# An Evolutionary Algorithm for Constrained Bi-Objective Optimization Using Radial Slots

Tapabrata, Ray[1] and Kok Sung, Won[2]

[1] School of Aerospace, Civil and Mechanical Engineering,
University of New South Wales,
Australian Defense Force Academy,
t.ray@adfa.edu.au
[2] Temasek Laboratories, National University of Singapore,
5 Sports Drive 2, Singapore 117508,
tslwks@nus.edu.sg

**Abstract.** In this paper, we introduce an evolutionary algorithm for constrained, bi-objective optimization. The objective space is divided into a predefined number of radial slots and solutions compete with members in the same slot for existence. The procedure creates a uniform spread of solutions across the slots and they collectively form the nondominated front. Constraints are handled using a standard min-max formulation. We report the performace of our algorithm on a set of seven constrained, bi-objective test problems (CTP1 to CTP7). The chosen test problems have been known to pose difficulties to *all* existing multiobjective algorithms.

## 1 Introduction

A number of evolutionary algorithms have been proposed in recent years and many of them have been successfully applied to solve multiobjective problems. A comprehensive review of multiobjective optimization (MO) algorithms appears in [1,2,3]. In contrast to single-objective optimization problems, a solution to a multi-objective problem is more of a concept rather than a theorem [3]. Typically, there is no single global solution, and it is often necessary to determine a set of points that fit a predetermined definition of an optimum. Thus before continuing our discussion, it is necessary to introduce the concept of nondominated solutions and Pareto solutions.

**Definition 1.** *A solution* $\mathbf{x}^* \in F$ *is termed Pareto optimal iff there does not exist another* $\mathbf{x} \in F$*, such that* $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ *for all* $i = 1, \ldots, k$ *objectives and* $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ *for at least one* $j$*. Here* $F$ *denotes the feasible space (i.e. regions where the constraints are satisfied) and* $f_j(\mathbf{x})$ *denotes the* $j^{th}$ *objective corresponding to the solution* $\mathbf{x}$.

All Pareto optimal points lie on the boundary of the feasible criterion space [4,5]. Often algorithms provide solutions that may not be Pareto optimal but may satisfy other criteria, making them *weakly* Pareto optimal. Furthermore, if the solution space is limited to only $m$ solutions instead of the entire space $F$, the set of solutions are termed as nondominated solutions. Since in practice, all the solutions in $F$ cannot be evaluated exhaustively, the goal in a multi-objective optimization problem is to arrive at the set of nondominated solutions with a hope that it is sufficiently close to the set of Pareto solutions. Diversity among these set of nondominated solutions is also a highly desirable feature as it means making a selection from a wider choice of solutions.

It is clear from the above discussion that MO algorithms should be able to achieve both the following:

1. Drive the set of nondominated solutions towards the Pareto optimal front.
2. Generate a set of nondominated solutions that is evenly spread across the front.

In order to study the performance of various MO algorithms, [6] proposed a set of seven tunable constrained, multi-objective test problems named CTP1 to CTP7. All existing state of the art MO algorithms have faced difficulties in solving these problems. The test functions assess an algorithm based on the two important features listed above. Deb et al., [6,7] observed the performance of NSGA-II and the MO algorithm of [8] on the above set of problems. The MO algorithm of [8] encountered much difficulties for most of the problems while NSGA-II could solve CTP1, CTP2, CTP3, CTP5, and CTP6 but had problems in solving CTP4 and CTP7 in terms of maintaining diversity and proximity to the Pareto front.

Jimenez et al., [9] introduced an algorithm called `ENORA` (An Evolutionary Algorithm of Nondominated Sorting with Radial Slots) and reported its performance on the set of above test problems. The results were indeed exceptionally good, but a number of parameters were not listed which made it difficult to compare its results. In this paper, we present our version of the algorithm along similar lines of those presented by [9] with complete details of the parameters and all the mechanisms involved. The remainder of this paper is organized as follows: section 2 provides the detailed description of the mathematical background for the algorithm. Section 3 showcases the numerical results culled from the simulation runs and Sect. 4 provides a summary and conclusion on the proposed MO algorithm.

## 2 Mathematical Details

A constrained, bi-objective optimization problem in the context of minimization can be presented as follows,

$$\text{Minimize:} \quad \mathbf{f} = \left[ f_1(\mathbf{x}) \quad f_2(\mathbf{x}) \right] \ . \tag{1}$$

Where $\mathbf{x} = \left[ x_1, x_2, \ldots, x_n \right]^T$ is the vector of $n$ design variables and $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ need to be minimized. Subject to:

$$g_i(\mathbf{x}) \geq a_i, i = 1, \ldots, q \ . \tag{2}$$

Where $q$ is the number of inequality constraints. The constraint vector $\mathbf{c}$ for each solution can be represented as,

$$\mathbf{c} = [c_1, c_2, \ldots, c_q]^T \quad . \tag{3}$$

Where $c_i = 0.0$ if the constraint is satisfied and $c_i = a_i - g_i(\mathbf{x})$ if the constraint is violated.

The pseudo-code of the algorithm is presented below and the individual components would be discussed in details in subsequent sections.

```
Initialize a Population of m Solutions. (Set Eval=m,
Number of Radial Slots=d)
Evaluate the Solutions of the Population.
Assign Solutions to the Radial Slots.
While Eval < MaxEvals
   Select Two Parents Using Uniform Random Sampling.
   Create Two Children Using Crossover and Mutation.
   Evaluate the Children.
   Survive or Die: The Children either Survives or Die.
End While
```

### 2.1 Initialization

The initial population of $m$ solutions is created as follows:

$$x_i = \left(x_i^{\mathrm{u}} - x_i^{\mathrm{l}}\right) \times R + x_i^{\mathrm{l}}, \quad \forall i = 1, \ldots, m \quad . \tag{4}$$

Where $x_i^{\mathrm{l}}$ and $x_i^{\mathrm{u}}$ are the lower and upper bounds of the $i^{th}$ variable and $R$ is a uniform random number between 0 and 1.

### 2.2 Evaluation

For each solution, the objective and the constraint vectors are evaluated, i.e. $\mathbf{f}$ and $\mathbf{c}$.

### 2.3 Assigning Slots

Given a set of $m$ solutions, their corresponding objective vectors and a predefined number of radial slots $d$, the assignment process is as follows:

1. Identify $f_1^{\mathrm{Min}}$, $f_1^{\mathrm{Max}}$, $f_2^{\mathrm{Min}}$, and $f_2^{\mathrm{Max}}$ where they correspond to the minimum and the maximum value of objective 1 and objective 2 respectively, among the set of $m$ solutions.
2. The objective vector of each solution $\mathbf{f} = [f_1(\mathbf{x}) \quad f_2(\mathbf{x})]$ is scaled to yield a scaled objective vector as follows:

$$\mathbf{sf} = [sf_1(\mathbf{x}) \quad sf_2(\mathbf{x})] = \left[\frac{f_1(\mathbf{x}) - f_1^{\mathrm{Min}}}{f_1^{\mathrm{Max}} - f_1^{\mathrm{Min}}} \quad \frac{f_2(\mathbf{x}) - f_2^{\mathrm{Min}}}{f_2^{\mathrm{Max}} - f_2^{\mathrm{Min}}}\right] \quad .$$

3. Compute $tan(\theta) = \frac{1 - sf_1(\mathbf{x})}{1 - sf_2(\mathbf{x})}$ .
4. If $0 \leq \theta < \frac{\pi}{2d}$: the solution belongs to slot 1, if $\frac{\pi}{2d} \leq \theta < \frac{\pi}{d}$: the solution belongs to slot 2 and so on.

### 2.4 Selection of Parents

Two parents are selected from the set of $m$ solutions using a uniform random sampling.

### 2.5 Creation of Children

Two children are created from the two selected parents. The steps involved in each child creation are as follow:

1. Select the parents $P_1$ and $P_2$ and perform crossover.
   (a) If $Q \leq P_{\text{UniCross}}$: use uniform crossover to create child 1 and child 2.
   (b) If $Q > P_{\text{UniCross}}$: use arithmetic crossover to create child 1 and child 2.
2. For every variable in child 1 and child 2, check and perform mutation.
   (a) If $S > P_{\text{Mut}}$: do not mutate the variable.
   (b) If $S \leq P_{\text{Mut}}$ and $T \leq P_{\text{UniMut}}$: use uniform mutation to mutate the variable and set flag $= 0$.
   (c) If $S \leq P_{\text{Mut}}$; flag $\neq 0$ and $R \leq P_{\text{NonUniMut}}$: use non-uniform mutation to mutate the variable and set flag $= 0$.
   (d) If $S \leq P_{\text{Mut}}$ and flag $\neq 0$: use minimum mutation to mutate the variable.

Where Q, R, S, T are random numbers generated using a uniform random number distribution, $P_{\text{UniCross}}$ is the probability of uniform crossover, $P_{\text{Mut}}$ is the probability of mutation in the variable, $P_{\text{UniMut}}$ is the probability of uniform mutation, and $P_{\text{NonUniMut}}$ is the probability of non-uniform mutation. All these probabilities of mutation have been designed to be user-defined to allow for more customization by the users.

**Uniform Crossover.** Each variable of child 1 is inherited either from parent 1 or parent 2 based on a uniform random selection and the remaining set of variables is assigned to child 2.

**Arithmetic Crossover.** Each variable of child 1 and child 2 is created as follows: $C_i^1 = P_i^1 \times d + (1-d) \times P_i^2$ and $C_i^2 = P_i^2 \times d + (1-d) \times P_i^1$, where $C_i^1$ denotes the $i^{th}$ variable of child 1, $d$ is a random number from a uniform distribution between 0 and 1, $P_i^1$ and $P_i^2$ are the $i^{th}$ variable of parent 1 and parent 2 respectively.

**Uniform Mutation.** Each variable of child 1 and child 2 is mutated as follows: $C_i^1 = x_i^l + (x_i^u - x_i^l) \times d$ where $d$ is a random number from a uniform distribution lying between 0 and 1.

**Non-uniform Mutation.** Each variable of child 1 or child 2 is mutated as follows:

1. If $R > 0.5$ then $C_i^1 = C_i^1 + (x_i^u - C_i^1) \times a$.
2. If $R \leq 0.5$ then $C_i^1 = C_i^1 - (C_i^1 - x_i^l) \times a$.

Where $R$'s are random numbers from a uniform distribution lying between 0 and 1, and $a$ is the perturbation scalar defined as follows:

$$a = 1 - d^{(1-t/T)^2} \quad . \tag{5}$$

Where $d$ is a random number between 0 and 1, $T$ is the maximum number of solutions to be evaluated and $t$ denotes the solution being evaluated.

**Minimum Mutation.** Each variable of child 1 and child 2 is mutated as follows:

1. If $R > 0.5$ then $C_i^1 = C_i^1 + (x_i^u - C_i^1) \times a$.
2. If $R \leq 0.5$ then $C_i^1 = C_i^1 - (C_i^1 - x_i^l) \times a$.

Where $R$'s are random numbers from a uniform distribution lying between 0 and 1, and $a$ is the perturbation scalar defined as follows:

$$a = 1 - d^{\epsilon} \quad . \tag{6}$$

Where $d$ is a random number between 0 and 1, and $\epsilon$ is set to $1e - 05$.

## 2.6 Replace or Die

This is the most important mechanism in the algorithm that accounts for diversity and collectively generates the nondominated front. Every child either survives and replaces an existing individual of the population or gets killed himself. The survival and replacement policy is outlined as follow:

1. If the child is infeasible and the population of $m$ solutions has infeasible solutions: eliminate the worst infeasible solution among $m + 1$ solutions. Fitness of an infeasible solution is determined by the maximum value of an element in the constraint vector $\mathbf{c}$.
2. If the child is infeasible and the population of $m$ solutions has no infeasible solutions: the child is killed.
3. If the child is feasible and the population of $m$ solutions has no infeasible solutions:
   (a) Child has objective 1 less than $f_1^{\text{Min}}$ or objective 2 less than $f_2^{\text{Min}}$: child survives, objective spaces are re-scaled using $m + 1$ solutions and the solutions are re-slotted. The solution from the densest slot that has the worst nondominance ranking is killed. If all the solutions in the densest slot are nondominated, a random one is killed.
   (b) If the objective function values of the child is within $f_1^{\text{Min}}$, $f_1^{\text{Max}}$, $f_2^{\text{Min}}$, and $f_2^{\text{Max}}$, find the slot that the child belongs to. Add this child to the solutions belonging to that slot and replace the solution with the worst nondominance ranking. If all the solutions are nondominated, replace a random one.

## 3 Numerical Examples

The definitions of the test problems CTP1 to CTP7, employed throughout our study are presented below. Besides CTP1 which is slightly different, as it has two constraints, CTP2 to CTP7 have the same formulations, with the difference being in the parameters. Note that the function $g(\mathbf{x})$ may be taken to be any arbitrary function, we have chosen a linear function in this paper. A non-linear function such as the Rastrigin's function may also be used.

**CTP1**

$$\text{Min.} \quad f_1(\mathbf{x}) = x_1 \ . \tag{7}$$

$$\text{Min.} \quad f_2(\mathbf{x}) = g(\mathbf{x}) \exp\left(-f_1(\mathbf{x})/g(\mathbf{x})\right) \ . \tag{8}$$

Subject to:

$$f_2(\mathbf{x}) - a_j \exp\left(-b_j f_1(\mathbf{x})\right) \geq 0, \quad j = 1,2 \ . \tag{9}$$

$$g(\mathbf{x}) = 1 + x_1 \ . \tag{10}$$

Where $0 \leq x_i \leq 1, \quad i = 1,\ldots,4$, $a_1 = 0.858$, $a_2 = 0.728$, $b_1 = 0.541$, and $b_2 = 0.295$.

**CTP2–7**

$$\text{Min.} \quad f_1(\mathbf{x}) = x_1 \ . \tag{11}$$

$$\text{Min.} \quad f_2(\mathbf{x}) = g(\mathbf{x}) \left(1 - \sqrt{\frac{f_1(\mathbf{x})}{g(\mathbf{x})}}\right) \ . \tag{12}$$

Subject to:

$$\begin{aligned} &\cos(\theta)\left(f_2(\mathbf{x}) - e\right) - \sin(\theta) f_1(\mathbf{x}) \geq \\ &\quad + a|\sin\left(b\pi\left(\sin(\theta)\left(f_2(\mathbf{x}) - e\right) + \cos(\theta) f_1(\mathbf{x})\right)^c\right)|^d \ . \end{aligned} \tag{13}$$

$$g(\mathbf{x}) = 1 + x_1 \ . \tag{14}$$

Where $0 \leq x_i \leq 1, \quad i = 1,\ldots,4$, and the parameters for CTP2 to CTP7 are listed in Table 1.

A popluation size of 100 was used for all the runs and all the algorithms were allowed to perform $50,000$ function evaluations. We have used four variables for all the test problems. It is worth noting that the results report in [6] are for five variables while that of [9] is for four variables. We further observe that the expression for $f_2(\mathbf{x})$ given by (12) for CTP2 to CTP7 does not have the square root in [6] while [9] uses the expression as we have presented. We believe our expression is correct and that there is a typographical error in [6]. The probability of crossover was set to 0.6, probability of mutation was set to 0.6, probability of uniform crossover as 0.5, probability of uniform mutation as 0.2, probability of non-uniform mutation as 0.5, and the probability of minimum mutation was set

**Table 1.** Parameters for the Test Problems CTP2 to CTP7

| | $\theta$ | $a$ | $b$ | $c$ | $d$ | $e$ |
|---|---|---|---|---|---|---|
| CTP2 | $-0.20\pi$ | 0.20 | 10.0 | 1 | 6.0 | 1 |
| CTP3 | $-0.20\pi$ | 0.10 | 10.0 | 1 | 0.5 | 1 |
| CTP4 | $-0.20\pi$ | 0.75 | 10.0 | 1 | 0.5 | 1 |
| CTP5 | $-0.20\pi$ | 0.75 | 10.0 | 2 | 0.5 | 1 |
| CTP6 | $0.10\pi$ | 40.00 | 0.5 | 1 | 2.0 | $-2$ |
| CTP7 | $-0.05\pi$ | 40.00 | 5.0 | 1 | 6.0 | 0 |

to 0.3. They were thus chosen as our experiences showed that such values gave reasonably good solutions.

As a comparison, we have also included the results of NSGA-II. The source code for NSGA-II may be downloaded from the following website: http://www.iitk.ac.in/kangal/soft.htm. The version that we have is the latest release, dated 10th April 2005. And to be consistent, a population size of 100 was similarly used, and the NSGA-II algorithm was allowed to perform also $50,000$ function evaluations. The crossover operator used was simulated binary crossover [10] with $\eta_c = 20$, and polynomial mutation operator [10] with $\eta_m = 20$ was used. A crossover probability of 0.9 and mutation probability of 0.25 were chosen, as suggested in [7].

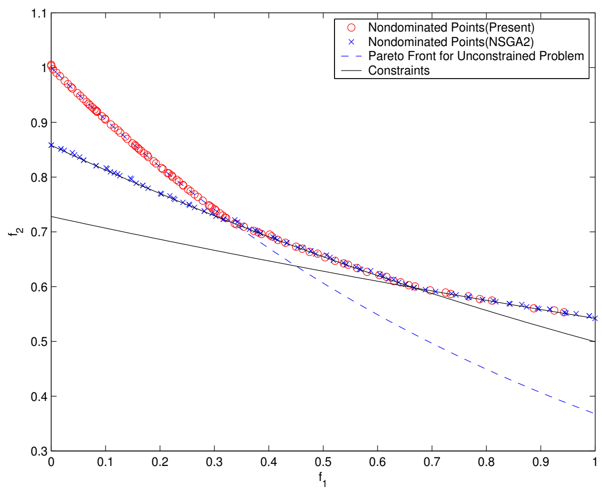

**Fig. 1.** Nondominated Solutions obtained on Test Problem CTP1

Figure 1 shows the results obtained for test problem CTP1. CTP1 is a two constraints problem, and with the presence of both constraints, about two-third portion of the original unconstrained Pareto-optimal region is now infeasible. This feasible two-third region of the constrained Pareto-optimal region now comes from the two constraints. Hence the constrained Pareto-optimal front has a kink at around $f_1 = 0.3$, and this is exactly what our algorithm has obtained, and the solutions have a good distribution along the front. On the other hand, NSGA-II failed to locate the Pareto-optimal front at the left one-third region, with its solutions falling into the infeasible region instead.

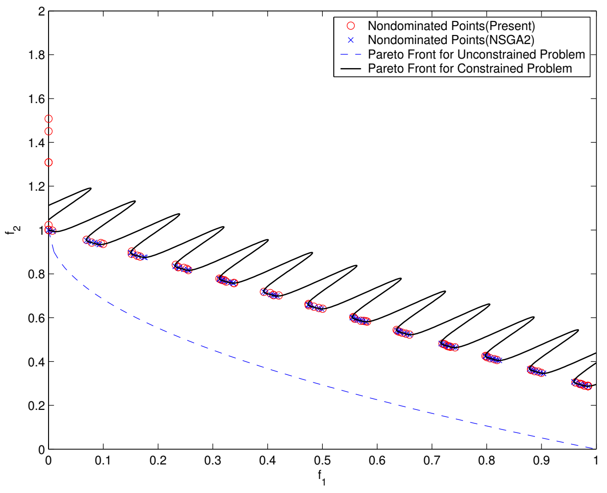

**Fig. 2.** Nondominated Solutions obtained on Test Problem CTP2

Figure 2 shows the nondominated solution front obtained by our algorithm for test problem CTP2. We show in our figures the theoretical pareto-optimal front for both the unconstrained (dashed line) and constrained problems (solid line). Unmistakably, the Pareto front for an unconstrained problem will always be lower. It is due to the presence of constraints that the Pareto front would be shifted upwards. As evident, our algorithm performed very well both in terms of arriving at the Pareto front as well as maintaining a good distribution along the front. For this test problem, NSGA-II also managed to arrive at the Pareto-optimal front.

Next we consider the test problem CTP3, where once again although most of the feasible search space is continuous, near the Pareto-optimal vicinity, the feasible regions are disconnected and each sub-region leading to only a singular feasible Pareto solution. As pointed out by [6], an MO algorithm will face difficulty in finding the discrete Pareto solutions because of the changing nature from continuous to discontinuous feasible space near the Pareto region. From Fig. 3, it can be seen that our algorithm performed resonably well too, by managing to locate all the discrete single Pareto solutions. The ability to successfully identify the discrete single Pareto solutions could be attributed to the unique property of assigning slots to the objective search space to maintain spread and diversity.

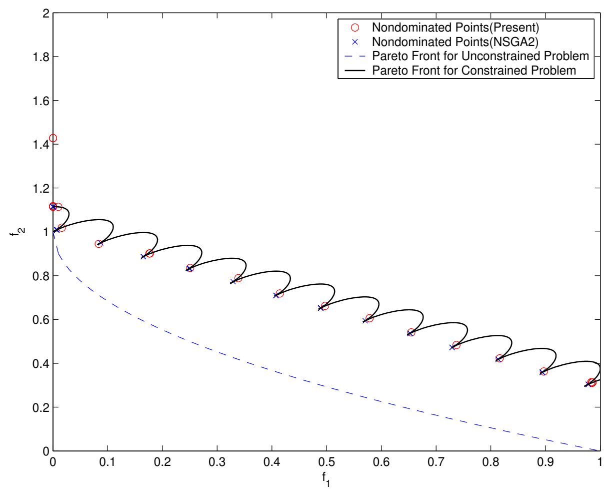

**Fig. 3.** Nondominated Solutions obtained on Test Problem CTP3

However, when we increase the value of parameter $a$ from 0.1 to 0.75, this has an effect of making the transition from continuous to discontinuous feasible region far away from the Pareto-optimal region. Thus the MO algorithm would now have to travel through a longer narrow feasible "tunnel" in search of the single discrete solution at the end of the "tunnel". Test problem CTP4 proved to be very difficult and Fig. 4 shows that our algorithm failed to locate the Pareto-optimal solutions. However, we note that the algorithm has successfully identified all the discrete discontinuous feasible regions just that it was not able to proceed out of the "tunnel" towards the Pareto-optimal solution at the end of it. NSGA-II performed slightly worse than our present algorithm as it failed to locate all the all the discrete discontinuous points. And for points that it managed to identify, it was trapped further up the "tunnel" as compared to our solutions.

Test problem CTP5 posed another difficulty in terms of clustering of Pareto-optimal solutions. By adjusting the value of parameter $c$, we can obtain a Pareto front with the solutions mose closely packed towards one end of the Pareto front. Figure. 5 shows the results obtained by our algorithm and NSGA-II. Once again, CTP5 proved to be fairly difficult, however an inspection shows that although the Pareto-optimal solutions are more clustered towards the right side, our algorithm managed to identify all of them. Another interesting phenomenon we observed is the availability of large number of infeasible solutions at the left end of the Pareto-optimal front, both for our algorithm and for NSGA-II. This probably also shows that both algorithms were not able to handle the varying widths of the discrete discontinuous Pareto front.

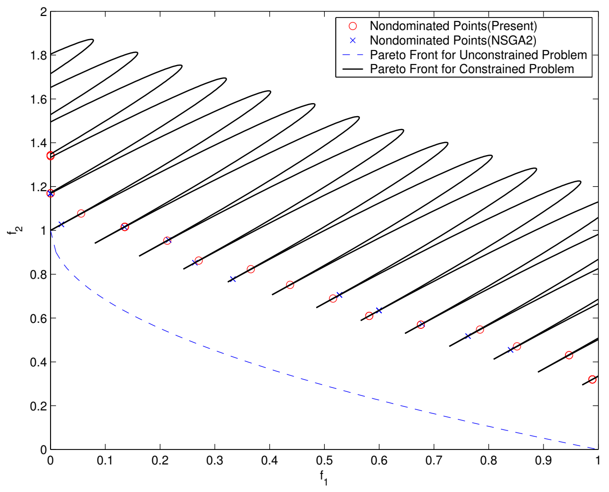

**Fig. 4.** Nondominated Solutions obtained on Test Problem CTP4

In comparison, test problem CTP6 was an easier challenge with our algorithm locating the Pareto-optimal front and at the same time maintaining a good distribution of nondominated solutions along the front. However, we wish to make a note here there, both our algorithm and NSGA-II were not able to generate any results for CTP6 using the definition of the $g$ function given by (14). Thus only for CTP6, we defined $g$ as the Rastrigin's function instead, given by: $g = 31 + \sum_{i=1}^{3} \left(x_i^2 - 10\cos(2\pi x_i)\right)$. The results with this new definition is shown in Fig. 6.

The final test problem CTP7 has infeasible solutions of differing widths towards the Pareto-optimal region. Since an MO algorithm has to overcome a number of such infeasible holes before coming to the region containing the Pareto-optimal front, many algorithms may face difficulty in solving this problem. Moreover, the unconstrained Pareto-optimal region is now infeasible, the entire constrained Pareto front lies on a part of the constrained boundary. Thus the entire search space now becomes patches of feasible and infeasible regions, with the MO algorithm easily being misled by the varying widths of the infeasible regions. Figure 7 shows the nondominated solutions obtained by our algorithm, it can be clearly seen that the entire Pareto-optimal front has been located and that in each sub-region, there is a respectable spread of solutions.

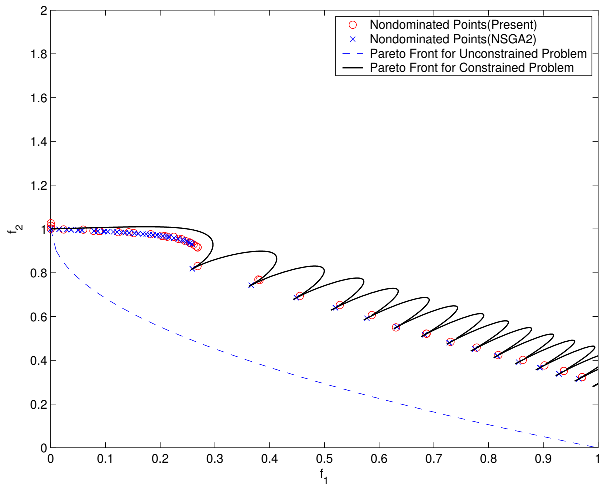

**Fig. 5.** Nondominated Solutions obtained on Test Problem CTP5

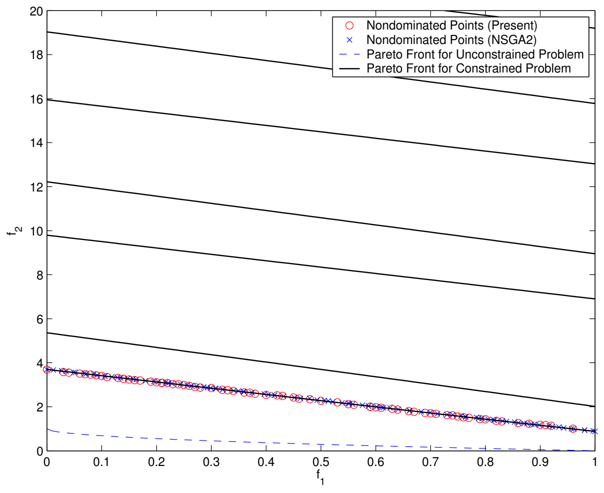

**Fig. 6.** Nondominated Solutions obtained on Test Problem CTP6

## 4 Summary and Conclusion

In this paper, we have presented a variant version of the algorithm along similar lines of those presented by [9] with complete details of the parameters and mechanisms involved. The results, however, were not so satisfactory in more complicated test problems, like those treated herein. However for other test prob-

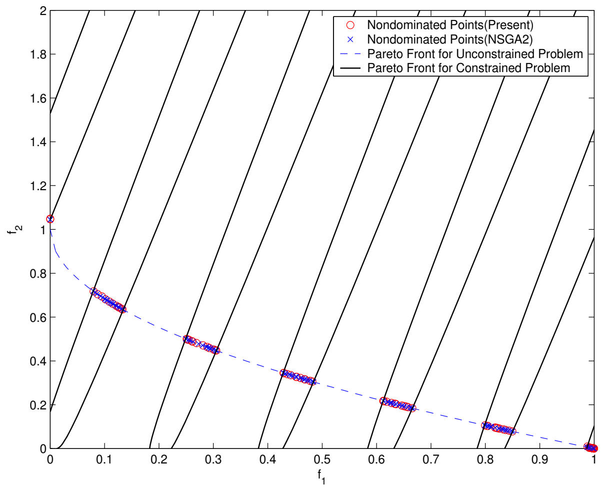

**Fig. 7.** Nondominated Solutions obtained on Test Problem CTP7

lems (e.g. CTP2), the proposed algorithm was able to simultaneously identify the Pareto front and also to maintain a good distribution along the front. The success of which has been fundamentally due to the slots assignment technique used.

The algorithm proposed incorporates a problem independent constraints satisfaction technique together with a powerful diversity mechanism which permits the identification of multiple nondominated solutions distributed uniformly along the Pareto-optimal fronts. The experimental results in test problems designed specifically to evaluate such algorithms show a considerable improvement on those obtained to date using other recently created algorithms, however for test problems CTP4 and CTP5, more research work need to be done to identify a more robust methodology in handling such difficult situations, e.g. where there is a long "tunnel" to a discrete Pareto-optimal solution at the end of it.

## References

1. Deb, K., (2001), *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons.
2. Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B., (2002), *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York.
3. Marler, R. T. and Arora, J. S., (2004), "Survey of Multi-objective Optimization Methods for Engineering," *Structural Multidisciplinary Optimization*, Vol. 26, pp. 369–395.
4. Athan, T. W. and Papalambros, P. Y., (1996), "A Note on Weighted Criteria Methods for Compromise Solutions in Multi-objective Optimization," *Eng. Optim.*, Vol. 27, pp. 155–176.

5. Chen, W., Sahai, A., Messac, A., and Sundararaj, G., (2000), “Exploration of the Effectiveness of Physical Programming in Robust Design,” *Journal of Mechanical Design*, Vol. 122, pp. 179–187.
6. Deb, K., Pratap, A., and Meyarivan, T., (2001), “Constrained Test Problems for Multi-objective Evolutionary Optimization,” *Proceeding of the First International Conference on Evolutionary Multi-Criterion Optimization* (EMO-2001), 7–9 March, Zurich, Switzerland, pp. 284–298.
7. Deb, K. and Goel, T., (2001), “Controlled Elitist Nondominated Sorting Genetic Algorithm for Better Convergence,” *Proceeding of the First International Conference on Evolutionary Multi-Criterion Optimization* (EMO-2001), 7–9 March, Zurich, Switzerland, pp. 67–81.
8. Ray, T., Tai, K., and Seow, K. C., (2001). “Multi-objective Design Optimization by an Evolutionary Algorithm,” *Engineering Optimization*, Vol. 33, No. 4, pp. 399–424.
9. Jimenez, F., Gomez-Skarmeta, A. F., Sanchez, G., and Deb, K., (2002), “An Evolutionary Algorithm for Constrained Multi-objective Optimization,” *Proceedings of IEEE World Congress on Computational Intelligence*, Congress on Evolutionary Computation (CEC 2002), Hawaii, 12–17 May.
10. Deb, K. and Agrawal, R. B., (1995), “Simulated Binary Crossover for Continuous Search Space,” *Complex Systems*, Vol. 9, pp. 115–148.