# Goal-constraint: Incorporating preferences through an evolutionary $\varepsilon$-constraint based method

Ricardo Landa
Information Technology Laboratory
CINVESTAV Tamaulipas
Cd. Victoria, Tamaulipas, MEXICO
Email: rlanda@tamps.cinvestav.mx

Carlos A. Coello Coello
Computer Science Department
CINVESTAV-IPN
Mexico, DF, MEXICO
Email: ccoello@cs.cinvestav.mx

Gregorio Toscano-Pulido
Information Technology Laboratory
CINVESTAV Tamaulipas
Cd. Victoria, Tamaulipas, MEXICO
Email: gtoscano@tamps.cinvestav.mx

***Abstract*—This paper presents the goal-constraint method for incorporating preferences in multiobjective optimization. The preferences are provided in the form of a vector of goals, which is familiar for decision makers and operations researchers. The portion of the Pareto front to be generated is totally defined by the vector of goals, regardless if such a vector is feasible or not. Once defined, it is feasible to experiment on many objective problems, because of the reduced cost of producing less points. The experimental results show good convergence properties, and the graphs illustrate the way the portion of front produced is related to the vector of goals.**

## I. INTRODUCTION

The multiobjective optimization problem consists of finding the decision vector $\mathbf{x}$ that optimizes (minimizes):

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ is the vector of $n$ decision variables and $\mathbf{f}$ is the vector of $m$ objective functions. The problem may or may not have constraints.

However, the concept of optimizing several functions simultaneously is not as simple as to find an optimum for each function, because the functions usually are in conflict to each other. Finding the optimum, then, can be interpreted as finding a good trade-off between all the objectives of the problem.

The most common way to find good trade-offs is through the concept of Pareto optimality. A point $\mathbf{x}^*$ is Pareto optimal, if does not exist any other point $\mathbf{x}$ that dominates it. That is to say, $\mathbf{x}^*$ is Pareto optimal if does not exist any other point $\mathbf{x}$ that fulfills:

$$f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$$

for all $i \in \{1, 2, \ldots, m\}$, and, for at least one $i$:

$$f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$$

After an optimization process has found several Pareto optimal points, the decision maker is the person who must choose the final solution to be implemented. Additionally, the decision maker can incorporate preferences before the optimization method starts. This may help to reduce computational costs, because producing the whole Pareto front is not necessary, and only solutions that follow the preferences are relevant.

Recently, evolutionary computation methods have achieved a great success on solving multiobjective optimization problems. Every year appear more approaches, each time more efficient (requiring less objective function evaluations for obtaining a good approximation of the Pareto optimum).

At the same time, hybrid approaches of evolutionary techniques and mathematical programming methods for multiobjective optimization are not very popular, probably due to the fact that mathematical programming techniques are frequently scalarizing functions, and thus require several executions of a single-objective optimizer to obtain the Pareto set or a sample of it. Conversely, most evolutionary multiobjective approaches obtain a set of nondominated solutions (*i.e.*, an approximation of the Pareto set) in a single run.

Nevertheless, and despite their disadvantages, some approaches have shown that hybrid techniques can be a very effective choice under certain conditions [1]. Those approaches show that mathematical programming techniques tend to produce points of very high quality (*i.e.*, tend to produce Pareto optimal points, or very good approximations of them), even when the problem may appear to be very difficult for most elitist multiobjective evolutionary algorithms based on Pareto ranking. This is maybe because in this case the search focuses on a single point, instead of aiming to converge to a set of them, and therefore, a better exploitation may take place.

In this work, we propose a hybrid of an $\varepsilon$-constraint-like method and a single-objective evolutionary algorithm. The modification to the $\varepsilon$-constraint is made to incorporate preferences, and simultaneously reduce the extension of the generated Pareto front.

The rest of the paper is organized as follows: Section II motivates the incorporation of preferences in an evolutionary algorithm; Section III introduces the goal-constraint approach for incorporating preferences and setting the portion of the Pareto front that will be generated, all through the use of a vector of goals; Section IV describes a procedure for populating the desired portion of the Pareto front after solving the goal-constraint problems; in Section V the experimental results are presented; finally in Section VI some conclusions are found, as well as some ideas for future work.

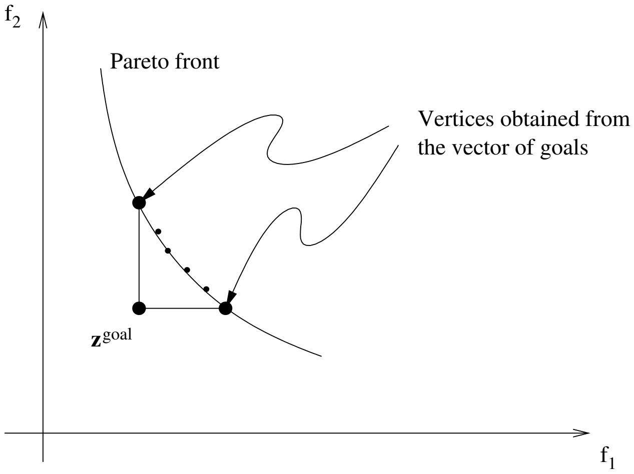

Fig. 1. Use of a vector of goals beyond the Pareto front

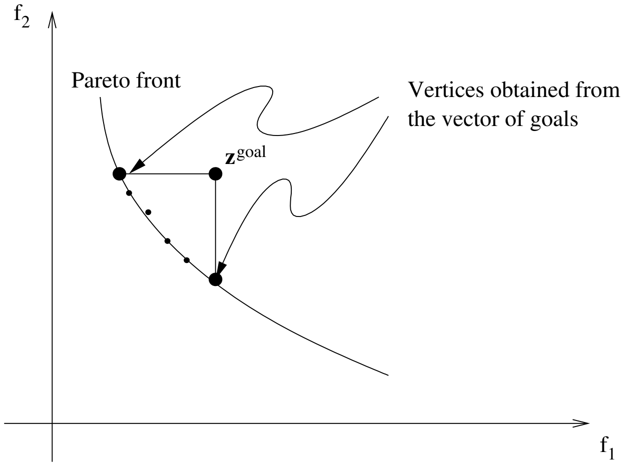

Fig. 2. Use of a vector of goals before the Pareto front

## II. INCORPORATION OF PREFERENCES

Incorporating preferences of the decision maker to a multiobjective optimization process is a common practice in operations research [2]. It consist of expressing the preference for some objectives over the others, or for some region of the objective space, aiming to reduce the total number of solutions representing different compromises, and thus helping to the decision making process.

Evolutionary algorithms are frequently designed to obtain a sample of the whole Pareto front, giving all the information to the decision maker and leaving to him/her the decision of which to implement (*a posteriori* methods) [3]. However, it may be useful to express preferences *a priori* or interactively to the optimization process, for example to reduce the computational cost of generating many points on regions of little interest.

## III. THE GOAL-CONSTRAINT APPROACH

The definition of a vector of goals may appear quite natural for a decision maker, because he/she provides only one point $\mathbf{z}^{\text{goal}} = (z_1^{\text{goal}}, \ldots, z_m^{\text{goal}})$ (one value for each of the $m$ objectives), and expects values near of it. The entries of the $\mathbf{z}^{\text{goal}}$ vector are frequently called *aspiration levels* [2] or *goals*, and there are several methods based on them in the operations research area (such as goal programming, or the reference points models). For this reason, this approach may result familiar to the decision maker.

Figures 1 and 2 show two cases of vectors of goals, one of them beyond the true Pareto front, and one before it. The first case represents an infeasible, unreachable solution, and the second is a feasible solution that can be improved. Both cases are possible if the decision maker has incomplete information of the problem (so he/she does not know where exactly is the Pareto front).

In both cases, an optimization per objective is performed, using a modified version of the $\varepsilon$-constraint problem, whose values for the constraints are taken from the vector of goals (for that reason, we call it *goal-constraint*):

$$\begin{array}{ll} \text{minimize} & f_i(\mathbf{x}) \\ \text{subject to} & f_j(\mathbf{x}) \leq z_j^{\text{goal}} \quad \forall j \in \{1, \ldots, m\}, j \neq i \end{array}$$

This process is carried out for all $i \in \{1, \ldots, m\}$, with the aim of obtaining the vertices of the region to be explored. The solution of each goal-constraint problem is the nearest weakly Pareto optimal solution less or equal to the set of goals $z_j^{\text{goal}}$ with $j \neq i$. If the solution of the problem is unique, it will be Pareto optimal, with the same proximity to the goals. These two characteristics can be proven the same way as the proofs of Pareto optimality for the $\varepsilon$-constraint problem [2].

It is possible that any of these optimization processes cannot find a feasible solution (depending on the shape of the Pareto front, and the vector of goals chosen), but an infeasible solution will work in most of the cases as an approximation for the next steps. If there is no feasible solution for one problem, it is possible to generate one by eliminating some of the $m-1$ constraints of the problem, but this may increase the computational cost of the approach. The only advise for this approach, is that the defined goals are not lower that the corresponding entry of the ideal objective vector (in that case, no feasible solution will be found, because there is no $\mathbf{x}$ that fulfills the constraints of the problem).

It may be the case that those $m$ solutions are enough for the user or decision maker, and the process can stop here. If he/she requires more points to make a decision, any alternative dispersion technique can be used (as that found in [1]). There are some recent proposals of continuation methods for generating points in the curve of the Pareto front given some starting points [4]. Other options are methods that require ranges for incorporation of preferences; in this case, once all the results of the goal-constraint problems have been obtained, the best and worst values for each objective are recorded, and these values will define the ranges for subsequent optimizations (this method is known as payoff table). Either the best or the worst point should be very near to $\mathbf{z}^{\mathrm{goal}}$, depending if such a vector of goals is before or beyond the true Pareto front.

For simplicity, we performed experiments with the $\varepsilon$-constraint method (as detailed in the next section). The $\varepsilon$-constraint method need intervals for varying the $\varepsilon$ values for each objective, and thus generate different Pareto points. Such intervals are commonly obtained from the ideal and nadir objective vectors. For its use with goal-constraint, the ranges are defined by the vector of goals, so no nadir and ideal vectors are needed.

We can only perform a few evaluations of the dispersion technique, to obtain at least one intermediate point. This process will produce several points at a reasonable computational cost, even for problems of ten or more objectives.

### *A. The cultured differential evolution as single-objective optimizer*

For solving each goal-constraint problem, and possibly some additional $\varepsilon$-constraint problems, a method for constrained single-objective optimization is needed. There are multiple proposals in literature, and the user can adopt any of them.

Perhaps the most important property of the selected method is a fast convergence, in order to keep the total number of objective function evaluations low. Here, we followed a previous work that coupled a cultured differential evolution with an $\varepsilon$-constraint method, showing this is a viable alternative, mainly for tackling challenging problems [1].

## IV. OPTIONALLY GENERATING MORE POINTS WITH $\varepsilon$-CONSTRAINT

As mentioned above, if the decision maker requires more than the $m$ solutions generated by the goal-constraint formulations, an alternative is to solve some extra $\varepsilon$-constraint problems. The main issue for $\varepsilon$-constraint formulations is that intervals for the objectives are needed beforehand. This is not a problem for us, because the intervals are defined for the vector of goals and the goal-constraint points obtained.

Let us now assume that it is available as the procedure $cde(f_l, \varepsilon, g)$, which performs the optimization process of the $\varepsilon$-constraint method during $e$ objective evaluations for the $f_l$ objective, and using the values of $\varepsilon$ for constraints. It returns the best point found. The pseudo-code for obtaining extra points is shown in Algorithm 1.

In Algorithm 1, the $\varepsilon$ values are updated with a $\delta$, which depends on the number of points in the Pareto front desired by the user (or decision maker). It is obtained as follows: $\delta_j = \frac{ub_j - lb_j}{p_j - 1}$. This way, we aim that the final points are equally spaced in their projection over the $f_2$ to $f_m$ axes. $e$ is an input parameter of the algorithm, but it is very important, because together with $p_j$, defines the total number of objective function evaluations required by the approach. The number of function evaluations is $e \prod_{j=2}^{m} p_j$.

Algorithm 1 shows $f_1$ as the objective to be optimized, and $f_2$ to $f_m$ as the constraints. However, one can interchange the roles of the objectives if the problem looks harder to solve in the original setting. In the experiments shown in this work, the original setting was always preserved, and $f_1$ was always taken as the objective to optimize, to allow a fair comparison. But, as a suggestion for better results, if it is known which of the objective functions is the most difficult to optimize, such objective function should be chosen to be optimized, and the rest should be adopted as constraints.

**Algorithm 1** $\varepsilon$-Constraint with CDE. CDE procedure can be replaced by any other evolutionary single-objective algorithm with constraint handling

```
$P = \emptyset$
$(\mathbf{lb}, \mathbf{ub}) = solve\_goal - constraint(\mathbf{f})$
$\varepsilon_{j=2,\ldots,m} = lb_j + \delta_j$
while $\varepsilon_m \leq ub_m$ do
  $\mathbf{x} = cde(f_1, \varepsilon, e)$
  if $\mathbf{x}$ is nondominated with respect to $P$ then
    $P = P - \{\mathbf{y} \in P \mid \mathbf{x} \succ \mathbf{y}\}$
    $P = P \cup \{\mathbf{x}\}$
  end if
  $\varepsilon_2 = \varepsilon_2 + \delta_2$
  for $j = 2$ to $m - 1$ do
    if $\varepsilon_j > ub_m$ then
      $\varepsilon_j = lb_j + \delta_j$
      $\varepsilon_{j+1} = \varepsilon_{j+1} + \delta_{j+1}$
    end if
  end for
end while
```

## V. RESULTS

For this approach, it is very difficult to make a comparison of results with another multiobjective approach, because even when there are some approaches that incorporate preferences of the decision maker [5], [6], [7], the resulting covered region of the Pareto front will be different. Thus, we decided to measure only convergence, adopting a unary performance measure. We use the generational distance, $GD$ [8], which measures the differences from the obtained points to the nearest points of the true Pareto front. As it measures distances, a value closer to zero is better.

The $GD$ measure has received some critics because it only measures distance from the true Pareto front, and not the dispersion or region covered. The inverted $GD$ measure somehow alleviates these issues, measuring the differences from every point of a good sample of the true Pareto front to the nearest obtained point. However, when only a portion of the Pareto front is intentionally produced, the inverted $GD$ will always report bad results, whilst the original $GD$ measure will measure convergence, regardless of the portion of the Pareto front generated.

### *A. Test problems*

To make evident the possible advantages of our approach, we looked within the current benchmarks for multiobjective problems that are particularly difficult to solve for current multiobjective evolutionary approaches. Recently, researchers have proposed problem sets that contain very difficult problems [9], [10]: this is, a state-of-the-art evolutionary approach (say, NSGA-II) cannot converge to the true Pareto front within 100,000 objective function evaluations (a typical budget) or

TABLE I. VECTORS OF GOALS ADOPTED FOR THE EXPERIMENTS

| Test Problem | Number of objectives | Vector of goals |
|---|---|---|
| WFG1 | 2 | (0.47, 1.03) |
| WFG2 | 2 | (1.08, 2.12) |
| WFG1 | 3 | (0.23, 2.44, 4.91) |
| WFG2 | 3 | (0.29, 0.79, 0.77) |
| WFG1 | 5 | (1.48, 1.80, 1.90, 4.78, 7.03) |
| WFG2 | 5 | (1.75, 0.58, 5.14, 0.91, 6.54) |
| WFG1 | 10 | (1.97, 3.35, 0.40, 1.41, 8.35, 2.77, 2.80, 4.02, 14.67, 7.18) |
| WFG2 | 10 | (1.50, 3.70, 5.90, 3.84, 9.84, 9.78, 1.33, 13.10, 17.44, 8.30) |

even more. It is precisely in these "hard" problems where a hybrid approach may appear advantageous, mainly when we use a fast convergence single-objective evolutionary approach for the hybridization.

We adopted a recent benchmark proposed by Huband et al. [11]. This benchmark was constructed using a block-oriented approach, where each block introduces a desired feature to the problem. For example, there are blocks for making the problem non-separable, deceptive, multimodal, etc. The shape of the Pareto front is also controlled with blocks, and it is possible to design linear, concave, convex, mixed or disconnected fronts.

In this benchmark we found very hard problems (WFG1 and WFG2), each of them with 24 decision variables. WFG1 is strongly biased toward small values of the first 4 variables and WFG2 is non-separable and also has a disconnected Pareto front.

We performed experiments on WFG1 and WFG2 with two and three objectives, to show the results graphically. Then, in order to show the potential of the technique, some experiments were performed on the same test problems with five and ten objectives.

### *B. Experimental setup*

The vectors of goals provided were obtained randomly between the ideal and nadir objective vectors. In Table I are shown the vectors of goals adopted. The rest of the parameters were adopted to obtain 20 points.

The parameters adopted for the two-objective problems were: $p = 5$, $e = 4,500$ for WFG1 and $e = 1,500$ for WFG2, with 10% of the population shared between optimizations (this is to share information between optimizations, since the intermediate problems are very similar). For the $cde$ procedure we used $\mu = 20$ (population size), $F = 0.7$, $CR = 0.5$ (strategy parameters). The maximum number of function evaluations for the dispersion technique was set to $7,500$. With these parameters, the approach performed $30,000$ function evaluations for WFG1 and $15,000$ for WFG2.

The parameters adopted for the three-objective problems were: $p = 3$, $e = 3,500$ for WFG1 and $e = 1,500$ for WFG2, with 10% of the population shared between optimizations (this 10% is chosen at random). For the $cde$ procedure we used $\mu = 20$, $F = 0.7$, $CR = 0.5$. The maximum number of function evaluations for the dispersion technique was set to $8,500$. With these parameters, the approach performed $40,000$ function evaluations for WFG1 and $22,000$ for WFG2.

TABLE II. RESULTS OF THE $GD$ MEASURE (A SMALLER VALUE IS BETTER)

| Test Problem | Number of objectives | $GD$ measure |
|---|---|---|
| WFG1 | 2 | 0.0142 |
| WFG2 | 2 | 0.0015 |
| WFG1 | 3 | 0.0511 |
| WFG2 | 3 | 0.0124 |
| WFG1 | 5 | 0.4176 |
| WFG2 | 5 | 0.2379 |
| WFG1 | 10 | 1.2867 |
| WFG2 | 10 | 0.6675 |

The parameters adopted for the five-objective problems were: $p = 2$, $e = 2,000$ for WFG1 and $e = 750$ for WFG2, with 10% of the population shared between optimizations (chosen at random). For the $cde$ procedure we used $\mu = 20$, $F = 0.7$, $CR = 0.5$. The maximum number of function evaluations for the dispersion technique was set to $8,000$. With these parameters, the approach performed $40,000$ function evaluations for WFG1 and $20,000$ for WFG2.

The parameters adopted for the ten-objective problems were: $p = 1$, $e = 40,000$ for WFG1 and $e = 15,000$ for WFG2, with 10% of the population shared between optimizations (chosen at random). For the $cde$ procedure we used $\mu = 20$, $F = 0.7$, $CR = 0.5$. The maximum number of function evaluations for the dispersion technique was set to $10,000$. With these parameters, the approach performed $50,000$ function evaluations for WFG1 and $25,000$ for WFG2.

### *C. Results*

The results for the two-objective problems adopted are shown in Figures 3 and 4, while the results for the three-objective problems are shown in Figures 5 and 6.

The results of the application of the $GD$ measure on all the instances adopted are in Table II.

According to the $GD$ measure, the approach decreases in quality as we increase the number of objectives. This sort of behavior is exhibited by most other evolutionary multiobjective approaches as well, and is subject of current research. It is often considered part of the "curse of dimensionality".

Another source for the degradation of the quality is that, as the number of objectives increases, the ranges for the higher objectives also increase (for the WFG problems), as can be seen from their expressions [11].

Regarding the different test problems, the technique obtain better results for WFG2, for all the number of objectives adopted here. These results are consistent with other works involving WFG1, which has shown to be a very difficult problem.

The overall results are very good, lower than 0.1 for three objectives, lower than 0.5 for five objectives, and lower than 1.5 for ten objectives. When comparing these values is important to keep in mind the increase of the ranges of higher objectives for WFG problems.

## VI. CONCLUSIONS AND FUTURE WORK

Hybrid evolutionary and mathematical programming methods for multiobjective optimization are typically considered

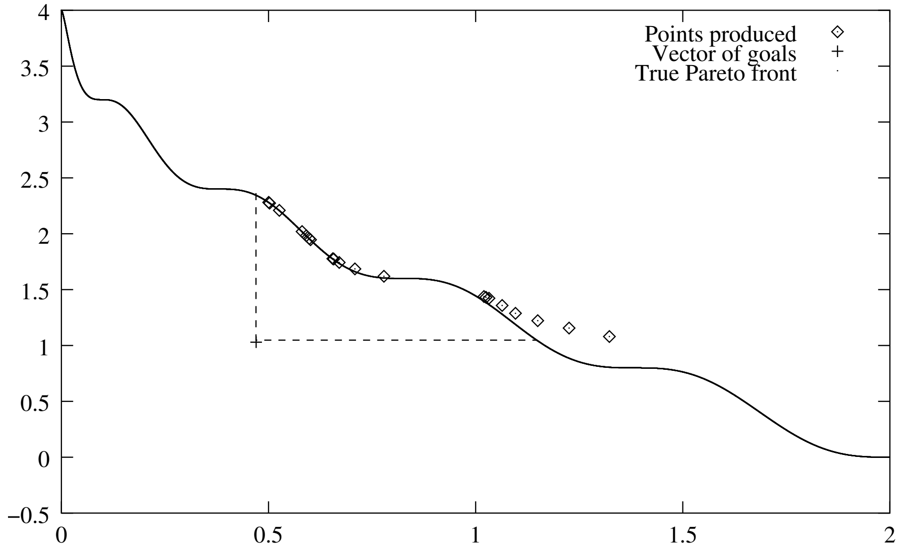

Fig. 3. Results for the two-objective WFG1

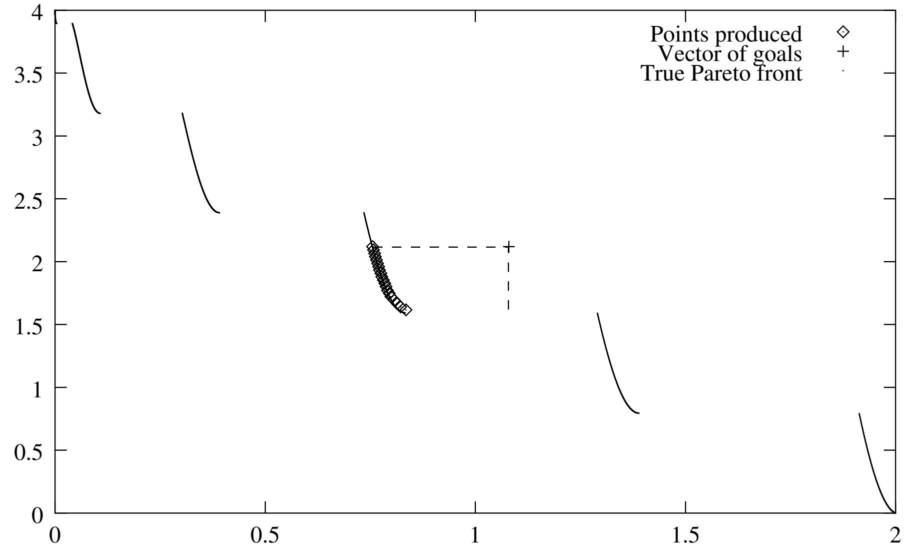

Fig. 4. Results form the two-objective WFG2

computationally expensive, but the cost can be reduced if preferences are incorporated. Such reduction takes place because the search is focused on a small region of the Pareto front, and fewer points are still acceptable.

The approach proposed here is based on $\varepsilon$-constraint, and this technique requires initial intervals for performing the single-objective optimizations. This problem is alleviated by using the same information of the vector of goals provided (the user preferences).

The incorporation of preferences with the goal-constraint method proposed here, has two main advantages compared with an $\varepsilon$-constraint based approach without preferences:

- To reduce the computational cost related to obtaining the ideal and nadir objective vectors for setting the intervals. The ideal and nadir objective vectors are not necessary when information about the region of desired solutions is available, because individual optimizations are carried out only inside that region.
- To alleviate the issues of the algorithm when solving many-objective problems. Because the number of desired solutions as outcome may be smaller (because they are closer to the desired values), is not necessary

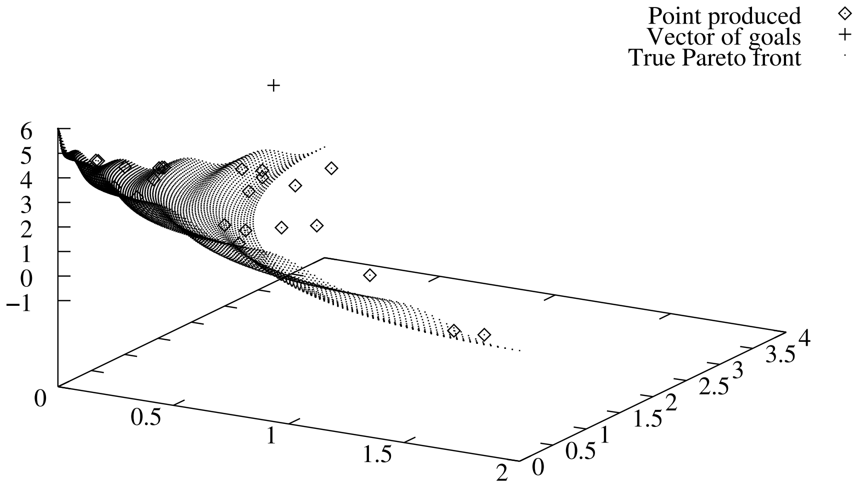

Fig. 5. Results for the three-objective WFG1

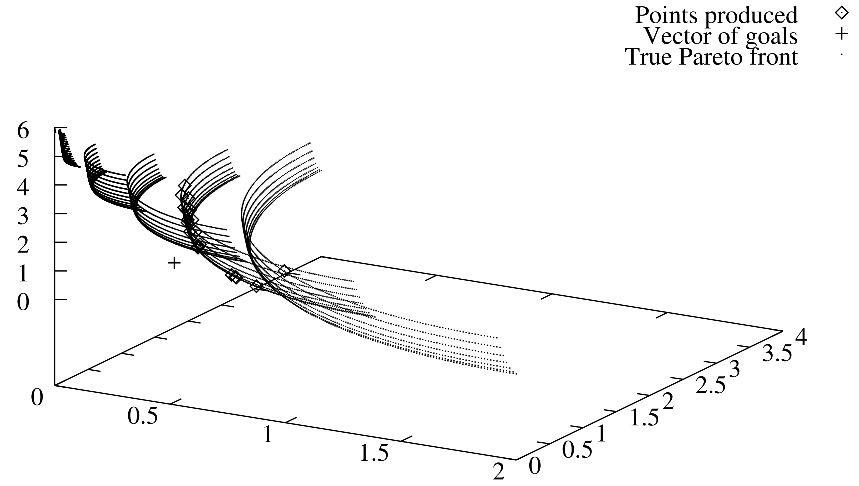

Fig. 6. Results for the three-objective WFG2

to perform an aggressive sub-division of the objective space. This advantage allow us to experiment in problems with up to ten objectives.

The obtained results are good regarding the $GD$ measure of convergence. No binary metrics were applied because comparisons with other algorithms at this stage are very difficult, due to the variety of techniques for incorporating preference information.

Comparisons are only possible if the aim is reaching the same portion of the Pareto front. In this sense, the use of the goal-constraint approach for defining such portion can be adopted for other approaches, allowing fair comparisons.

As part of the future work is the efficient generation of intermediate solutions, once the extrema are determined. Some recent approaches suggest the use of interpolation-based operators for this task, with very good results.

Finally, the vectors of goals were just generated at random in this work. Setting such vectors can be done using some methodology, maybe dependent on the problem.

## VII. ACKNOWLEDGMENT

The third author gratefully acknowledges support from CONACyT through project 105060.

## REFERENCES

[1] R. Landa Becerra, C. A. Coello Coello, and A. G. Hernandez-Diaz, “Alternative techniques to solve hard multi-objective optimization problems,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO’2007)*. ACM Press, 2007.

[2] K. Miettinen, M. M. Mäkelä, and J. Mäkinen, "Handling Constraints with Penalty Techniques in Genetic Algorithms - A Numerical Comparison," University of Jyväskylä, Department of Mathematical Information Technology, Series B, Scientific Computing, Tech. Rep. B10/1999, 1999.

[3] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer Academic Publishers, May 2002, iSBN 0-3064-6762-3.

[4] O. Schütze, A. Lara, and C. A. C. Coello, "Evolutionary continuation methods for optimization problems," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2009)*, 2007, pp. 651–658.

[5] D. Cvetković and I. C. Parmee, "Preferences and their Application in Evolutionary Multiobjective Optimisation," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 42–57, February 2002.

[6] E. Fernández and J. C. Leyva, "A method based on multiobjective optimization for deriving a ranking from a fuzzy preference relation," *European Journal of Operational Research*, vol. 154, no. 1, pp. 110–124, April 2004.

[7] B. Rekiek, P. de Lit, and A. Delchambre, "Hybrid assembly line design and user's preferences," *International Journal of Production Research*, vol. 40, no. 5, pp. 1095–1111, March 2002.

[8] D. A. V. Veldhuizen and G. B. Lamont, "On Measuring Multiobjective Evolutionary Algorithm Performance," in *2000 Congress on Evolutionary Computation*, vol. 1. Piscataway, New Jersey: IEEE Service Center, July 2000, pp. 204–211.

[9] S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, October 2006.

[10] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff, "On Test Functions for Evolutionary Multi-objective Optimization," in *Parallel Problem Solving from Nature - PPSN VIII*, X. Yao *et al.*, Eds. Birmingham, UK: Springer-Verlag. LNCS Vol. 3242, September 2004, pp. 792–802.

[11] S. Huband, L. Barone, L. While, and P. Hingston, "A Scalable Multi-objective Test Problem Toolkit," in *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, C. A. Coello Coello *et al.*, Eds. Guanajuato, México: Springer. LNCS Vol. 3410, March 2005, pp. 280–295.