Multiobjective Optimization using a Micro-Genetic Algorithm

Carlos A. Coello Coello
CINVESTAV-IPN
Depto. de Ing. Elec./Secc. Computación
Av. Instituto Politécnico Nacional No. 2508
Sn. Pedro Zacatenco, México, D. F. 07300
ccoello@cs.cinvestav.mx

Gregorio Toscano Pulido
Maestría en Inteligencia Artificial
LANIA-Universidad Veracruzana
Sebastián Camacho No. 5
Xalapa, Veracruz, México 91090
gtoscano@mia.uv.mx

Abstract

In this paper, we propose a micro genetic algorithm with three forms of elitism for multiobjective optimization. We show how this relatively simple algorithm coupled with an external file and a diversity approach based on geographical distribution can generate efficiently the Pareto fronts of several difficult test functions (both constrained and unconstrained). A metric based on the average distance to the Pareto optimal set is used to compare our results against two evolutionary multiobjective optimization techniques recently proposed in the literature.

1 INTRODUCTION

Despite the considerable volume of research on evolutionary multiobjective optimization (see for example [4, 1, 16]), until recently, little emphasis had been placed on developing efficient techniques. The usual approach has been to use a ranking procedure to classify a population of individuals based on their Pareto dominance. This ranking procedure normally consumes most of the running time of an evolutionary multiobjective optimization technique[1]. Pareto ranking is $O(kM^2)$, where k is the number of objective functions and M is the size of the population. Additionally, an extra mechanism is required to preserve diversity (some form of fitness sharing [3] is normally adopted). This generally implies the use of another process that is $O(M^2)$.

Some authors have recently addressed efficiency issues in the context of evolutionary multiobjective optimization (e.g., [10, 2]). Knowing the sources of inefficiency of traditional evolutionary multiobjective optimization techniques, several researchers have focused their recent efforts on reducing the checkings for nondominance and in the development of efficient approaches to keep diversity. Regarding the first issue, the main emphasis has been on using an external file that stores nondominated vectors found during the evolutionary process. These vectors are put back into the population at later generations (this can be seen as a form of elitism in the context of multiobjective optimization [6, 18]). Regarding the second issue, the main emphasis has been on using clustering techniques [2] or approaches based on geographical positioning of individuals in an adaptive grid [10].

Also, some researchers have suggested the use of a distributed GA in which Pareto dominance is applied only to neighbors within a certain region [13]. Such sort of approach can handle the two problems previously mentioned simultaneously. The approach is efficient because Pareto dominance is applied in parallel to small groups of individuals. Diversity does not require an extra mechanism, since it naturally emerges from the distributed population. However, to take advantage of these features of the algorithm, a parallel architecture is necessary.

Our approach was to use a GA with a very small population size and a reinitialization process (a micro-GA) to solve multiobjective optimization problems of different degrees of complexity. To validate the performance of our approach, we used a metric previously defined in the literature to compare our results against two techniques that are representative of the state-of-the-art in evolutionary multiobjective optimization algorithms: the Nondominated Sorting Genetic Algorithm II (NSGA II) [2] and the Pareto Archived Evolution Strategy (PAES) [10].

[1] This is obviously considering the academic test functions that most researchers have used so far. In real-world problems, most of the computational time is normally spent evaluating the fitness functions of the problem.

2 PREVIOUS WORK

The term micro-genetic algorithm (micro-GA) refers to a small-population genetic algorithm with reinitialization. The approach was derived from some theoretical results obtained by Goldberg [5], according to which a population size of three 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 [11], 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 [11] 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). He 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 (e.g., [8, 17]). However, the work reported in this paper represents, to the best of our knowledge, the first attempt to use a micro-GA for multiobjective optimization.

Regarding similar work, we are only aware of an approach developed by Jaszkiewicz [7] in which a small population initialized from a large external memory is used for a short period of time. However, to the best of our knowledge, this approach has been used only for multiobjective combinatorial optimization. Some could also argue that the multi-membered versions of PAES can be seen as a form of micro-GA. However, the authors of PAES concluded that the addition of a population did not, in general, improve the performance of their approach, and increased the computational overhead in an important way [10].

3 DESCRIPTION OF OUR APPROACH

The way in which our technique works is illustrated in Figure 1. First, a random population is generated. This random population feeds the population memory, which is divided in two parts: a replaceable and a non-replaceable portion. 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. In contrast, the replaceable portion will experience changes after each cycle of the micro-GA.

The 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 so that we can have a mixture of randomly generated individuals (non-replaceable portion) and evolved individuals (replaceable portion).

During each cycle, the micro-GA undergoes conventional genetic operators (binary representation is used in our implementation): tournament selection, two-point crossover, uniform mutation, and elitism. After the micro-GA finishes one cycle, we choose two nondominated vectors[2] from the final population 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 it against the vectors in this external memory, then they are included there (i.e., in the external memory). This is our historical archive of nondominated vectors. All dominated vectors contained in the external memory are eliminated.

The same two vectors previously mentioned are also compared against two elements from the replaceable portion of the population memory. If either of these vectors dominates to its match in the population memory, then it replaces it. Otherwise, the vector is discarded. Over time, the replaceable part of the population memory will tend to have more nondominated vectors, some of which will be used in some of the initial populations of the micro-GA.

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

[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.

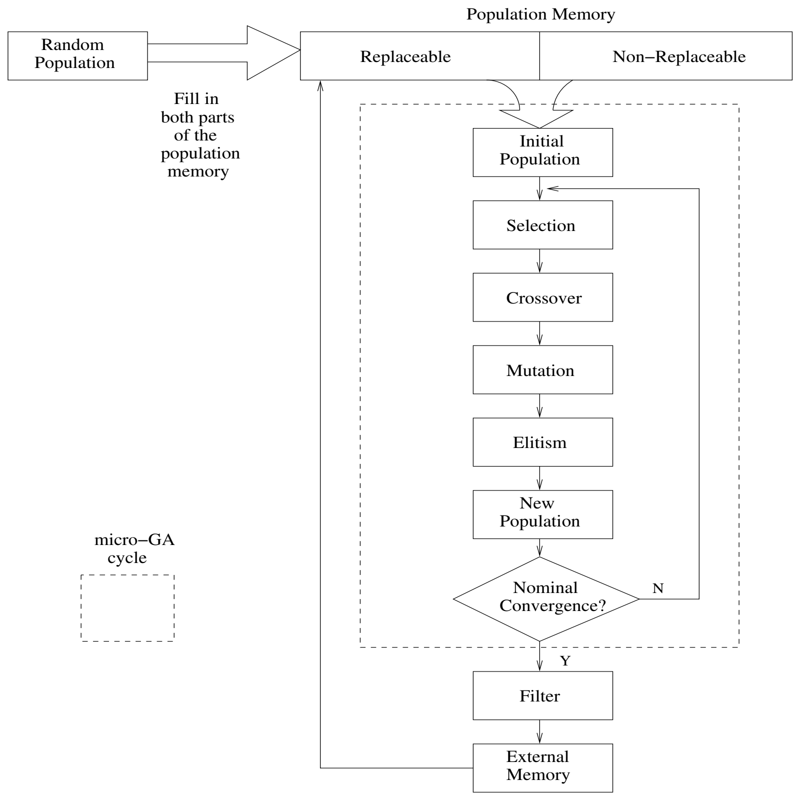

Figure 1: Diagram that illustrates the way in which our micro-GA works.

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. Nominal convergence, in our case, is defined in terms of a certain (low) number of generations (two to five in our case). 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 micro-GA, 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 & Corne [10]. 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. 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 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 its occurrence.

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. Additionally, we decided to use one of the metrics defined in objective space by Zitzler et al. [18]:

$$M_1^* = \frac{1}{\mid Y' \mid} \sum_{\mathbf{d}' \in \mathbf{Y}'} \min \; \{\parallel \mathbf{d}' - \bar{\mathbf{d}} \parallel^*; \bar{\mathbf{d}} \in \bar{\mathbf{Y}}\} \qquad (1)$$

where: $Y', \bar{Y} \subseteq Y$ are the sets of objective vectors that correspond to a set of pairwise nondominating decision vectors $X', \bar{X} \subseteq X$, respectively, and X corresponds to the decision variables of the problem. It should be obvious that M_1^* gives the average distance to the Pareto optimal set. Therefore, we should aim to minimize this value (see [18] for further details).

Since the main aim of this approach has been to increase efficiency, we additionally decided to compare running times of our micro-GA against two very fast algorithms: the NSGA II [2] and PAES[3] [10].

In the following examples, the NSGA II was run using a population size of 100, a crossover rate of 0.8, 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 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.

For constrained functions, we used a very simple approach. Whenever two individuals were compared, we checked their constraints. If both were feasible, nondominance was directly applied. If one was feasible and the other was infeasible, the feasible would dominate. If both were infeasible, then the one with the lowest amount of constraint violation would dominate the other. This same approach was used in PAES. The NSGA II has its own constraint-handling mechanism, so we did not have to implement one for it.

To allow a fair comparison of running times, all the experiments were performed on a PC with a Pentium III processor running at 650 MHz, 128 Mb of RAM and a hard drive of 15 Gbytes. Our implementation was compiled using GNU C running under Linux Red Hat release 6.2.

Several test functions were used to validate our approach, but due to space limitations, only the results corresponding to the four test functions shown in Table 1 were included in this paper. In all our experiments, our micro-GA used a crossover rate of 0.7, 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 micro-GA itself) of four individuals, and 25 subdivisions of the adaptive grid. The other parameters used are shown in Table 2. Note that the mutation rate was always $1/L$ (L = length of the chromosomic string).

Figures 2, 3, 4, and 5, show the results produced by the NSGA II, PAES and our micro-GA in the four test functions adopted. The true Pareto fronts of each problem are also shown in each figure.

Results are summarized in Table 3. In all the unconstrained test functions used, the micro-GA obtained the lowest CPU time and the lowest value of the metric M_1^*. For the constrained test functions (such as functions three and four), the NSGA II obtained the lowest value of the metric, and the micro-GA placed second. However, note that for the third test function, the micro-GA only took a third of the running time than the NSGA II and it covered most of the Pareto front of the problem unlike the other two algorithms.

5 ANALYSIS OF RESULTS

There are a few things that we can say about the observed behavior of the three algorithms compared. The NSGA II is a very good algorithm that provides elegant solutions and a good performance (in terms of CPU time). However, we have found that in some test functions the NSGA II is not able to cover properly the whole Pareto front. We believe that its exploratory capabilities could be improved, and that its main strength is its extraordinary capability to exploit a promising region of the search space, once it finds it. This last point is in fact the main weakness of our micro-GA in its current form. However, our micro-GA has compared relatively well in terms of the metric adopted and it obtained the lowest computational costs in all the test functions used. In fact, for the case of the unconstrained test functions used, the micro-GA exhibited the best overall performance. That is also an

[3] Readers interested in reproducing these experiments may download the source code of the NSGA II and PAES (original versions from their corresponding authors) from the EMOO repository located at `http://www.lania.mx/~ccoello/EMOO/EMOOsoftware.html`. The code of the micro-GA is available from the authors upon request.

Table 1: Test Functions used to validate our micro-GA

TEST FUNCTION	OBJECTIVES	SOURCE
1	$\text{Min } f_1(x) = \begin{cases} -x & \text{if } x \leq 1 \\ -2+x & \text{if } 1 < x \leq 3 \\ 4-x & \text{if } 3 < x \leq 4 \\ -4+x & \text{if } x > 4 \end{cases}$ $\text{Min } f_2(x) = (x-5)^2$ $-5 \leq x \leq 10$	[14]
2	$\text{Min } f_1(\vec{x}) = \sum_{i=1}^{n-1} \left(-10 \exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right)$ $\text{Min } f_2(\vec{x}) = \sum_{i=1}^{n} \left(\lvert x_i\rvert^{0.8} + 5\sin(x_i)^3\right)$ $-5 \leq x_1, x_2, x_3 \leq 5$	[12]
3	$\text{Max } f_1(x,y) = -x^2 + y$ $\text{Max } f_2(x,y) = \frac{1}{2}x + y + 1$ $\frac{1}{6}x + y - \frac{13}{2} \leq 0$ $\frac{1}{2}x + y - \frac{15}{2} \leq 0$ $5x + y - 30 \leq 0$	[9]
4	$\text{Max } f_1(x,y) = (x-2)^2 + (y-1)^2 + 2$ $\text{Max } f_2(x,y) = 9x - (y-1)^2$ $x^2 + y^2 - 225 \leq 0$ $x - 3y + 10 \leq 0$	[15]

Table 2: Some of the parameters used by our micro-GA for each of four test functions (TF) used to validate our approach (the other parameters were kept constant in all our experiments)

PARAMETER	TF1	TF2	TF3	TF4
number of iterations	150	3000	2500	1500
mutation rate	0.056	0.019	0.0217	0.0192
replacement cycle (iterations)	25	50	50	100

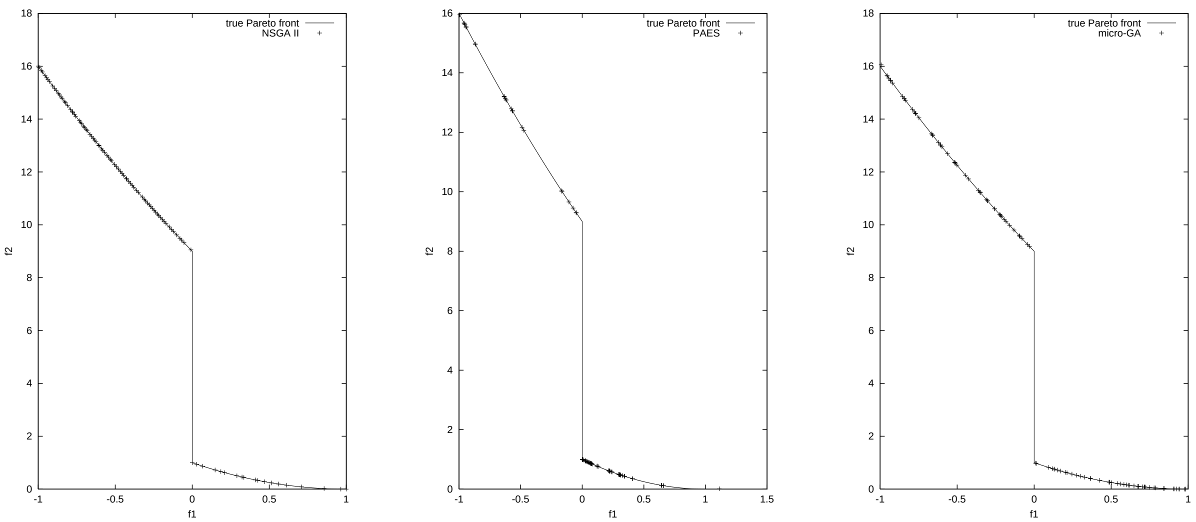

Figure 2: Pareto fronts produced by the NSGA II (left), PAES (middle), and our micro-GA (right) for the first test function

Table 3: Comparison of results. Results are reported over 20 runs

	TEST FUNCTION 1		
Perf. measure	**NSGA II**	**PAES**	**micro-GA**
average M_1^*	0.00161422	0.0675201338	0.001530162
variance of M_1^*	0.0000000200	0.0169825683	0.0000005886
average running time	0.282s	0.107s	0.017s
fitness function evals.	1,200	1,200	1,200
	TEST FUNCTION 2		
average M_1^*	0.13777005	0.42445655	0.13460185
variance of M_1^*	0.0000288497	0.0645582672	0.0000479001
average running time	6.481s	2.195s	0.704s
fitness function evals.	24,000	24,000	24,000
	TEST FUNCTION 3		
average M_1^*	0.04684924	0.399809545	0.26210439
variance of M_1^*	0.0064993289	0.2463272535	0.0622687130
average running time	6.4857s	68.937s	2.6896s
fitness function evals.	20,000	20,000	20,000
	TEST FUNCTION 4		
average M_1^*	0.2951232	5.6864776	0.4046362
variance of M_1^*	0.0015191459	384.12725166	0.0214295578
average running time	4.038s	56.6706s	3.4679s
fitness function evals.	12,000	12,000	12,000

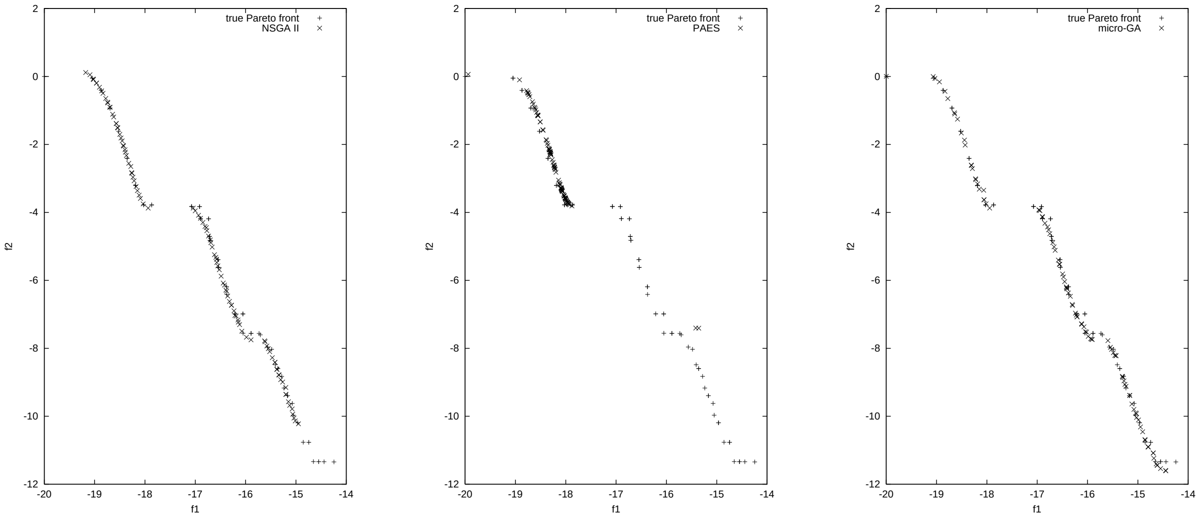

Figure 3: Pareto fronts produced by the NSGA II (left), PAES (middle), and our micro-GA (right) for the second test function

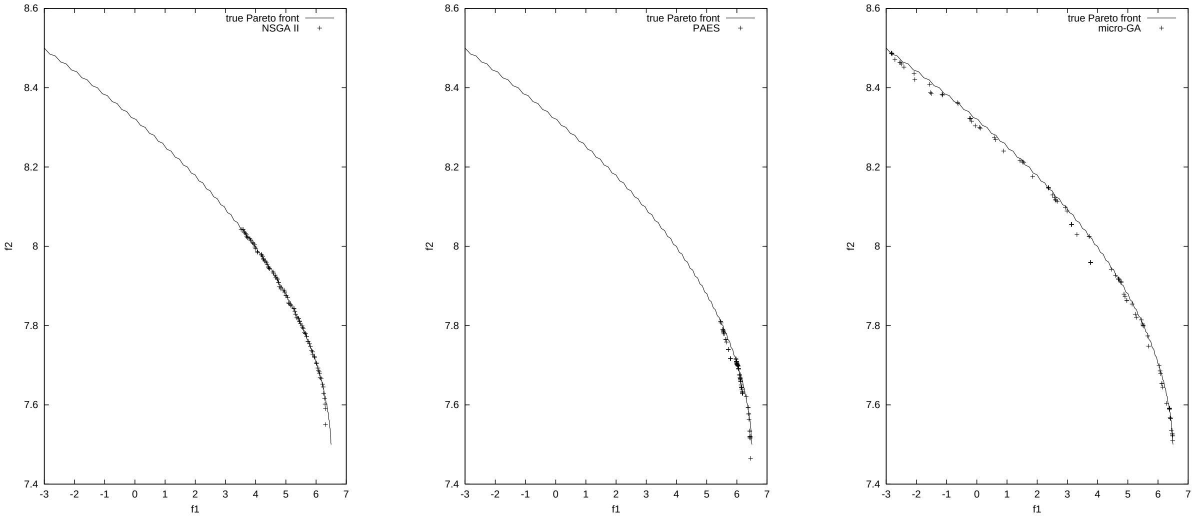

Figure 4: Pareto fronts produced by the NSGA II (left), PAES (middle), and our micro-GA (right) for the third test function

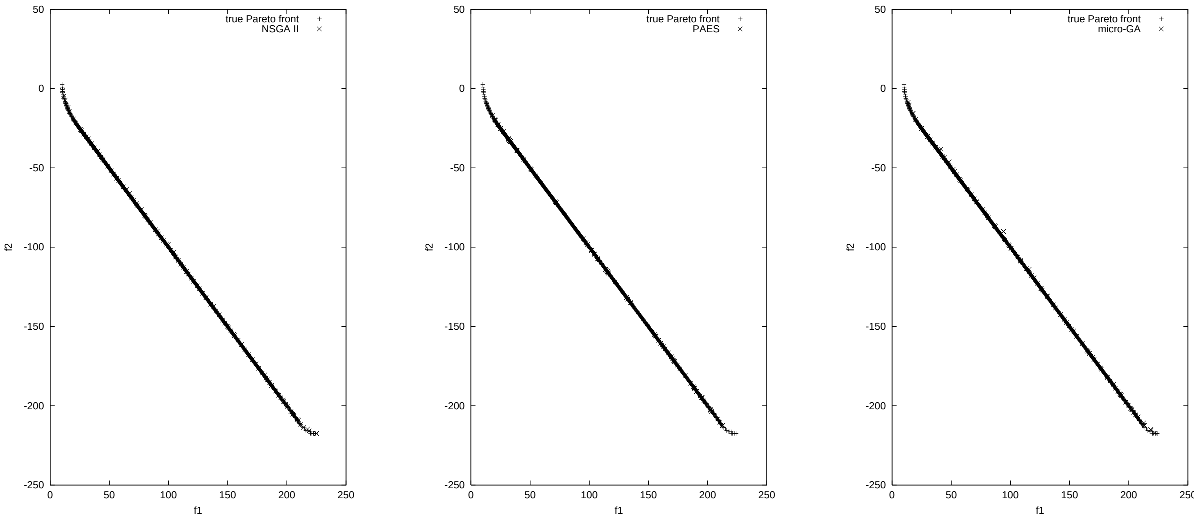

Figure 5: Pareto fronts produced by the NSGA II (left), PAES (middle), and our micro-GA (right) for the fourth test function

indicative that our approach to incorporate constraints may not be the most appropriate for the micro-GA and we are studying other techniques. Finally, PAES has difficulty with disconnected Pareto fronts, since in those cases it exhibited its worst behavior. Also, the approach used to handle constraints with PAES (the same adopted for the micro-GA) may not be the most appropriate and it probably had some impact on its performance. Nevertheless, a more comprehensive study is still necessary (using more test functions and other metrics) to derive more general conclusions.

6 THE PARAMETERS OF OUR APPROACH

Since our micro-GA 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 nondominated 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 micro-GA, 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.

7 CONCLUSIONS AND FUTURE WORK

We have proposed the use of a GA with a very small population size (only four individuals) and a reinitialization process to solve multiobjective optimization problems. Our approach has been compared against the NSGA II and PAES in several test functions. In the unconstrained test functions used, our approach has been able to converge faster (in terms of CPU time) to the true Pareto front than the two other algorithms analyzed. Also, it has performed better than them in terms of a metric previously proposed in the specialized literature. In the constrained functions, however, its performance has not been as good as that of the NSGA II, although it has been better than PAES. Also, in some cases, it produced a better distribution along the Pareto front than any of the other two algorithms analyzed.

Our initial future work will be to analyze other approaches to handle constraints in our micro-GA and to study the capabilities of our algorithm to exploit a certain promising region of the search space (the main advantage of the NSGA II over our approach). We will also perform a careful sensitivity study of the parameters of the algorithm so that we can provide more general guidelines to set them up, and we also aim to eliminate some of the parameters currently used.

Acknowledgements

The authors would like to thank the four anonymous reviewers for their valuable comments that helped them improve this paper.

The first author gratefully acknowledges support from CONACyT through project 34201-A to perform this work. 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] 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.

[2] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. 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*, pages 849–858. Springer, 2000.

[3] Kalyanmoy Deb and David E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, San Mateo, California, June 1989. George Mason University, Morgan Kaufmann Publishers.

[4] Carlos M. Fonseca and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

[5] 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.

[6] 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.

[7] Andrzej Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. Technical Report RA-014/98, Institute of Computing Science, Poznan University of Technology, 1998.

[8] 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.

[9] Hajime Kita, Yasuyuki Yabumoto, Naoki Mori, and Yoshikazu Nishikawa. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature—PPSN IV*, Lecture Notes in Computer Science, pages 504–512, Berlin, Germany, September 1996. Springer-Verlag.

[10] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[11] Kalmanje Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. In *SPIE Proceedings: Intelligent Control and Adaptive Systems*, volume 1196, pages 289–296, 1989.

[12] 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.

[13] Jon Rowe, Kevin Vinsen, and Nick Marvin. Parallel GAs for Multiobjective Functions. In Jarmo T. Alander, editor, *Proceedings of the Second Nordic Workshop on Genetic Algorithms and Their Applications (2NWGA)*, pages 61–70, Vaasa, Finland, August 1996. University of Vaasa.

[14] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms.* PhD thesis, Vanderbilt University, 1984.

[15] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, fall 1994.

[16] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.

[17] 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.

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