StrucOpt manuscript No.
(will be inserted by the editor)

Multiobjective Structural Optimization using a Micro-Genetic Algorithm

Carlos A. Coello Coello and Gregorio Toscano Pulido

Abstract In this paper, we present a genetic algorithm with a very small population and a reinitialization process (a micro genetic algorithm) for solving multiobjective optimization problems. Our approach uses three forms of elitism, including an external memory (or secondary population) to keep the nondominated solutions found along the evolutionary process. We validate our proposal using several engineering optimization problems taken from the specialized literature, and we compare our results with respect to two other algorithms (the NSGA-II and PAES) using three different metrics. Our results indicate that our approach is very efficient (computationally speaking) and that performs very well in problems with different degrees of complexity.

Key words evolutionary multiobjective optimization, genetic algorithms, multiobjective optimization, vector optimization

Received: date / Revised version: date

Carlos A. Coello Coello and Gregorio Toscano Pulido
CINVESTAV-IPN
Evolutionary Computation Group
Depto. de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México, D. F. 07360
e-mail: ccoello@cs.cinvestav.mx
gtoscano@computacion.cs.cinvestav.mx
Send offprint requests to: Carlos A. Coello Coello

1 Introduction

The idea of using techniques based on the emulation of the mechanism of natural selection to solve problems can be traced as long back as the 1930s (Cannon 1932). However, it was not until the 1960s that the three main techniques based on this notion were developed: genetic algorithms (Holland 1962), evolution strategies (Schwefel 1965) and evolutionary programming (Fogel 1966). These approaches, which are now collectively denominated "evolutionary algorithms" (EAs), have been very effective for single-objective optimization (Goldberg 1989*a*; Schwefel 1981; Fogel 1999). Despite this considerably large volume of research, new areas remain to be explored with sufficient depth. One of them is the use of

evolutionary algorithms to solve multiobjective optimization problems.

The first implementation of a multi-objective evolutionary algorithm (MOEA) dates back to the mid-1980s (Schaffer 1984, 1985). Since then, a considerable amount of research has been done in this area, now known as evolutionary multiobjective optimization (EMO for short) (Coello Coello et al. 2002). Evolutionary algorithms seem particularly desirable for solving multiobjective optimization problems because they deal simultaneously with a set of possible solutions (the so-called population) which allows us to find several members of the Pareto optimal set in a single run of the algorithm, instead of having to perform a series of separate runs as in the case of the traditional mathematical programming techniques. Additionally, evolutionary algorithms are less susceptible to the shape or continuity of the Pareto front (e.g., they can easily deal with discontinuous and concave Pareto fronts), whereas these two issues are a real concern for mathematical programming techniques (Coello Coello 1999).

Despite the important volume of research on evolutionary multiobjective optimization in the last few years[1] (see for example (Coello Coello et al. 2002; Fonseca and Fleming 1995; Van Veldhuizen and Lamont 2000)), until recently, little emphasis had been placed on stressing efficiency when designing multi-objective evolutionary algorithms (MOEAs). This paper deals precisely with this issue, since we propose the use of a genetic algorithm with a very small population size and a reinitialization process (a micro genetic algorithm) to solve (quite efficiently) multiobjective optimization problems. Our approach is validated using several engineering optimization problems taken from the specialized literature and its performance is compared against two techniques that are representative of the state-of-the-art in EMO: the Nondominated Sorting Genetic Algorithm II (NSGA II) (Deb et al. 2000, 2002) and the Pareto Archived Evolution Strategy (PAES) (Knowles and Corne 2000). Our results indicate that our approach is a viable alternative for efficient multiobjective optimization.

[1] The first author maintains the EMOO repository which currently contains over 1900 bibliographic references on evolutionary multiobjective optimization. The EMOO repository is located at: `http://delta.cs.cinvestav.mx/~ccoello/EMOO`

2 Basic Concepts

We are interested in the general nonlinear programming problem in which we want to:

$$\text{Find } \mathbf{x} \text{ which optimizes } f(\mathbf{x}) \tag{1}$$

subject to:

$$g_i(\mathbf{x}) \leq 0, \;\; i = 1, \ldots, n \tag{2}$$

$$h_j(\mathbf{x}) = 0, \;\; j = 1, \ldots, p \tag{3}$$

where $\mathbf{x}$ is the vector of solutions $\mathbf{x} = [x_1, x_2, \ldots, x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or non-linear).

If we denote with $\mathcal{F}$ to the feasible region and with $\mathcal{S}$ to the whole search space, then it should be clear that $\mathcal{F} \subseteq \mathcal{S}$.

For an inequality constaint that satisfies $g_i(\mathbf{x}) = 0$, then we will say that is active at $\mathbf{x}$. All equality constraints h_j (regardless of the value of $\mathbf{x}$ used) are considered active at all points of $\mathcal{F}$.

Now, we will define some basic concepts from multiobjective optimization.

Definition 1 (General Multiobjective Optimization Problem): *Find the vector* $\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ *which will satisfy the* m *inequality constraints:*

$$g_i(\mathbf{x}) \leq 0 \quad i = 1, 2, \ldots, m \tag{4}$$

the p *equality constraints*

$$h_i(\mathbf{x}) = 0 \quad i = 1, 2, \ldots, p \tag{5}$$

and will optimize the vector function

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T \tag{6}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ *is the vector of decision variables.* □

Having several objective functions, the notion of "optimum" changes, because in multiobjective optimization problems, the aim is to find good compromises (or "trade-offs") rather than a single solution as in global optimization. The notion of "optimum" that is most commonly adopted is that originally proposed by Edgeworth (1881) and later generalized by Pareto (1896). This notion is normally referred to as "Pareto optimality" and is defined next.

Definition 2 (Pareto Optimality:): *A point* $\mathbf{x}^* \in \mathcal{F}$ *is* **Pareto optimal** *if for every* $\mathbf{x} \in \mathcal{F}$ *and* $I = \{1, 2, \ldots, k\}$,

$$\forall_{i \in I}(f_i(\mathbf{x}^*) \leq f_i(\mathbf{x})) \tag{7}$$

and there is at least one $i \in I$ *such that*

$$f_i(\mathbf{x}^*) < f_i(\mathbf{x}) \tag{8}$$

□

In words, this definition says that $\mathbf{x}^*$ is Pareto optimal if there exists no feasible vector $\mathbf{x}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. The phrase "Pareto optimal" is considered to mean with respect to the entire decision variable space unless otherwise specified.

Other important definitions associated with Pareto optimality are the following:

Definition 3 (Pareto Dominance): *A vector* $\mathbf{u} = (u_1, \ldots, u_k)$ *is said to dominate* $\mathbf{v} = (v_1, \ldots, v_k)$ *(denoted by* $\mathbf{u} \preceq \mathbf{v}$*) if and only if u is partially less than v, i.e.,* $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$. □

Definition 4 (Pareto Optimal Set): *For a given multiobjective optimization problem,* $\mathbf{f}(x)$*, the Pareto optimal set (*$\mathcal{P}^*$*)*

is defined as:

$$\mathcal{P}^* := \{x \in \mathcal{F} \mid \neg\exists\ x' \in \mathcal{F}\ \ \mathbf{f}(x') \preceq \mathbf{f}(x)\}. \quad (9)$$

□

Definition 5 (Pareto Front:): *For a given multiobjective optimization problem* $\mathbf{f}(x)$ *and Pareto optimal set* $\mathcal{P}^*$*, the Pareto front* ($\mathcal{PF}^*$) *is defined as:*

$$\mathcal{PF}^* := \{\mathbf{u} = \mathbf{f} = (f_1(x), \ldots, f_k(x)) \mid x \in \mathcal{P}^*\}. \quad (10)$$

□

3 Previous Work

The term micro-genetic algorithm (microGA) refers to a small-population genetic algorithm with reinitialization. The approach was derived from some theoretical results obtained by Goldberg (1989*b*), according to which a population size of three is sufficient to converge, regardless of the chromosomic length adopted. The process suggested by Goldberg was to start with a small randomly generated population, then apply to it the genetic operators until reaching *nominal convergence* (e.g., when all the individuals have their genotypes either identical or very similar), and then to generate a new population by transferring the best individuals of the converged population to the new one. The remaining individuals would be randomly generated.

The first to report an implementation of a microGA was Krishnakumar (1989), who used a population size of five, a crossover rate of one and a mutation rate of zero. His approach also adopted an elitist strategy that copied the best string found in the current population to the next generation. Selection was performed by holding four competitions between strings that were adjacent in the population array, and declaring to the individual with the highest fitness as the winner. Krishnakumar (1989) compared his microGA against a simple GA (with a population size of 50, a crossover rate of 0.6 and a mutation rate of 0.001). He reported faster and better results with his microGA on two stationary functions and a real-world engineering control problem (a wind-shear controller task). After him, several other researchers have developed applications of microGAs (e.g., (Johnson and Abushagur 1995; Xiao and Yabe 1998)). Despite the existence of a considerable amount of research on the use of genetic algorithms for multiobjective optimization (see for example (Coello Coello et al. 2002; Deb 2001; Coello Coello 1999)), the work reported in this paper represents, to the best of our knowledge, the first attempt to use a microGA (i.e., a genetic algorithm with a very small population size and a reinitialization process) for multiobjective optimization.

Related ideas have, however, been proposed in the literature. For example, Jaszkiewicz (2002) proposed the multiple-objective genetic local search (MOGLS) algorithm which uses for a short time a small population initialized from a large external memory. This approach, however, dues its good performance to the use of local search and a lot of computer memory. It could also be argued that the multi-membered versions of PAES (Knowles and Corne 2000) can be seen as a form of

microGA. However, the authors of PAES concluded that the addition of a population did not, in general, improve the performance of their approach, and instead increased the computational overhead in an important way (Knowles and Corne 2000).

Our approach differs from these two proposals because it does not make an excessive use of memory, it does not use special local search mechanisms and it does not behave as an evolution strategy (which is the evolutionary algorithm adopted in PAES).

4 Description of Our Approach

The way in which our technique works is the following. First, we generate a set of random solutions and we place them in what we call the "population memory". This memory is divided in two parts, one replaceable and another one non-replaceable (see Figure 1).

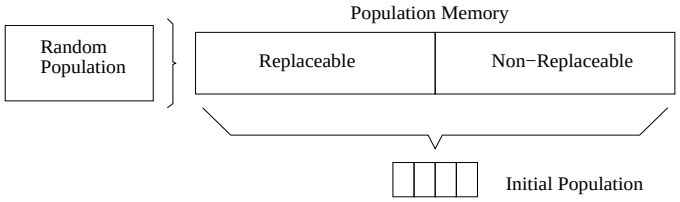

Fig. 1 Population memory of our microGA.

As indicated by its name, the *non-replaceable* part of the population memory will never change during the execution of the algorithm, and its main goal is to maintain diversity. In contrast, the *replaceable* part of the population memory will change after each microGA cycle as indicated below.

The microGA will obtain its small working population from these two portions of the population memory to evolve it during a few generations (the use of a certain (fixed) number of generations is the criterion that we adopt to define *nominal convergence* in this case). The microGA works the same as a simple genetic algorithm that uses a selection scheme that favors nondominated solutions (the so-called Pareto-ranking scheme). The microGA uses tournament selection, two-point crossover and uniform mutation. Upon reaching nominal convergence (i.e., after performing a certain number of iterations (defined *a priori* by the user), the microGA retains only two nondominated solutions.[2] Then, these two solutions are compared with respect to two contenders taken from the replaceable memory. If any of these two solutions dominates its adversary, then it replaces it in the replaceable memory.

Our microGA also uses an external archive that keeps all the nondominated solutions found along the evolutionary process. The main goal of this external archive is to ensure that the nondominated solutions found by our microGA are retained through generations. Otherwise, the use of genetic operators (i.e., crossover and mutation) could destroy such nondominated solutions.

The same two solutions indicated before (i.e., those compared with two adversaries from the replaceable memory) are also compared with respect to the contents of an external population. In case these two solutions are not dominated by the contents of the external population, they are both stored in

[2] This is of course, in case there is more than one nondominated solution.

such external archive. Note that any of these solutions that turns out to be dominated by any individual stored in the external archive is discarded. Also, if any of these solutions dominates any solution stored in the external archive, then such a solution is deleted from the archive.

After storing our nondominated solutions found in the external archive, we start another cycle of the microGA all over again. For that sake, our microGA again takes (randomly) solutions from both portions of the population memory to conform its initial population. This process is repeated until a stop condition is reached.

The pseudo-code of our approach is the following:

function MicroGA
begin
Generate starting population P of size N
and store its contents in the population memory M
/* Both portions of M will be filled with
random solutions */
i=0
while i $<$ Max **do**
begin
Get the initial population for
the microGA (P) from M
repeat
begin
Apply binary tournament selection
based on nondominance
Apply two-point crossover
and uniform mutation
to the selected individuals
Apply elitism (retain only one
nondominated vector per generation)
Produce the next generation
end
until nominal convergence is reached
Copy two nondominated vectors from P
to the external memory E (first form of elitism)
if E is full when trying to insert $individual$
then adaptive_grid($individual$)
Copy two nondominated vectors from P to M
(second form of elitism)
if i **mod** replacement_cycle
then apply third form of elitism
(Move points from E to the replaceable memory)
i = i+1
end while
end function

As indicated in the previous pseudo-code, our approach uses three forms of elitism:

1. The first form of elitism is based on the notion that if we store in the external archive the nondominated vectors produced from each cycle of the microGA, we will not lose any valuable information obtained from the evolutionary process.
2. The second form of elitism is based on the idea that if we replace the (replaceable portion of the) population memory by the nominal solutions (i.e., the best solutions found

when nominal convergence is reached), we will gradually converge, since crossover and mutation will have a higher probability of reaching the true Pareto front of the problem over time. This notion was hinted by Goldberg (1989*b*).

3. The third type of elitism is applied at certain intervals (defined by a parameter called "replacement cycle"). We take a certain number of points from all the regions of the Pareto front generated so far and we use them to fill the replaceable memory. Depending on the size of the replaceable memory, we choose as many points from the Pareto front as necessary to guarantee a uniform distribution. This process allows us to use the best solutions generated so far as the starting point for the microGA, so that we can improve them (either by getting closer to the true Pareto front or by getting a better distribution).

To keep diversity in the Pareto front, we use an approach similar to the adaptive grid proposed by Knowles and Corne (2000) (see Figure 2). Once the archive that stores nondominated solutions has reached its limit, we divide the objective search space that this archive covers, assigning a set of coordinates to each solution. Then, each newly generated nondominated solution will be accepted only if the geographical location to where the individual belongs has fewer individuals than the most crowded location of the grid. Alternatively, the new nondominated solution could also be accepted if the individual belongs to a location outside the previously specified boundaries.

The adaptive grid requires two parameters: the expected size of the Pareto front and the number of positions in which we will divide the solution space for each objective. The first parameter is defined by the size of the external memory. We have found that our approach is not very sensitive to the second parameter (e.g., in our experiments a value between 15 and 25 provided very similar results). The process of determining the location of a certain individual has a low computational cost (it is based on the values of its objectives as indicated before). However, when the individual is out of range, we have to relocate all the positions. Nevertheless, this last situation does not occur too often, and we allocate a certain amount of extra room in the first and last locations of the grid to minimize its occurrence.

4.1 Handling Constraints

Our microGA incorporates a very simple constraint-handling scheme, which has effect on its selection mechanism, the adaptive grid and the population memory.

In the selection mechanism, constraints are handled when checking Pareto dominance. When we compare two individuals, we first check their feasibility. If one is feasible and the other is infeasible, the feasible individual wins. If both are infeasible, then the individual with the lowest amount of (total) constraint violation wins. If they both have the same amount of constraint violation (or if they are both feasible), then the comparison is done using Pareto dominance. Note that we avoided the use of penalty functions (Richardson et al. 1989),

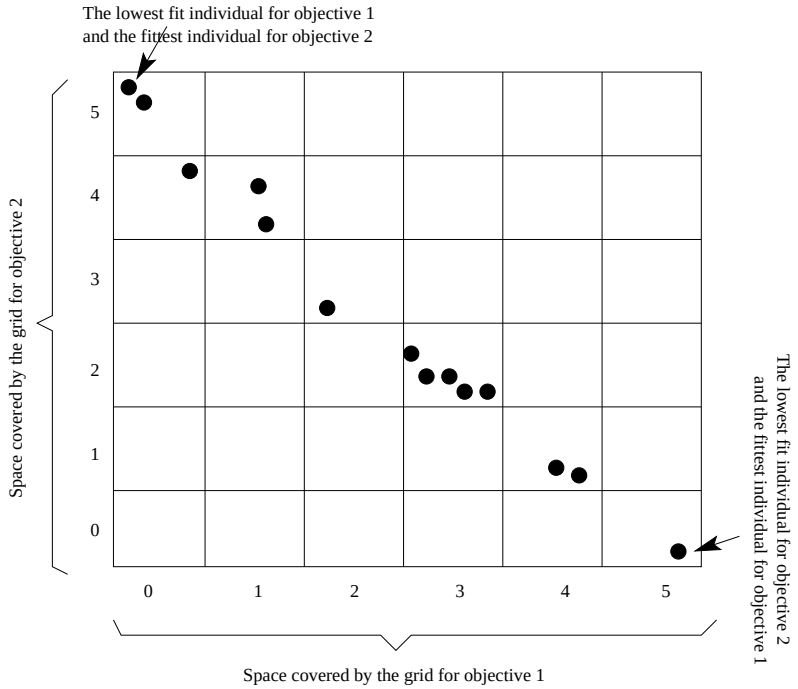

Fig. 2 The adaptive grid used to handle the external memory of the microGA.

because it is well-known that the performance of such type of approach is highly dependent on the type of penalty function and penalty factors adopted, which are normally defined in an *ad-hoc* manner for each specific problem to be solved (Coello Coello 2002; Michalewicz and Schoenauer 1996).

We never store an infeasible individual in the external population, because it is not a valid solution. However, it is possible to store an infeasible individual within the population memory (if it dominates its competitor). This is done in order to allow the possibility of evolving it as to reach the feasible region of the problem.

5 Comparison of Results

Several test functions were taken from the specialized literature to compare our approach. In order to allow a quantitative assessment of the performance of an MOEA, three issues are normally taken into consideration (Zitzler et al. 2000):

1. Minimize the distance of the Pareto front produced by our algorithm with respect to the global Pareto front (assuming we know its location).
2. Maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible.
3. Maximize the number of elements of the Pareto optimal set found.

Based on this notion, we adopted one metric to evaluate each of the three aspects previously indicated:

1. **Error Ratio** (ER): This metric was proposed by Van Veldhuizen (1999) to indicate the percentage of solutions (from

the nondominated vectors found so far) that are not members of the true Pareto optimal set:

$$ER = \frac{\sum_{i=1}^{n} e_i}{n}, \tag{11}$$

where n is the number of vectors in the current set of nondominated vectors available; e_i = 0 if vector i is a member of the Pareto optimal set, and $e_i = 1$ otherwise. It should then be clear that $ER = 0$ indicates an ideal behavior, since it would mean that all the vectors generated by our algorithm belong to the Pareto optimal set of the problem. This metric addresses the third issue from the list previously provided.

2. **Generational Distance** (GD): The concept of generational distance was introduced by Van Veldhuizen and Lamont (1998) as a way of estimating how far are the elements in the set of nondominated vectors found so far from those in the Pareto optimal set and is defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{12}$$

where n is the number of vectors in the set of nondominated solutions found so far and d_i is the Euclidean distance (measured in objective space) between each of these and the nearest member of the Pareto optimal set. It should be clear that a value of $GD = 0$ indicates that all the elements generated are in the Pareto optimal set. Therefore, any other value will indicate how "far" we are from the global Pareto front of our problem. This metric addresses the first issue from the list previously provided.

3. **Spacing** (SP): Here, one desires to measure the spread (distribution) of vectors throughout the nondominated vectors found so far. Since the "beginning" and "end" of the current Pareto front found are known, a suitably defined metric judges how well the solutions in such front are distributed. Schott (1995) proposed such a metric measuring the range (distance) variance of neighboring vectors in the nondominated vectors found so far. This metric is defined as:

$$S \triangleq \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\overline{d} - d_i)^2}\ , \tag{13}$$

where $d_i = \min_j (\mid f_1^i(\mathbf{x}) - f_1^j(\mathbf{x}) \mid + \mid f_2^i(\mathbf{x}) - f_2^j(\mathbf{x}) \mid)$, $i, j = 1, \ldots, n$, $\overline{d}$ is the mean of all d_i, and n is the number of nondominated vectors found so far. A value of zero for this metric indicates all members of the Pareto front currently available are equidistantly spaced. This metric addresses the second issue from the list previously provided.

Additionally, CPU times were also evaluated (using the same hardware platform and the exact same environment for each of the algorithms) in order to establish if our microGA was really faster than the other techniques as we hypothesized, since that was one of its design goals.

In order to know how competitive was our approach, we decided to compare it against two multiobjective evolutionary algorithms that are representative of the state-of-the-art in the area:

1. **Nondominated Sorting Genetic Algorithm II**: Proposed by Deb et al. (2000, 2002), this algorithm is a revised version of the Nondominated Sorting Genetic Algorithm proposed by Srinivas and Deb (1994). The original NSGA is based on several layers of classifications of the individuals as suggested by Goldberg (1989*a*). Before selection is performed, the population is ranked on the basis of nondomination: all nondominated individuals are classified into one category (with a dummy fitness value, which is proportional to the population size, to provide an equal reproductive potential for these individuals). Then this group of classified individuals is ignored and another layer of nondominated individuals is considered. The process continues until all individuals in the population are classified. Since individuals in the first front have the maximum fitness value, they always get more copies than the rest of the population. This allows to search for nondominated regions, and results in convergence of the population toward such regions. The NSGA-II is more efficient (computationally speaking) than the original NSGA, uses elitism and a crowded comparison operator that keeps diversity without specifying any additional parameters (the original NSGA used fitness sharing). This algorithm uses $(\mu + \lambda)$-selection as its elitist mechanism.
2. **Pareto Archived Evolution Strategy**: This algorithm was introduced by Knowles and Corne (2000). PAES consists of a (1+1) evolution strategy (i.e., a single parent that generates a single offspring) in combination with a historical archive that records some of the nondominated solutions previously found. This archive is used as a reference set against which each mutated individual is being compared. Such a historical archive is the elitist mechanism adopted in PAES. However, an interesting aspect of this algorithm is the procedure used to maintain diversity which consists of a crowding procedure that divides objective space in a recursive manner. Each solution is placed in a certain grid location based on the values of its objectives (which are used as its "coordinates" or "geographical location"). A map of such grid is maintained, indicating the number of solutions that reside in each grid location. Since the procedure is adaptive, no extra parameters are required (except for the number of divisions of the objective space).

In the following examples, the NSGA-II was run using a population size of 100, a crossover rate of 0.8 (uniform crossover was adopted), tournament selection, and a mutation rate of $1/L$, where L = chromosome length (binary representation was adopted). The microGA used a crossover rate of 0.8, an external memory of 100 individuals, a number of iterations to achieve nominal convergence of two, a population memory of 50 individuals, a percentage of non-replaceable memory of 0.3, a population size (for the microGA itself) of four individuals, and 25 subdivisions of the adaptive grid. The mutation rate was set to $1/L$ ($L =$ length of the chromosomic string). PAES was run using an adaptive grid with a depth of five, a size of the archive of 100, and a mutation rate of $1/L$, where L refers to the length of the chromosomic string that encodes the decision variables.

To avoid any bias or misinterpretation when implementing each of the two other approaches used for our comparative study, we adopted the public-domain versions of PAES and the NSGA-II, which are available at `http://delta.cs.cinvestav.mx/~ccoello/EMOO/EMOOsoftware.html`. It is worth indicating that our microGA is also available in the public-domain at the same URL previously indicated. These three algorithms (i.e., PAES, the NSGA-II and our microGA) were run on the same hardware and software platform.

Note that in all the following examples, we generated the true (or global) Pareto front by enumeration, using parallel processing techniques (some of these fronts require a considerably high computational effort to be generated by enumeration). This is necessary to compute some of the metrics previously defined and to allow a quantitative assessment of results.

For constrained test functions, our microGA and PAES (which doesn't have an explicit mechanism to handle constraints) use the mechanism described in Section 4.1. The NSGA-II has its own constraint-handling mechanism (Deb et al. 2002), so we did not have to implement one for it.

6

Example 1

Our first test function was proposed by Kursawe (1991):

$$\text{Minimize } f_1(\mathbf{x}) = \sum_{i=1}^{n-1} \left(-10 \exp\left(-0.2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right)$$

ER	microGA	NSGA-II	PAES
Best	0.18	0.06	0.1
Worst	0.36	1.01	1.01
Average	0.2655	0.56	0.8145
Median	0.245	0.495	0.975
Std. Dev.	0.05394685	0.38451610	0.27633837

Table 1 Results of the Error Ratio metric for the first test function.

GD	microGA	NSGA-II	PAES
Best	0.00680344	0.00690487	0.0146701
Worst	0.0103437	0.103095	0.157191
Average	0.008456311	0.029255159	0.054914365
Median	0.008489235	0.01735665	0.0493581
Std. Dev.	0.00098659	0.02716975	0.03074372

Table 2 Results of the Generational Distance metric for the first test function.

(14)

$$\text{Minimize } f_2(\mathbf{x}) = \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5 \sin(x_i)^3 \right) \tag{15}$$

where:

$$-5 \leq x_1, x_2, x_3 \leq 5 \tag{16}$$

In this example, the total number of fitness function evaluations was set to 12000.

Figures 3, 4 and 5 show the graphical results produced by the microGA, the NSGA-II and PAES, respectively. The true Pareto front of the problem is shown as a continuous

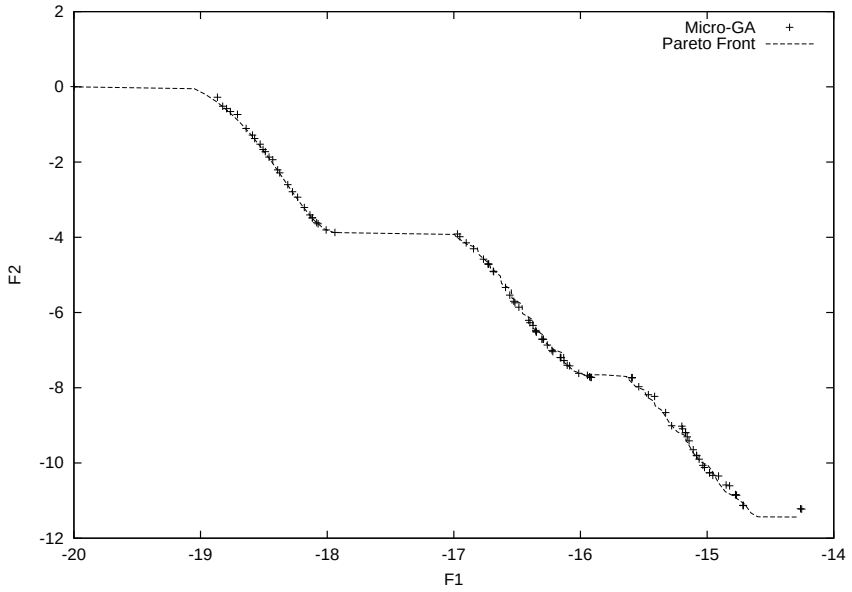

Fig. 3 Pareto fronts produced by the microGA for the first test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric.

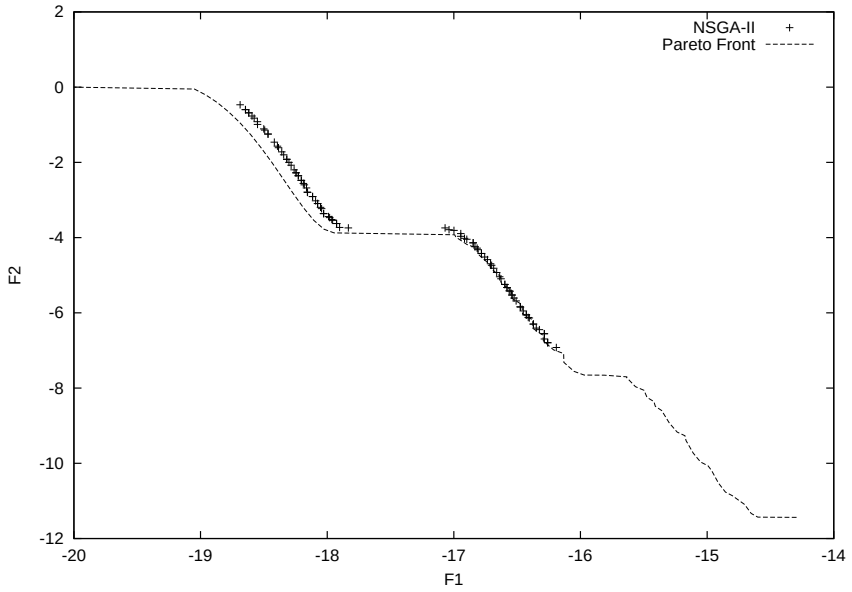

Fig. 4 Pareto fronts produced by the NSGA-II for the first test function. The true Pareto front is shown as a continuous line.

line. Tables 1, 2, 3 and 4 show the comparison of results among the three algorithms considering the metrics previously described. The microGA obtained the best average values with respect to generational distance and error ratio, and was outperformed by the NSGA-II with respect to spacing. Note however in Figures 3, 4 and 5 how the microGA covers the largest segment of the true Pareto front of the problem. Both the NSGA-II and PAES missed several segments of the true Pareto front. Therefore, irrespective of the values of the metric, it should be clear that the microGA had the best over-

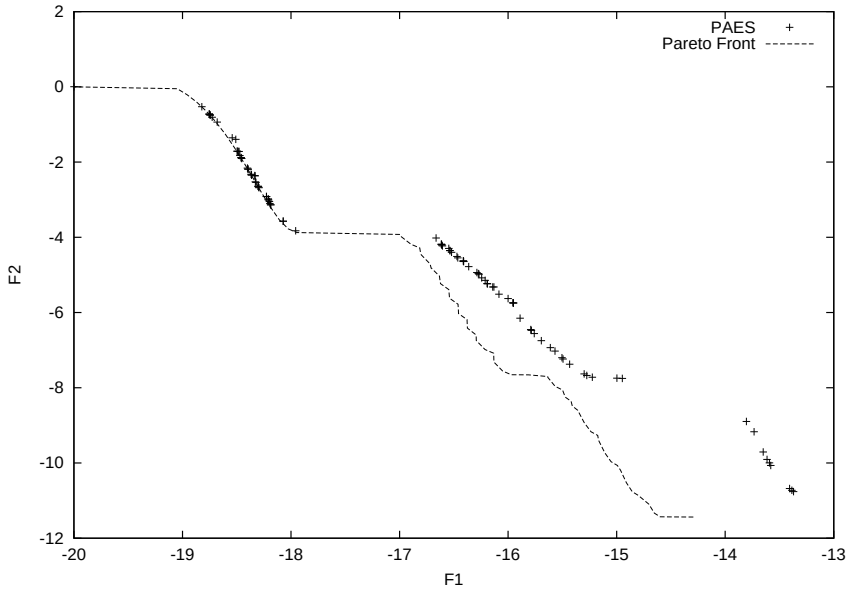

Fig. 5 Pareto fronts produced by PAES for the first test function. The true Pareto front is shown as a continuous line.

SP	microGA	NSGA-II	PAES
Best	0.0716859	0.0184178	0.0641144
Worst	0.203127	0.0657118	0.340955
Average	0.12889499	0.036135605	0.1975222
Median	0.126655	0.03608505	0.186632
Std. Dev.	0.02993154	0.01097740	0.07301957

Table 3 Results of the Spacing metric for the first test function.

Time	microGA	NSGA-II	PAES
Best	0.295	2.181	0.938
Worst	0.345	2.693	1.39
Average	0.32695	2.42435	1.12615
Median	0.3325	2.454	1.121
Std. Dev.	0.01481277	0.16680441	0.10522420

Table 4 Computational time (in seconds) required by each algorithm for the first test function.

all performance in the first test function. Also note that in this case the microGA was (on average) three times faster than PAES and eight times faster than the NSGA-II.

7

Example 2

Our second test function is to optimize the four-bar plane truss shown in Figure 6. The problem is the following (Cheng and Li 1999):

Minimize

$$f_1(\mathbf{x}) = L(2x_1 + \sqrt{2x_2} + \sqrt{x_3} + x_4) \tag{17}$$

$$f_2(\mathbf{x}) = \frac{FL}{E}\left(\frac{2}{x_2} + \frac{2\sqrt{2}}{x_2} - \frac{2\sqrt{(2)}}{x_3} + \frac{2}{x_4}\right) \tag{18}$$

such that

$$(F/\sigma) \leq x_1 \leq 3 \times (F/\sigma)$$

$$\sqrt{2}(F/\sigma) \leq x_2 \leq 3 \times (F/\sigma)$$

$$\sqrt{2}(F/\sigma) \leq x_3 \leq 3 \times (F/\sigma)$$

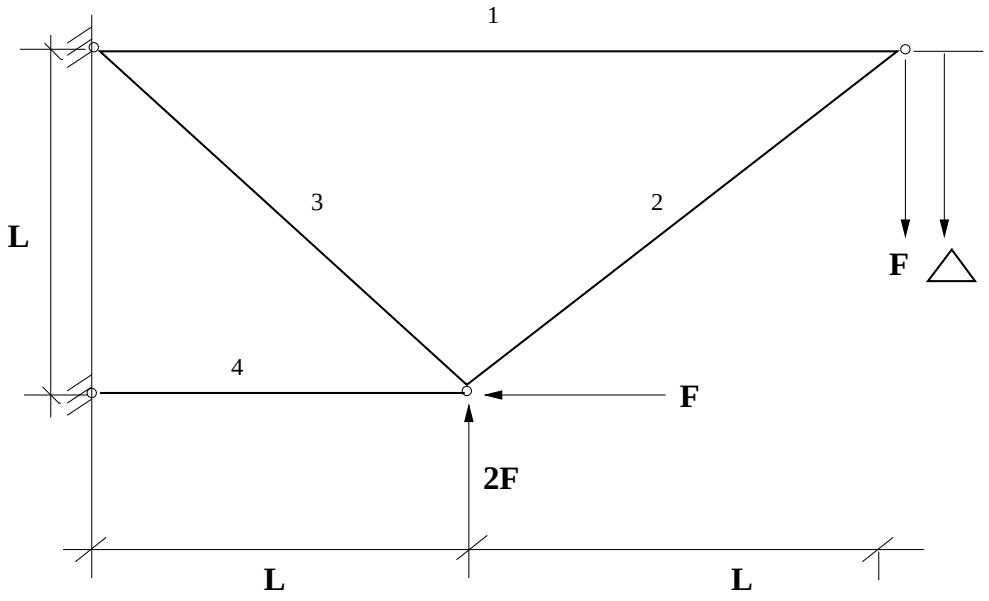

Fig. 6 Plane truss used for the second example. The structural volume and the joint displacement (Δ) are to be optimized.

ER	microGA	NSGA-II	PAES
Best	0.85	0.99	1.01
Worst	1.01031	1.01	1.01
Average	0.980464	0.9995	1.01
Median	0.995	1	1.01
Std. Dev.	0.04009190	0.00944513	0

Table 5 Results of the Error Ratio metric for the second test function.

GD	microGA	NSGA-II	PAES
Best	0.0580338	7.93903	3.93581
Worst	0.0918631	11.4831	14.2827
Average	0.075033695	9.8997555	9.5249935
Median	0.0757498	9.877115	9.417545
Std. Dev.	0.00828227	1.14986135	2.717377174

Table 6 Results of the Generational Distance metric for the second test function.

$(F/\sigma) \leq x_4 \leq 3 \times (F/\sigma)$

where:

$F = 10kN$, $E = (2)10^5 kN/cm^2$, $L = 200cm$ $\sigma = 10kN/cm^3$

In this example, the total number of fitness function evaluations was set to 12000.

SP	microGA	NSGA-II	PAES
Best	5.32187	6.41401	1.71862
Worst	8.81188	8.46951	47.932
Average	6.8609385	7.3116655	17.434317
Median	6.77402	7.239515	17.25515
Std. Dev.	1.03065513	0.53557575	14.98498216

Table 7 Results of the Spacing metric for the second test function.

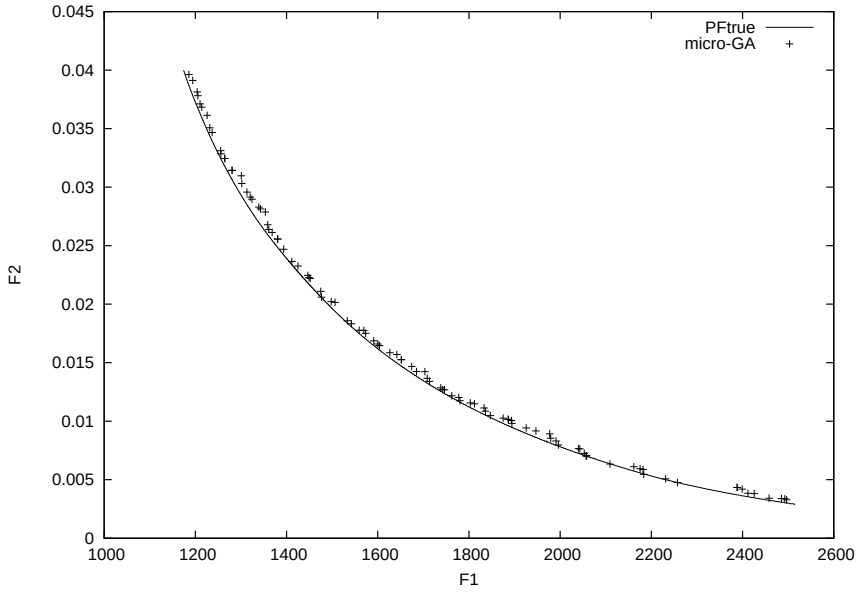

Fig. 7 Pareto fronts produced by the microGA for the second test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

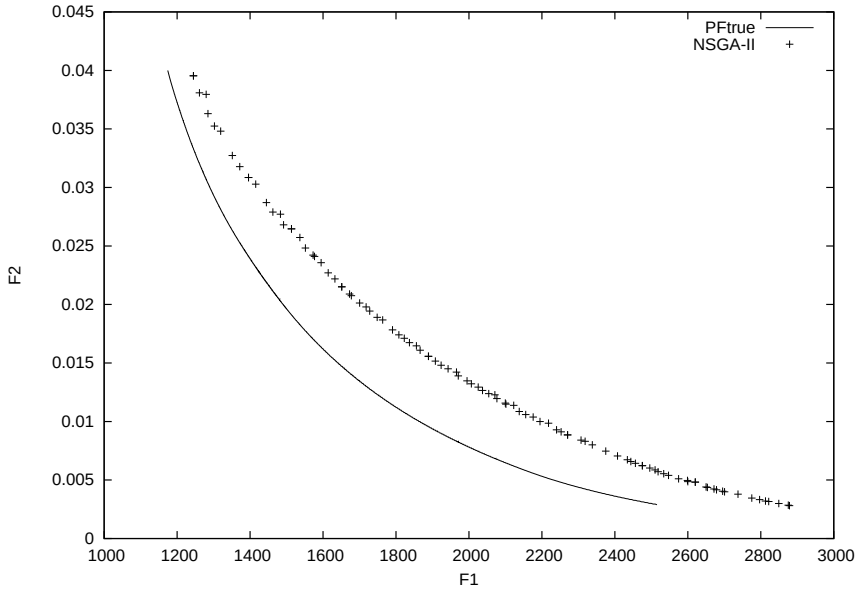

Fig. 8 Pareto fronts produced by the NSGA-II for the second test function. The true Pareto front is shown as a continuous line.

Figures 7, 8 and 9 show the graphical results produced by the microGA, the NSGA-II and PAES in the second test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables 5, 6, 7 and 8 show the comparison of results among the three algorithms considering the metrics previously described. In this case, the microGA had the best numerical values with respect to all the metrics. Graphically, it is also clear that both the NSGA-II and PAES completely missed the true Pareto front and converged to a false front.

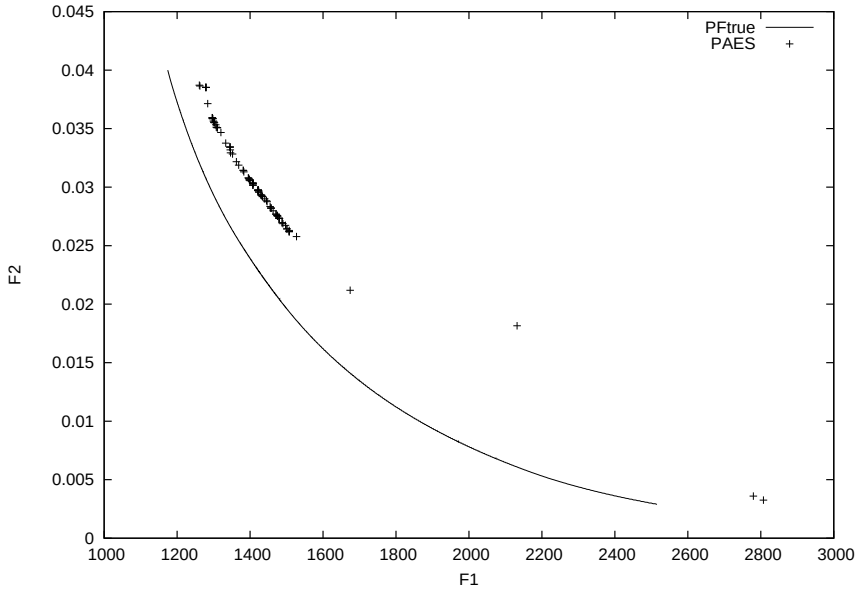

Fig. 9 Pareto fronts produced by PAES for the second test function. The true Pareto front is shown as a continuous line.

Time	microGA	NSGA-II	PAES
Best	0.258	3.063	2.684
Worst	0.282	3.086	2.958
Average	0.26315	3.06825	2.82805
Median	0.262	3.0675	2.8055
Std. Dev.	0.00580630	0.00501445	2.717377174

Table 8 Computational time (in seconds) required by each algorithm for the second test function.

In terms of computational time, the microGA was ten times faster than PAES and twelve times faster than the NSGA-II.

8

Example 3

Our third example consists of optimizing the two-bar plane truss shown in Figure 10. The mathematical description of the problem is the following (Cheng and Li 1999):

Minimize

$$f_{volume} = f_1(\mathbf{x}) = x_1(16 + y^2)^{0.5} + x_2(1 + y^2)^{0.5} \quad (19)$$

$$f_{stress,AC} = f_2(\mathbf{x}) = \frac{20(16 + y^2)^{0.5}}{yx_1} \quad (20)$$

such that

$f_{volume} \leq 0.1$

$f_{stress,AC} \leq 100000$

$f_{stress,BC} \leq 100000$

where:

$1 \leq y \leq 3$

$x_1, x_2 > 0$

$f_{stress,BC} = \frac{80(1+y^2)^{0.5}}{yx_2}$

In this example, the total number of fitness function evaluations was set to 12,000.

Figures 11, 12 and 13 show the graphical results produced by the microGA, the NSGA-II and PAES in the third test function chosen. The true Pareto front of the problem is shown

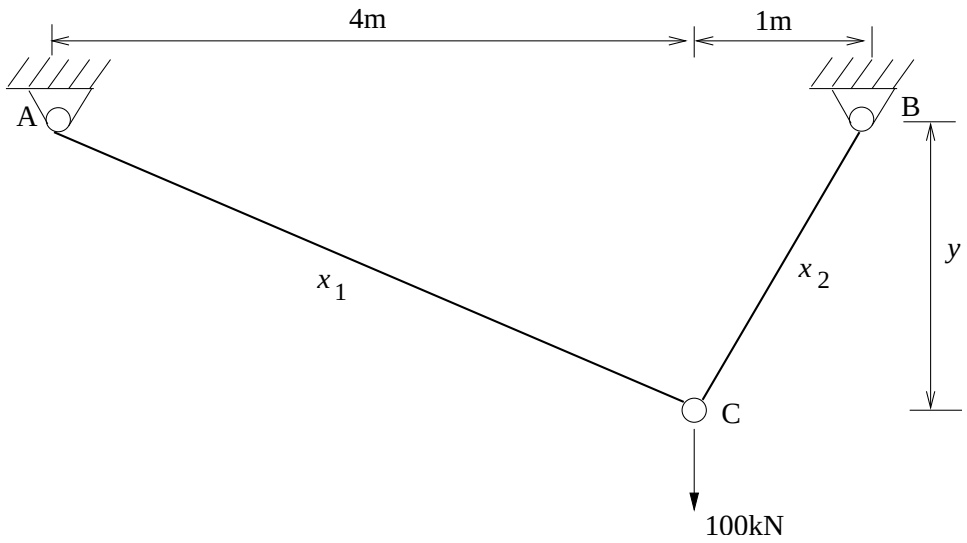

Fig. 10 Plane truss used for the third example. The structural volume and the joint displacement (Δ) are to be minimized.

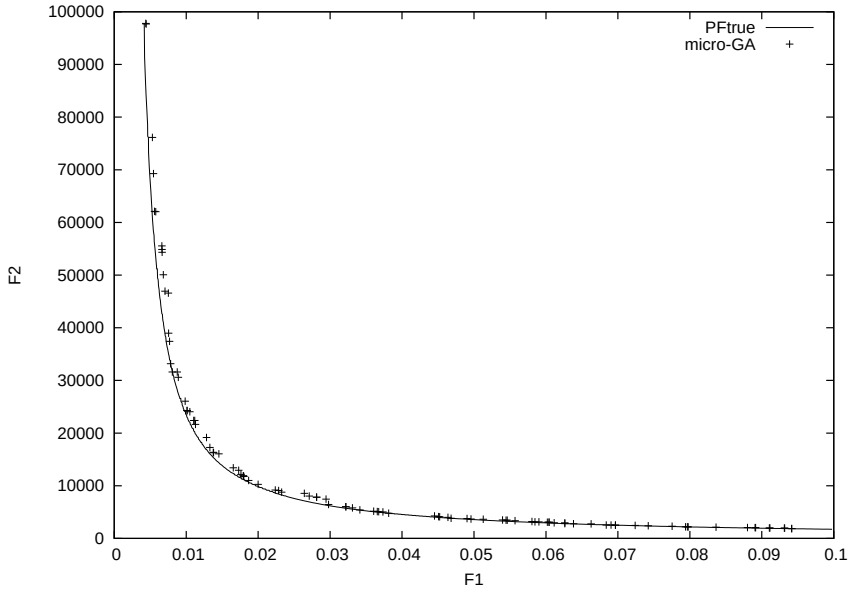

Fig. 11 Pareto fronts produced by the microGA for the third test function. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

as a continuous line. Tables 9, 10, 11 and 12 show the comparison of results among the three algorithms considering the metrics previously described. In this case, the microGA had the best average value with respect to generational distance, and the second best average value with respect to error ratio. With respect to spacing, it had the poorest result. However, if we look at the graphical results, it is clear that the NSGA-II totally missed the true Pareto front and therefore its good spacing value becomes irrelevant. PAES has a good error ratio value and a better spacing than the NSGA-II, but it generated less points of the true Pareto front. Therefore, we can again conclude that, regardless of the numerical values of the met-

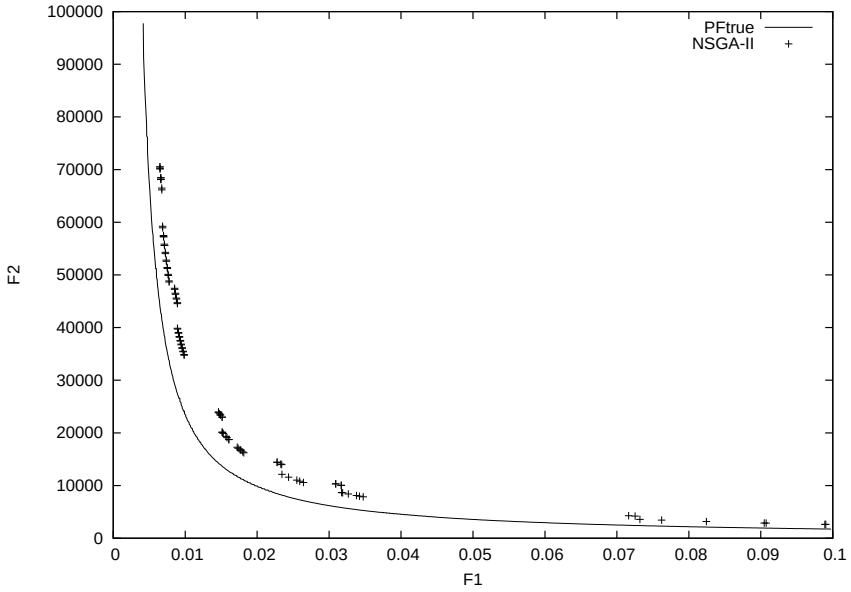

Fig. 12 Pareto fronts produced by the NSGA-II for the third test function. The true Pareto front is shown as a continuous line.

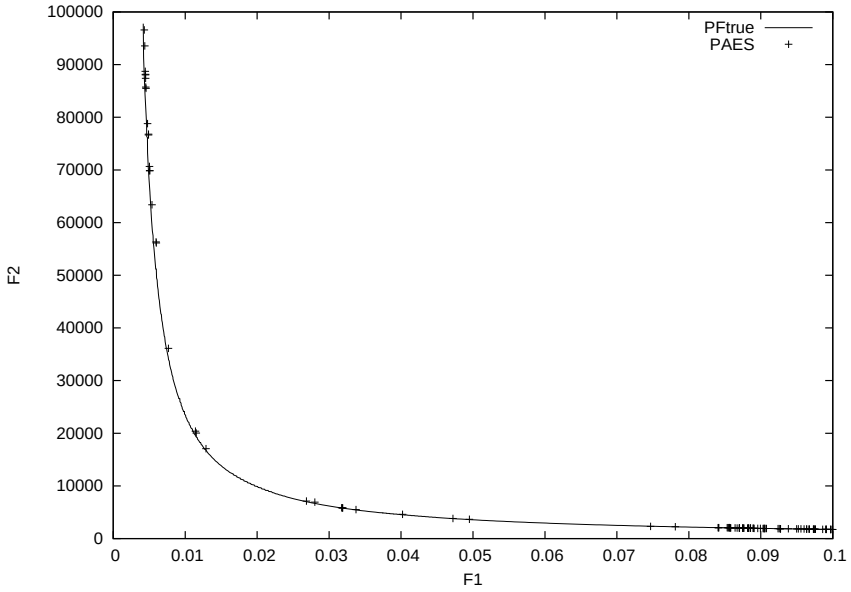

Fig. 13 Pareto fronts produced by PAES for the third test function. The true Pareto front is shown as a continuous line.

rics, the microGA had the best overall performance in this problem. Also note that in terms of computational time, the microGA was almost three times faster than the NSGA-II and it was twenty five times faster than PAES.

8.1

Example 4

Our fourth example consists of the speed reducer problem shown in Figure 14. The problem is the following (Wu 2001):

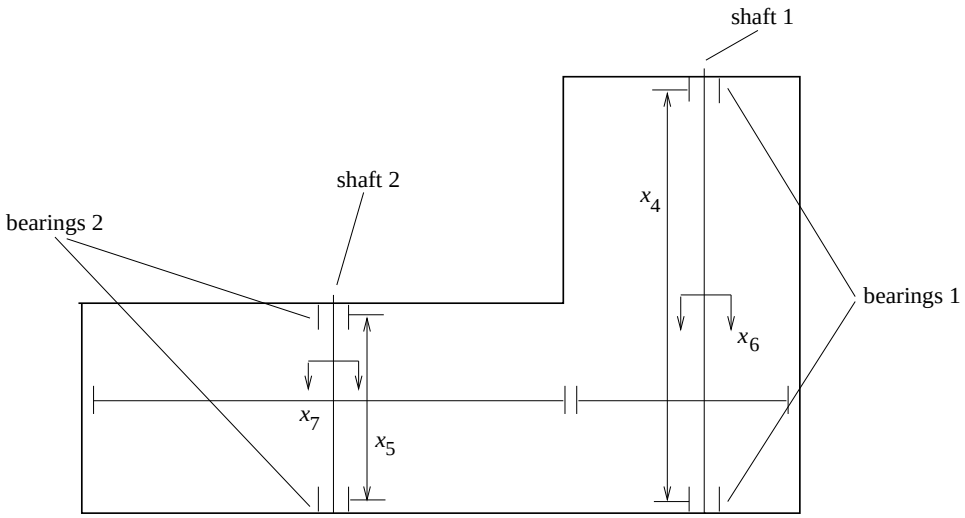

Fig. 14 Speed reducer used for the fourth example. The weight and stress are to be minimized.

ER	microGA	NSGA-II	PAES
Best	0.29	0.29	0.27
Worst	1.04762	1.01	0.93
Average	0.87489485	0.943	0.557
Median	0.985784	1.01	0.535
Std. Dev.	0.22558222	0.18342071	0.17553227

Table 9 Results of the Error Ratio metric for the third test function.

SP	microGA	NSGA-II	PAES
Best	379.427	4.90527	360.437
Worst	7321.23	442.208	3462.49
Average	1891.865	181.2533155	1220.87415
Median	1182.675	121.6995	1040.385
Std. Dev.	1807.102861787	155.566882103	652.87969651

Table 11 Results of the Spacing metric for the third test function.

GD	microGA	NSGA-II	PAES
Best	9.71281	0.947	10.6856
Worst	41.2999	78.143	58.1152
Average	20.0494855	33.38484245	36.7977
Median	17.9617	19.2609	36.156
Std. Dev.	7.69435095	30.09812074	13.65416231

Table 10 Results of the Generational Distance metric for the third test function.

Time	microGA	NSGA-II	PAES
Best	0.669	2.769	25.193
Worst	1.976	3.147	30.607
Average	1.2012	3.1125	27.5981
Median	1.2165	3.135	27.371
Std. Dev.	0.45024152	0.08220033	1.68822828

Table 12 Computational time (in seconds) required by each algorithm for the third test function.

Minimize

$$f_{weight} = f_1(\mathbf{x}) =$$

$$0.7854x_1x_2^2(10x_3^2/3 + 14.933x_3 - 43.0934) -$$

$$1.508x_1(x_6^2 + x_7^2) +$$

$$7.477(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x7^2)$$

$$f_{stress} = f_2(\mathbf{x}) = \frac{\sqrt{(745.0x_4/x_2x_3)^2 + 1.6910^7}}{0.1x_6^3} \qquad (21)$$

such that

$g_1: \frac{1.0}{x_1 x_2^2 x_3} - \frac{1.0}{27.0} \leq 0$

$g_2: \frac{1.0}{x_1 x_2^2 x_3^2} - \frac{1.0}{397.5} \leq 0$

$g_3: \frac{x_4^3}{x_2 x_3 x6^4} - \frac{1.0}{1.93} \leq 0$

$g_4: \frac{x5^3}{x_2 x_3 x7^4} - \frac{1.0}{1.93} \leq 0$

$g_5: x_2 x_3 - 40.0 \leq 0$

$g_6: x_1/x_2 - 12.0 \leq 0$

$g_7: 5.0 - x_1/x_2 \leq 0$

$g_8: 1.9 - x_4 + 1.5x6 \leq 0$

$g_9: 1.9 - x5 + 1.1x7 \leq 0$

$g_{10}: \frac{\sqrt{(745x_4/x_2x_3)^2 + 1.6910^7}}{0.1x6^3} \leq 1300)$

$g_{11}: \frac{\sqrt{(745x5/x_2x_3)^2 + 1.57510^8}}{0.1x7^3} \leq 1100)$

where:

$2.6 \leq x_1 \leq 3.6$

$0.7 \leq x_2 \leq 0.8$

$17 \leq x_3 \leq 28$

$7.3 \leq x_4 \leq 8.3$

$7.3 \leq x_5 \leq 8.3$

$2.9 \leq x_6 \leq 3.9$

$5.0 \leq x_7 \leq 5.5$

In this example, the total number of fitness function evaluations was set to 24,000. This increase was due to the complexity of the search space of this problem (its Pareto front is more difficult to be generated than in the previous examples).

Figures 15, 16 and 17 show the graphical results produced by the microGA, the NSGA-II and PAES in the fourth test function chosen. The true Pareto front of the problem is shown as a continuous line. Tables 13, 14, 15 and 16 show the comparison of results among the three algorithms considering the metrics previously described. Regarding generational distance and error ratio, the microGA obtained the best average results in this problem. With respect to spacing, the NSGA-II had the best average value. However, in the graphical representation of the results we can clearly see that both the NSGA-II and PAES missed the true Pareto front of the problem (in fact,

ER	microGA	NSGA-II	PAES
Best	0.33	0.96	1.01
Worst	1.02857	1.01	1.01
Average	0.9347823	1.0055	1.01
Median	1.00595	1.01	1.01
Std. Dev.	0.16143897	0.01394538	0

Table 13 Results of the Error Ratio metric for the fourth example.

GD	microGA	NSGA-II	PAES
Best	1.08871	2.05287	70.7167
Worst	9.07369	28.3722	87.8481
Average	3.117536	9.843702	77.99834
Median	2.700905	7.357945	78.63515
Std. Dev.	1.67810867	7.08103039	4.21026087

Table 14 Results of the Generational Distance metric for the fourth example.

SP	microGA	NSGA-II	PAES
Best	14.5077	0.0784511	9.4861
Worst	137.044	12.7075	27.2249
Average	47.80098	2.765449155	16.20129
Median	41.3446	1.95639	15.21375
Std. Dev.	32.80151572	3.53493787	4.26842769

Table 15 Results of the Spacing metric for the fourth example.

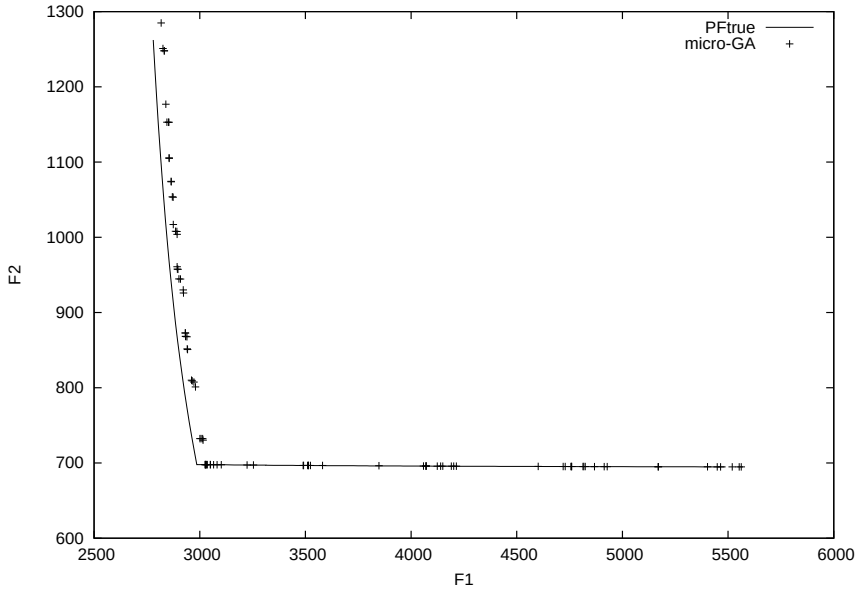

Fig. 15 Pareto front produced by the microGA for the fourth example. The true Pareto front is shown as a continuous line. The solutions displayed correspond to the median result with respect to the generational distance metric for each of the algorithms compared.

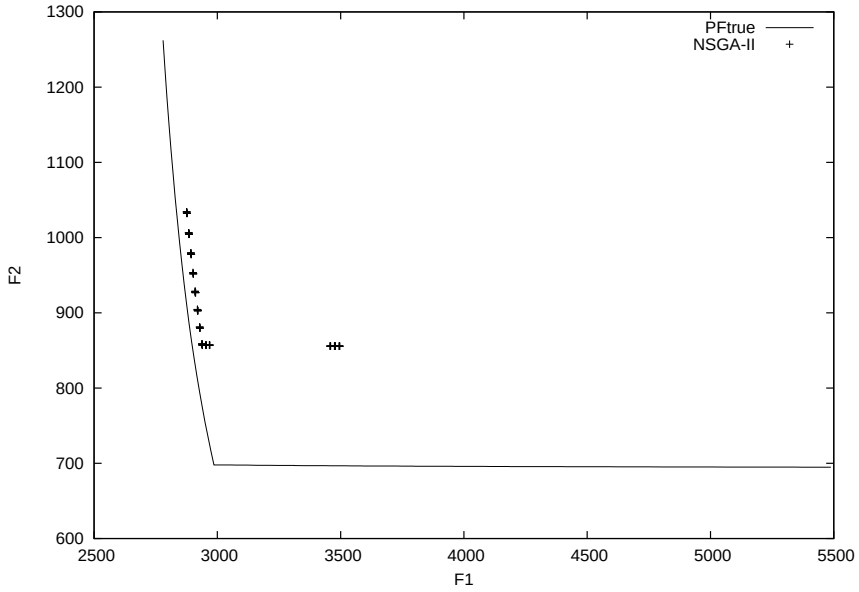

Fig. 16 Pareto fronts produced by the NSGA-II for the fourth example. The true Pareto front is shown as a continuous line.

PAES found nondominated solutions which are very far away from the true Pareto front). Therefore, we again conclude that the microGA had the best overall performance in this case. Regarding computational time, the microGA was about 1.6 times faster than the NSGA-II and about thirteen times faster than PAES.

Summarizing results, we can clearly see that the microGA outperformed the two other algorithms against which it was compared. It is worth noticing that both the NSGA-II and

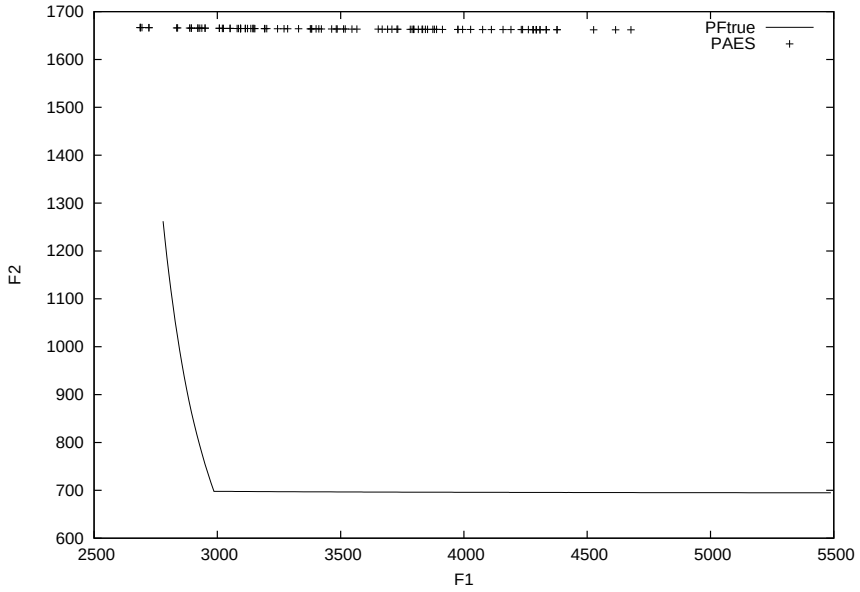

Fig. 17 Pareto fronts produced by PAES for the fourth example. The true Pareto front is shown as a continuous line.

Time	microGA	NSGA-II	PAES
Best	4.043	7.314	56.799
Worst	5.503	7.644	67.963
Average	4.6485	7.40275	59.5992
Median	4.5545	7.388	59.4325
Std. Dev.	0.37922101	0.07438228	2.45812146

Table 16 Computational time (in seconds) required by each algorithm for the fourth example.

PAES were chosen for this comparative study because they are representative of the state-of-the-art in evolutionary multiobjective optimization. Nevertheless, they performed rather poorly in some of the test functions chosen, which gives an idea of their high degree of complexity.

We attribute the good performance of our microGA to the fact that it is able to produce a good balance between exploration and exploitation by combining the search engine provided by a simple genetic algorithm with three forms of elitism (as explained in Section 4). The non-replaceable memory of our microGA is its main source of diversity (a key component of any multiobjective evolutionary algorithm) and the replaceable memory combined with the external archive encourage that the nondominated solutions produced converge towards the Pareto optimal set and, at the same time, are uniformly distributed. Because each of the components of our microGA is very simple (computationally speaking), our approach requires, in several cases, a considerably lower computational time than the two other algorithms compared. The quality of the solutions obtained seems to corroborate our hypothesis regarding the effectiveness of such mechanisms in solving multiobjective optimization problems.

It is worth noticing, however, that the previous results, although encouraging, do not necessarily guarantee that our microGA will perform equally well in more complex, large-scale problems (this, however, cannot be guaranteed for any

evolutionary algorithm because of their stochastic nature (Wolpert and Macready 1997)). However, it would be expected that our microGA would remain at least competitive with respect to other multiobjective evolutionary algorithms when used in more complex problems.

9 Parameters fine-tuning

Since our microGA uses several parameters that are not typical of evolutionary multiobjective optimization approaches, we performed several experiments to try to determine a set of values that can be used by default (i.e., when nothing about the problem is known).

The size of the external memory is a parameter that should be easy to setup, since it corresponds to the number of non-dominated vectors that the user wishes to find.

Regarding the size of the population memory, we recommend to set it to 50% of the size of the external memory. The reason is that if a larger percentage is used, the number of individuals to undergo evolution becomes too large. On the other hand, if the percentage is lower, we can easily lose diversity.

For the number of iterations of the microGA, we found that a value between two and five seems to work well. It is important to be aware of the fact that a larger value for this parameter implies a greater CPU cost for the algorithm. However, a larger value provides Pareto fronts with a better spread. Therefore, the setup of this parameter is really a trade-off between efficiency and quality of the solutions found.

Regarding the number of subdivisions of the adaptive grid, the recommended range is a value between 5 and 100. As a default value, we suggest 25, which is the value that provided the best overall performance in our experiments. Larger values for this parameter will provide a better spread of the Pareto front, but will sacrifice efficiency and memory requirements.

For the percentage of non-replaceable memory, we suggest to use 0.3, since this value ensures that for each pair of individuals evolved, one will be randomly selected (i.e., this promotes diversity).

Finally, for the replacement cycle, we suggest to use a value between 25 and n (where n is the total number of iterations). We have used values between 25 and 200 for this parameter. However, this is a parameter that requires special attention and we intend to study its behavior in more detail to try to derive more general values within a narrower range. This value is also critical for our algorithm, because if it is too small, the algorithm may converge to a local Pareto front. If it is too large, the replacement of the population at each cycle may not be enough to guarantee the necessary diversity. So far, the value proposed has been empirically set up for each particular problem.

10 Conclusions and Future Work

We have proposed the use of a genetic algorithm with a very small population size (a micro genetic algorithm) and a reinitialization process to solve multiobjective optimization prob-

lems. The algorithm has been validated using several structural optimization problems and has been compared to the NSGA-II and PAES using three metrics taken from the specialized literature. In all cases, the microGA outperformed the two other algorithms and was considerably faster than them.

Perhaps the main current drawback of our microGA for multiobjective optimization is the fact that it requires several parameters. Although we have provided some guidelines to set them up, we are currently working in a new version of our algorithm that uses on-line adaptation to make it unnecessary to fine tune any of these parameters. We are also analyzing different approaches to incorporate preferences from the decision maker as to narrow the search performed by our microGA (Coello Coello 2000).

Acknowledgements

The authors gratefully acknowledge the valuable comments provided by the anonymous reviewers which greatly helped them to improve the contents of this paper.

The first author acknowledges support from CONACyT through project 42435-Y to perform this work. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at CINVESTAV-IPN's Electrical Engineering Department.

References

Cannon, W. D. (1932), *The Wisdom of the Body*, Norton and Company, New York, NY.

Cheng, F. and Li, X. (1999), 'Generalized Center Method for Multiobjective Engineering Optimization', *Engineering Optimization* **31**, 641–661.

Coello Coello, C. A. (1999), 'A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques', *Knowledge and Information Systems. An International Journal* **1**(3), 269–308.

Coello Coello, C. A. (2000), Handling Preferences in Evolutionary Multiobjective Optimization: A Survey, *in* '2000 Congress on Evolutionary Computation', Vol. 1, IEEE Service Center, Piscataway, New Jersey, pp. 30–37.

Coello Coello, C. A. (2002), 'Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art', *Computer Methods in Applied Mechanics and Engineering* **191**(11-12), 1245–1287.

Coello Coello, C. A., Van Veldhuizen, D. A. and Lamont, G. B. (2002), *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York. ISBN 0-3064-6762-3.

Deb, K. (2001), *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK. ISBN 0-471-87339-X.

Deb, K., Agrawal, S., Pratab, A. and Meyarivan, T. (2000), A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II, *in* 'Proceedings of the Parallel Problem Solving from Nature VI Conference', Springer, pp. 849–858.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002), 'A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II', *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197.

Edgeworth, F. Y. (1881), *Mathematical Physics*, P. Keagan, London, England.

Fogel, L. J. (1966), *Artificial Intelligence through Simulated Evolution*, John Wiley, New York, NY.

Fogel, L. J. (1999), *Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming*, John Wiley & Sons, New York, NY.

Fonseca, C. M. and Fleming, P. J. (1995), 'An Overview of Evolutionary Algorithms in Multiobjective Optimization', *Evolutionary Computation* **3**(1), 1–16.

Goldberg, D. E. (1989*a*), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Reading, Massachusetts.

Goldberg, D. E. (1989*b*), Sizing Populations for Serial and Parallel Genetic Algorithms, *in* J. D. Schaffer, ed., 'Proceedings of the Third International Conference on Genetic Algorithms', Morgan Kaufmann Publishers, San Mateo, California, pp. 70–79.

Holland, J. H. (1962), 'Outline for a logical theory of adaptive systems', *Journal of the Association for Computing Machinery* **9**, 297–314.

Jaszkiewicz, A. (2002), 'Genetic local search for multiple objective combinatorial optimization', *European Journal of Operational Research* **137**(1), 50–71.

Johnson, E. and Abushagur, M. (1995), 'Micro-Genetic Algorithm Optimization Methods Applied to Dielectric Gratings', *Journal of the Optical Society of America* **12**(5), 1152–1160.

Knowles, J. D. and Corne, D. W. (2000), 'Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy', *Evolutionary Computation* **8**(2), 149–172.

Krishnakumar, K. (1989), Micro-genetic algorithms for stationary and non-stationary function optimization, *in* 'SPIE Proceedings: Intelligent Control and Adaptive Systems', Vol. 1196, pp. 289–296.

Kursawe, F. (1991), A variant of evolution strategies for vector optimization, *in* H. P. Schwefel and R. Männer, eds, 'Parallel Problem Solving from Nature. 1st Workshop, PPSN I', Vol. 496 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Germany, pp. 193–197.

Michalewicz, Z. and Schoenauer, M. (1996), 'Evolutionary Algorithms for Constrained Parameter Optimization Problems', *Evolutionary Computation* **4**(1), 1–32.

Pareto, V. (1896), *Cours D'Economie Politique*, Vol. I and II, F. Rouge, Lausanne.

Richardson, J. T., Palmer, M. R., Liepins, G. and Hilliard, M. (1989), Some Guidelines for Genetic Algorithms with Penalty Functions, *in* J. D. Schaffer, ed., 'Proceedings of the Third International Conference on Genetic Algorithms', Morgan Kaufmann Publishers, George Mason University, pp. 191–197.

Schaffer, J. D. (1984), Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, PhD thesis, Vanderbilt University.

Schaffer, J. D. (1985), Multiple objective optimization with vector evaluated genetic algorithms, *in* 'Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms', Lawrence Erlbaum, Hillsdale, NJ, pp. 93–100.

Schott, J. R. (1995), Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization, Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Schwefel, H.-P. (1965), 'Kybernetische evolution als strategie der experimentellen forschung inder strömungstechnik', Dipl.-Ing. thesis. (in German).

Schwefel, H.-P. (1981), *Numerical Optimization of Computer Models*, Wiley, Chichester, UK.

Srinivas, N. and Deb, K. (1994), 'Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms', *Evolutionary Computation* **2**(3), 221–248.

Van Veldhuizen, D. A. (1999), Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

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

Van Veldhuizen, D. A. and Lamont, G. B. (2000), 'Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art', *Evolutionary Computation* **8**(2), 125–147.

Wolpert, D. H. and Macready, W. G. (1997), 'No Free Lunch Theorems for Optimization', *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82.

Wu, J. (2001), Quality Assisted Multiobjective and Multidisciplinary Genetic Algorithms, PhD thesis, Department of Mechanical Engineering, University of Maryland at College Park, College Park, Maryland.

Xiao, F. and Yabe, H. (1998), 'Microwave Imaging of Perfectly Conducting Cylinders from Real Data by Micro Genetic Algorithm Coupled with Deterministic Method', *IEICE Transactions on Electronics* **E81-C**(12), 1784–1792.

Zitzler, E., Deb, K. and Thiele, L. (2000), 'Comparison of Multiobjective Evolutionary Algorithms: Empirical Results', *Evolutionary Computation* **8**(2), 173–195.