Using Gradient-Based Information to Deal with Scalability in Multi-Objective Evolutionary Algorithms

Adriana Lara Carlos A. Coello Coello Oliver Schütze

***Abstract*—This work introduces a hybrid between an elitist multi-objective evolutionary algorithm and a gradient-based descent method, which is applied only to certain (selected) solutions. Our proposed approach requires a low number of objective function evaluations to converge to a few points in the Pareto front. Then, the rest of the Pareto front is reconstructed using a method based on rough sets theory, which also requires a low number of objective function evaluations. Emphasis is placed on the effectiveness of our proposed hybrid approach when increasing the number of decision variables, and a study of the scalability of our approach is also presented.**

I. Introduction

A multi-objective optimization problem (MOP) is stated as: minimize

$$F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T$$

where $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ for all $i \in 1, \ldots, m$ (m is the number of objective functions). Evolutionary algorithms have been found to be very effective for solving MOPs, presenting several advantages with respect to mathematical programming techniques (e.g., they are less susceptible to the shape and continuity of the Pareto front) [5].

In spite of the increasing popularity of multi-objective evolutionary algorithms (MOEAs), they tend to require a relatively large number of iterations to produce reasonably good approximations of the Pareto optimal set of a MOP. This has motivated the hybridization of MOEAs (which are known to be good global search engines) with local search engines of different types (see [16]), aiming to speed up convergence.

Gradient-based methods are a possible choice for designing local search engines, in cases in which the objective functions are differentiable. Indeed, this is an interesting choice (when possible), since the use of gradient information can give us much more precise descent directions for the search, than the only use of the operators of an evolutionary algorithm. In fact, this sort of hybrid between a MOEA and a gradient-based method has been previously studied by a number of researchers (e.g., [21], [1], [11], [22], [14]). The hybrids proposed by these authors normally replace, add or modify existing evolutionary operators, such that the gradient information is used to guide the search.

MOEAs hybridized with gradient-based methods present several advantages, from which the most remarkable one is their capability to speed up convergence. However, one advantage that has not been documented so far (to the authors' best knowledge) is their ability to scale better than traditional MOEAs in the presence of many decision variables. It has been recently found that the performance of state-of-the-art MOEAs degrades quickly as the number of decision variables increases [8]. This may limit the applicability of MOEAs in real-world problems, since their use may require a prohibitively high number of objective function evaluations. Thus, we suggest here that hybrids of MOEAs with gradient-based methods can be a suitable choice (when applicable) to deal with this scalability issue. Note that we do not refer here to scalability in objective function space which is a relatively popular research topic [3], but which we do not cover in this paper.

In order to validate our hypothesis, we will present here our own proposed hybrid of a MOEA and a gradient-based method, whose design emphasis is efficiency (measured in terms of the objective function evaluations performed), and which will be validated using scalable problems (in decision variable space).

The remainder of this paper is organized as follows. In Section II, we introduce the basic concepts on which the gradient-based descent method is based. In Section III, we introduce a two-stage algorithm named *Gradient-Based Multi-objective Evolutionary Strategy* (GBMES), which is our proposed hybrid of a MOEA and a gradient-based method. The results obtained with our proposed hybrid approach and a state-of-the-art MOEA (NSGA-II [7]) in two scalable test problems are presented in Section IV. Finally, our conclusions and some possible paths for further work are stated in Section V.

II. Gradient-Based Descent

During the search of solutions that minimize a continuous and differentiable function, the gradient of the function provides information about its growth or decrement. It is well known that the maximum decrement of the function f is obtained when following the direction $-\nabla f(x)$, departing from the point x.

In the multi-objective case, and along the same line, we want to find a movement direction in $\mathbb{R}^n$ that brings simultaneously the maximum decrement —or at least a decrement— in all the f_i components of F. Hence, we will call *descent direction* a unit vector $\hat{u} \in \mathbb{R}^n$ that intuitively represents a decrement in all the f_i, or in most of them, and keeps the same value in the others. For our purposes we found suitable to use the descent direction proposed by

The authors are with the Departamento de Computación, CINVESTAV-IPN, Av. IPN No. 2508, Col. San Pedro Zacatenco, México, D.F. 07360, MEXICO (email: ccoello@cs.cinvestav.mx). The second author is also with the UMI-LAFMIA 3175 CNRS.

Fliege in [9]. The basic details of this method are explained next after providing some basic definitions.

Using the canonical coordinatewise order $\leq$ in $\mathbb{R}^n$ (this is, $a \leq b$ if $a_i \leq b_i$ for all $i \in \{i, .., n\}$), a point $x \in \mathbb{R}^n$ *dominates* $y \in \mathbb{R}^n$, denoted as $x \prec y$, if $F(x) \leq F(y)$ with $x \neq y$. If $y \not\prec x$ and $x \not\prec y$, we say that x and y are *not comparable* (not dominated) denoted by $x||y$.

One point $x \in A \subseteq \mathbb{R}^n$ is a *Pareto optimum* if there does not exist another point $z \in A \subseteq \mathbb{R}^n$ such that $z \prec x$.

Let $JF(x) = \frac{\delta f_i}{\delta x_j}(x)$ be the Jacobian matrix of F and $\mathbb{R}^-$ the set of negative real numbers. Then, a necessary condition [9] for the point $x \in \mathbb{R}^n$ to be a local Pareto point is that

$$range(JF(x)) \cap (\mathbb{R}^-)^m = \emptyset \tag{1}$$

holds. The above expression means that if we have a point x which is not dominated by any other point in a certain neighborhood, in a Pareto sense, then it is not possible that a direction v exists, for which the directional derivative of each f_i could be negative —which would turn it into a descent direction. Note that the condition (1) can be fulfilled by Pareto optimal points and by *critical points* as well.

Setting the above in terms of our purposes, we will assume that a solution x should be improved during the solution of a real-valued minimization MOP. Then, if x is not a critical point, it is possible to choose a descent direction v holding

$$JF(x)v \in (\mathbb{R}^-)^m$$

If this is the case, v can be computed [9] using the information provided by the Jacobian matrix of the problem evaluated on x. As Fliege presents in [9], it is necessary to solve the next quadratic programming problem such that we:

$$\text{minimize} \quad \alpha + \frac{1}{2}||v||^2 \tag{2}$$

$$\text{subject to} \quad (JF(x)v)_i \leq \alpha, \quad \text{for all} \quad i \in \{1, \ldots, m\}$$

Once the above problem produces a solution (v^*, α^*), the descent direction that we are looking for is v^*, and as Fliege suggests, the step length for the movement can be obtained by an Armijo's rule, by decreasing t until the condition

$$F(x + tv) \leq F(x) + \beta t JF(x)v$$

is fulfilled. The value $\beta \in (0, 1)$ is a control parameter to decide how fine grained, numerically speaking, the descent will be. At this point of the process, having $x^1 = x + tv$, we are in condition of repeating the movement by calculating a new descent direction for x^1 or, if this is not possible, we can assume that a critical point has been achieved.

This method, constructed by Fliege, automatically triggers a condition to know if x^1 fulfills condition (1), which is the case when $\alpha^* = 0$. In practice, it is necessary to set a tolerance parameter τ, $\tau < 0$ to stop the descent when $\tau \leq \alpha^*$ holds; note that by construction $\alpha^* \leq 0$.

Several ways to calculate descent directions have been proposed [19], [1], [2], [4], [12]. A study of the efficiency of each of them for the method proposed here is left as future work. Even though following descent directions leads directly to good candidates for local Pareto points, performing a gradient-based descent could be very expensive in terms of function calls. For example, according to the results, regarding the Automatic Differentiation method, reported in [10], we are counting in our experiments one Jacobian matrix computation as the equivalent to five times one function-call effort. Then, if we have a three-objective MOP, and perform twenty times the descent step for just a single point, we need at least $5 \times 3 \times 20 = \mathbf{300}$ function calls —besides the necessary calls to calculate the step size length. That is why in practice, and for the sake of implementing this algorithm, we bound the number of descent steps spent for each point. This situation will be explained in the next section.

III. The Proposed Algorithm

Our proposed *Gradient-Based Multi-objective Evolutionary Strategy* (GBMES) is conformed by two stages. The aim of the first stage is getting a small set of Pareto optimal points. In the case of facing moderate multi-frontality, the evolutionary part of the hybrid algorithm can deal with the critical points that are not optimum; then, we assume that, by the end of this first stage, all the points in this set are part of the global front. The second stage looks for the reconstruction of the entire front, starting from a few points lying on it.

A. First Stage

For the first stage of the algorithm we use several populations, in a way analogous to the Micro-GA for Multiobjective Optimization [6]. We adopt an external population P with a replaceable part Pr and a non-replaceable part Pn. The former population will evolve over time and the latter will introduce diversity into the process. We use a small population Pt of parents, $|Pt| = \mu$ randomly chosen from $P = Pr \cup Pn$. The individuals from Pt are recombined to produce a set P_{off} of λ descendants. Unlike a traditional evolution strategy, we set $\lambda \approx \mu$ because we need to bound the number of function calls in this phase —in order to spend most of them in the gradient-based descent part. The individuals used for recombination are randomly chosen from Pt. We use two types of recombination, arithmetic and discrete, choosing one of them in a random way, with a certain (predefined) probability. We set a higher probability for the discrete recombination at this stage (the proportion is 2:1). For the second stage of GBMES, arithmetic recombination plays a more important role.

Once we have selected the nondominated parents P_{nd} from $Pt \cup P_{off}$, we perform the next insertion process in order to obtain the secondary population S and the elite population E, $E \subseteq S$. This insertion process has two variants.
In the initial generation $S = \emptyset$, and for each $p \in P_{nd}$ we:

1) Perform the gradient-based descent for p and get p' as the final point of the descent.

2) Insert the element: $S \leftarrow S \cup \{p'\}$. If the necessary condition for being in E (see below) holds then $E \leftarrow E \cup \{p'\}$.

The next generations the process is slightly different, as we describe next:

1) Set P'_{nd} as the elements from P_{nd} that are not dominated by any $s \in E$.
2) For each $p \in P'_{nd}$:
   - Perform the gradient-based descent for p and get p' as the final point of the descent.
   - Remove $s \in S$ such that $p' \prec s$.
   - Insert p' in S (in E if applicable).
   - Remove $q \in P'_{nd}$ such that $p' \prec q$.

In both cases we use the set of inserted points p' to replace the dominated individuals from P_r.

The secondary population is directly conformed by the points obtained after the descent. The condition for a point $x \in S$ to be in E is having finished the descent with an α^* (see Section II) value of zero —which means it is a candidate to be a local Pareto point. Note that the latter could not be the case for all the points in S. Thus, this separation is made because the descent could be stopped before obtaining a local Pareto point. This happens because, in practice, we restrict the number of steps used in the descent. Additionally, we could also stop the descent if a certain tolerance value between the initial and the final points in the movement is achieved. Also, in order to handle box constraints, when the computed steepest descent direction leads outside the feasible region, the process must quit exploiting that solution, and then take the point nearest to the boundary, between the feasible and infeasible regions, as the final point of the movement. This is necessary, since it is possible that the movement of the descent points leads to a local optimum located in the infeasible region or to a local optimum located in the boundary between the feasible and the infeasible region. In both cases, it is unnecessary to perform a fine-grained steepest descent, and the cost of doing it could lead to an important increase in the number of function calls. Then, during the procedure these end points from the descent would enter the elite population only if the elite population is empty. The algorithm's implementation should detect the case when, after certain number of generations, no point has entered the elite population. In this case, we perform uniform mutation on the individuals in the secondary population and we only retain the nondominated solutions as candidates for becoming the outcome of the first-stage. In order to feed the replaceable population, we take P''_{nd}, which is conformed by individuals from P'_{nd} that entered the secondary population at that generation, and we use them to replace those individuals from Pr that are dominated. To select the μ parents to conform Pt for the next generation, we take $min\{mu/3, |P''_{nd}|\}$ individuals from P''_{nd} and the remainder are randomly selected from P.

The general process of the first stage is described next:

1) Initialize the replaceable P_r and non-replaceable P_{nr} parts of the population.
2) Select μ parents for P_t.
3) Recombine the parents to produce λ offspring Off.
4) Select P_{nd} nondominated individuals from the union of parents and offspring.
5) Perform the insertion process with the gradient-based descent, and feed the secondary population S, the elite population E, and the replaceable population P_r.
6) Repeat steps 2 to 5 until having at least four individuals in the elite population or until running out of the function-calls budget allowed for this first stage.

For all the problems tested here, the values for μ and λ were 20, the initial population size was of 130 individuals, and the proportion between the non-replaceable and the replaceable parts of the initial population was 3:10. Regarding the value of μ, in general, it must be a trade-off because it has to be big enough in order to allow a good sampling of the population for the evolution strategy but not too large, in order to avoid that too many function calls are spent.

B. Second Stage

Once the Pareto front has been approached, the next step consists of applying a technique that performs a reconstruction of the entire front departing from a few points from such front. One possibility at this point is to use continuation methods [20], [15]; if this is the case, only one point in the front is necessary to start, but we would have to be able to afford the computational cost of calculating the second derivatives of the functions, which is high, even if clever techniques are devised [13].

In this work, we propose using Rough Sets [17], [18] for the second stage, and we reserve at least one third of our function-calls budget for this part. This approach (rough sets) consists of a stochastic technique which uses information about individuals that were dominated in a previous iteration, in order to construct new solutions close to the nondominated individuals and far away from the dominated ones. This aims to generate new nondominated solutions and, as a consequence, fill up the missing parts of the Pareto front. Next, we briefly describe the approach that we use.

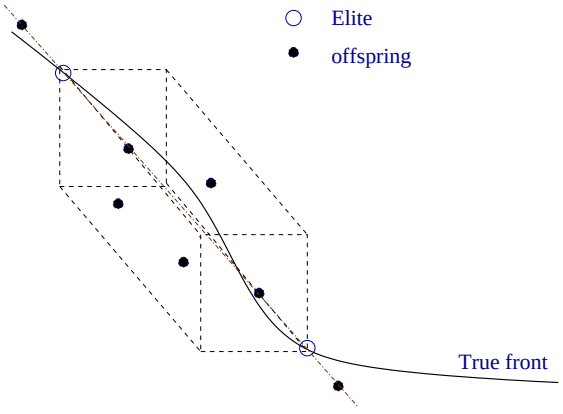

Fig. 1. This figure shows the box formed by two elements in the elite set. Their offspring lie on the line that joins them, and in the neighborhood bounded by their own coordinates (n-dimensional box).

1) Preliminary Phase: Our rough sets approach requires an initial population ($P1$) which is close to the true Pareto front. This population is partitioned into two sets: DS, which contains the dominated solutions, and NS which contains the nondominated solutions.

At the end of the first stage of the GBMES, we assume that we have a few (at least four) individuals in the true Pareto front. Then, to start the rough sets procedure, we can generate a small population in the neighborhood of these solutions in the following way: For each point in the elite population, we generate an $n-$dimensional box with each point as one vertex and its nearest neighbor located in the opposite corner. Then, we generate two types of offspring: first, we apply total arithmetic recombination to get s descendants in the line that joins the reference point and its nearest neighbor and we produce another offspring outside the box, but in the same direction as before (see Figure 1). Next, the second kind of offspring consists of t descendants which are randomly built inside the box. For the examples presented here, we used $s = 2$ and $t = 3$.

2) Rough Sets: Once the population is close enough to the true Pareto front, and once is partitioned into the NS and DS sets, we perform a set of iterations until reaching the maximum allowable by our function-calls budget. At each iteration, we build a grid with k dominated points from DS, which serve as vertices. We also take q individuals from the set of nondominated solutions NS and we apply bounded mutation to them. The bounds for these mutations are set from half of the distance over each coordinate to the limits of the grid (See Figure 2). If there is no edge bounding the mutation operator, we set the limit for that specific coordinate as the natural limits of the box constraints from the problem. The procedure is the following:

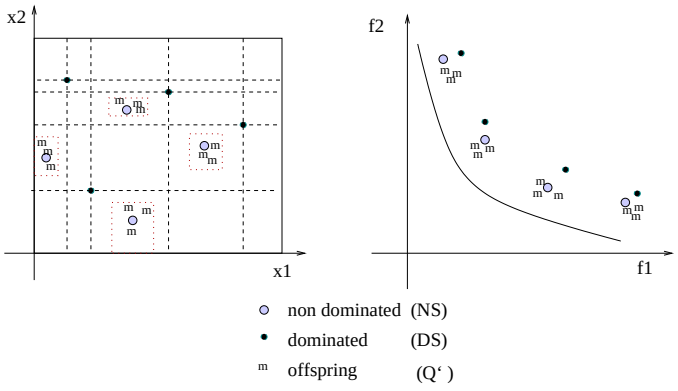

Fig. 2. This figure shows the grid for the rough sets method formed with the elements of DS serving as vertices (marked as black dots in the figure). The nondominated individuals (taken from NS), which are used as the reference solutions, are marked with gray circles. Their descendents, produced by mutation, are marked with the letter 'm'.

1) Divide the population $P1$ into the DS and NS sets.
2) Randomly choose k elements from DS to set the limits of the grid.
3) Randomly choose q elements from NS to form Q (the set of parents).
4) Apply u mutations to each parent and conform Q' with them.
5) Divide the population $P1 \cup Q'$ into the new DS and NS sets.
6) If $v < |NS|$, we use a crowding or a clustering technique to reduce the size to v. v is a user-defined parameter.
7) If $v' < |DS|$, we keep the solutions that were non-dominated in the last iteration, and we choose the rest randomly from the dominated solutions in the population, until reaching the maximum size allowable for v'. v' is a user-defined parameter.
8) Repeat steps 2 to 7 until a certain (predefined) number of function-calls is performed.

For the examples we present here, we used $k = 20$, $q = 10$, $u = 4$, $v = 100$ for the bi-objective problem, $v = 150$ for the three-objective problem, and $v' = 100$ in both cases.

IV. Experimental Results

The performance measures used here are described next. In the following, δ_i denotes the minimum Euclidean distance from the image $F(x_i) \in PF_{known}$ of a solution $x_i, i = 1, \ldots, q = |PF_{known}|$, to the true Pareto front PF_{true}; d_{ij} is the Euclidean distance between $F(x_i)$ and $F(x_j)$. Finally, we define

$$d_i := \min_{\substack{j=1,\ldots,q \\ i \neq j}} d_{ij}; \quad \overline{d} := \frac{1}{q}\sum_{i=1}^{q} d_i.$$

For $u \in R^n, A, B \subseteq R^n$

$$dist(u, A) := \inf_{v \in A} ||u - v||$$

and

$$dist(B, A) := \sup_{u \in B} dist(u, A).$$

The performance measures are:

- Generational Distance (**GD**, **IGD**)

$$GD = \frac{1}{q}\sqrt{\sum_{i=1}^{q} \delta_i^2}$$

  The Inverted Generational Distance (**IGD**) is analogous to GD but measured from PF_{true} to PF_{known}.
- Spacing (**S**)

$$S = \sqrt{\frac{1}{q-1}\sum_{i=1}^{q}(d_i - \overline{d})^2}$$

- Hausdorff's distance

$$d_H := \max\{dist(PF_{true}, PF_{known}), dist(PF_{known}, PF_{true})\}$$

Since we want to test the scalability of our proposed hybrid approach, we adopted the two problems defined in Table I, which have two and three objectives, respectively. They are scalable in decision variable space. The final population for both algorithms was set to 100 individuals for the problem with two objectives and to 150 individuals for the problem

TABLE I
MOPS ADOPTED FOR OUR EXPERIMENTS.

Problem 1

$f_1(x) = (x_1 - 1)^4 + \sum_{i=2}^{n}(x_i - 1)^2$

$f_2(x) = \sum_{i=1}^{n}(x_i + 1)^2$

with $n = 10$

Problem 2 (DTLZ2)

$f_1(x) = \cos(\frac{x_1\pi}{2})\cos(\frac{x_2\pi}{2})\ldots\cos(\frac{x_{k-1}\pi}{2})(1 + g(x))$

$f_2(x) = \cos(\frac{x_1\pi}{2})\cos(\frac{x_2\pi}{2})\ldots\sin(\frac{x_{k-1}\pi}{2})(1 + g(x))$

$\vdots$

$f_{k-1}(x) = \cos(\frac{x_1\pi}{2})\sin(\frac{x_2\pi}{2})(1 + g(x))$

$f_k(x) = \sin(\frac{x_1\pi}{2})(1 + g(x))$

$g(x) = \sum_{i=k}^{n}(x_i - \frac{1}{2})^2$

$0 \leq x_i \leq 1,\ i = 1, \ldots, n$

with $k = 3$, $n = 12$

with three objectives (a larger value was adopted in this case because of the higher number of objectives). Our results are compared with respect to those generated by the NSGA-II [7] using 100 and 150 individuals, respectively (same as our approach), and performing the same number of evaluations as our proposed hybrid approach. Tables II and III show our comparison of results after performing 3,000 objective function evaluations for both problems. In this case, we use $n = 10$ for Problem 1, and $n = 12$ for Problem 2. From Tables II and III, we can see that our GBMES achieves much better convergence than the NSGA-II. This can be corroborated by looking at Figure 3, in which it is clear that the NSGA-II is unable to converge, even when Problem 1 only has ten decision variables.

Then, we increased the number of decision variables of the problem and we focused our analysis on the convergence of each approach (ours and the NSGA-II). Figures 3, 4, 5 and 6 show the plots of the final population, corresponding to the run in the mean obtained for IGD over 30 runs, for Problem 1, using $n \in \{10, 30, 60, 100\}$ decision variables. We can observe that, as we increase the number of decision variables, our proposed GBMES is still able to generate an important portion of the Pareto front (e.g., it is able to generate the "knee" in all cases) with the same number of evaluations as before (3,000). The values of the performance measures (shown in Tables IV, V and VI) indicate that both approaches suffer a performance degradation as we increase the number of decision variables. This behavior is consistent in the case of Problem 2, as well. Although the performance of our proposed GBMES degrades as we increase the number of decision variables, such degradation is less significant than the one suffered by the NSGA-II. Also, in all cases (for both problems), our approach outperforms the NSGA-II with respect to all the performance measures used to assess convergence (see Tables IV, V and VI). This can be better appreciated in Figures 7 to 9 for Problem 1 and from Figures 10 to 12 for Problem 2, in which we show a graphical comparison of the performance (regarding convergence) of the two approaches (ours and the NSGA-II) as we increase the number of decision variables.

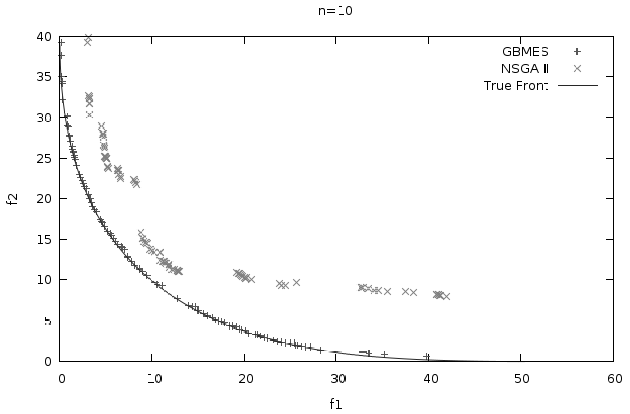

Fig. 3. This graph shows the Pareto fronts generated by our GBMES and NSGA-II for Problem 1 (with $n = 10$), after 3,000 function evaluations.

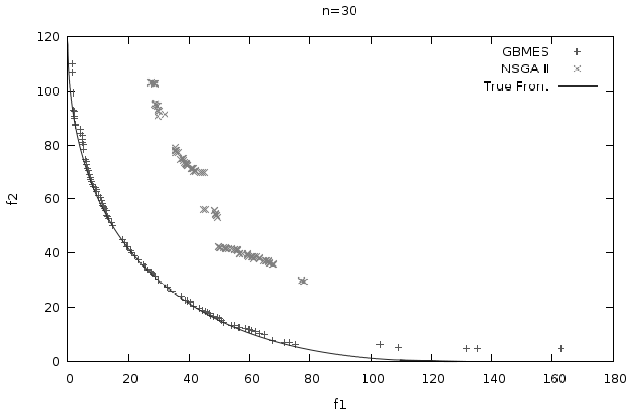

Fig. 4. This graph shows the Pareto fronts generated by our GBMES and NSGA-II for Problem 1 (with $n = 30$), after 3,000 function evaluations.

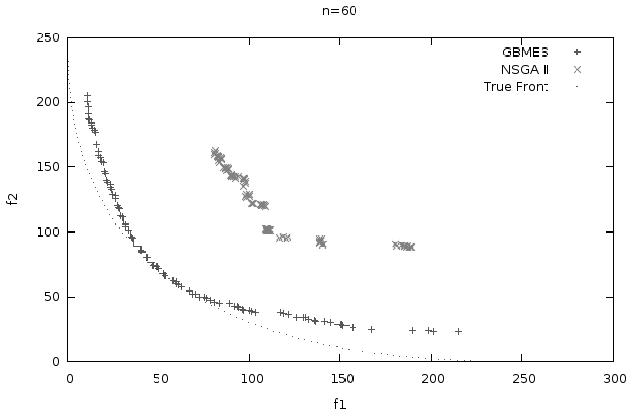

Fig. 5. This graph shows the Pareto fronts generated by our GBMES and NSGA-II for Problem 1 (with $n = 60$), after 3,000 function evaluations.

TABLE II
COMPARISON OF RESULTS FOR PROBLEM 1, USING $n = 10$, AND PERFORMING 3000 FUNCTION EVALUATIONS. STATISTICS WERE GATHERED FROM 30 INDEPENDENT RUNS. THE BEST RESULTS ARE SHOWN IN **boldface**.

GD			
Method	Best	Mean	Worst
GBMES	**0.023375**	**0.049261**	**0.103776**
NSGA II	0.242919	0.415592	0.539319
IGD			
Method	Best	Mean	Worst
GBMES	**0.092534**	**0.222116**	**0.549907**
NSGA II	0.367473	0.565376	0.863660
Spacing			
Method	Best	Mean	Worst
GBMES	0.004457	0.029559	0.081939
NSGA II	**0.002789**	**0.022826**	**0.070167**
Hausdorff's distance			
Method	Best	Mean	Worst
GBMES	**3.493272**	**9.823990**	**21.668491**
NSGA II	7.022420	15.473644	25.665400

TABLE III
COMPARISON OF RESULTS FOR PROBLEM 2, USING $n = 12$, AND PERFORMING 3000 FUNCTION EVALUATIONS. STATISTICS WERE GATHERED FROM 30 INDEPENDENT RUNS. THE BEST RESULTS ARE SHOWN IN **boldface**.

GD			
Method	Best	Mean	Worst
GBMES	**0.003234**	**0.019648**	**0.032110**
NSGA II	0.027392	0.035683	0.043271
IGD			
Method	Best	Mean	Worst
GBMES	**0.000421**	**0.000643**	**0.001017**
NSGA II	0.001422	0.001746	0.002200
Spacing			
Method	Best	Mean	Worst
GBMES	**0.000737**	**0.004314**	**0.009814**
NSGA II	0.001360	0.005380	0.032598
Hausdorff's distance			
Method	Best	Mean	Worst
GBMES	**0.385372**	0.842908	1.335189
NSGA II	0.556158	**0.798111**	**1.170840**

V. CONCLUSIONS AND FUTURE WORK

We have introduced a hybrid approach called GBMES, which is designed to take advantage of gradient-based information extracted for multi-objective optimization problems. Since MOEAs are known to perform well in problems with high multi-frontality, our method focuses instead on problems in which the gradient descent is a good option to speed up the first stages of the search. Although obtaining gradient information is an expensive process (because it requires several objective function evaluations), it is possible to design a gradient-based hybrid which is very efficient. For this sake, it is important to devise a careful interleaving between the MOEA and the gradient-based search engine, so that we do not exceed a modest function-calls budget. Such balance is achieved by our proposed GBMES, which only performs a total of 3,000 objective function evaluations for the problems included here.

An interesting aspect of our proposed approach (which we believe that is shared by other hybrids between MOEAs and gradient-based methods[1]) is that it scales well as we increase the number of decision variables of a MOP. This is illustrated in the paper by two examples in which we use up to 100 decision variables. Our proposed approach is found to degrade significantly less than a state-of-the-art MOEA (the NSGA-II), while still performing 3,000 objective function evaluations.

TABLE IV
COMPARISON OF RESULTS REGARDING GD FOR BOTH PROBLEMS WHEN USING n =*original*,30, 60, 100, AND AFTER PERFORMING 3,000 FUNCTION EVALUATIONS. THE VALUE CORRESPONDS TO THE MEAN OVER 30 INDEPENDENT RUNS. THE *original* VALUES ADOPTED FOR n IN EACH PROBLEM ARE THOSE DEFINED IN TABLE I. THE BEST RESULTS ARE SHOWN IN **boldface**.

GD	Problem 1		Problem 2	
n	GBMES	NSGAII	GBMES	NSGAII
original	**0.0493**	0.4156	**0.019648**	0.035683
30	**0.3063**	2.6406	**0.032565**	0.131245
60	**0.8925**	6.8851	**0.015376**	0.308811
100	**1.7495**	12.9756	**0.029423**	0.567924

TABLE V
COMPARISON OF RESULTS REGARDING IGD FOR BOTH PROBLEMS WHEN USING n =*original*,30, 60, 100, AND AFTER PERFORMING 3,000 FUNCTION EVALUATIONS. THE VALUE CORRESPONDS TO THE MEAN OVER 30 INDEPENDENT RUNS. THE *original* VALUES ADOPTED FOR n IN EACH PROBLEM ARE THOSE DEFINED IN TABLE I. THE BEST RESULTS ARE SHOWN IN **boldface**.

IGD	Problem 1		Problem 2	
n	GBMES	NSGAII	GBMES	NSGAII
original	**0.2221**	0.5654	**0.000643**	0.001746
30	**0.7449**	3.0245	**0.000829**	0.006448
60	**1.6436**	7.6595	**0.001116**	0.016010
100	**2.8417**	14.6620	**0.001249**	0.029858

As part of our future work, we are interested in extending our hybrid approach in several ways. For example, for problems with many critical (non Pareto-optimum) points, our approach cannot distinguish between such points and those which are Pareto optima. Thus, the gradient-based information is not very effective in this case, and the MOEA should be used in order to deal with this situation (e.g., using a higher mutation rate to avoid getting stuck). Otherwise, the hybrid turns out to be too expensive (computationally speaking). Thus, a more careful algorithmic design is required to deal with this sort of situation.

Additionally, the rough sets mechanism can be improved by introducing gradient-based information into it. However, this should be done very carefully, because of the high computational cost associated with obtaining this information. For this sake, it is possible to take advantage of the construction explained in Section III-B1. Such construction could be selectively repeated and mixed with a gradient-based descent applicable only to a few selected individuals.

[1]This depends on the sort of method adopted to approximate the derivates of the functions.

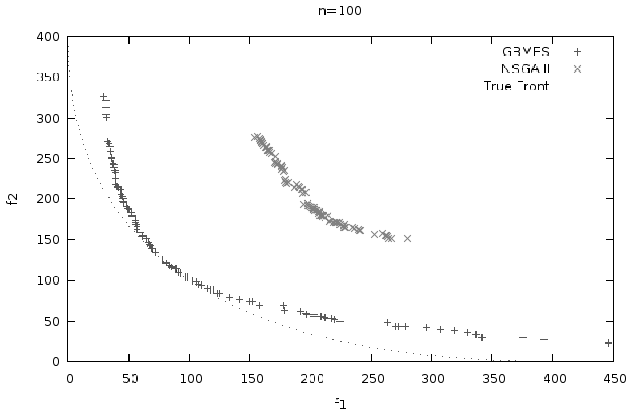

Fig. 6. This graph shows the Pareto fronts generated by our GBMES and NSGA-II for Problem 1 (with $n = 100$), after 3,000 function evaluations.

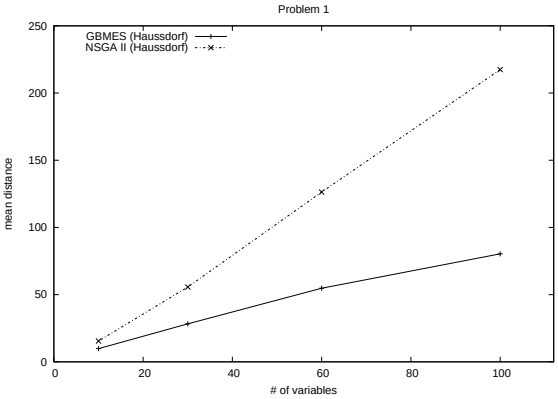

Fig. 9. Graphical illustration of the performance of our GBMES and the NSGA-II in Problem 1, regarding Hausdorff's distance, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

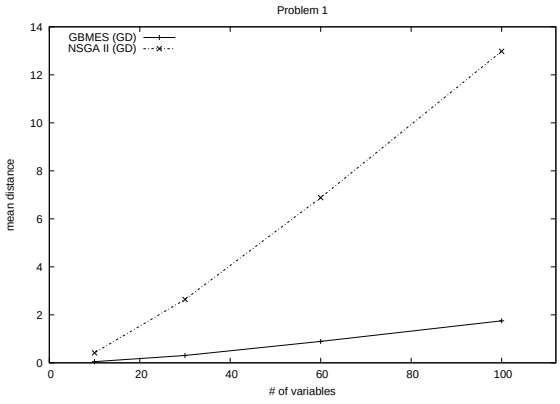

Fig. 7. Graphical illustration of the performance of our GBMES and the NSGA-II in Problem 1, regarding the GD performance measure, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

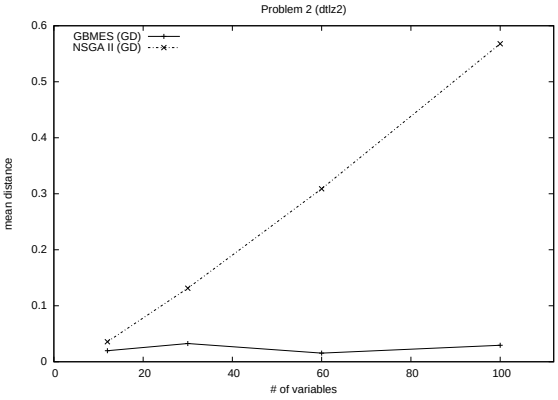

Fig. 10. Graphical illustration of the performance of our GBMES and the NSGA-II in Problem 2, regarding the GD performance measure, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

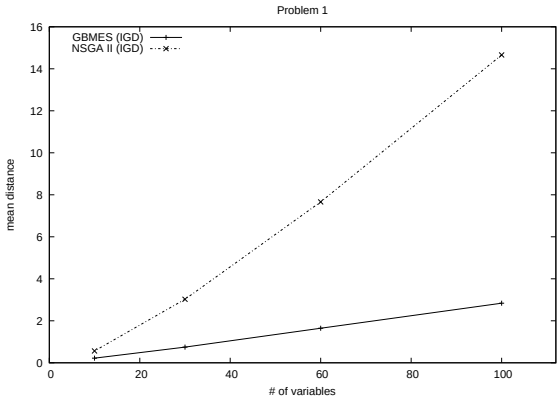

Fig. 8. Graphical illustration of the performance of our GBMES and the NSGA-II in Problem 1, regarding the IGD performance measure, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

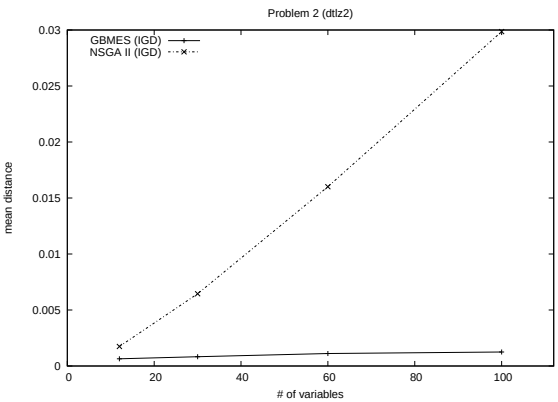

Fig. 11. Graphical illustration of the performance of our GBMES and the NSGA-II in Problem 2, regarding the IGD performance measure, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

TABLE VI

COMPARISON OF RESULTS REGARDING HAUSDORFF'S DISTANCE FOR BOTH PROBLEMS WHEN USING n =*original*,30, 60, 100, AND AFTER PERFORMING 3,000 FUNCTION EVALUATIONS. THE VALUE CORRESPONDS TO THE MEAN OVER 30 INDEPENDENT RUNS. THE *original* VALUES ADOPTED FOR n IN EACH PROBLEM ARE THOSE DEFINED IN TABLE I. THE BEST RESULTS ARE SHOWN IN **boldface**.

Haus. dist	Problem 1		Problem 2	
n	GBMES	NSGA-II	GBMES	NSGA-II
original	**9.8240**	15.4736	0.842908	**0.798111**
30	**28.3035**	55.6760	**1.625643**	2.354311
60	**54.7381**	126.3136	**1.164723**	4.914662
100	**80.3792**	217.4507	**2.065171**	8.331932

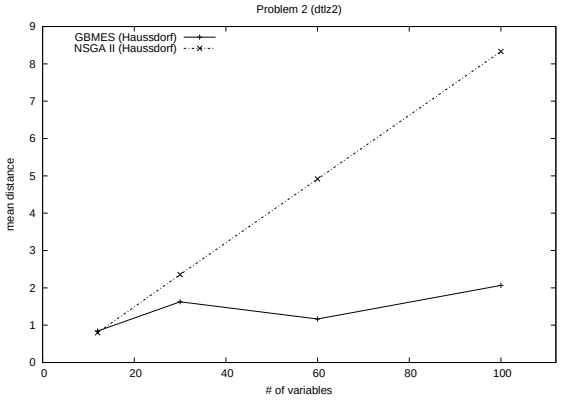

Fig. 12. Graphical illustration of the performance of our GBMES and the NSGA-II in Problem 2, regarding Hausdorff's distance, as we increase the number of decision variables. In all cases, we performed 3,000 function evaluations.

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] Dimo Brockhoff, Tobias Friedrich, Nils Hebbinghaus, Christian Klein, Frank Neumann, and Eckart Zitzler. Do Additional Objectives Make a Problem Harder? In Dirk Thierens, editor, *2007 Genetic and Evolutionary Computation Conference (GECCO'2007)*, volume 1, pages 765–772, London, UK, July 2007. ACM Press.

[4] Martin Brown and R.E. Smith. Effective use of directional information in multi-objective evolutionary computation. In *Genetic and Evolutionary Computation GECCO 2003*, volume Volume 2723/2003 of *Lecture Notes in Computer Science*, pages 778–789. Springer Berlin / Heidelberg, 2003.

[5] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic Algorithms and Evolutionary Computation. Springer, 2nd edition, Sept 2007.

[6] Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[7] 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.

[8] Juan J. Durillo, Antonio J. Nebro, Carlos A. Coello Coello, Francisco Luna, and Enrique Alba. A Comparative Study of the Effect of Parameter Scalability in Multi-Objective Metaheuristics. In *2008 Congress on Evolutionary Computation (CEC'2008)*, pages 1893–1900, Hong Kong, June 2008. IEEE Service Center.

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

[10] Andreas Griewank. On Automatic Differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108. Kluwer Academic Publishers, 1989.

[11] Ken Harada, Kokolo Ikeda, and Shigenobu Kobayashi. Hybridizing of Genetic Algorithm and Local Search in Multiobjective Function Optimization: Recommendation of GA then LS. In Maarten Keijzer et al., editor, *2006 Genetic and Evolutionary Computation Conference (GECCO'2006)*, volume 1, pages 667–674, Seattle, Washington, USA, July 2006. ACM Press. ISBN 1-59593-186-4.

[12] Ken Harada, Jun Sakuma, and Shigenobu Kobayashi. Local search for multiobjective function optimization: pareto descent method. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 659–666, New York, NY, USA, 2006. ACM Press.

[13] Ken Harada, Jun Sakuma, and Shigenobu Kobayashi. Uniform Sampling of Local Pareto-Optimal Solution Curves by Pareto Path Following and its Applications in Multi-objective GA. In Dirk Thierens, editor, *2007 Genetic and Evolutionary Computation Conference (GECCO'2007)*, volume 1, pages 813–820, London, UK, July 2007. ACM Press.

[14] Alfredo G. Hernandez-Diaz, Carlos A. Coello Coello, Luis V. Santana-Quintero, Fatima Perez, Julian Molina, and Rafael Caballero. On the use of Projected Gradients for Constrained Multiobjective Optimization Problems. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature–PPSN X*, pages 712–721. Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Alemania, September 2008.

[15] Claus Hillermeier. *Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach*, volume 135 of *International Series of Numerical Mathematics*. Birkhäuser, 2001.

[16] Joshua Knowles and David Corne. Memetic Algorithms for Multiobjective Optimization: Issues, Methods and Prospects. In William E. Hart, N. Krasnogor, and J.E. Smith, editors, *Recent Advances in Memetic Algorithms*, pages 313–352. Springer. Studies in Fuzziness and Soft Computing, Vol. 166, 2005.

[17] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(1):341–356, Summer 1982.

[18] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991. ISBN 0-471-87339-X.

[19] S. Schäffler, R. Schultz, and K. Weinzierl. A stochastic method for the solution of unconstrained vector optimization problems. *Journal of Optimization Theory and Applications*, 2002.

[20] O. Schütze, A. Dell'Aere, and M. Dellnitz. On continuation methods for the numerical treatment of multi-objective optimization problems. In Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Ralph E. Steuer, editors, *Practical Approaches to Multi-Objective Optimization*, number 04461 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005. <http://drops.dagstuhl.de/opus/volltexte/2005/349>.

[21] 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.

[22] Karthik Sindhya, Kalyanmoy Deb, and Kaisa Miettinen. A Local Search Based Evolutionary Multi-objective Optimization Approach for Fast and Accurate Convergence. In Günter Rudolph, Thomas Jansen, Simon Lucas, Carlo Poloni, and Nicola Beume, editors, *Parallel Problem Solving from Nature–PPSN X*, pages 815–824. Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Alemania, September 2008.