A Micro-Genetic Algorithm for Multiobjective Optimization

Carlos A. Coello Coello[1] and Gregorio Toscano Pulido[2]

[1] CINVESTAV-IPN
Depto. de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México, D. F. 07300
ccoello@cs.cinvestav.mx

[2] Maestría en Inteligencia Artificial
LANIA-Universidad Veracruzana
Xalapa, Veracruz, México 91090
gtoscano@mia.uv.mx

Abstract. In this paper, we propose a multiobjective optimization approach based on a micro genetic algorithm (micro-GA) which is a genetic algorithm with a very small population (four individuals were used in our experiment) and a reinitialization process. We use three forms of elitism and a memory to generate the initial population of the micro-GA. Our approach is tested with several standard functions found in the specialized literature. The results obtained are very encouraging, since they show that this simple approach can produce an important portion of the Pareto front at a very low computational cost.

1 Introduction

A considerable number of evolutionary multiobjective optimization techniques have been proposed in the past [3, 20]. However, until recently, little emphasis had been placed on efficiency issues. It is well known that the two main processes that consume most of the running time of an EMOO algorithm are: the ranking of the population (i.e., the comparisons required to determine non-dominance) and the mechanism to keep diversity (evolutionary algorithms tend to converge to a single solution because of stochastic noise; it is therefore necessary to avoid this with an additional mechanism).

Recent research has shown ways of improving the efficiency of an EMOO technique (see for example [14, 6]). The main emphasis has been on using an external file that stores nondominated vectors found during the evolutionary process which are reinserted later in the population (this can be seen as a form of elitism in the context of multiobjective optimization [10, 17, 22]).

Following the same line of thought of this current research, we decided to develop an approach in which we would use a GA with a very small population size and a reinitialization process (the so-called micro-GA) combined with an external file to store nondominated vectors previously found. Additionally, we decided to include an efficient mechanism to keep diversity (similar to the adaptive grid method of Knowles & Corne [14]). Our motivation was to show that a micro-GA carefully designed is sufficient to generate the Pareto front of a multiobjective optimization problem. Such approach not only reduces the amount of comparisons required to generate the Pareto front (with respect to traditional EMOO approaches based on Pareto ranking), but also allows us to control the amount of points that we wish to obtain from the Pareto front (such amount is in fact a parameter of our algorithm).

2 Related Work

The term micro-genetic algorithm (micro-GA) refers to a small-population genetic algorithm with reinitialization. The idea was suggested by some theoretical results obtained by Goldberg [8], according to which a population size of 3 was sufficient to converge, regardless of the chromosomic length. 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 micro-GA was Krishnakumar [15], who used a population size of 5, a crossover rate of 1 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 4 competitions between strings that were adjacent in the population array, and declaring to the individual with the highest fitness as the winner. Krishnakumar [15] compared his micro-GA against a simple GA (with a population size of 50, a crossover rate of 0.6 and a mutation rate of 0.001). Krishnakumar [15] reported faster and better results with his micro-GA 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 micro-GAs [13, 7, 12, 21]. However, to the best of our knowledge, the current paper reports the first attempt to use a micro-GA for multiobjective optimization, although some may argue that the multi-membered versions of PAES can be seen as a form of micro-GA[1] [14]. However, Knowles & Corne [14] concluded that the addition of a population did not, in general, improve the performance of PAES, and increased the computational overhead in an important way. Our technique, on the other

[1] We recently became aware of the fact that Jaszkiewicz [11] proposed an approach in which a small population initialized from a large external memory and utilized it for a short period of time. However, to the best of our knowledgem this approach has been used only for multiobjective combinatorial optimization.

hand, uses a population and traditional genetic operators and, as we will show in a further section, it performs quite well.

3 Description of the Approach

In this paper, we propose a micro-GA with two memories: the **population memory**, which is used as the source of diversity of the approach, and the **external memory**, which is used to archive members of the Pareto optimal set. Population memory is respectively divided in two parts: a **replaceable** and a **non-replaceable** portion (the percentages of each can be regulated by the user).

The way in which our technique works is illustrated in Fig. 1. First, an initial random population is generated. This population feeds the population memory, which is divided in two parts as indicated before. The non-replaceable portion of the population memory will never change during the entire run and is meant to provide the required diversity for the algorithm. The initial population of the micro-GA at the beginning of each of its cycles is taken (with a certain probability) from both portions of the population memory as to allow a greater diversity.

During each cycle, the micro-GA undergoes conventional genetic operators: tournament selection, two-point crossover, uniform mutation, and elitism (regardless of the amount of nondominated vectors in the population only one is arbitrarily selected at each generation and copied intact to the following one). After the micro-GA finishes one cycle (i.e., when nominal convergence is achieved), we choose two nondominated vectors[2] from the final population (the first and last) and compare them with the contents of the external memory (this memory is initially empty). If either of them (or both) remains as nondominated after comparing against the vectors in this external memory, then they are included there. All the dominated vectors are eliminated from the external memory. These two vectors are also compared against two elements from the replaceable portion of the population memory (this is done with a loop, so that each vector is only compared against a single position of the population memory). If either of these vectors dominates to its match in the population memory, then it replaces it. The idea is that, over time, the replaceable part of the population memory will tend to have more nondominated vectors. Some of them will be used in the initial population of the micro-GA to start new evolutionary cycles.

Our approach uses three types of elitism. The first is based on the notion that if we store the nondominated vectors produced from each cycle of the micro-GA, we will not lose any valuable information obtained from the evolutionary process. The second is based on the idea that if we replace 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.

[2] This is assuming that we have two or more nondominated vectors. If there is only one, then this vector is the only one selected.

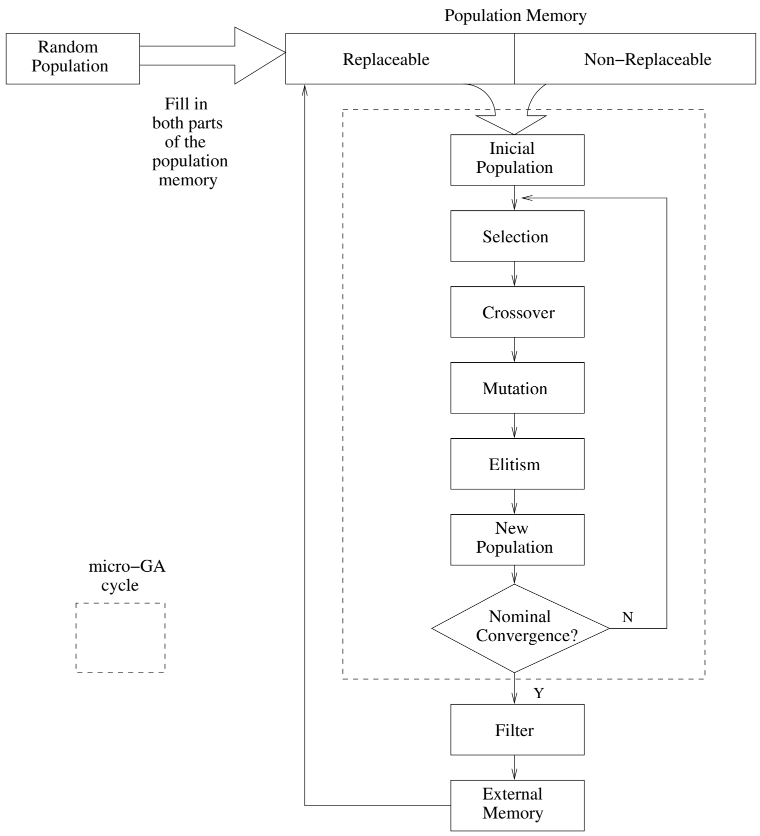

Fig. 1. Diagram that illustrates the way in which our micro-GA works.

This notion was hinted by Goldberg [8]. Nominal convergence, in our case, is defined in terms of a certain (low) number of generations (typically, two to five in our case). However, similarities among the strings (either at the phenotypical or genotypical level) could also been used as a criterion for convergence. The third type of elitism is applied at certain intervals (defined by a parameter called "replacement cycle"). What we do is to take a certain amount of points from all the regions of the Pareto front generated so far and we use them to fill in 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 intends to use the best solutions generated so far as the starting point for the micro-GA, so that we can improve them (either by getting closer to the true Pareto front or by getting a better distribution). This also avoids that the contents of the replaceable memory becomes homogeneous.

To keep diversity in the Pareto front, we use an approach similar to the adaptive grid proposed by Knowles & Corne [14]. The idea is that once the archive that stores nondominated solutions has reached its limit, we divide the 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 is less populated than the most crowded location. Alternatively, the new nondominated solution could also be accepted if the individual belongs to a location outside the previously speficied boundaries. In other words, the less crowded regions are given preference so that the spread of the individuals on the Pareto front can be more uniform.

The pseudo-code of the algorithm is the following:

function Micro-GA
    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 micro-GA (P_i) 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)
                Produce the next generation
              end
            until nominal convergence is reached
          Copy two nondominated vectors from P_i
          to the external memory E

if E is full when trying to insert ind_b
then adaptive_grid(ind_b)
Copy two nondominated vectors from P_i to M
if i **mod** replacement_cycle
then apply second form of elitism
i=i+1
end while
end function

The adaptive grid requires two parameters: the expected size of the Pareto front and the amount 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 and it is provided by the user. The second parameter (the amount of positions in which we will divide the solution space) has to be provided by the user as well, although we have found that our approach is not very sensitive to it (e.g., in most of our experiments a value of 15 or 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 the occurrence of this situation.

When the external memory is full, then we use the adaptive grid to decide what nondominated vectors will be eliminated. The adaptive grid will try to balance the amount of individuals representing each of the elements of the Pareto set, so that we can get a uniform spread of points along the Pareto front.

4 Comparison of Results

Several test functions were taken from the specialized literature to compare our approach. In all cases, we generated the true Pareto fronts of the problems using exhaustive enumeration (with a certain granularity) so that we could make a graphical comparison of the quality of the solutions produced by our micro-GA.

Since the main aim of this approach has been to increase efficiency, we additionally decided to compare running times of our micro-GA against those of the NSGA II [6] and PAES [14]. In the following examples, the NSGA was run using a population size of 100, a crossover rate of 0.8 (using SBX), tournament selection, and a mutation rate of 1/vars, where vars = number of decision variables of the problem. In the following examples, PAES was run using a depth of 5, a size of the archive of 100, and a mutation rate of 1/bits, where bits refers to the length of the chromosomic string that encodes the decision variables. The amount of fitness function evaluations was set such that the NSGA II, PAES and the micro-GA could reasonably cover the true Pareto front of each problem.

4.1 Test Function 1

Our first example is a two-objective optimization problem defined by Deb [5]:

$$\text{Minimize } f_1(x_1, x_2) = x_1 \tag{1}$$

$$\text{Minimize } f_2(x_1, x_2) = \frac{g(x_2)}{x_1} \tag{2}$$

where:

$$g(x_2) = 2.0 - \exp\left\{-\left(\frac{x_2 - 0.2}{0.004}\right)^2\right\} - 0.8\exp\left\{-\left(\frac{x_2 - 0.6}{0.4}\right)^2\right\} \tag{3}$$

and $0.1 \leq x_1 \leq 1.0$, $0.1 \leq x_2 \leq 1.0$.

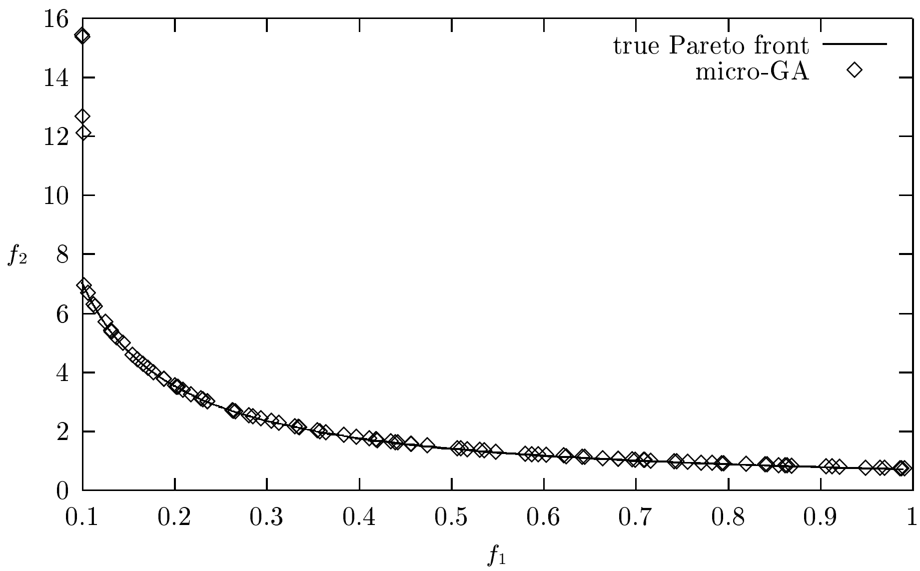

Fig. 2. Comparison of results for the first test function.

The parameters used by the micro-GA for this example were: size of the external memory = 100, size of the population memory = 50, number of iterations = 1500, number of iterations of the micro-GA (to achieve nominal convergence) = 2, number of subdivisions of the adaptive grid = 25, crossover rate = 0.7, mutation rate = 0.029, percentage of non-replaceable memory = 0.3, population size (of the micro-GA) = 4, replacement cycle at every 50 iterations.

Our first test function has a local Pareto front to which a GA can be easily attracted. Fig. 2 shows the true Pareto front for this problem with a continuous line, and the results found by the NSGA II, PAES and our micro-GA are shown with points. Similar fronts were found by the three approaches. For this example, both the NSGA II and PAES performed 12,000 evaluations of the fitness function. The average running time of each algorithm (over 20 runs) were the following: 2.601 seconds for the NSGA II (with a standard deviation of 0.33555913), 1.106 seconds for PAES (with a standard deviation of 0.25193672) and only 0.204 seconds for the micro-GA (with a standard deviation of 0.07764461).

4.2 Test Function 2

Our second example is a two-objective optimization problem proposed by Schaffer [18] that has been used by several researchers [19, 1]:

$$\text{Minimize } f_1(x) = \begin{cases} -x & \text{if } x \le 1 \\ -2 + x & \text{if } 1 < x \le 3 \\ 4 - x & \text{if } 3 < x \le 4 \\ -4 + x & \text{if } x > 4 \end{cases} \tag{4}$$

$$\text{Minimize } f_2(x) = (x - 5)^2 \tag{5}$$

and $-5 \le x \le 10$.

The parameters used for this example were: size of the external memory = 100, size of the population memory = 50, number of iterations = 150, number of iterations of the micro-GA (to achieve nominal convergence) = 2, number of subdivisions of the adaptive grid = 25, crossover rate = 0.7, mutation rate = 0.056 (1/L, where L=18 bits in this case), percentage of non-replaceable memory = 0.3, population size (of the micro-GA) = 4, replacement cycle at every 25 iterations.

This problem has a Pareto front that is disconnected. Fig. 3 shows the true Pareto front for this problem with a continuous line (the vertical line is obviously not part of the true Pareto front, but it appears because we used linear segments to connect every pair of nondominated points). We used points to represent the solutions found by the NSGA II, PAES and our micro-GA.

Again, similar Pareto fronts were found by the three approaches. For this example, both the NSGA II and PAES performed 1,200 evaluations of the fitness function. The average running time of each algorithm (over 20 runs) were the following: 0.282 seconds for the NSGA II (with a standard deviation of 0.00014151), 0.107 seconds for PAES (with a standard deviation of 0.13031718) and only 0.017 seconds for the micro-GA (with a standard deviation of 0.0007672).

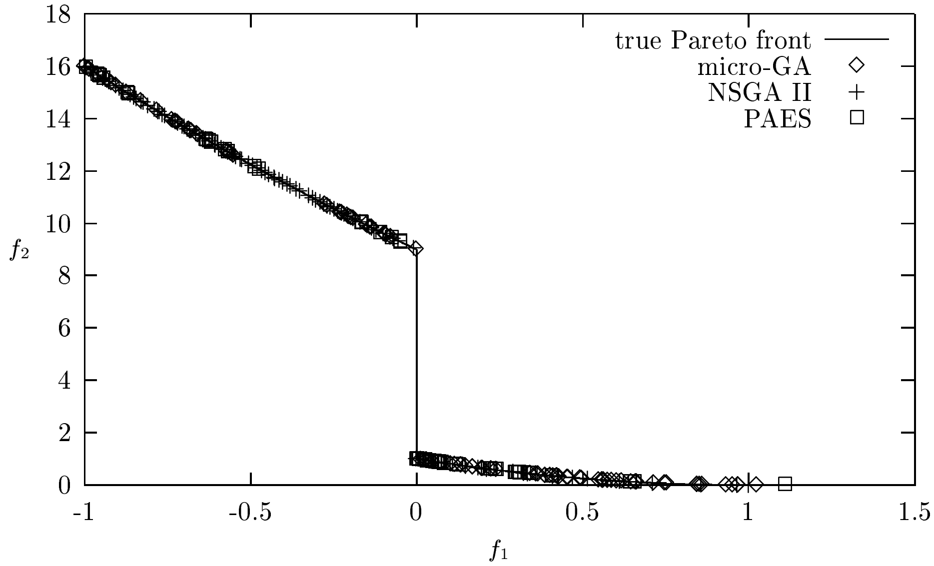

Fig. 3. Comparison of results for the second test function.

4.3 Test Function 3

Our second example is a two-objective optimization problem defined by Deb [5]:

$$\text{Minimize } f_1(x_1, x_2) = x_1 \tag{6}$$

$$\text{Minimize } f_2(x_1, x_2) = g(x_1, x_2) \cdot h(x_1, x_2) \tag{7}$$

where:

$$g(x_1, x_2) = 11 + x_2^2 - 10 \cdot \cos(2\pi x_2) \tag{8}$$

$$h(x_1, x_2) = \begin{cases} 1 - \sqrt{\frac{f_1(x_1, x_2}{g(x_1, x_2}} & \text{if } f_1(x_1, x_2) \leq g(x_1, x_2) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

and $0 \leq x_1 \leq 1$, $-30 \leq x_2 \leq 30$.

This problem has 60 local Pareto fronts. Fig. 4 shows the true Pareto front for this problem with a continuous line. The results obtained by the NSGA II, PAES and our micro-GA are displayed as points.

The parameters used by the micro-GA for this example were: size of the external memory = 100, size of the population memory = 50, number of iterations = 700, number of iterations of the micro-GA (to achieve nominal convergence)

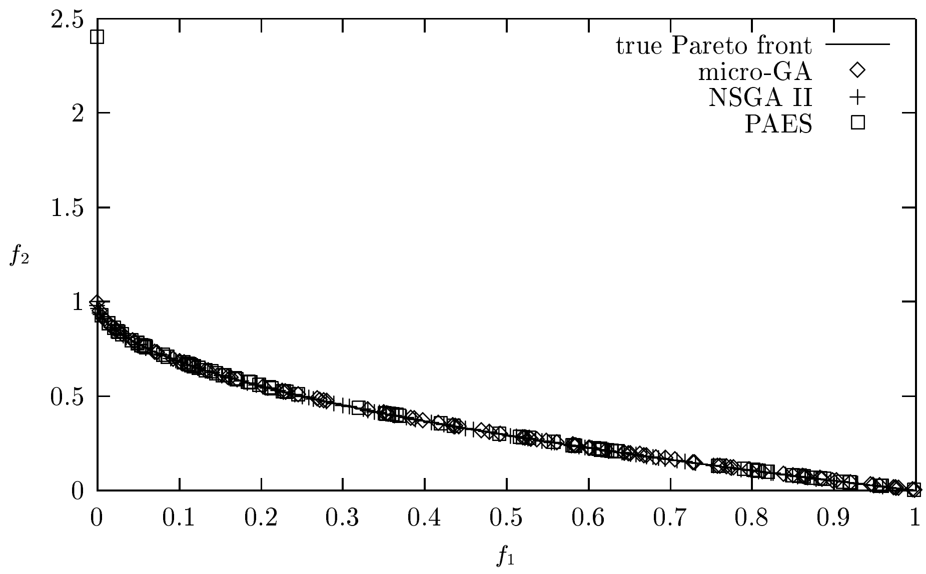

Fig. 4. Comparison of results for the third test function.

= 4, number of subdivisions of the adaptive grid = 25, crossover rate = 0.7, mutation rate = 0.029, percentage of non-replaceable memory = 0.3, population size (of the micro-GA) = 4, replacement cycle at every 50 iterations.

Once again, the fronts produced by the three approaches are very similar. For this example, both the NSGA II and PAES performed 11,200 evaluations of the fitness function. The average running time of each algorithm (over 20 runs) were the following: 2.519 seconds for the NSGA II (with a standard deviation of 0.03648403), 2.497 seconds for PAES (with a standard deviation of 1.03348519) and only 0.107 seconds for the micro-GA (with a standard deviation of 0.00133949).

4.4 Test Function 4

Our fourth example is the so-called "unitation versus pairs" problem [9], which involves the maximization of two functions over bit strings. The first function, f_1 is the number of pairs of adjacent complementary bits found in the string, and the second function, f_2 is the numbers of ones found in the string. The Pareto front in this case is discrete. We used a string length of 28, and therefore, the true Pareto front is composed of 15 points.

The parameters used for this example were: size of the external memory = 100, size of the population memory = 15, number of iterations = 1250, number of iterations of the micro-GA (to achieve nominal convergence) = 1, number of subdivisions of the adaptive grid = 3, crossover rate = 0.5, mutation rate

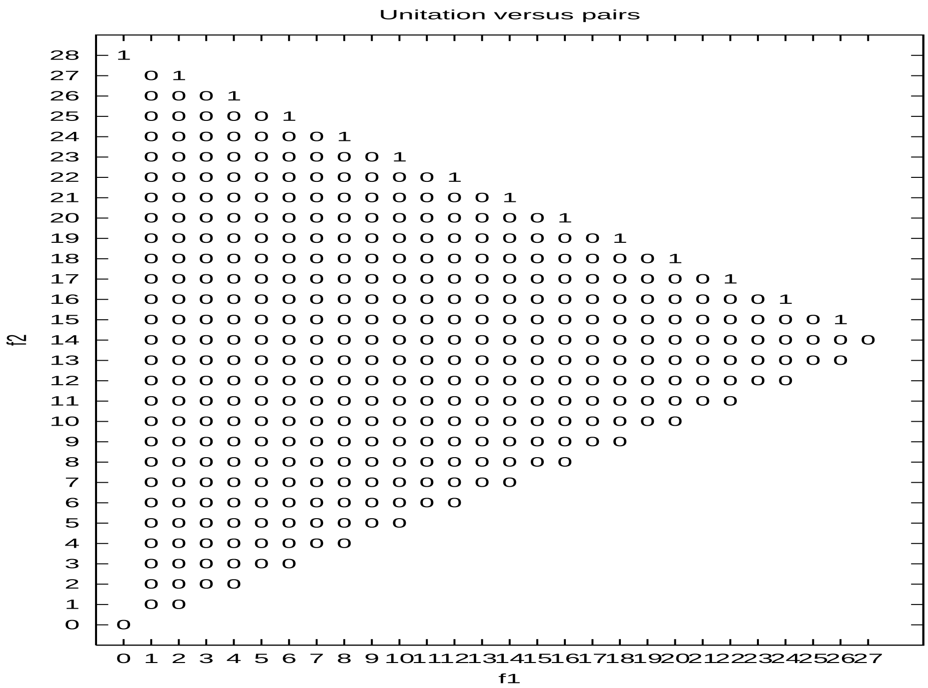

Fig. 5. Results of the micro-GA for the fourth test function.

= 0.035, percentage of non-replaceable memory = 0.2, population size (of the micro-GA) = 4, replacement cycle at every 25 iterations.

Fig. 5 shows the results obtained by our micro-GA for the fourth test function. A total of 13 (out of 15) elements of the Pareto optimal set were found on average (only occasionally was our approach able to find the 15 target elements). PAES was also able to generate 13 elements of the Pareto optimal set on average, and the NSGA II was only able to generate 8 elements on average.

For this example, both the NSGA II and PAES performed 5,000 evaluations of the fitness function. The average running time of each algorithm (over 20 runs) were the following: 2.207 seconds for the NSGA II, 0.134 seconds for PAES and only 0.042 seconds for the micro-GA.

Borges & Barbosa [2] reported that were able to find the 15 elements of the Pareto optimal set for this problem, using a population size of 100 and 5,000 evaluations of the fitness function, although no actual running times of their approach were reported.

4.5 Test Function 5

Our fifth example is a two-objective optimization problem defined by Kursawe [16]:

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

$$\text{Minimize } f_2(\boldsymbol{x}) = \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5\sin(x_i)^3 \right) \tag{11}$$

where:

$$-5 \le x_1, x_2, x_3 \le 5 \tag{12}$$

Fig. 6 shows the true Pareto front for this problem as points. The results obtained by the NSGA II, PAES and our micro-GA are also shown as points. It is worth mentioning that PAES could not eliminate some of the dominated points in the runs performed.

The parameters used for this example were: size of the external memory = 100, size of the population memory = 50, number of iterations = 3000, number of iterations of the micro-GA (to achieve nominal convergence) = 2, number of subdivisions of the adaptive grid = 25, crossover rate = 0.7, mutation rate = 0.019, percentage of non-replaceable memory = 0.3, population size (of the micro-GA) = 4, replacement cycle at every 50 iterations.

For this example, both the NSGA II and PAES performed 2,400 evaluations of the fitness function. The average running time of each algorithm (over 20 runs) were the following: 6.481 seconds for the NSGA II (with a standard deviation of 0.053712), 2.195 seconds for PAES (with a standard deviation of 0.25408319) and only 0.704 seconds for the micro-GA (with a standard deviation of 0.00692099).

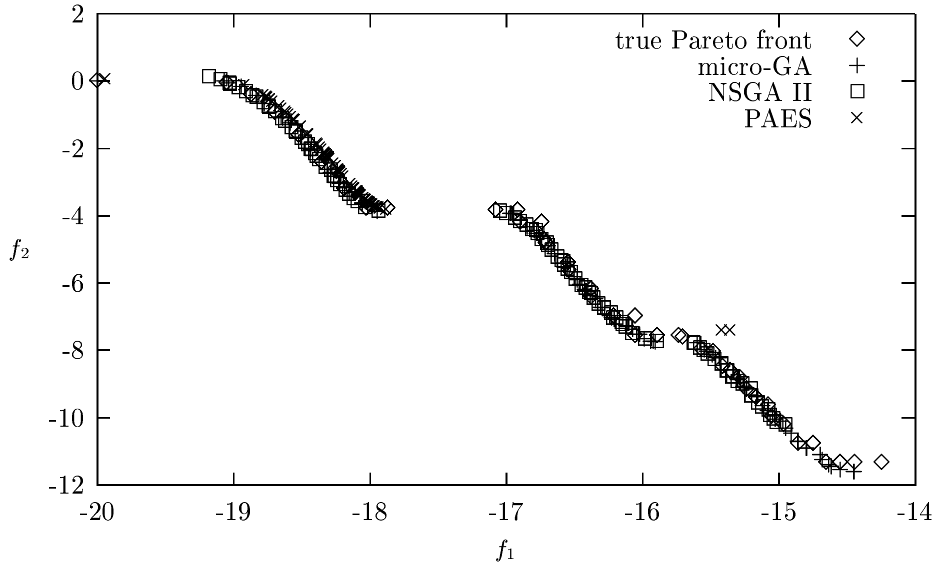

Fig. 6. Comparison of results for the fifth test function.

5 Conclusions and Future Work

We have introduced an approach that uses a GA with a very small population and a reinitialization process to generate the Pareto front of a multiobjective optimization problem. The technique has a exhibited a low computational cost when compared to the NSGA II and PAES in a few test functions. The approach uses three forms of elitism, including an external file of nondominated vectors and a refilling process that allows us to approach the true Pareto front in a successive manner. Also, we use an adaptive grid (similar to the one used by PAES [14]) that maintains diversity in an efficient way.

The approach still needs more validation (particularly, with MOPs that have more decision variables and constraints), and needs to be compared with other EMOO approaches (under similar conditions) using some of the metrics that have been proposed in the literature (see for example [22]). We only provided running times produced on a PC, but a more exhaustive comparison is obviously lacking. However, the preliminary results presented in this paper, indicate the potential of the approach.

Some other future work will be to refine part of the process, so that we can eliminate some of the additional parameters that the approach needs. Since some of them are not very critical (e.g., the number of grid subdivisions, or the amount of iterations to reach critical convergence), we could probably automatically preset them to a reasonable value so that the user does not need to provide them.

Finally, we are also interested in using this approach as a basis to develop a model of incorporation of preferences from the decision maker [4].

Acknowledgements

We thank the anonymous referees for their comments that helped us improve the contents of this paper. The first author gratefully acknowledges support from CONACyT through project 34201-A. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Maestría en Inteligencia Artificial of LANIA and the Universidad Veracruzana.

References

1. P. J. Bentley and J. P. Wakefield. Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, Part 5, pages 231–240, London, June 1997. Springer Verlag London Limited. (Presented at the 2nd On-line World Conference on Soft Computing in Design and Manufacturing (WSC2)).
2. Carlos C.H. Borges and Helio J.C. Barbosa. A Non-generational Genetic Algorithm for Multiobjective Optimization. In *2000 Congress on Evolutionary Computation*, volume 1, pages 172–179, San Diego, California, July 2000. IEEE Service Center.
3. Carlos A. Coello Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, August 1999.
4. Carlos A. Coello Coello. Handling Preferences in Evolutionary Multiobjective Optimization: A Survey. In *2000 Congress on Evolutionary Computation*, volume 1, pages 30–37, Piscataway, New Jersey, July 2000. IEEE Service Center.
5. Kalyanmoy Deb. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205–230, Fall 1999.
6. Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.
7. G. Dozier, J. Bowen, and D. Bahler. Solving small and large scale constraint satisfaction problems using a heuristic-based microgenetic algorithm. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 306–311, 1994.
8. David E. Goldberg. Sizing Populations for Serial and Parallel Genetic Algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 70–79, San Mateo, California, 1989. Morgan Kaufmann Publishers.
9. Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

10. Hisao Ishibuchi and Tadahiko Murata. Multi-Objective Genetic Local Search Algorithm. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE.
11. Andrzej Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. Technical Report RA-014/98, Institute of Computing Science, Poznan University of Technology, 1998.
12. E.G. Johnson and M.A.G. Abushagur. Micro-Genetic Algorithm Optimization Methods Applied to Dielectric Gratings. *Journal of the Optical Society of America*, 12(5):1152–1160, 1995.
13. Charles L. Karr. Air-Injected Hydrocyclone Optimization via Genetic Algorithm. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, pages 222–236. Van Nostrand Reinhold, New York, 1991.
14. Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
15. K. Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. In *SPIE Proceedings: Intelligent Control and Adaptive Systems*, pages 289–296, 1989.
16. Frank Kursawe. A variant of evolution strategies for vector optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.
17. Geoffrey T. Parks and I. Miller. Selective Breeding in a Multiobjective Genetic Algorithm. In A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature — PPSN V*, pages 250–259, Amsterdam, Holland, 1998. Springer-Verlag.
18. J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.
19. N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, fall 1994.
20. David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.
21. Fengchao Xiao and Hatsuo Yabe. 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, December 1998.
22. Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.