A Non-generational Genetic Algorithm for Multiobjective Optimization

Carlos C.H. Borges
COPPE/UFRJ
Prog. de Engenharia Civil – PO Box 68506
21945 970 Rio de Janeiro RJ, BRAZIL
cchb@lncc.br

Helio J.C. Barbosa
LNCC/CNPq
Rua Getulio Vargas 333
25651 070 Petropolis RJ, BRAZIL
hcbm@lncc.br – Corresponding author

Abstract- In this paper a non-generational genetic algorithm for multi-objective optimization problems is proposed. For each element in the population a domination count is defined together with a neighborhood density measure based on a sharing function. Those two measures are then non-linearly combined in order to define the individual's fitness. Numerical experiments with four test-problems taken from the evolutionary multi-objective literature are performed and the results are compared with those obtained by other evolutionary techniques.

1 Introduction

A number of optimization problems (specially design problems) are actually multiobjective optimization problems (MOPs) where the objectives are usually incommensurable and conflicting. As a result, there is usually not a single solution which optimizes all objectives simultaneously and a number of techniques have been developed in order to find a "good" solution to the MOP.

The set of solutions which are such that no improvement can be made in one objective without deteriorating at least one of the other objectives is called the Pareto-optimal set and an approximation of it would be very useful in order to get insight into the problem thus assisting the decision making process.

Genetic algorithms (GAs) have features that make them attractive for the approximation of the Pareto set: they are population-based, require only objective function evaluations, use probabilistic transition rules which make then less prone to local optimum entrapment and allow for several types of parallel implementations.

The first application of GAs to MOPs dates back to the mid-eighties[1, 2] and numerous papers on the application of evolutionary techniques to MOPs have been published since then (see [3, 4, 5, 6]).

Although other techniques have been used for finding a suitable solution to MOPs we are interested here in those that try to approximate the Pareto set. The most popular techniques seem to be MOGA, due to Fonseca and Fleming[7], NPGA, due to Horn and Nafpliotis[8] and NSGA, due to Srinivas and Deb[9].

Almost all the work with GAs for MOPs involved algorithms of the so called generational type, where a large portion of the current population is replaced by newly generated individuals. To the best of our knowledge, among those that try to build the Pareto set, the only non-generational GA proposed for MOPs is the one due to Valenzuela-Rendon and Uresti-Charre[10], hereafter denoted by VR-UC.

In this paper, a non-generational GA is presented and applied to some test-problems available in the literature.

The paper is structured as follows: Section 2 introduces some definitions concerning MOPs, Section 3 summarizes the VR-UC algorithm, Section 4 presents the proposed algorithm and in Section 5 numerical results are presented and discussed. The paper ends with some conclusions.

2 Multiobjective Optimization and Genetic Algorithms

In an unconstrained MOP one seeks to optimize the m components of a vector of objective functions $F(x) = (f_1(x), f_2(x), \ldots, f_m(x))$ where $x \in \Omega \subset R^n$ is the vector of design (or decision) variables which belong to the admissible set Ω. The function $F : \Omega \mapsto \Lambda$ maps solution (or design) vectors $x = (x_1, x_2, \ldots, x_n)$ to vectors $y = (y_1, y_2, \ldots, y_m)$, with $y_i = f_i(x)$, in the objective function space.

Here, the optimization problem associated to each objective will be considered to be a minimization one, without loss of generality. As the objectives are in general conflicting there is usually not a solution which minimizes all m objectives simultaneously. This motivates the concept of *dominance*: a vector $a = (a_1, a_2, \ldots, a_n)$ dominates a vector $b = (b_1, b_2, \ldots, b_n)$ if $a_i \leq b_i \, \forall i$ and there is j such that $a_j < b_j$. One then defines Pareto optimality: a solution $x \in D$ is said to be Pareto-optimal in D if and only if there is no $x' \in D$ such that its image $F(x')$ dominates $F(x)$. The Pareto-optimal set, $\mathcal{P}$, is thus the set of all $x \in D$ such that $F(x)$ is non-dominated in Λ. The Pareto front $\mathcal{P}_F$ is the image of the Pareto-optimal set in the objective function space.

As pointed out by Deb[11] the two major issues that should be addressed when applying a GA to a MOP are: (i) how to define fitness and perform selection in order to drive the population towards the Pareto set and (ii) how to maintain diversity in order to prevent premature convergence and achieve a well distributed Pareto front.

With regard to (i), Horn[12] classifies the methods as performing criterion selection (like VEGA[2]), aggregation selection (as in [13]) and Pareto selection (as in MOGA[7], NPGA[8] and NSGA[9]).

3 The VR-UC algorithm

The decision to investigate a non-generational GA for MOPs was inspired by the similarities that Valenzuela-Rendón and Uresti-Charre[10] observed between the tasks faced by a GA when dealing with a learning classifier system (LCS) using the Michigan approach and when dealing with a MOP, which are substantially different from that of the more common case of a single objective problem. In the later case, the GA, although using a population of candidate solutions, is expected to produce a good approximation to the global optimum. In the former case, the GA must evolve a co-adapted rule set in which classifiers cooperate with each other in order to achieve payoff for the LCS, while a good approximation to the potentially infinite set of non-dominated solutions should be produced when dealing with the MOP. The belief that the characteristics that require a non-generational GA for LCSs are also present in the MOP, led to the design of the VR-UC algorithm.

In the VR-UC algorithm, the fitness of an individual i is given by

$$f_i = c_\delta \delta_i + c_\omega \omega_i \tag{1}$$

where δ (the domination count) is a running weighted average of the number of individuals that have dominated this individual so far and ω (the niche count) is a running weighted average of the number of individuals that lie close according to the sharing function

$$\mathrm{sh}(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\sigma_s})^\alpha, & \text{if } d_{ij} < \sigma_s; \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where d_{ij} is the distance between elements i and j. By using pair-wise comparisons the dominance count is adjusted by

$$\delta(t+1) = \delta(t) - k_\delta \delta(t) + D(t) \tag{3}$$

where $D(t)$ is 1 if the individual is dominated in comparison t and zero otherwise. The moving niche count is adjusted by

$$\omega(t+1) = \omega(t) - k_\omega \omega(t) + \mathrm{sh}(d) \tag{4}$$

where d is the distance between the two individuals being compared.

The GA thus seeks individuals with low values of dominance and moving niche counts which are linearly combined, through Equation 1, into a single objective.

During the initialization phase, each individual is compared to n_1 randomly selected individuals. In the main cycle n_2 comparisons are made and values of δ, ω and f are adjusted using Equations 3, 4 and 1. Each new offspring is initially evaluated performing n_1 comparisons.

It is seen that 6 new parameters must be defined in the VR-UC algorithm: $c_\delta, c_\omega, k_\delta, k_\omega, n_1$ and n_2. In [10] the authors set $k_\delta = 0$ and $k_\omega = \frac{1}{n_1}$ leaving the determination of the other 4 open.

The VR-UC algorithm was compared with the NPGA algorithm[8] in three problems of increasing difficulty and it was observed[10] that: (i) it explored a smaller portion of the search space, (ii) produced a better Pareto front and (iii) its final population contained less individuals that do not belong to the Pareto set.

The non-generational algorithm proposed here is discussed in the next section.

4 A non-generational GA for MOP

The non-generational GA for MOP proposed here has one point in common with the VR-UC algorithm just described, namely, the reduction of all objectives to two measures related with dominance and population distribution. However, the use of those measures in the VR-UC algorithm is inspired by LCS techniques and is thus substantially different from the use made in the proposed algorithm.

The domination measure used here, $\mathcal{D}(i)$, expresses the state of domination of the $i-$th individual with respect to the current population:

$$\mathcal{D}(i) = \sum_{j=1}^{p} \mathrm{nd}(i,j)$$

where p is the number of individuals in the population and $\mathrm{nd}(i,j)$ is one if element j dominates element i, and zero otherwise.

The neighbor density measure $\mathcal{N}(i)$ associated with element i is defined as

$$\mathcal{N}(i) = \sum_{j=1}^{p} \mathrm{sh}(d_{ij}).$$

The fitness (to be minimized) of a given element is then defined as the non-linear function:

$$\mathcal{F}(i) = (1+\mathcal{D}(i))^\beta (1+\mathcal{N}(i)) \tag{5}$$

The exponent β is usually set to one. However, it can be given a higher value in order to increase the role of the domination measure in exceptional cases (see Test-case 4).

The algorithm can be written as:

```
Begin
  Generate initial population
  Compute f_i^k   i = 1,...,p   k = 1,...,m
  Compute D(i) and N(i) for i = 1,...,p
  Rank population according to F
  repeat
    Select two individuals p_1 and p_2
    Generate offsprings o_1 and o_2
    Compute D(o_j), N(o_j) and F(o_j),   j = 1,2
    Select o = argmin{F(o_1), F(o_2)}
    if F(o) < max_i{F(i)} then
      Replace worst element by o
      Update D and N
      Rank population according to F
    endif
  until(termination criteria satisfied)
  Stop
End
```

It can be seen that the proposed algorithm differs from the VR-UC by:

- using the whole population in the computation of the dominance and neighbor distribution measures for each individual thus avoiding noise introduced by the sampling procedure adopted in the VR-UC (which follows the LCS technology)

- using the current values of those measures in the present population instead of running weighted averages inspired in LCS
- avoiding the use of 6 parameters ($c_\delta, c_\omega, k_\delta, k_\omega, n_1$ and n_2) which have to be set by the user in a given application
- using a non-linear function of the dominance and neighbor density measures when defining the fitness of an individual thus avoiding the need to introduce (and find adequate numerical values of) constants (like c_δ and c_ω in the VR-UC algorithm) required to properly balance the two uncommensurable measures $\mathcal{D}$ and $\mathcal{N}$.
- introducing a new offspring in the population only if it is really better than the worst one, that is, it has a lower value for at least one of the measures $\mathcal{D}(i)$ or $\mathcal{N}(i)$.

It is expected that the use of:

1. a non-generational scheme,
2. the proposed fitness function (Eq. 5) and
3. the proposed replacement policy

(i) minimizes the disruption in the Pareto front (when compared with a generational scheme), (ii) tends to emphasize the retention of non-dominated solutions and (iii) improves the distribution of those solutions along the Pareto front.

5 Numerical experiments

In order to investigate the feasibility and performance of the approach proposed here, some examples available in the literature were considered. The results shown correspond to typical runs; similar results were obtained using other values for the seed supplied to the pseudo-random number generator. For a given test-problem, the results shown for the proposed algorithm correspond to the same seed so that comparisons can be made.

5.1 Test-problem 1

The first test problem, the so-called "unitation versus pairs"[8], involves the maximization of two functions over bit strings. Function f_1 is the number of pairs of adjacent complementary bits found in the string while function f_2 is the numbers of ones found in the string. For instance, the string 011101 has $f_1 = 3$ and $f_2 = 4$. This is a convenient test which permits the exact construction of the (discrete) Pareto front. Here the string length was taken equal to 28 and the Pareto-optimal front is composed of 15 points. In Figure 1 the final Pareto front obtained using NPGA[8] with a population of 400 individuals evolving for 200 generations is displayed. The figure shows all possible points in the objective function space and displays the number of individuals in the population which achieve those values of f_1 and f_2. It is seen that the point corresponding to $f_1 = 27$ and $f_2 = 14$ has not been found and that some individuals in the final population do not belong to the Pareto set.

The results corresponding to the algorithm proposed here are shown in Figure 2 using the same number of function evaluations, namely 80000. It is seen that (i) all points of the Pareto front are found, (ii) all individuals in the final population belong to the Pareto optimal set and (iii) the distribution of solutions along the Pareto front is quite uniform, with a minimum of 21 and a maximum of 28 individuals per point in the Pareto front.

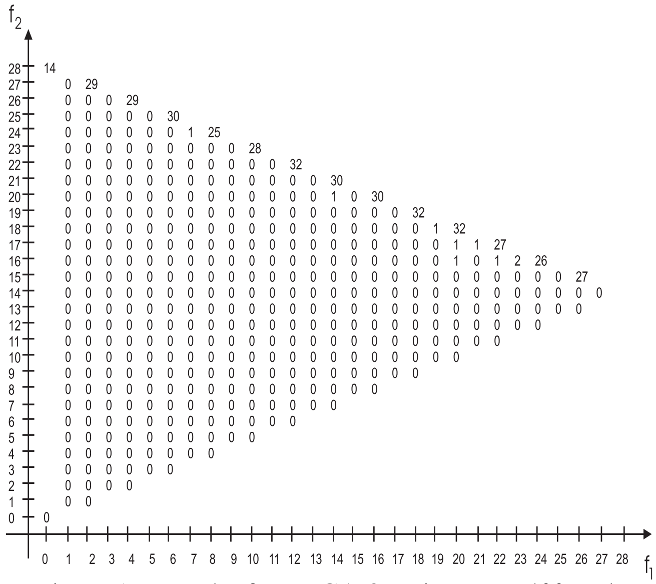

Figure 1 - Results for NPGA[8] using $p = 400$ and 200 generations.

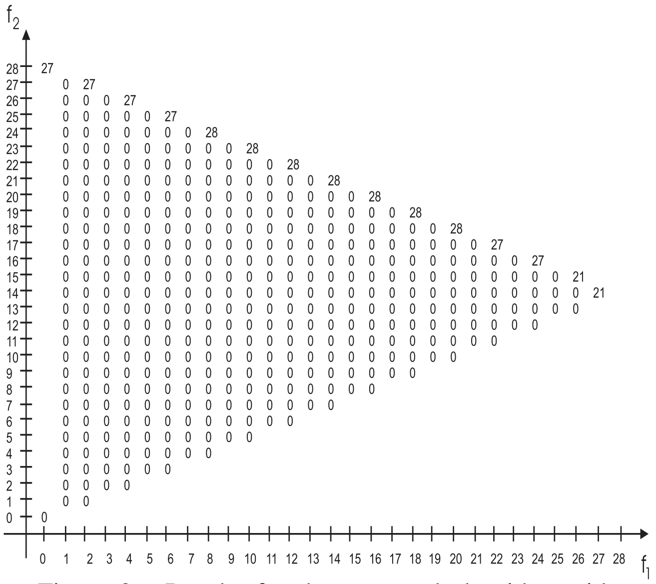

Figure 2 - Results for the proposed algorithm with $p = 400$ and 80000 evaluations.

Using 100 elements – one fourth of the previous population size – the algorithm is now run allowing for 10000 and 5000 function evaluations, respectively. The results are shown in Figure 3 and 4, respectively. It can be seen that the observations (i), (ii) and (iii) above are still valid with a number of function evaluations reduced by a factor of 16.

The population size is now reduced to 50 and the algorithm is allowed only 5000 and 2500 function evaluations, respectively. The results are displayed in Figures 5 and 6, respectively.

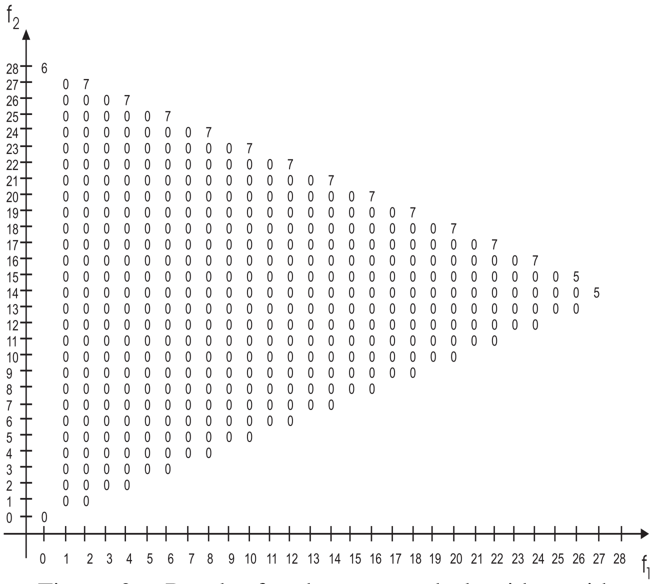

Figure 3 - Results for the proposed algorithm with $p = 100$ and 10000 evaluations.

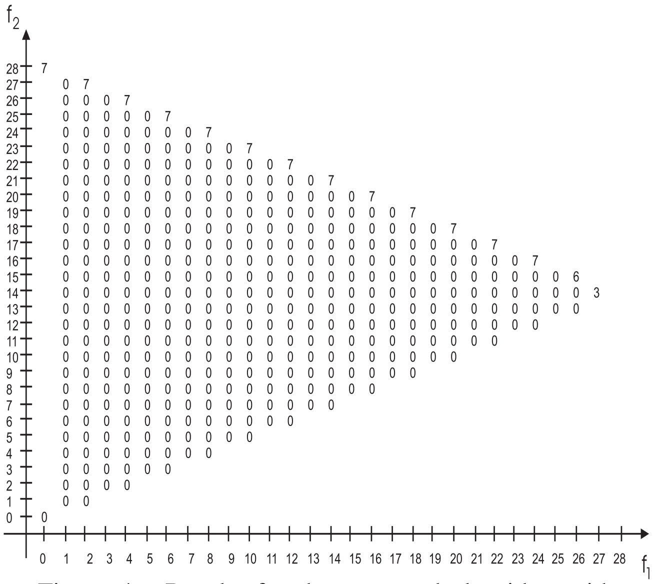

Figure 4 - Results for the proposed algorithm with $p = 100$ and 5000 evaluations.

In both runs with 50 individuals the Pareto front point (26,15) was not obtained. This means that it was not present in the initial population nor generated during the evolution.

For all cases sharing was performed in the objective function space using $\sigma_s = 2$.

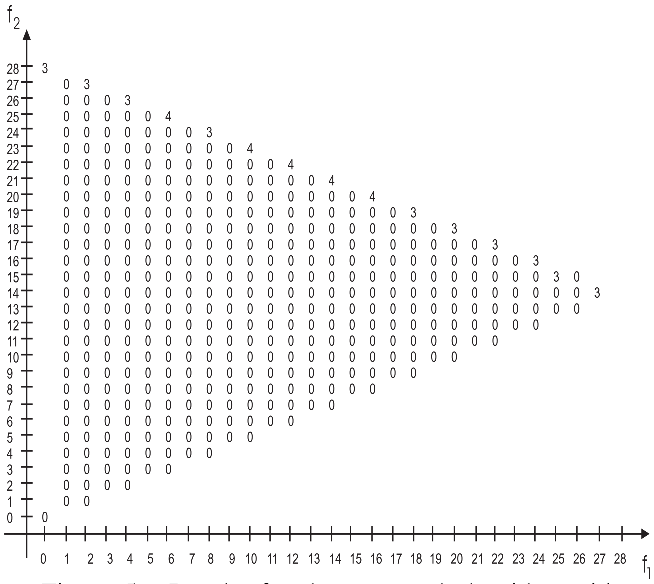

Figure 5 - Results for the proposed algorithm with $p = 50$ and 5000 evaluations.

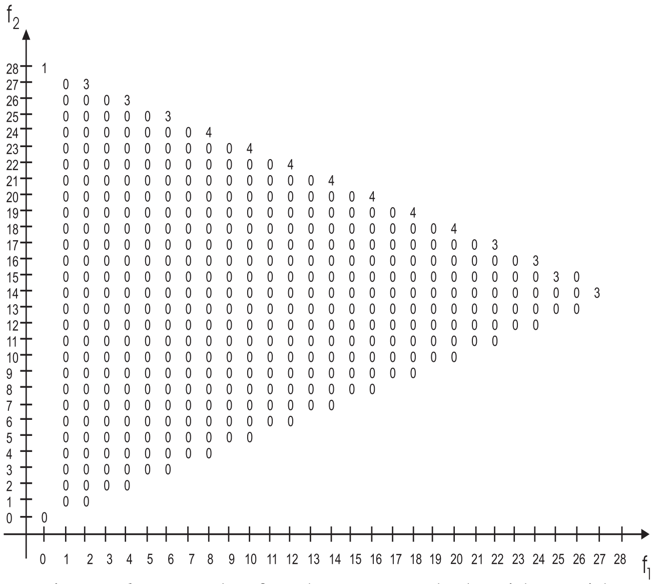

Figure 6 - Results for the proposed algorithm with $p = 50$ and 2500 evaluations.

5.2 Test-problem 2

In the second test-problem[9], the functions f_1 and f_2 to be minimized are given by

$$f_1 = \begin{cases} -x & \text{if } x \leq 1 \\ -2 + x & \text{if } 1 < x \leq 3 \\ 4 - x & \text{if } 3 < x \leq 4 \\ -4 + x & \text{if } x > 4 \end{cases}$$

$$f_2 = (x-5)^2$$

Figure 7 plots f_1 e f_2, for each value of $x \in [-10, 10]$.

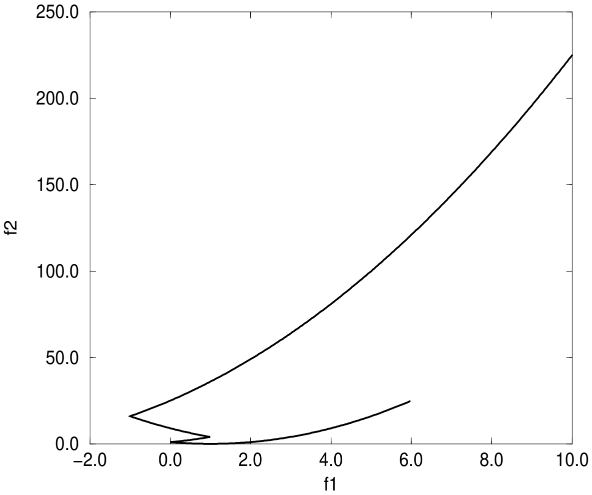

Figure 7 - Values of f_1 and f_2.

In this problem the Pareto optimal set is disjoint: $\mathcal{P} = x \in \{[1,2] \cup [4,5]\}$ and the Pareto front is discontinuous. The population size was set to 100 and the chromosome has 32 bits. In [9] the recombination probability was taken as 1 and the mutation rate was set to zero. Those values were also used here and Figure 8 displays the initial population for a typical run.

The results after 1000, 10000 and 50000 function evaluations are shown in Figures 8, 9 and 10 respectively, and they do not deteriorate, as when using VEGA in this problem[9], when the number of generations increase. Also, by visual inspection, the proposed algorithm provides a better definition of the Pareto front when compared to the results given by NSGA in [9].

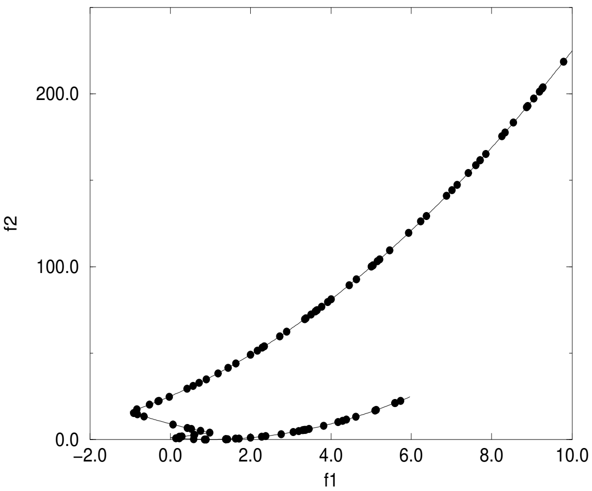

Figure 8 - Initial population

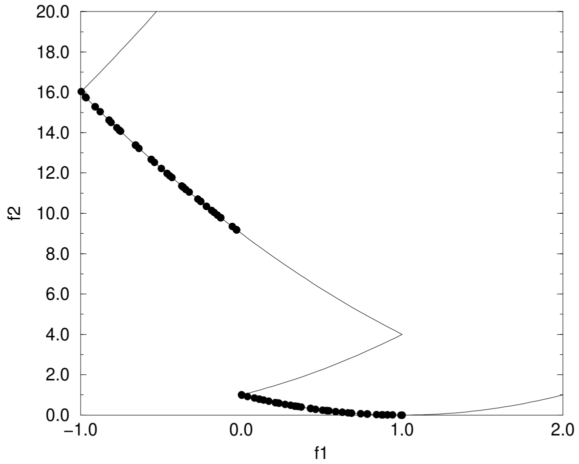

Figure 9 - Results for the proposed algorithm after 1000 evaluations.

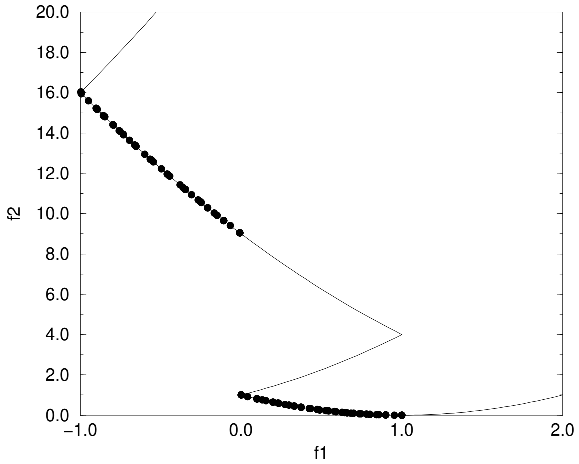

Figure 10 - Results for the proposed algorithm after 10000 evaluations.

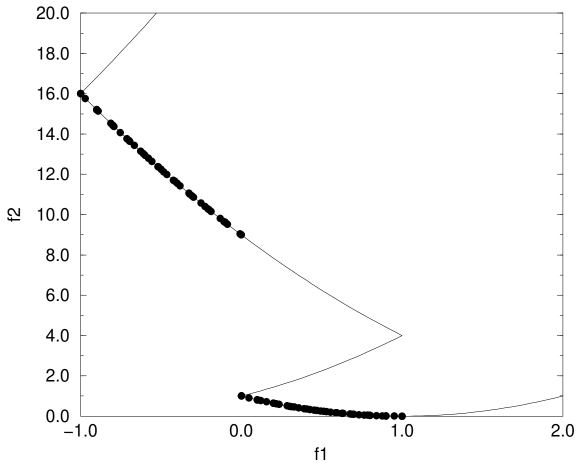

Figure 11 - Results for the proposed algorithm after 50000 evaluations.

5.3 Test-problem 3

Now the test-problem considered the most difficult among those solved in [10] is analyzed. The functions f_1 and f_2 to be minimized are given by

$$f_1 = x + y + 1$$
$$f_2 = x^2 + 2y - 1$$

In this problem f_1 and f_2 are functions of two variables $x \in [-3, 3]$ and $y \in [-3, 3]$ instead of only one as in the previous case. For comparative purpose the parameters used in the algorithm are the same used in [10], with a chromosome length of 20 bits, two-point crossover with 0.85 probability, mutation rate equal to 0.033, a population of 60 individuals with 660 function evaluations allowed. The value of σ_s is set to 0.1. In the Figure 12 the results are shown using the neighborhood density computed in the design-variable space. The same case is displayed in Figure 13 with the neighborhood density computed now in the objective-function space.

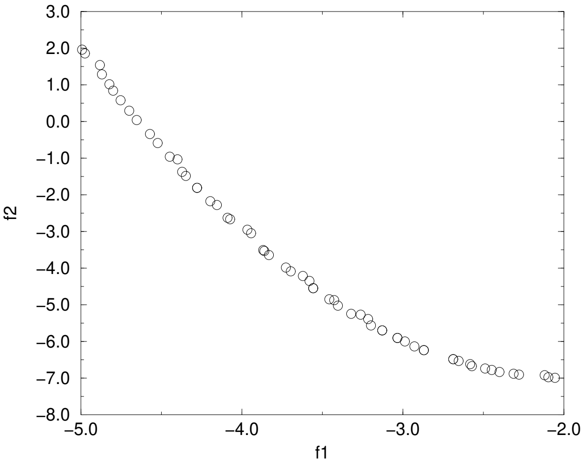

Figure 12 - Results for the proposed algorithm with density computed in the design-variable space

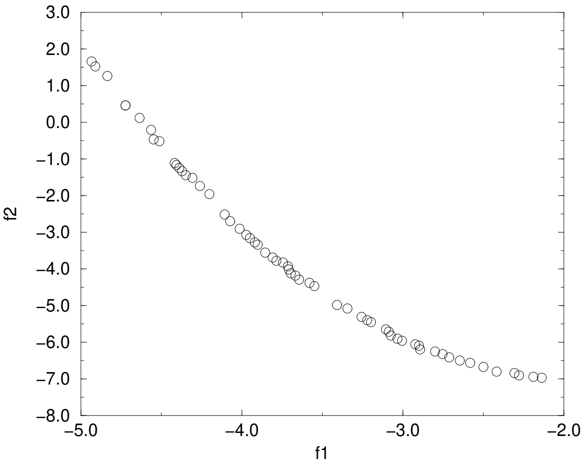

Figure 13 - Results for the proposed algorithm with density computed in the objective-function space

The results are similar for both forms of sharing. For a better distribution of the population over the Pareto front, 2000 evaluations are now allowed. The Figures 14 and 15 show the results for sharing in design-variable and objective-function spaces, respectively.

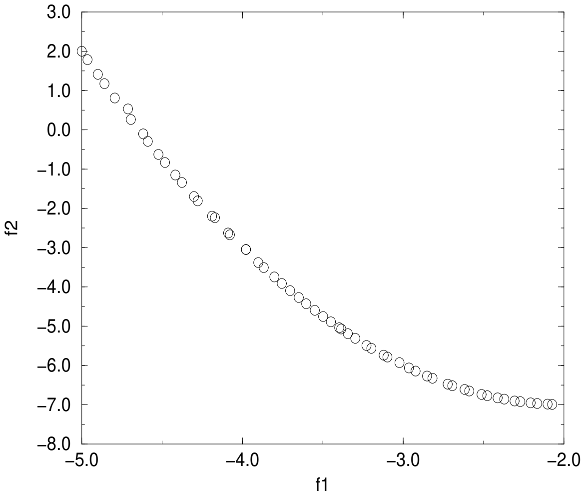

Figure 14 - Results after 2000 evaluations with density computed in the design-variable space

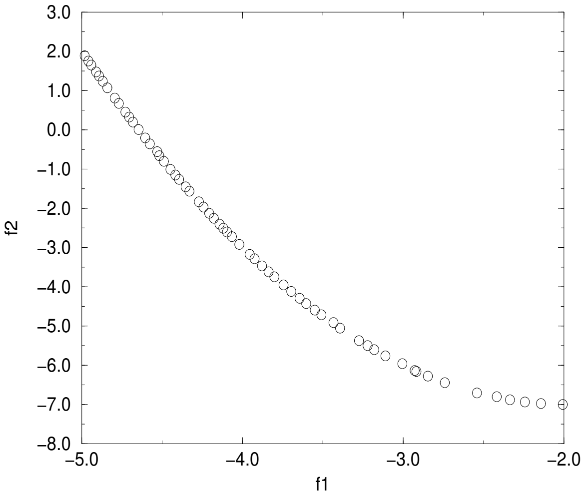

Figure 15 - Results after 2000 evaluations with density computed in the objective-function space

5.4 Test-problem 4

More recently, a set of test-problems for multi-objective optimization has been proposed in the evolutionary computation literature by Deb[14] and the last test-problem is taken from there.

The objectives considered are

$$f_1(x) = x_1$$
$$f_2(x) = g(x_2, \ldots, x_n)h(f_1(x_1), g(x_2, \ldots, x_n))$$

with

$$g(x_2, \ldots, x_n) = 1 + 9\sum_{j=2}^{n} \frac{x_j}{n-1} \quad \text{and}$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$$

This particular example illustrates a situation where it is necessary to increase the value of β in Equation 5.

As in[15] we adopted $n = 30$, with 30 bits per variable (a 900 bit long chromosome), with $x_i \in [0, 1]$. The population is formed by 100 individuals evolving in 25000 evaluations. The other parameters used are: two-point crossover (with $p_c = 0.8$), p_m=0.01 and σ_s=0.48862 (using objective-function density measure). The result for one run is shown in Figure 16 for $\beta = 1$. The figure also shows the value of $g(x)$ for each point of the Pareto front achieved. In the optimal-Pareto front one has $g(x)$=1.

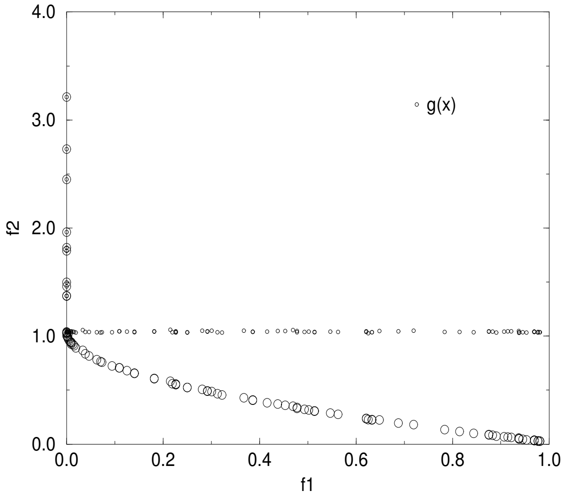

Figure 16 - Results for $\beta = 1$ and $\sigma_s = 0.48862$.

We can observe that several dominated points along the vertical axis are still present in the final population. This happens because, in this problem, it is possible to obtain points in the vertical axis with low domination count and low density which are not removed from the population during the evolution. To circumvent this inconvenience it suffices to set β=3. The results are displayed in Figure 17.

Now the dominated points along the vertical axis disappear, allowing a best distribution of the population over the optimal Pareto front.

In the next experiment it is shown that by using a more adequate value for σ_s the default value $\beta = 1$ can be used. The new value for σ_s corresponds to

$$\sigma_s = \frac{\left|f_1^{max}(x) - f_1^{min}(x)\right| + \left|f_2^{max}(x) - f_2^{min}(x)\right|}{p}$$

following [8], leading to $\sigma_s = 0.02$. The results are displayed in Figure 18.

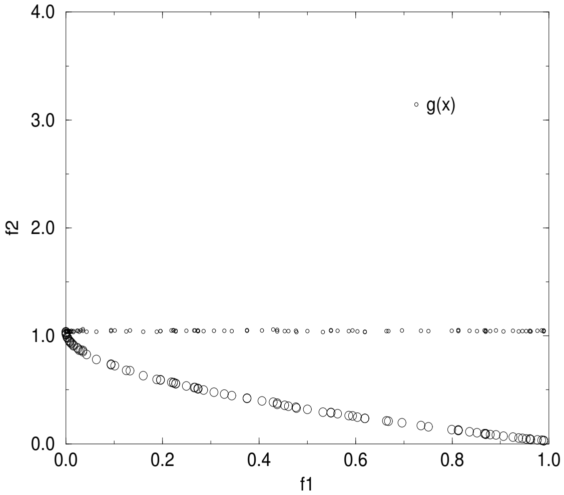

Figure 17 - Results for $\beta = 3$ and $\sigma_s = 0.48862$.

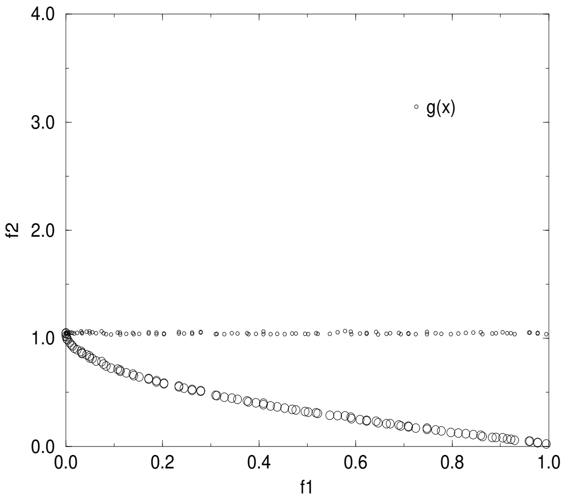

Figure 18 - Results for $\beta = 1$ and $\sigma_s = 0.02$.

The algorithm has also been applied to a larger MOP in structural mechanics[16] with quite satisfactory results. Furthermore, experiments with the other test-problems from [14] also produced very good results.

6 Conclusions

In this paper a non-generational genetic algorithm for multi-objective optimization problems is proposed. For each element in the population a domination count is defined together with a neighborhood density measure based on a sharing function. Those two measures are then non-linearly combined in order to define the individual's fitness. The replacement policy is such that an offspring replaces the worst individual in the current population only if it is better than it.

Numerical experiments with four test-problems taken from the literature were performed and the results obtained were very good in all cases, suggesting that the non-generational scheme, combined with the other algorithmic design choices adopted here, lead to an efficient technique for multi-objective optimization problems.

Bibliography

[1] J.D. Schaffer. *Some Experiments in Machine Learning using Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, TN, 1984.

[2] J.D. Schaffer. "Multiple objective optimization with vector evaluated Genetic Algorithms". In *Proceedings of the 1st Int. Conf. on Genetic Algorithms*, pages pp. 93–100, 1985.

[3] C.M. Fonseca and P.J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

[4] Jeffrey Horn. *Multicriterion Decision Making*, volume 1, pages F1.9:1 – F1.9:15. IOP Publishing Ltd. and Oxford University Press, 1997.

[5] D.A. Van Veldhuizen and G.B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[6] C.A.C. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, 1999.

[7] C.M. Fonseca and P.J. Fleming. "Genetic Algorithm for multiobjective optimization: formulation, discussion and generalization". In *Proceedings of the 5th Int. Conf. on Genetic Algorithms*, pages pp. 416–423, Sao Mateo, CA, 1993.

[8] J. Horn and N. Nafpliotis. "Multiobjective optimization using the niched Pareto Genetic Algorithm". Technical Report 93005, University of Illinois, Illinois Genetic Algorithms Lab., 1993.

[9] N. Srinivas and K. Deb. "Multiobjective optimization using nondominated sorting in Genetic Algorithms". *Evolutionary Computation*, 2(3): pp. 221–248, 1995.

[10] M. Valenzuela-Rendón and E. Uresti-Charre. "A non-generational Genetic Algorithm for multiobjective optimization". In *Proceedings of the 7th Int. Conf. on Genetic Algorithms*, pages pp. 658–665, Sao Mateo, CA, 1997.

[11] Kalyanmoy Deb. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Technical Report CI-49/98, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, 1998.

[12] Jeffrey Horn. *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. PhD thesis, University of Illinois at Urbana Champaign, Urbana, Illinois, 1997.

[13] P. Hajela and C. Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.

[14] Kalyanmoy Deb. "Construction of test problems for multiobjective optimization". In *Proc. of the Genetic and Evolutionary Conf.*, pages pp. 164–171, Orlando, FL, 1999.

[15] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Technical Report 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, December 1999.

[16] C.C.H. Borges. *Genetic Algorithms for Structural Dynamics Optimization Problems*. PhD thesis, Prog. de Engenharia Civil, COPPE/UFRJ, 1999 (In Portuguese).