  $R = 10$



**Figure 9(b)**  $f_1$  vs.  $f_2$  values for generation number,  $Ng = 600$  and  $R = 10$



**Figure 10(a)**  $f_1$  vs.  $f_2$  for Penalty parameter,  $R = 100$  and  $Ng = 200$



**Figure 10(b)**  $f_1$  vs.  $f_2$  for Penalty parameter,  $R = 1000$  and  $Ng = 200$

## V. DISCUSSION

As shown in Figure 3 and 5, DE is able to generate uniform distribution of solutions even at high generations and at high penalty parameter values. This robustness of DE is attributed to its mutation operator. A real coded GA generates mutation vectors by adaptively scaling and correlating the output of pre-defined, multivariate probability distribution, whereas DE just randomly samples the object vector population for them. As Eq. (25) shows, DE mutates an object vector by adding the weighted difference of a randomly chosen pair of object vectors to it:

$$\begin{aligned}
& r_1, r_2 \in \{1, 2, \dots, NP\}, \\
& \text{randomly selected,} \\
& \text{except: } r_1 \neq r_2 \neq i \\
& u_{i,G+1} = x_{i,G} + F(x_{r_1,G} - x_{r_2,G})
\end{aligned} \tag{25}$$

where,  $u_{i,G+1}$  is the offspring formed and  $Ng$  represents the generation number.

The possibility of  $r_1 = r_2$  can safely be eliminated, since no mutation ever occurs with this combination of indices. Also, the cases of  $r_1 = i$  and  $r_2 = i$  are excluded because they effectively transform Eq. (25) into a crossover procedure. In fact, DE's simple mutation scheme is often mistaken for real coded GA crossover and a comparison of Eq.s (25) and (26) shows why:

$$\begin{aligned}
& r_3 \in \{1, 2, \dots, NP\}, \\
& \text{randomly selected,} \\
& \text{except: } r_3 \neq i \\
& u_{i,G+1} = x_{i,G} + k_x(x_{r_3,G} - x_{i,G})
\end{aligned} \tag{26}$$

Eq. (25) is the DE mutation operation, whereas Eq. (26) is the formula for real coded GA crossover. The difference, of course, is that Eq. (26) is a general linear combination of two vectors, whereas Eq. (25) is a special case of linear combination of three vectors. In both cases, new vectors are generated that are linear combinations of existing object vectors.

Despite their similarity, the processes defined by Eqs. (25) and (26) search the space of object vectors in very different ways. For example, the GA crossover Eq. (26) provides a way for  $x_{i,G}$  to become more like another vector in the population. In particular, the special case  $k_x = 1$  effectively transform Eq. (26) into a replacement operation that attempts the wholesale substitution of  $x_{r_3,G}$  for  $x_{i,G}$ . For other values of  $k_x$ , Eq. (26) limits the search for a new  $x_{i,G}$  to points along one of the  $NP-1$  axes that join  $x_{i,G}$  with every other vector in the population. The randomly chosen index  $r_3$  determines which axis will be searched, while  $k_x$  determines which points on the chosen axes is actually tested. Furthermore, when added to  $x_{i,G}$ , the increment:  $k_x(x_{r_3,G} - x_{i,G})$  moves  $u_{i,G}$  either toward, or away from  $x_{r_3,G}$ . As such, any constant value of  $k_x$  other than zero introduces a bias into the search by constantly moving each vector either closer to ( $0 < k_x \leq 1$ ), or further away from ( $k_x < 0, k_x > 1$ ), other object vectors in the population.

Unlike the steps generated by Eq. (26), the mutation vectors produced by Eq. (25) do not contain any reference to  $x_{i,G}$ . In its place is a second, randomly chosen object

vector, the inclusion of which expands the number of search axes to:  $(NP-1)(NP-2)/2$ . Further more, since random sampling assures that each differential,  $(x_{r_1,G} - x_{r_2,G})$ , occurs as often as does its opposite,  $(x_{r_2,G} - x_{r_1,G})$ , the distribution that Eq. (25) generates is guaranteed to exhibit a zero mean. Consequently, a constant value of  $F$  scales the magnitude of the step sizes but does not bias the search like  $k_x$  does in Eq. (26). Additionally, there is no value of  $F$  for which Eq. (25) becomes a replacement operator, so the population's diversity cannot be threatened in this way. As has been seen, DE is a consistent evolutionary approach to real parameter optimization in which both the mutation and recombination are simple operations that randomly sample linear combination of object vectors. Thus instead of falling easy prey to the generic pathologies, DE's simple mutation scheme has proven itself to be remarkably resistant to them.

## VI. CONCLUSIONS

In this paper, a Differential Evolution approach is presented for multi-objective optimization problems. The algorithm developed is tested for four standard problems and it is found that DE algorithm (MODE) is more successful in giving good spread of solutions maintaining both diversity and convergence. The MODE results are compared with NSGA and it is shown that MODE gives a uniform distribution of solutions even when run for higher iterations and also when the penalty parameter is over emphasized. The crossover and mutation rates are selected based on trial runs, however it is seen that larger number of non-dominated solutions are obtained at low crossover rates.

## VII. REFERENCES

- [1] K. Deb, "Multi-Objective Optimization using Evolutionary Algorithms," New York: John Wiley & Sons Limited, 2001.
- [2] H.A. Abbas, R. Sarker, and C. Newton, "PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems," in *Proceedings of the 2001 Congress on Evolutionary Computation, IEEE, Piscataway, NJ, USA. ISBN 0-7803-6657-3*, 2001, Vol. 2, pp. 971-978.
- [3] A.K.Y. Yee, A.K. Ray, and G.P. Rangaiah, "Multiobjective optimization of an industrial styrene reactor," *Computers & Chemical Engineering*, vol. 27, pp. 111-130, 2003.
- [4] C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowledge and Information Systems*, vol. 3, pp. 269-308, 1999.
- [5] J.K. Rajesh, S.K. Gupta, G.P. Rangaiah, and A.K. Ray, "Multi-objective Optimization of Steam Reformer Performance Using Genetic Algorithm," *Industrial and Engineering Chemistry Research*, vol. 39, pp. 707-717, 2000.
- [6] J.K. Rajesh, S.K. Gupta, G.P. Rangaiah, and A.K. Ray, "Multi-objective Optimization of Industrial Hydrogen

- Plants," *Chemical Engineering Science*, vol. 56, pp. 999-1010, 2001.
- [7] P.P. Oh, G.P. Rangaiah, and A.K. Ray, "Simulation and Multi-objective Optimization of an Industrial Hydrogen Plant Based on Refinery Off-gas," *Industrial and Engineering Chemistry Research*, vol. 41, pp. 2248-2261, 2002.
- [8] Bhaskar, S.K.Gupta, and A.K. Ray, "Multiobjective optimization of an industria wiped film PET reactor," *American Institute of Chemical Engineering Journal*, vol. 46(5), pp. 1046-1058, 2000.
- [9] Bhaskar, S.K.Gupta, and A.K. Ray, "Multiobjective optimization of an industria wiped film poly(ethylene terephthalate) reactor: some further insights," *Computers and Chemical Engineering*, vol. 25, pp. 391-407, 2001.
- [10] B.V. Babu and S.A. Munawar (2000, March 11), "Differential Evolution for the Optimal Design of Heat Exchangers," in *Proceedings of All-India seminar on Chemical Engineering Progress on Resource Development: A Vision 2010 and Beyond*, IE (I).
- [11] B.V. Babu and S.A. Munawar (2001), "Optimal Design of Shell & Tube Heat Exchanger by Different strategies of Differential Evolution," *PreJournal.com - The Faculty Lounge, Article No. 003873*, Available: <http://www.prejournal.com>.
- [12] B.V. Babu and R. Angira (2001, December 19), "Optimization of Thermal Cracker Operation using Differential Evolution," in *Proceedings of International Symposium & 54th Annual Session of IChE (CHEMCON-2001)*.
- [13] B.V. Babu, R. Angira, and A. Nilekar, "Differential Evolution for Optimal Design of an Auto-Thermal Ammonia Synthesis Reactor," *Proceedings of The Eighth World Multi-Conference on Systemics, Cybernetics and Informatics (SCI-2004)*, July 18-21, 2004, Orlando, Florida, USA.
- [14] B.V. Babu and R. Angira (2002, November 18), "A Differential Evolution Approach for Global Optimization of MINLP Problems," in *Proceedings of 4<sup>th</sup> Asia Pacific Conference on Simulated Evolution and Learning (SEAL-2002)*, Vol. 2, pp. 880-884.
- [15] K. Price and R. Storn, "Differential Evolution - A simple evolution strategy for fast optimization," *Dr. Dobb's Journal*, vol. 22 (4), pp.18 – 24 and 78, 1997.
- [16] D.E. Goldberg, "Genetic Algorithms in search, Optimization, and Machine learning," MA: Addison-Wesley, 1989.
- [17] K. Price and R. Storn, 2003, Home Page of Differential Evolution, Available: <http://www.ICSI.Berkeley.edu/~storn/code.html>.
- [18] B.V. Babu and R. Angira, "New Strategies of Differential Evolution for Optimization of Extraction Process," Presented at International Symposium & 56th Annual Session of IChE (CHEMCON-2003), Bhubaneswar, 2003.
- [19] R. Storn, "Differential Evolution design of an IIR-filter with requirements for magnitude and group delay," *International Computer Science Institute*, TR-95-026, 1995.
- [20] J.P. Chiou and F.S. Wang, "Hybrid Method of Evolutionary Algorithms for Static and Dynamic Optimization Problems with Application to a Fed-batch Fermentation Process," *Computers & Chemical Engineering*, vol. 23, pp. 1277-1291, 1999.
- [21] F. S. Wang and W.M. Cheng, "Simultaneous optimization of feeding rate and operation parameters for fed-batch fermentation processes," *Biotechnology Progress*, vol.15 (5), pp. 949-952, 1999.
- [22] B.V. Babu and K.K.N. Sastry (1999), "Estimation of Heat-transfer Parameters in a Trickle-bed Reactor using Differential Evolution and Orthogonal Collocation," *Computers & Chemical Engineering*, vol. 23, pp. 327-339.
- [23] B.V. Babu and R.P. Singh, "Synthesis & optimization of Heat Integrated Distillation Systems Using Differential Evolution," in *Proceedings of All-India seminar on Chemical Engineering Progress on Resource Development: A Vision 2010 and Beyond*, IE (I), 2000.
- [24] B.V. Babu and C. Gaurav (2000, December 18), "Evolutionary Computation Strategy for Optimization of an Alkylolation Reaction," in *Proceedings of International Symposium & 53rd Annual Session of IChE (CHEMCON-2000)*.
- [25] B.V. Babu and R. Angira (2001, January 10), "Optimization of Non-linear functions using Evolutionary Computation," in *Proceedings of 12<sup>th</sup> ISME Conference*, India, 153-157.
- [26] B.V. Babu and R. Angira, "Optimization of Water Pumping System Using Differential Evolution Strategies," *Proceedings of The Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems (CIRAS-2003)*, Singapore, 2003.
- [27] B.V. Babu and A.S. Chaurasia, "Optimization of Pyrolysis of Biomass Using Differential Evolution Approach," *Proceedings of The Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems (CIRAS-2003)*, Singapore, 2003.
- [28] D. Dasgupta, and Z. Michalewicz, "Evolutionary algorithms in Engineering Applications," Germany: Springer, 1997, pp. 3 - 23.
- [29] G.C. Onwubolu and B.V. Babu, "New Optimization Techniques in Engineering," Germany: Springer-Verlag, 2004.
- [30] B.V. Babu, "Process Plant Simulation", New York: Oxford University Press, 2004.

**Note: The pdf files of the references [10, 11, 12, 13, 14, 18, 22, 24, 25, 26, 27] are available at <http://discovery.bits-pilani.ac.in/discipline/chemical/BVb/publications.html>**