An Evolutionary Algorithm for Constrained Optimization

Tapabrata Ray, Tai Kang and Seow Kian Chye

Center for Advanced Numerical Engineering Simulations

School of Mechanical and Production Engineering

Nanyang Technological University

Singapore 639798

Email : mtray@ntu.edu.sg, mktai@ntu.edu.sg, mkcseow@ntu.edu.sg

Tel : 65 7904058

Abstract

In this paper we present an evolutionary algorithm for constrained optimization. The algorithm is based on nondominance of solutions *separately* in the objective and constraint space and uses effective mating strategies to improve solutions that are weak in either. Since the methodology is based on nondominance, scaling and aggregation affecting conventional penalty function methods for constraint handling does not arise. The algorithm incorporates intelligent partner selection for cooperative mating. The diversification strategy is based on niching that result in a wide spread of solutions in the parametric space. Preliminary results of the algorithm for constrained single and multiobjective test problems are presented and compared to illustrate the efficiency of the algorithm in solving constrained optimization problems.

1 INTRODUCTION

Evolutionary computation (EC) methods have received considerable attention over the years as optimization methods for complex functions. EC methods are essentially unconstrained search techniques that require a scalar measure of quality or fitness. The presence of constraints significantly affects the performance of an optimization algorithm, including evolutionary search methods. There have been a number of approaches to handle constraints including rejection of infeasible solutions, penalty functions and their variants, repair methods, use of decoders, separate treatment of constraints and objectives and hybrid methods incorporating knowledge of constraint satisfaction. A comprehensive review on constraint handling methods is provided by Michalewicz [1]. All the methods have limited success as they are problem dependent and require a number of additional inputs. Penalty functions using static, dynamic or adaptive concepts have been developed over the years. All of them still suffer from common problems of aggregation and scaling. Repair methods are based on additional function evaluations, while the decoders and special operators or constraint satisfaction methods are problem specific and cannot be used to model a generic constraint. Separate treatment of constraints and objectives is an interesting concept that eliminates the problem of scaling and aggregation.

Constraint handling using a pareto ranking scheme is a relatively new concept having its origin in multiobjective optimization. Fonseca and Flemming [2] proposed a pareto ranking scheme to handle multiple objectives. Jimenez and Verdegay [3] used a nondominated sorting genetic algorithm (NSGA) ranking scheme to deal with multiple objectives while a separate evaluation function was used for infeasible solutions. The NSGA used by Jimenez and Verdegay [3] was introduced by Srinivas and Deb [4]. Surry et al. [5] applied a pareto ranking scheme among constraints while fitness was used in the objective function space for the optimization of gas supply networks. Fonseca and Flemming [6] proposed a unified formulation to handle multiple constraints and objectives based on pareto ranking scheme. All the above attempts successfully eliminate the drawbacks of aggregation and scaling that exist with the penalty function methods. In addition, they do not require any additional input and are problem independent. However, none of the above methods incorporate concepts of cooperative learning through parent matching which is expected to improve the efficiency of the algorithm. An interesting attempt to incorporate the knowledge of constraint satisfaction during mating was proposed by Hinterding and Michalewicz [7]. In an attempt to match *the beauty with the brains*, constraint matching was employed during partner selection. A single measure (sum of squares of violation) was used to compute a solution's infeasibility. The algorithm does not include any niching or diversification mechanism to ensure a uniform spread of points along the pareto frontier for multiobjective

problems. Moreover, a single aggregate measure of infeasibility fails to incorporate the knowledge of individual constraint satisfaction/violation and in addition leads to scalability and aggregation problems.

The proposed evolutionary algorithm eliminates all the drawbacks as discussed above. Solutions are ranked *separately* based on objectives, constraints and combined matrices (described in Section 2). This process eliminates the problems of scaling and aggregation. Moreover, since the constraints are handled separately, the true objective function is optimized rather than some transformed evaluation function. The rank of a solution in the objective, constraint or the combined matrix is used for intelligent mating between solutions to improve those that are weak in either constraint satisfaction or objective performance. The mating process within the proposed evolutionary algorithm incorporates the knowledge of every individual constraint satisfaction/violation and objective performance. Strategies to handle highly constrained and moderately constrained problems are outlined. Section 2 provides a detailed description of the algorithm. Four examples comprising of three constrained single objective and one multiobjective constrained optimization problem are presented to illustrate the performance of the algorithm, identifying a faster convergence through cooperative learning.

2 PROPOSED ALGORITHM

The proposed evolutionary algorithm is described in the context of multiobjective optimization. A single objective problem is handled in the same formulation by assigning k=1. A general constrained multiobjective optimization problem (in minimization sense) is presented as:

$$\begin{aligned} &\text{Minimize} && \mathbf{f} = [f_1(\mathbf{x}) \quad f_2(\mathbf{x}) \quad \ldots \quad f_k(\mathbf{x})] \\ &\text{subject to} && g_i(\mathbf{x}) \geq a_i \quad , i = 1,2,\ldots,q \\ & && h_j(\mathbf{x}) = b_j \quad , j = 1,2,\ldots,r \end{aligned}$$

where $\mathbf{f}$ is a vector of k objectives to be minimized subject to q inequality and r equality constraints. $\mathbf{x} = [x_1 \quad x_2 \quad \ldots \quad x_n]$ is the vector of n design variables.

The **OBJECTIVE** matrix for a population of M solutions assumes the form

$$\mathbf{OBJECTIVE} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1k} \\ f_{21} & f_{22} & \cdots & f_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ f_{M1} & f_{M2} & \cdots & f_{Mk} \end{bmatrix}$$

It is common practice to transform the equality constraints (with a tolerance δ) to a set of inequalities and use a unified formulation for all constraints:

$h_j(\mathbf{x}) \leq b_j + \delta$ which is same as $-h_j(\mathbf{x}) \geq -b_j - \delta$ and $h_j(\mathbf{x}) \geq b_j - \delta$

Thus r equality constraints will give rise to $2r$ inequalities, and the total number of inequalities for the problem is denoted by s, where $s=q+2r$.

For each solution, $\mathbf{c}$ denotes the constraint satisfaction vector given by $\mathbf{c} = [c_1 \quad c_2 \quad \ldots \quad c_s]$ where

$$c_i = \begin{cases} 0 & \text{if } i^{\text{th}} \text{ constraint satisfied , } i = 1,2,\ldots, s \\ a_i - g_i(\mathbf{x}) & \text{if } i^{\text{th}} \text{ constraint violated , } i = 1,2,\ldots, q \\ b_i - \delta - h_i(\mathbf{x}) & \text{if } i^{\text{th}} \text{ constraint violated , } i = q+1, q+2,\ldots, q+r \\ -b_i - \delta + h_i(\mathbf{x}) & \text{if } i^{\text{th}} \text{ constraint violated , } i = q+r+1, q+r+2,\ldots, s \end{cases}$$

For the above c_i's, $c_i = 0$ indicates the i^{th} constraint is satisfied, whereas $c_i > 0$ indicates the violation of the constraint.

The **CONSTRAINT** matrix for a population of M solutions assumes the form

$$\mathbf{CONSTRAINT} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{M1} & c_{M2} & \cdots & c_{Ms} \end{bmatrix}$$

A **COMBINED** matrix that is a combination of objective and constraint matrix assumes the form

$$\mathbf{COMBINED} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1k} & c_{11} & c_{12} & \cdots & c_{1s} \\ f_{21} & f_{22} & \cdots & f_{2k} & c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{M1} & f_{M2} & \cdots & f_{Mk} & c_{M1} & c_{M2} & \cdots & c_{Ms} \end{bmatrix}$$

2.1 PARETO RANKING

From a population of M solutions, all nondominated solutions are assigned a rank of 1. The rank 1 individuals are removed from the population and the new set of nondominated solutions is assigned a rank of 2. The process is continued until every solution in the population is assigned a rank. Rank=1 in any of the objective, constraint or combined matrices indicate that the solution is nondominated.

The pareto rank of each solution in the population is computed individually in the **OBJECTIVE**, **CONSTRAINT** and **COMBINED** matrix and are stored in vectors **RankObj**, **RankCon** and **RankCom** respectively.

Having described the general formulation of the constraint and the objective function matrices and the concept of pareto ranking, the pseudo code of the algorithm is introduced.

Algorithm

Initialize M solutions to form a population

Do {

Compute Pareto Ranking based on **OBJECTIVE** matrix to yield a vector **RankObj**

Compute Pareto Ranking based on **CONSTRAINT** matrix to yield a vector **RankCon**

Compute Pareto Ranking based on **COMBINED** matrix to yield a vector **RankCom**

Multiobjective Optimization: Select individuals from the population in this generation if (**RankCom** = 1) & (Feasible) and put them into the population for the next generation.

Single Objective Optimization: Select individuals from the population in this generation if (**RankCom** is better than allowable rank) & (Feasible) and put them into the population for the next generation (Allowable rank=maximum rank of an individual in the population/2).

To generate the remaining members of the population for the next generation

Do {

Select an individual A and its partner from the population at this generation.

Mate A with its partner.

Put parents and children into the population for the next generation.

} while the population is not full.

Remove duplicate points in parametric space and shrink population

} while the maximum number of generations is not attained.

2.2 INITIALIZATION

The initialization is based on a random generation of M starting solutions using uniform random number generator and the variable bounds (side constraints). A solution

$\mathbf{x} = [x_1 \quad x_2 \quad \ldots \quad x_n]$ is generated as follows:

$$x_i = (x_{i,upper\ bound} - x_{i,lower\ bound}) * R + x_{i,lower\ bound}$$

where $x_{i,lower\ bound}$ and $x_{i,upper\ bound}$ are the lower and upper bounds of the i^{th} variable and R is a random number between 0 and 1.

2.3 SELECTION PROBABILITY

The probability of selection of an individual is based on the vectors **RankObj**, **RankCon** or **RankCom** and denoted as **ProbObj**, **ProbCon** and **ProbCom** respectively.

As an example, the vector **ProbObj** is computed as follows:

The vector **RankObj** with element values varying from 1 to P_{max} is transformed to a fitness vector **FitObj** with elements varying from P_{max} to 1 using a linear scaling (P_{max} denotes the rank of the worst solution).

The probability of selection **ProbObj** of an individual is then computed based on this fitness vector **FitObj** using the roulette wheel selection scheme. The process ensures that solutions that are fitter have a higher probability of being selected.

2.4 CHOOSING A PARTNER FOR MATING

A mating is performed between a solution A and its partner (B or C). The process of partner selection is dependent on the type of the constrained problem. Problems are classified into the following:

1. Unconstrained problem (Objective-Objective Mating)
2. Moderately constrained problem (Objective-Constraint Mating)
3. Highly constrained problem (Constraint-Constraint Mating)

For an unconstrained problem, the selection of A, B and C is based on **ProbObj**.

For a moderately constrained problem, selection of A is based on **ProbObj** while the selection of B and C is based on **ProbCon**. Such a mating between solutions that are *good* in objective function with that of solutions that are *good* in constraint satisfaction is analogous to mating between the *beauty and the brains.*

For a highly constrained problem, selection of A, B and C is based on **ProbCon**. Since finding a feasible solution is quite difficult for highly constrained problems, the selection of mating partners is based on the solution's ability towards constraint satisfaction.

The process of partner selection for a moderately-constrained problem is outlined below for a greater understanding of the selection process.

2.4.1 Moderately Constrained: Partner selection

Select first individual A based on **ProbObj**

Select potential mating candidate B based on **ProbCon**

Select potential mating candidate C based on **ProbCon**

Partner of A is either B or C, depending upon Condition 1, 2 or 3.

Condition 1 : If B and C are both feasible

If RankObj_B < RankObj_C

then : Partner is B

else : Partner is C.

If RankObj_B = RankObj_C

then : Choose the one with the minimum adaptive niche count (to be explained in Section 2.6).

where, RankObj_B denotes the rank of solution B in the vector **RankObj**.

Condition 2 : If B and C are both infeasible

If RankCon_B < RankCon_C

then : Partner is B

else : Partner is C.

If RankCon_B = RankCon_C

then : Choose the one with minimum overlapping constraint satisfaction with A (to be explained in Section 2.7).

Condition 3: If one is feasible and the other is not.

If B is feasible while C is not

then : Partner is B

else : Partner is C.

2.5 MATING

Every mating generates 3 additional solutions unlike conventional process of crossover generating two children. Out of the three solutions, one is generated by uniform crossover between A and its partner while the other two are generated using random mix and move. Every mating will place 2 parents and 3 additional solutions to the population for the next generation. The process of random mix and move is as follows:

For i=1: n

Action 1 : Randomly pick A or its partner and denote it as base.

Action 2 : Randomly pick a number Q for direction (<0.5 is negative, positive otherwise)

Action 3 : Randomly pick a number R between 0 and 1.

Condition 1: $x_{i,A} < x_{i,partner}$

A is base and Q <0.5:

New var. = $x_{i,A} - R(x_{i,A} - x_{i,lower\ bound})$

A is base and Q ≥0.5:

New var. = $x_{i,A} + R(x_{i,partner} - x_{i,A})$

Partner is base and Q <0.5:

New var. = $x_{i,partner} - R(x_{i,partner} - x_{i,A})$

Partner is base and Q ≥0.5:

New var. = $x_{i,partner} + R(x_{i,upper\ bound} - x_{i,partner})$

Condition 2: $x_{i,A} > x_{i,partner}$

A is base and Q <0.5:

New var. = $x_{i,A} - R(x_{i,A} - x_{i,partner})$

A is base and Q ≥0.5:

New var. = $x_{i,A} + R(x_{i,upper\ bound} - x_{i,A})$

Partner is base and Q <0.5:

New var. = $x_{i,partner} - R(x_{i,partner} - x_{i,lower\ bound})$

Partner is base and Q ≥0.5:

New var. = $x_{i,partner} + R(x_{i,A} - x_{i,partner})$

Condition 3: $x_{i,A} = x_{i,partner}$

Q <0.9:

New var. = $x_{i,A}$

Q ≥0.9:

New var. = $x_{i,A} + R(x_{i,upper\ bound} - x_{i,lower\ bound})$

End

The process of random mix and move will ensure that any feasible variable value can be generated even if it does not exist in either A or its partner. Generation of a large number of initial solutions to maintain all possible variable values is not considered favorable as those solutions are generated without any knowledge of the search process and adds on to a computational overhead. The proposed method as illustrated can be used with relatively small population size as the process of generating solutions comes along with random mix and move.

2.6 ADAPTIVE NICHE COUNT

Adaptive niche count of a solution is the number of solutions in that population which are within the average distance metric and is computed as follows:

For i=1: M

Compute the Euclidean distances between it and all other M-1 solutions

Compute the average Euclidean distance

Count the number of solutions that are within the average distance

End

A solution with a small niche count as compared to another physically means that there are few solutions in its neighborhood. Such solutions are preferred over others and is the diversification strategy used in the algorithm.

2.7 NON-OVERLAPPING CONSTRAINT SATISFACTION

The strategy is based on the philosophy that a solution is allowed to mate with another if one complements the other towards constraint satisfaction. Such a mating between the *beauty* and the *brains* is incorporated with a hope to generate solutions with better constraint satisfaction. The concept of non-overlapping constraint satisfaction is incorporated as follows:

With reference to the **CONSTRAINT** matrix discussed earlier, each of the solutions A, B and C has an associated constraint satisfaction vector $\mathbf{c_A}$, $\mathbf{c_B}$ and $\mathbf{c_C}$ respectively.

The sets $\{S_A\}$,$\{S_B\}$ and $\{S_C\}$ denote the set of constraints satisfied by solution A, B and C respectively.

The selection of either B or C is based on the following condition:

If $(\{S_A\} \cap \{S_B\}) > (\{S_A\} \cap \{S_C\})$

then : the partner is C.

If $(\{S_A\} \cap \{S_B\}) < (\{S_A\} \cap \{S_C\})$

then : the partner is B.

If $\{ \{S_A\} \cap \{S_B\}) = (\{S_A\} \cap \{S_C\})$

then : the partner is randomly chosen between B and C.

2.8 POPULATION SHRINKING

After each new population is full, a screening is done to remove identical points in the parametric (variable) space to give room for new and different solutions.

3 RESULTS AND DISCUSSION

The performance of the algorithm is reported for four constrained optimization problems. Examples 1, 2 and 3 are single objective problems while Example 4 is a multiobjective problem.

Example 1: The first example is a constrained single objective optimization problem. It has five variables, a single quadratic objective function and is subjected to six nonlinear inequalities [8]. The ratio of feasible points to sampled number of points for a 1,000,000 point random sampling was reported to be 0.52123 [8]. The above ratio indicates that the problem is moderately constrained and hence an objective-constraint-mating scheme was employed for the solution.

The optimum solution is (78.0, 33.0, 29.995, 45.0, 36.776) with an objective function value of -30665.5. Two constraints (upper bound of the first inequality and the lower bound of the third inequality) are active at the optimum.

$$\begin{aligned}
&\textit{Minimize} \quad f(\mathbf{x}) = 5.3578547x_3^{\,2} + 0.8356891x_1x_5 + \\
&\qquad 37.293239x_1 - 40792.141 \\
&\textit{Subject to} \quad 0 \le 85.334407 + 0.0056858x_2x_5 + \\
&\qquad 0.0006262x_1x_4 - 0.0022053x_3x_5 \le 92 \\
&\qquad 90 \le 80.51249 + 0.0071317x_2x_5 + \\
&\qquad 0.0029955x_1x_2 + 0.0021813x_3^{\,3} \le 110 \\
&\qquad 20 \le 9.300961 + 0.0047026x_3x_5 + \\
&\qquad 0.0012547x_1x_3 + 0.0019085x_3x_4 \le 25 \\
&78 \le x_1 \le 102,\ 33 \le x_2 \le 45,\ 27 \le x_i \le 45,\ i = 3,4,5.
\end{aligned}$$

Table 1 : Comparison of Results

	Number of Function Evaluations	Function Value
Reported [8]	350,000	-30617.0 (worst) -30643.8 (avg.) -30662.5 (best)
Present	13,370 13,364 13,139	-30626.123 -30640.969 -30615.689
Reported [8]	1,400,000	-30645.9 (worst) -30655.3 (avg.) -30664.5 (best)
Present	35,207 35,348 35,669	-30647.105 -30619.047 -30651.662

The objective function values as obtained from 3 successive trials using the present algorithm are compared with the best, worst and the average of 20 trials reported in Reference [8]. It can be observed from Table 1 that the proposed algorithm obtained comparable objective function values using a significantly smaller number of function evaluations.

Example 2: The second example is a constrained single objective optimization problem. It has two variables, a single cubic objective function and is subjected to two

nonlinear inequalities [8]. The ratio of feasible points to sampled number of points for a 1,000,000 point random sampling was reported to be 0.000066 [8].

The above ratio indicates that the problem is highly constrained and hence a constraint-constraint-mating scheme was employed for the solution. It is also interesting to note that an objective-constraint mating scheme fails to identify any feasible solution after 75245 function evaluations.

$$\begin{aligned} &\textit{Minimize} \quad f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \\ &\textit{Subject to} \quad (x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0 \\ &\qquad -(x_1 - 5)^2 - (x_2 - 5)^2 + 82.81 \geq 0 \\ &\qquad 13 \leq x_1 \leq 100, \quad 0 \leq x_2 \leq 100. \end{aligned}$$

The optimum solution is (14.095, 0.84296) with an objective function value of -6961.81381. The first two constraints are active at the optimum. The objective function values as obtained from 3 successive trials using the present algorithm are compared with the best, worst and the average of 20 trials reported in Reference [8]. Table 2 provides a comparison of results for the above example.

Table 2 : Comparison of Results

	Number of Function Evaluations	Function Value
Reported [8]	350,000	-4236.7 (worst) -6191.2 (avg.) -6901.5 (best)
Present	38,231 38,234 39,164	-6773.0078 -6525.8374 -6819.0391
Reported [8]	1,400,000	-5473.9 (worst) -6342.6 (avg.) -6952.1 (best)
Present	75,245 73,445 74,987	-6744.0864 -6852.5630 -6737.0479

Example 3: The third example is a constrained single objective problem [8] with three variables. The feasible region of the search space consists of 125 disjoint spheres (all of them having a radius of 0.5). The global maximum is located at (5, 5, 5) with the objective function value of 1.00.

The objective function values as obtained from 3 successive trials using the present algorithm are compared with the best, worst and the average of 20 trials reported in Reference [8]. It can be seen from Table 3, that the algorithm arrived at a solution (4.9702, 5.0193, 5.0082) with an objective value of 1.00 in 3958 function evaluations using the objective-constraint mating scheme.

It also reached (4.9828, 5.000, 4.9977) with an objective value of 1.00 in 3910 function evaluations using the constraint-constraint mating scheme.

$$\begin{aligned} &\textit{Maximize} \quad f_1(\mathrm{x}) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - \\ &\qquad (x_3 - 5)^2)/100 \\ &\textit{Subject to} \quad (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 \leq 0.25 \\ &\text{for at least one set of p, q, r} \\ &\textit{where} \quad p, q, r = 1,3,5,7 \textit{ and } 9. \\ &\qquad 0 \leq x_i \leq 10, \quad i = 1,2,3. \end{aligned}$$

Table 3 : Comparison of Results

	Number of Function Evaluations	Function Value
Reported [8]	35,000	0.999694591 (worst) 0.99993476 (avg.) 1.000 (best)
Present (Obj-Con)	3958 3961 3963	1.00 1.00 1.00
Present (Con-Con)	3910 3922 3914	1.00 1.00 1.00

Example 4: This is a two-variable constrained bi-objective problem [9].

$$\begin{aligned} &\textit{Minimize} \quad f_1(\mathbf{x}) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2 \\ &\qquad f_2(\mathbf{x}) = 9x_1 - (x_2 - 1)^2 \\ &\textit{Subject to} \quad g(\mathbf{x}) \equiv 225 - {x_1}^2 - {x_2}^2 \geq 0 \\ &\qquad g(\mathbf{x}) \equiv 3x_2 - x_1 - 10 \geq 0 \\ &\textit{where} \quad -20 \leq x_1, x_2 \leq 20 \end{aligned}$$

The initial population is presented in Figure 1 while the final pareto optimal front is presented in Figure 2. The final pareto front as presented in Figure 2 was obtained after 1153 function evaluations and consists of 92 pareto optimal points. The above problem was solved using a

objective-constraint mating scheme with a initial population size of 200.

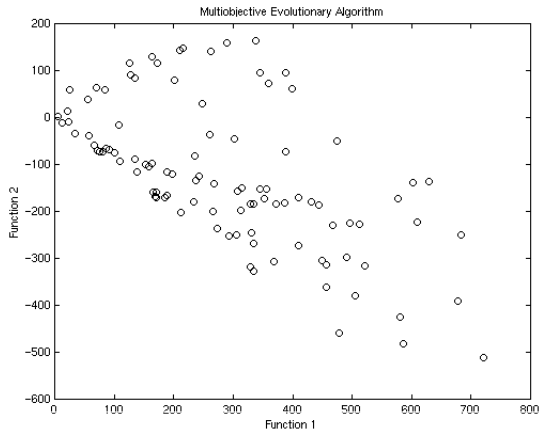

Figure 1: Initial population

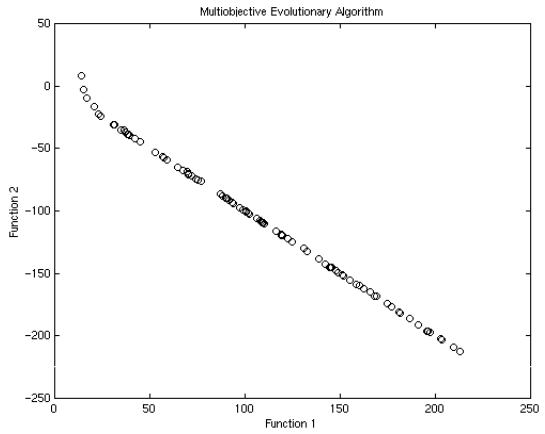

Figure 2: Final pareto front

4 SUMMARY AND CONCLUSIONS

This paper presents an evolutionary algorithm for constrained optimization. The method is problem independent and can handle any computable constraint and in addition optimizes the true objective function and not some transformed function. The performance of the algorithm on both constrained single and multiobjective problems show a significant decrease in the number of function evaluations for comparable objective function values. It can be seen from Example 2 that a constraint-constraint mating is effective for highly constrained problems where finding even a single feasible solution might be difficult. In the same example an objective-constraint mating fails to locate a solution. On the other hand, for problems where the feasible space is large, an objective-constraint mating results in comparable solutions in a significantly less number of function evaluations. The presence of cooperative learning through objective-constraint or constraint-constraint mating results is a faster convergence while the presence of niching allows the solution to be evenly distributed on the pareto front (Figure 2.) The algorithm is currently being tested on a wide range of single and multiobjective constrained test problems to establish its suitability as a generic constrained optimization methodology.

Acknowledgment

The authors would like to acknowledge the support for this work received from the Institute of High Performance Computing and the National Science and Technology Board (RICURF research fund EMT/98/013).

References

1. Michalewicz, Z. (1995). A Survey of Constraint Handling Techniques in Evolutionary Computation Methods, Proceedings of the 4th Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, pp. 135-155.
2. Fonseca, C.M. and Flemming, P.J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computation, 3(1), pp. 1-16.
3. Jimenez, F. and Verdegay, J.L. (1998). Constrained Multiobjective Optimization by Evolutionary Algorithms, Procs. Of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98), Spain, pp. 266-271.
4. Srinivas, N. and Deb, K. (1994). Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation, 2(3), pp. 221-248.
5. Surry, P., Radcliffe, N.J. and Boyd, I. (1995). A multiobjective approach to constrained optimization of gas supply networks, Procs. of the AISB-95 Workshop on Evolutionary Computing, Springer Verlag, Vol. 993, pp. 166-180.
6. Fonseca, C.M. and Flemming, P.J. (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms--Part I: A Unified Formulation. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 28(1), pp. 26-37.
7. Hinterding, R. and Michalewicz, Z. (1998). Your Brains and My Beauty: Parent Matching for Constrained Optimisation, Procs. of the 5th International Conference on Evolutionary Computation, Alaska, pp.810-815.
8. Koziel, S. and Michalewicz, Z. (1999). Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization, Evolutionary Computation, Vol.7, No.1, pp.19-44.
9. Deb, K. (1999). Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design, Procs. of Evolutionary Algorithms in Engineering and Computer Science (EUROGEN-99),Finland.