Seeding the Initial Population of a Multi-Objective Evolutionary Algorithm using Gradient-Based Information

Alfredo G. Hernández-Díaz, Carlos A. Coello Coello, Fátima Pérez,
Rafael Caballero, Julián Molina, Luis V. Santana-Quintero

***Abstract*— In the field of single-objective optimization, hybrid variants of gradient-based methods and evolutionary algorithms have been shown to perform better than an evolutionary method by itself. This same idea has been recently used in Evolutionary Multiobjective Optimization (EMO), obtaining also very promising results. In most cases, gradient information is used along the whole process, which involves a high computational cost, mainly related to the computation of the step lengths required. In contrast, in this paper we propose the use of gradient information only at the beginning of the search process. We will show that this sort of scheme maintains results of good quality while considerably decreasing the computational cost. In our work, we adopt a steepest descent method to generate some nondominated points which are then used to seed the initial population of a multi-objective evolutionary algorithm (MOEA), which will spread them along the Pareto front. The MOEA adopted in our case is the NSGA-II, which is representative of the state-of-the-art in the area. To validate our proposal, we adopt box-constrained continuous problems (the ZDT test suite). The gradients required are approximated using quadratic regressions. Our proposed approach performs a total of 2000 objective function evaluations, which is much lower than the number of evaluations normally adopted with the ZDT test suite in the specialized literature. Our results are compared with respect to the "pure" NSGA-II (i.e., without using gradient-based information) so that the potential benefit of these initial solutions fed into the population can be properly assessed.**

I. Introduction

MOEAS have been very successful in the solution of a wide variety of problems, mainly during the last few years [4]. However, for certain types of applications, MOEAs turn out to be particularly expensive (computationally speaking), since they require a relatively large number of objective function evaluations in order to produce an acceptable approximation of the true Pareto front.

On the other hand, the classical (exact) methods for (multi-objective) optimization (gradient-based methods) normally require a low number of evaluations, but can get easily trapped in local optima and require a lot of assumptions about the problems to be solved: continuity, differentiability, explicit mathematical formulation, etc.

Also, it is well known that, under proper assumptions, Newton's method is quadratically convergent, but its efficiency is reduced by its expensive computational cost, especially, for mid- and large-scale problems. The key point to make this sort of method of practical use is to evaluate the gradient and the Hessian matrix efficiently. For that sake, two different approaches are available:

- **Use of analytical derivatives:** The first option is obtaining the exact (analytical) derivatives of each function by hand and evaluate them. But this is only possible if an explicit mathematical formulation is available for the objective functions. This is precisely the main weakness of this approach, since many interesting problems may not be solvable with it (e.g., simulation-based problems, design problems, etc.). On the other hand, obtaining the derivatives by hand is an error-prone process, since it is relatively easy to make a mistake when obtaining the derivative of a complicated function.

- **Use of estimated derivatives:** In this category, we can find the Newton-like methods, where derivatives are estimated in some efficient way (e.g., using finite differences). These methods do not require the derivatives to be obtained by hand, but require more objective function evaluations to compute the estimation of the derivatives (or the Hessian matrix).

In this work, we use **"global" estimated derivatives** through quadratic regression, but consuming the lowest possible number of evaluations (using them only at the beginning) while maintaining a high quality on the results. The regression is done on the whole decision space, aiming to obtain a global movement direction instead of a proper (or precise) derivative (which can get easily trapped in local optima). On the other hand, instead of using this gradient-based information along the entire evolutionary process (which would consume too many evaluations) we will just use it at the beginning to seed the MOEA. Thus, the main role of this gradient-based method will be driving the MOEA directly to the exact Pareto front and then let it spread solutions along the Pareto front using its own diversification mechanisms.

Alfredo G. Hernández-Díaz is with the Department of Quantitative Methods, Pablo de Olavide University, Spain (email: agarher@upo.es).

Carlos A. Coello Coello is with the Departamento de Computación, CINVESTAV-IPN, México (email: ccoello@cs.cinvestav.mx).

Fátima Pérez is with the Department of Applied Economics, University of Málaga, Spain (email: f_perez@alu.uma.es).

Rafael Caballero is with the Department of Applied Economics, University of Málaga, Spain (email: r_caballero@uma.es).

Julián Molina is with the Department of Applied Economics, University of Málaga, Spain (email: julian.molina@uma.es).

Luis V. Santana-Quintero is with the Departamento de Computación, CINVESTAV-IPN, México (email: lvspenny@hotmail.com).

II. Related work

Several hybrids between a MOEA and a mathematical programming technique have been proposed in the specialized literature in the last few years. The main idea has normally been to use the MOEA to guide the search towards a good search region (i.e., to perform a global search) and then use a mathematical programming technique (e.g., gradient information) to perform a fine-grained search (i.e., local search) that identifies the exact location of the optima. Next, we will briefly discuss the papers that are more closely related to our work:

- In [9], on each generation, for several randomly selected solutions in the population, they convert the multi-objective optimization problem into a single-objective optimization problem through the use of the ε-constraint method (see for example [14]) and solve it with a **Newton-like** method (the Sequential Quadratic Programming (SQP) method), in order to improve this solution. They obtain very good results in terms of quality, but consume a lot of objective function evaluations in some cases.
- In [7], they use a multilevel subdivision technique that subdivides the search space, and perform local search in each subspace. This local search is based on a similar derivation of a single descent direction used in [12]. Again, **exact derivatives** are used, and some problems can be found if the objectives have different ranges, because the largest direction of simultaneous descent will be biased towards the objective with the largest range.
- In [1], they analytically describe the complete set of nondominated simultaneously improving directions using the **exact gradient** of each objective function, and this set is considered as a multi-objective gradient. In order to use this information, at the end of a generation, a set of candidate solutions is determined. The gradient-based local search operator is then applied with each of these candidate solutions as a starting point. The performance of this approach turns out to be good in bi-objective problems, but not so good in problems with more than 2 objectives, as explained by the authors. On the other hand, they find problems when moving a solution towards the boundary between the feasible and the infeasible regions, and the number of objective function evaluations consumed by the approach is also high.
- In [2], they use **exact derivatives**, and try to answer a key question: what is the best way to integrate the use of gradient techniques in the cycle of a MOEA? They propose an adaptive resource-allocation scheme that uses three gradient techniques: 1) a conjugate-gradient algorithm, which is applied to a randomly chosen objective, 2) an alternating-objective repeated line-search and 3) a combined-objectives repeated line-search. During the optimization process, the effectivity of the gradient techniques is monitored and the available computational resources are redistributed to allow the (currently) most effective operator to spend the most resources. The quality of the results varies a lot, but again, a high number of objective function evaluations is consumed by the approach. Additionally, the exact derivatives of the objectives are required.
- In [13], two methods for unconstrained multi-optimization problems are used as a mutation operator in a state-of-the-art MOEA. These operators require gradient information which is **estimated** using a finite differences method and a stochastic perturbation technique requiring few function evaluations. The results are very promising, but the number of evaluations is still high, since the gradient-based operator is used along the entire optimization process.
- In [3], they design a population-based **estimation** of the multi-objective gradient, although a complete algorithm is not described in this paper. Also, no experimentation is provided, because their aim is to give an indication of the power of using directional information.
- In [8], the Multiobjective Steepest Descent Method (MSDM) defines the degree of improvement in each objective function when a solution is moved in a direction as the inner product of the direction and the steepest descent direction (using **exact derivatives**) of the corresponding objective function. MSDM finds the direction that maximizes the minimum degree of improvement of all the objective functions by solving a quadratic programming problem and moves the solution in that direction. When a solution is on a feasible region boundary, it incorporates the boundary information into the quadratic programming problem to exclude infeasible directions. MSDM is computationally expensive, since a quadratic programming problem has to be solved to find a single direction.

As we will see in the remainder of this paper, by seeding the initial population of a MOEA with gradient-based information, our proposed approach consumes a lower number of objective function evaluations than any of the proposals previously discussed, while still producing reasonably good results.

III. Definitions and basic concepts

A. Multi-objective Optimization

We consider multiobjective optimization problems (MOPs) of the form

$$\begin{array}{ll} \text{minimize} & (f_1(x), f_2(x), \ldots, f_p(x)) \\ \text{subject to} & x \in X \subseteq \mathbf{R}^n, \end{array} \tag{1}$$

where:

- n is the number of decision variables, $x = (x_1, x_2, \ldots, x_n)$,
- $X \subset \mathbf{R}^n$ is the set of feasible solutions, and
- $(f_1, f_2, \ldots, f_p)$ is the vector with the p objective functions.

A feasible solution $x^* \in X$ is Pareto optimal for problem (1) if it does not exist any other solution $x \in X$, such that

$$f_i(x) \leq f_i(x^*) \text{ for all } i = 1, \ldots, p$$

with at least one $j \in \{1, \ldots, p\}$ such that $f_j(x) < f_j(x^*)$. The set of all Pareto optimal solutions of (1) is called the Pareto front.

B. Single-objective Optimization

Given a function $f : \mathbf{R}^n \rightarrow \mathbf{R}$, for $x \in \mathbf{R}^n$ the gradient of f at x, $\nabla f(x)$, is an n-dimensional vector with entries:

$$(\nabla f(x))_i = \frac{\partial f}{\partial x_i}(x).$$

A direction $v \in \mathbf{R}^n$ is a *descent direction* if:

$$\nabla f(x)v < 0. \tag{2}$$

A generalized gradient method can be summarized in the following equation:

$$x^{k+1} = x^k + \alpha^k v^k$$

where v^k is a descent direction and α^k is the step size. One of the most commonly used choice for the descent direction is the following (*steepest descent*):

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k).$$

Choosing the optimum step size α^k is desirable, but it may be computationally expensive. Hence, some other set of rules, which has good properties (i.e., convergence), is more efficient. One of the most efficient is the Armijo's rule:

Let $\beta \in (0, 1)$ be a prespecified value, let v be a descent direction and let x be the current point. The condition to accept t (the step size) is:

$$f(x + tv) \leq f(x) + \beta t \nabla f(x) v$$

where we start with $t = 1$ and while this condition is not satisfied we set $t := t/2$.

The choice of β can be critical because the bigger the value of β, the bigger the steps we can implement at the beginning; but the more evaluations that can be consumed if too many reductions of t must be done to achieve the condition.

Armijo's rule is mathematically correct and the "t" value always exists. However, this value could be very small, which would be translated into an insignificant progress (this is, in fact, the main disadvantage of Armijo's rule). This problem is more significant for box-constrained problems (and/or for constrained problems, which are not considered in this work) when the current solution is close to the boundary between the feasible and infeasible regions, or to the boundary of one of the decision variables, and the descent direction moves it out of the feasible space. In those cases, we apply the following rules for each decision variable i:

- If $x_i^{k+1} < linf[i]$, then $x_i^{k+1} = \frac{linf[i]+x_i^k}{2}$.
- If $x_i^{k+1} > lsup[i]$, then $x_i^{k+1} = \frac{lsup[i]+x_i^k}{2}$.

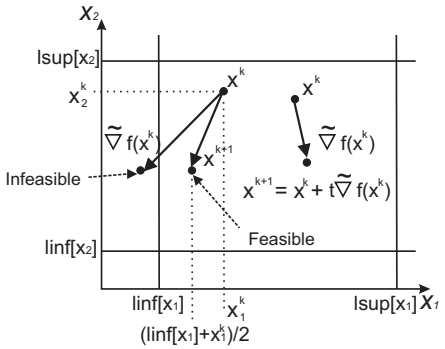

Fig. 1. Adapted Armijo's rule for box-constrained problem allows two types of movements, for feasible and infeasible solutions.

where $linf[i]$ and $lsup[i]$ denote the lower limit and the upper limit for variable x_i, for $i = 1, 2, \ldots n$. This way, the current solution x^k is not always moved in the same direction induced by the estimated gradient but, however, it follows the signs of the gradient. In some sense, this adapted Armijo's rule is considering different step sizes in each coordinate.

In Figure 1 we have represented the two possibilities adopted:

- If $x^{k+1} = x^k + t\nabla f(x^k)$ is infeasible, because its $i-th$ coordinate is out of range, it is replaced by the mean value.
- A feasible x^{k+1}.

IV. Gradient-based method for MOPs

The main idea of our proposed approach is based on the Fritz-John optimality condition for unconstrained MOPs (see for example [8]):

Given a point $x \in X$, a necessary condition for this point to be a Pareto optimal solution is the existence of $\vec{\lambda} \geq 0$ such that:

$$\sum_{i=1}^{p} \lambda_i \nabla f_i(x) = 0.$$

Thus, for a bi-objective optimization problem, this condition means that for any Pareto optimal solution, we can find some $\lambda \geq 0$ such that:

$$\nabla f_1(x) = -\lambda \nabla f_2(x).$$

This is, for any Pareto optimal point, gradients of both objective functions are parallel but in the opposite direction. It means that if we are placed in the minimum of one of the objectives (for example the minimum of f_1) and we follow the direction of $\nabla f_2(x)$, we will remain in the Pareto front. This is shown graphically in Figure 2.

This idea was used in [11], where they link $p + 1$ local searches (more precisely, tabu searches). The first local search starts from an arbitrary point and attempts to find the optimal solution to the problem with the single objective f_1. Let x_1 be the last point visited at the end of this search. Then, a local search is applied again to find the

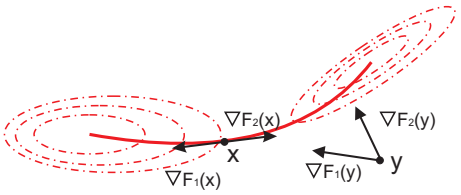

Fig. 2. Pareto front on a bi-objective problem

best solution to the problem with the single objective f_2 using x_1 as the initial solution. This process is repeated until all the single-objective problems associated with the p objectives have been solved. At this point, they solve again the problem with the first objective f_1 starting from x_p, to finish a cycle around the efficient set. This phase yields the p efficient points that approximate the best solutions to the single-objective problems that result from ignoring all but one objective function, and additional efficient solutions may be found during this phase because all visited points are checked for inclusion in the approximation of the Pareto front, as probably most of the intermediate points will lie on the Pareto front. This way, an initial set of efficient points to be used as the initial population for a MOEA are generated in [11].

In this work, we are going to use the same idea, namely to link $p+1$ single objective local searches, but using a single-objective gradient based method instead of a tabu search. The next subsection is devoted to show the main features on this gradient-based local search mechanism.

A. Single-objective gradient-based method

For our local search engine, we are going to use a steepest descent method, this is, given the current point x^k, the next point will be computed as follows:

$$x^{k+1} = x^k - t \cdot \widetilde{\nabla} f(x^k)$$

where $\widetilde{\nabla} f(x^k)$ is an estimation of $\nabla f(x^k)$, and the step length (t) will be computed following our adapted Armijo's rule with $\beta = 0.1$ and starting with the value of $t = 1$. The reason to choose a low value for β is the fact that small steps are also interesting for us while we are on the Pareto front, as we are checking every intermediate solution for being included in the final approximation. This is, we are not only interested in the final point of each search, but also in the intermediate points.

To estimate the gradient of a function f, we will use a quadratic approximation:

$$f(x) \approx \beta_0 + \sum_{i=1}^{n} \beta_i^1 \cdot x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} \beta_{i,j}^2 \cdot x_i \cdot x_j$$

The number of parameters (N) to adjust such an approximation for a function with n variables is:

$$N = 1 + n + \frac{n(n+1)}{2} = \frac{n^2 + 3n + 2}{2}$$

N represents the minimum number of points needed to adjust such an approximation. For a problem with 30 variables, for example, at least 496 will be needed. In order to generate these N points efficiently, we used Latin-Hypercubes [10], which is a method that guarantees a good distribution of the initial population in a multidimensional space, as it is required in order to better fit the function with this quadratic approximation. Once these points are generated and evaluated, we compute the values of each parameter solving the corresponding system of equations using a pseudo-inverse (X is not always a square matrix). This system of equations can be formulated using matrices:

$$X \cdot B = Y$$

or, equivalently:

$$\left(\begin{array}{c|c|c} 1 & (x_i^1) & (x_i^1 \cdot x_j^1) \\ 1 & (x_i^2) & (x_i^2 \cdot x_j^2) \\ \vdots & \vdots & \vdots \\ 1 & (x_i^N) & (x_i^N \cdot x_j^N) \end{array}\right) \left(\begin{array}{c} \beta_0 \\ \beta_i^1 \\ \vdots \\ \beta_{i,j}^2 \end{array}\right) = \left(\begin{array}{c} f(\vec{x_1}) \\ f(\vec{x_2}) \\ \vdots \\ f(\vec{x_N}) \end{array}\right)$$

In case the X matrix is not square, this system of linear equations can be solved by using the pseudo-inverse of X, that is:

$$B = (X^t X)^{-1} \cdot X^t \cdot Y$$

Finally, we assumed the following stopping conditions:

1) The step is too small: $t \cdot \|\nabla f(x_k)\| < \epsilon$, or
2) The improvement is too small: $|f(x_{k+1}) - f(x_k)| < 0.1 * \epsilon$

The complete method is summarized in Algorithm 1.

Algorithm 1 Pseudo-code of our Multi-Objective Gradient-Based method

```
Generate a set InitPop with N initial points using Latin-Hypercubes.
Send each point in InitPop to the list of nondominated solutions: PF.
Use the set InitPop to adjust a quadratic approximation of each objective function over all its domain.
while iteration < NumIter do
    for each objective function f_i (repeating the first one) do
        x_0 = last point visited or random solution in PF when i = 0
        while stopping conditions = FALSE do
            Obtain x_{k+1} through the single-objective gradient-based method for f_i.
            Send x_{k+1} to PF.
        end while
    end for
end while
```

V. Hybridization and Preliminary Results

In order to show some preliminary results, we have used our proposed Multi-Objective Gradient-Based method to seed the NSGA-II [6], which is a MOEA representative of the state-of-the-art in the area. The seeding procedure consumes about 1000 objective function evaluations (500 are required for the quadratic regression) while the NSGA-II consumes another 1000 objective function evaluations.

The first phase is not focused on approximating the Pareto front but it tries to obtain some good points, while consuming few evaluations. After that, this initial set of solutions is used in the NSGA-II as its initial population. If the resulting set of solutions is too small or the number of points is not a multiple of 4, it is completed with randomly generated solutions until the population size is 52 (we experimentally found that this is the minimum size population needed by the NSGA-II in order for this approach to show a good performance). If the population size is bigger than 52, it is completed until the nearest multiple of 4 value. In the second phase, the NSGA-II is seeded by this population and runs for a certain number of generations.

In order to allow a fair comparison of results, the seeded NSGA-II is compared with two versions of the NSGA-II: (1) one that uses a random initial population size of 52 solutions and (2) another one that uses a random initial population of 100 points. This second version was adopted, because this is the standard population size that several other authors have used in their experiments.

To assess the performance of the seeded NSGA-II, we used five test problems from the **ZDT** set [15]: ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6.[1] Table I shows the definitions of these problems. We first run Newton's method until 1000 evaluations are consumed. Then, the NSGA-II uses this initial population to complete the approximation of the Pareto front during 1000 additional evaluations.

The first phase of our approach uses two parameters: $NumIter$ and ϵ, that were set as follows: $NumIter = 2$ and $\epsilon = 0.001$ for all cases. For the NSGA-II, we adopted real-numbers encoding and the following parameters setting: crossover rate = 0.9, mutation rate = 1/num_var (num_var = number of decision variables), η_c = 15, $\eta_m = 20$. For the number of generations, we have three different settings: for the seeded NSGA-II (the population size is 52) the maximum number of generations is set to 20; for the 52-population-size random NSGA-II the number of generations is 40 and for the 100-population-size random NSGA-II the number of generations is 20. Thus, in all cases, the total number objective function evaluations is 2000.

In order to allow a quantitative comparison of results, we adopted three performance measures which are described next. Note that although the issue of more interest to us is to measure convergence, we have also included a performance measure related to the spread of the solutions obtained:

- **Size of the space covered (SSC)**: This performance measure was proposed by Zitzler and Thiele [16], and it measures the hypervolume of the portion of the objective space that is dominated by the approximation set A, which is to be maximized. In other words, SSC measures the volume of the dominated points. Hence, the larger the SSC value, the better. This measure is computed by using a Monte Carlo process with 1,000,000 randomly generated points.

$$SSC = |\{x \in X | \exists z \in A \text{ such that } z_i \leq x_i \forall i\}|$$

- **Unary additive epsilon indicator ($I^1_{\varepsilon+}$)**: The epsilon indicator family has been introduced by Zitzler et al. [17] and comprises a multiplicative and an additive version. We decided to use the unary additive epsilon indicator ($I^1_{\varepsilon+}$). The unary additive epsilon indicator of an approximation set A ($I^1_{\varepsilon+}(A)$) gives the minimum factor ϵ by which each point in the real front R can be added such that the resulting transformed approximation set is dominated by A:

$$I^1_{\varepsilon+}(A) = inf_{\epsilon \in \mathbf{R}}\{\forall z^2 \in R \backslash \exists z^1 \in A : z^1_i \leq z^2_i + \epsilon \; \forall i\}.$$

  $I^1_{\varepsilon+}(A)$ is to be minimized and a value equal to 0 implies that A strictly dominates the real front R.

- **Spread (Δ)**: In order to measure both the spread of the approximation set A and the distances from the extreme points of A to the extremes of the real Pareto front R, we use *Spread* ([5]):

$$\Delta = \frac{\sum_{m=1}^{2} d_m^e + \sum_{i=1}^{|A|} |d_i - \overline{d}|}{\sum_{m=1}^{m} d_2^e + |A| \cdot \overline{d}}$$

  where d_i has been taken to be the Euclidean distance of the $i-th$ point in A to the $i+1-th$ point in A (once these points are ranked in ascending order), $\overline{d}$ is the mean value of d_i, d_m^e is the Euclidean distances between the extreme solutions of both fronts corresponding to the $m-th$ objective function ($m = 1, 2$). So, $0 \leq \Delta \leq \infty$ and the lower the value of Δ, the better the distribution of vectors in A. A perfect distribution, that is $\Delta = 0$, means that $d_i = \overline{d}$ for all i and $d_m^e = 0$ for all m (so the extremes of the true Pareto front have been achieved).

Table II shows a summary of our results. For each test problem, we performed 11 independent runs per algorithm. The results reported in Table II show the mean values for each of the three performance measures and the standard deviation of the 11 runs performed. The best mean values in each case are shown in **boldface** in Table II. Also, the last column in Table II shows the mean sizes of the final nondominated solution sets.

It can be observed in Table II that the seeded NSGA-II produced the best mean values in most cases. Regarding SSC and the unary additive epsilon indicator, there was only one case in which the NSGA-II outperformed our approach. Finally, regarding the Spread measure, the random NSGA-II outperformed our approach only in two cases. This is certainly remarkable if we consider the fact that the seeding procedure is only focused on convergence aspects. Thus, it

[1] ZDT5 was excluded, because it is a binary problem.

Test Function	Objectives	Bounds
ZDT1	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}, g) = 1 - \sqrt{f_1/g(\vec{x})}$ where: $g(\vec{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	n = 30 $0 \leq x_i \leq 1$ $i = 1, 2, \ldots, 30$
ZDT2	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}, g) = 1 - (f_1/g(\vec{x}))^2$ where: $g(\vec{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	n = 30 $0 \leq x_i \leq 1$ $i = 1, 2, \ldots, 30$
ZDT3	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}, g) = 1 - \sqrt{f_1/g(\vec{x})} - (f_1/g)sin(10\pi f_1)$ where: $g(\vec{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	n = 30 $0 \leq x_i \leq 1$ $i = 1, 2, \ldots, 30$
ZDT4	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}, g) = 1 - \sqrt{f_1/g(\vec{x})}$ where: $g(\vec{x}) = 1 + 10(m-1) + \sum_{i=2}^{n}(x_i^2 - 10cos(4\pi x_i))$	n = 10 $-5 \leq x_i \leq 5$ $i = 1, 2, \ldots, 10$
ZDT6	$f_1(\vec{x}) = 1 - exp(-4x_1)sin^6(6\pi x_1)$ $f_2(\vec{x}, g) = 1 - (f_1/g(\vec{x}))^2$ where: $g(\vec{x}) = 1 + 9(\sum_{i=2}^{n} x_i/(n-1))^{0.25}$	n = 10 $0 \leq x_i \leq 1$ $i = 1, 2, \ldots, 30$

TABLE I

OBJECTIVE FUNCTIONS AND BOUNDS OF THE DECISION VARIABLES FOR EACH OF THE TEST PROBLEMS ADOPTED FOR OUR EXPERIMENTAL STUDY.

was expected that the random NSGA-II would be favored by this performance measure.

Figures 3, 4, 5, 6 and 7 show the nondominated solutions obtained by the four algorithms. These plots correspond to the run in the median value with respect to the unary additive epsilon indicator. We are denoting by "newton" the seeding procedure, by "newton+nsga2" the seeded NSGA-II, by "nsga2-52" the random NSGA-II with a population size of 52 and by "nsga2-100" the random NSGA-II with a population size of 100.

We can clearly see that in problems ZDT1, ZDT2 and ZDT6, the NSGA-II is very far from the true Pareto front, whereas the seeded NSGA-II (newton+nsga2) has already converged to the true Pareto front after only 2000 fitness function evaluations. In fact, the seeding procedure was able to produce nondominated solutions on (or very close to) the real Pareto front, so the NSGA-II spread the solutions and tried to move them closer to the true Pareto front. For ZDT4, the most difficult problem in our experiments, the seeded NSGA-II is able to find better nondominated solutions (regarding convergence) than the other (standard versions of the) NSGA-II. Finally, for ZDT3, results are very similar among the different methods, while the NSGA-II with the smallest population performs slightly better.

It is important to note that 500 of the evaluations are consumed by the Latin-Hypercubes (for the regression model). Thus, the gradient-based method is consuming only around 500 evaluations. As indicated by our previous results, this initial set of nondominated solutions generated at the first stage of our approach has been found to be a good alternative to seed a MOEA. In total, our approach requires 2000 evaluations and produces results that are competitive with respect to a state-of-the-art MOEA in terms of both convergence and spread of solutions.

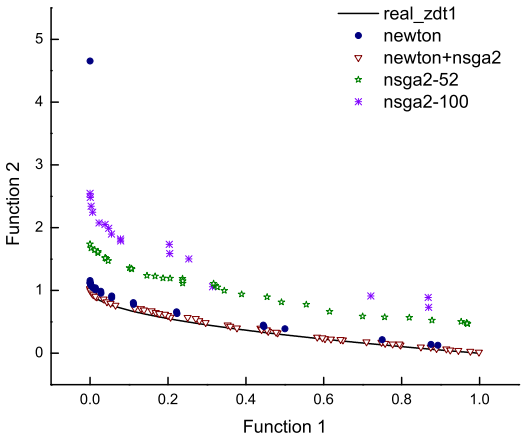

Fig. 3. Pareto fronts generated, using 2000 objective function evaluations, by the seeded NSGA-II (newton+nsga2) and the standard NSGA-II with population sizes of 52 points (nsga2-52) and 100 points (nsga2-100) for ZDT1. Newton refers to the points obtained by the seeding procedure after 1000 evaluations.

		SSC		$I^1_{\varepsilon+}$		Δ		NonDom Sol.
Function	**A**lgorithm	**M**ean	σ	**M**ean	σ	**M**ean	σ	**M**ean
ZDT1	Newton	0.8954	0.0318	0.0563	0.0290	1.0962	0.1000	34.00
ZDT1	Newton+NSGA2	**0.9203**	0.0157	**0.0233**	0.0180	**0.4571**	0.0887	53.91
ZDT1	NSGA2-52	0.8266	0.0155	0.1085	0.0144	0.7592	0.0589	40.18
ZDT1	NSGA2-100	0.7604	0.0159	0.1780	0.0153	0.8093	0.0304	41.18
ZDT2	Newton	0.8753	0.0036	0.0353	0.0109	1.2394	0.1032	53.09
ZDT2	Newton+NSGA2	**0.8870**	0.0008	**0.0104**	0.0064	**0.4074**	0.0872	58.91
ZDT2	NSGA2-52	0.2251	0.0547	0.6009	0.0897	0.9703	0.0277	8.00
ZDT2	NSGA2-100	0.6765	0.0199	0.2727	0.0225	0.9246	0.0439	9.00
ZDT3	Newton	0.5009	0.0640	0.3239	0.0594	0.9459	0.0534	32.36
ZDT3	Newton+NSGA2	0.6849	0.0257	0.1769	0.0195	0.7954	0.0376	42.36
ZDT3	NSGA2-52	**0.7318**	0.0190	**0.1361**	0.0237	0.8114	0.0447	46.18
ZDT3	NSGA2-100	0.6752	0.0123	0.1817	0.0098	**0.7848**	0.0640	47.18
ZDT4	Newton	0.7817	0.0520	0.2185	0.0518	1.0900	0.0834	15.91
ZDT4	Newton+NSGA2	**0.9562**	0.0108	**0.0448**	0.0104	0.9972	0.0664	10.20
ZDT4	NSGA2-52	0.9524	0.0118	0.0470	0.0125	0.9832	0.1146	9.55
ZDT4	NSGA2-100	0.9075	0.0190	0.0915	0.0179	**0.9291**	0.0908	10.55
ZDT6	Newton	0.4286	0.1114	0.5311	0.0898	1.1611	0.1820	46.55
ZDT6	Newton+NSGA2	**0.9215**	0.0162	**0.0291**	0.0181	1.0198	0.1470	49.36
ZDT6	NSGA2-52	0.5713	0.0168	0.3007	0.0213	**0.9441**	0.0180	8.09
ZDT6	NSGA2-100	0.4281	0.0241	0.4831	0.0284	0.9523	0.0254	9.09

TABLE II

COMPARISON OF RESULTS BETWEEN THE SEEDED NSGA-II (NEWTON+NSGA2) AND THE OTHER 3 ALTERNATIVES FOR THE FIVE TEST PROBLEMS. THE BEST VALUES ARE IN **boldface**. σ REFERS TO THE STANDARD DEVIATION OVER THE 11 RUNS PERFORMED.

VI. CONCLUSIONS AND FUTURE WORK

We have introduced a new approach that hybridizes a gradient-based method with a MOEA in order to generate solutions to multi-objective optimization problems with a low number of objective function evaluations. The main focus of our work has been to keep the number of evaluations as low as possible. For this sake, each objective function is fitted by a quadratic function in order to estimate a global improvement direction. The solutions generated are then used to seed a MOEA, so that the gradient-based information is not invoked throughout the run of the MOEA (which would be computationally expensive), which is the sort of scheme normally adopted in the specialized literature.

Our results indicate that our proposed scheme can produce a significant reduction in the computational cost of the approach, while producing results of good quality.

As part of our future work, we plan to experiment with other hybridization schemes that may allow a further reduction in the number of evaluations performed. We also plan to extend our approach to problems with more than two objectives and to add it a constraint-handling mechanism.

ACKNOWLEDGMENTS

The second author gratefully acknowledges support from CONACyT project 45683-Y. The last author acknowledges support from CONACyT to pursue graduate studies in computer science at CINVESTAV-IPN.

REFERENCES

[1] Peter A.N. Bosman and Edwin D. de Jong. Exploiting Gradient Information in Numerical Multi-Objective Evolutionary Optimization. In Hans-Georg Beyer et al., editor, *2005 Genetic and Evolutionary Computation Conference (GECCO'2005)*, volume 1, pages 755–762, New York, USA, June 2005. ACM Press.

[2] Peter A.N. Bosman and Edwin D. de Jong. Combining Gradient Techniques for Numerical Multi-Objective Evolutionary Optimization. In Maarten Keijzer et al., editor, *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, volume 1, pages 627–634, Seattle, Washington, USA, July 2006. ACM Press. ISBN 1-59593-186-4.

[3] Martin Brown and Robert E. Smith. Effective Use of Directional Information in Multi-objective Evolutionary Computation. In Erick Cantú-Paz et al., editor, *Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I*, pages 778–789. Springer. Lecture Notes in Computer Science Vol. 2723, July 2003.

[4] Carlos A. Coello Coello and Gary B. Lamont, editors. *Applications of Multi-Objective Evolutionary Algorithms*. World Scientific, Singapore, 2004. ISBN 981-256-106-4.

[5] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[6] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[7] M. Dellnitz, O. Schütze, and T. Hestermeyer. Covering Pareto sets by multilevel subdivision techniques. *Journal of Optimization Theory and Applications*, 124(11):113–136, January 2005.

[8] Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

[9] Xiaolin Hu, Zhangcan Huang, and Zhongfan Wang. Hybridization of the Multi-Objective Evolutionary Algorithms and the Gradient-based Algorithms. In *Proceedings of the 2003 Congress on Evolutionary*

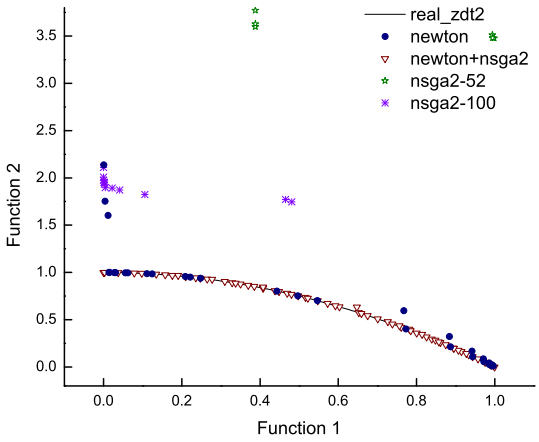

Fig. 4. Pareto fronts generated, using 2000 objective function evaluations, by the seeded NSGA-II (newton+nsga2) and the standard NSGA-II with population sizes of 52 points (nsga2-52) and 100 points (nsga2-100) for ZDT2. Newton refers to the points obtained by the seeding procedure after 1000 evaluations.

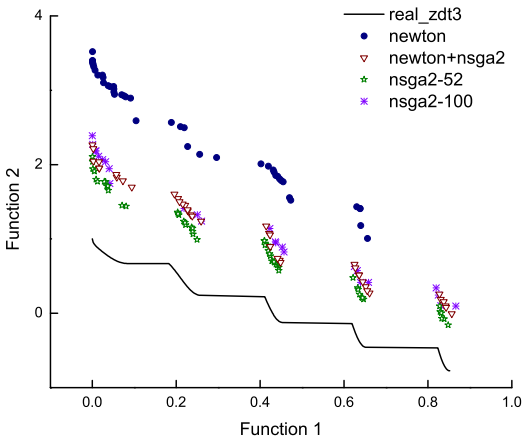

Fig. 5. Pareto fronts generated, using 2000 objective function evaluations, by the seeded NSGA-II (newton+nsga2) and the standard NSGA-II with population sizes of 52 points (nsga2-52) and 100 points (nsga2-100) for ZDT3. Newton refers to the points obtained by the seeding procedure after 1000 evaluations.

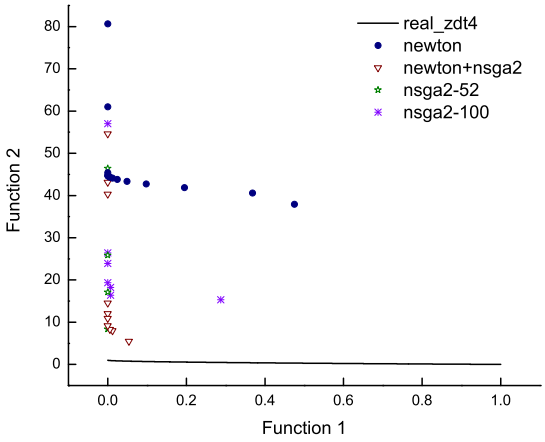

Fig. 6. Pareto fronts generated, using 2000 objective function evaluations, by the seeded NSGA-II (newton+nsga2) and the standard NSGA-II with population sizes of 52 points (nsga2-52) and 100 points (nsga2-100) for ZDT4. Newton refers to the points obtained by the seeding procedure after 1000 evaluations.

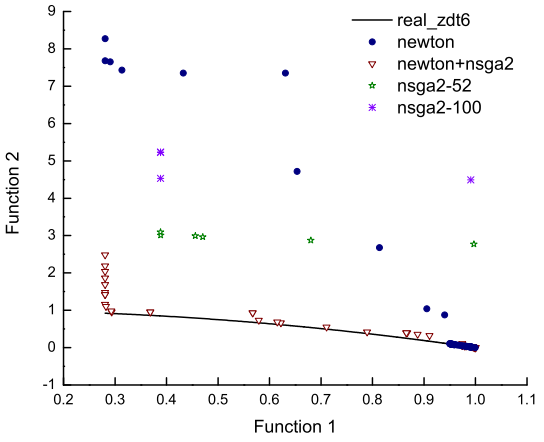

Fig. 7. Pareto fronts generated, using 2000 objective function evaluations, by the seeded NSGA-II (newton+nsga2) and the standard NSGA-II with population sizes of 52 points (nsga2-52) and 100 points (nsga2-100) for ZDT6. Newton refers to the points obtained by the seeding procedure after 1000 evaluations.

Computation (CEC'2003), volume 2, pages 870–877, Canberra, Australia, December 2003. IEEE Press.

[10] M.D. McKay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[11] Julian Molina, Manuel Laguna, Rafael Martí, and Rafael Caballero. SSPMO: A Scatter Tabu Search Procedure for Non-Linear Multiobjective Optimization. *INFORMS Journal on Computing*, 19(1):91–100, January 2007.

[12] S. Schäffler, R. Schultz, and K. Weinzierl. Stochastic method for the solution of unconstrained vector optimization problems. *Journal of Optimization Theory and Applications*, 114(1):209–222, 2002.

[13] Pradyumn Kumar Shukla. On Gradient Based Local Search Methods in Unconstrained Evolutionary Multi-objective Optimization. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007*, pages 96–110, Matshushima, Japan, March 2007. Springer. Lecture Notes in Computer Science Vol. 4403.

[14] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley, New York, 1986.

[15] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[16] Eckart Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.

[17] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.