The Micro Genetic Algorithm 2: Towards On-Line Adaptation in Evolutionary Multiobjective Optimization

Gregorio Toscano Pulido and Carlos A. Coello Coello

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. 07300
`gtoscano@computacion.cs.cinvestav.mx`
`ccoello@cs.cinvestav.mx`

Abstract. In this paper, we deal with an important issue generally omitted in the current literature on evolutionary multiobjective optimization: on-line adaptation. We propose a revised version of our micro-GA for multiobjective optimization which does not require any parameter fine-tuning. Furthermore, we introduce in this paper a dynamic selection scheme through which our algorithm decides which is the "best" crossover operator to be used at any given time. Such a scheme has helped to improve the performance of the new version of the algorithm which is called the micro-GA2 (μGA^2). The new approach is validated using several test function and metrics taken from the specialized literature and it is compared to the NSGA-II and PAES.

1 Introduction

One of the research topics that has been only scarcely covered in the current literature on evolutionary multiobjective optimization has been (on-line or self-) adaptation [4]. This is an important issue since evolutionary multiobjective optimization techniques normally require more parameters than a traditional evolutionary algorithm (e.g., a niche radius or sharing threshold). In some of our recent research, we have emphasized the importance of designing computationally efficient evolutionary multiobjective optimization approaches. One of our proposals in this direction was the micro-GA for multiobjective optimization, which uses a very small internal population (four individuals) combined with three forms of elitism and a reinitialization process [3]. The micro-GA has been found to be highly competitive with respect to other techniques that are representative of the state-of-the-art in the area (namely, the Nondominated Sorting Genetic Algorithm or NSGA-II [6] and the Pareto Archived Evolution Strategy or PAES [8]). However, its main drawback is that the micro-GA requires several parameters whose fine-tuning may not be intuitive for a newcomer.

This paper presents a revised version of the micro-GA for multiobjective optimization [3] which does not require any parameters fine-tuning from the user. The new approach, called the micro-GA2 (μGA^2), tends to perform better than our original micro-GA without needing any non-intuitive parameters. To validate our proposal, we compare it against PAES [8], the NSGA-II [6] and our original micro-GA [3] using several test functions of high degree of difficulty which have been recently proposed in the specialized literature.

2 Related Work

There have been very few attempts in the literature to produce an evolutionary multiobjective optimization technique that adapts its parameters during the evolutionary process and that, therefore, does not require any fine-tuning from the user. One of the earliest attempts to incorporate self-adaptive mechanisms in evolutionary multiobjective optimization was Kursawe's proposal of providing individuals with a set of step sizes for each objective function such that his multiobjective evolution strategy could deal with a dynamic environment [10]. Laumanns et al. [11] showed that a standard self-adaptive evolution strategy had problems to converge to the true Pareto set of a multiobjective optimization problem and proposed alternative self-adaptation mechanisms that, however, were applied only to an aggregating fitness function. Tan et al. [13] proposed the incrementing multiobjective evolutionary algorithm (IMOEA), which uses a dynamic population size that adapts based on the tradeoffs produced so far and the desired population distribution density. The IMOEA relies on a measure of convergence based on population domination and progress ratio [14]. The IMOEA also uses dynamic niches (i.e., no sharing factor needs to be defined). Another interesting proposal is the idea of Büche et al. [2] of using self-organizing maps of Kohonen [9] to adapt the mutation step size of an evolutionary multiobjective optimization algorithm. The authors also define a recombination operator using self-organizing maps (something similar to intermediate recombination). Abbass [1] recently proposed a differential evolution algorithm used to solve multiobjective problems that self-adapts its crossover and mutation rates. Zhu and Leung [16] proposed an asynchronous self-adjustable island genetic algorithm in which certain information about the current search status of each island in a parallel evolutionary algorithm is used to focus the search effort into non-overlapping regions.

The approach proposed in this paper introduces on-line adaptation into a genetic algorithm (which uses Pareto ranking and an external memory) in order to make unncessary to fine tune its parameters. Unlike the proposals previously discussed, our approach is mainly focused on performing an appropriate exploration and exploitation of the search space relying on very simple information and statistical measures obtained from the evolutionary process itself.

3 The μGA^2

Since this paper presents a variation of the micro-GA reported in [3], we consider convenient to describe our original proposal first. The way in which our micro-GA works is the following: 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

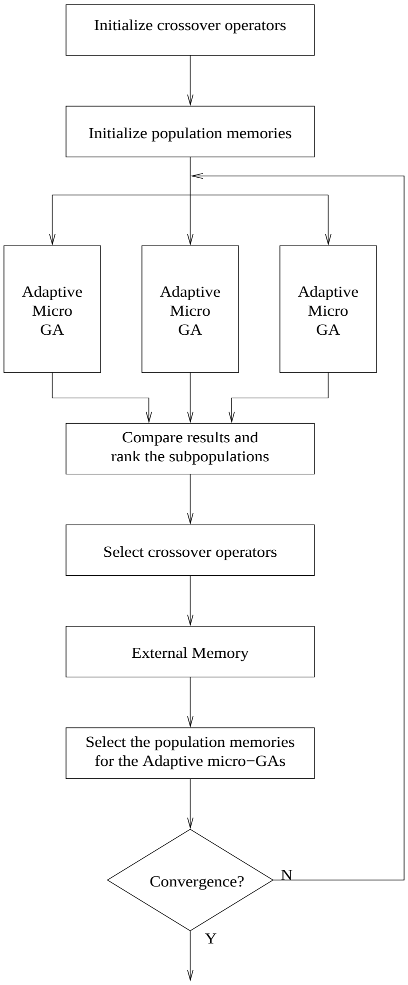
Initialize crossover operators
Initialize population memories
Adaptive Micro GA
Adaptive Micro GA
Adaptive Micro GA
Compare results and rank the subpopulations
Select crossover operators
External Memory
Select the population memories for the Adaptive micro−GAs
Convergence?
N
Y

Fig. 1. Diagram that illustrates the way in which our μGA^2 works.

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. After the micro-GA finishes one cycle, we choose two non-dominated vectors 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 non-dominated 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 non-dominated 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 non-dominated vectors, some of which will be used in some of the initial populations of the micro-GA.

The main difference between the μGA^2 and its ancestor is that we now provide on-line adaptation mechanisms. The way of which μGA^2 works is illustrated in Figure 1. Since one of the main features of the new approach is the use of a parallel strategy to adapt the crossover operator (i.e., we have several micro-GAs which are executed in parallel), we will start by describing this mechanism. First, the initial crossover operator to be used by each micro-GA is selected. The three crossover operators available in our approach are: 1) SBX, 2) two-point crossover, and 3) a crossover operator proposed by us. The behavior of the crossover operator that we proposed depends on the distance between each variable of the corresponding parents: if the variables are closer than the mean variance of each variable, then intermediate crossover is performed; otherwise, a recombination that emphasizes solutions around the parents is applied. These crossover operators were selected because they exhibited the best overall performance in an extensive set of experiments that we conducted.

Once the crossover operator has been selected, the population memories of the internal micro-GAs are randomly generated. Then, all the internal micro-GAs are executed, each one using one of the crossover operators available (this is a deterministic process). The nondominated vectors found by each micro-GA are compared against each other and we rank the contribution of each crossover operator with respect to its effectiveness to produce nondominated vectors. At this point, the crossover operator which exhibits the worst performance is replaced by the one with the best performance. The external memory stores the globally nondominated solutions, and we fill the new population memories (of every internal micro-GA) using this external memory. The new external memories of the micro-GA are identical to this external memory.

When all these processes are completed, we check if the algorithm has converged. For this sake, we assume that convergence has been reached when none of the internal micro-GAs can improve the solutions previously reached (see below for details). The rationale here is that if we have not found new solutions in a certain (reasonably large) amount of time, it is fruitless to continue the search.

The μGA^2 works in two stages: the first one starts with a conventional evolutionary process and it concludes when the external memory of each slave process is full or when at least one slave has reached convergence (as assumed in the previous paragraph). We finish the second stage when global convergence (i.e., when all of the slaves have converged) is reached.

An interesting aspect of the μGA^2 is that it attempts to balance between exploration and exploitation by changing the priorities of the genetic operators. This is done during each of the two stages previously described. During the first stage, we emphasize exploration and during the second we emphasize exploitation, which are performed in the following way:

- **Exploration stage**: At this stage, mutation has more importance than crossover so that we can locate the most promising regions of the search space. We use at this point a low crossover rate and the mutation operator is the main responsible of directing the search. We also decrease the nominal convergence (i.e., the internal cycle of the micro-GA), since we are not interested in recombining solutions at this point.
- **Exploitation stage**: At this stage, the crossover operator has more importance and therefore nominal convergence is increased to reach better results.

Since the main drawback of the micro-GA is that it requires several additional parameters which have to be fine-tuned by hand [3], the main goal of the μGA^2 was the elimination of these parameters. With this goal in mind, we divided the parameters of the micro-GA into two groups: parameters which cannot be adapted on-line and parameters that can be adapted. The first class is composed by those parameters that depend on the problem characteristics and it includes: bounds for the decision variables, number of decision variables, number of objectives, and number of nondominated vectors that we aim to find (this defines the size of the external memory which will be called T_{pareto}).

For those parameters that can be adapted, we followed a rationale based on their possible dependences. Certain parameters such as the mutation rate can be easily fixed to one divided by the number of genes (or decision variables). After a careful analysis of the parameters of the micro-GA susceptible of adaptation, we decided the following:

- **Crossover rate**: It is important that it behaves in such a way that we can explore more at the beginning of the evolutionary process and we can exploit more at later stages of the search. In the first stage of the μGA^2 we use only a 50% for the crossover rate, while during the exploitation stage, we use a 100% for the crossover rate. As indicated before, we adapt the crossover operator using a parallel strategy.
- **Size of the population memory**: The size of the population was set to $T_{pareto} \div 2$.
- **Percentage of non-replaceable memory**: Since the μGA^2 is a parallel algorithm, we decided to decrease the percentage of the non-replaceable memory to a 10% (with respect to the 30% used in the original micro-GA [3]) of the size of the population memory.

- **Total number of iterations**: This refers to the external cycle of the micro-GA [3]. This parameter is set such that we finish after the external memory has been replaced $T_{pareto} \times 2$ times without having any dominated individual nor any replacement of the limits of the adaptive grid.
- **Replacement cycle**: We replace the replaceable memory whenever T_{pareto} individuals had been evaluated. Also note that when the replaceable memory is refreshed, we also refresh the non-replaceable memory, but using randomly generated individuals (as it was done at the begining of the execution of the algorithm).
- **Number of subdivisions of the adaptive grid**: The number of subdivisions is treated with respect to the number of individuals desired per region. This value is set such that it never exceeds (on average) three individuals per region and is never less than 1.5 individuals per region. Thus, the number of subdivisions (and therefore the number of individuals per hypercube) is either increased or decreased in consequence.

The constraint-handling approach of the original micro-GA was kept intact (see [3] for details).

4 Metrics Adopted

In order to give a numerical comparison of our approach, we adopted three metrics: generational distance [15], error ratio [14] and spacing [12]. The description and mathematical representation of each metric are shown below.

1. **Generational Distance (GD)**: The concept of generational distance was introduced by Van Veldhuizen & Lamont [15] 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:
$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{1}$$
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.
2. **Error Ratio (ER)**: This metric was proposed by Van Veldhuizen [14] to indicate the percentage of solutions (from the nondominated vectors found so far) that are not members of the true Pareto optimal set:
$$ER = \frac{\sum_{i=1}^{n} e_i}{n}, \tag{2}$$
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.

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 [12] proposed such a metric measuring the range (distance) variance of neighboring vectors in the nondominated vectors found so far. This metric is defined as:

$$S \triangleq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d}-d_i)^2}\ , \tag{3}$$

   where $d_i = \min_j(\mid f_1^i(\boldsymbol{x}) - f_1^j(\boldsymbol{x}) \mid + \mid f_2^i(\boldsymbol{x}) - f_2^j(\boldsymbol{x}) \mid), i, j = 1, \ldots, n$, $\overline{d}$ 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.

5 Test Functions and Numerical 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 and metric-based comparison of the quality of the solutions produced by the μGA^2. We also compared our results with respect to the NSGA-II [6], with respect to the PAES [8] and with respect to our original micro-GA [3]. In the following examples, the NSGA-II 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. 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. 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, 25 subdivisions of the adaptive grid, and a mutation rate of $1/L$ ($L =$ length of the chromosomic string).

The number of fitness function evaluations for the original micro-GA, the NSGA-II and PAES was the closest value to the average number of fitness function evaluations obtained from performing 20 runs with the μGA^2.

5.1 Test Function 1

Our first example is a n-objective, n-variable test function proposed by *Deb et al.* [7]:

$$
\begin{array}{rl}
\text{Minimize} & f_1(\mathbf{x}) = x_1, \\
\vdots & \vdots \\
\text{Minimize} & f_{M-1}(\mathbf{x}) = x_{M-1}, \\
\text{Minimize} & f_M(\mathbf{x}) = (1 + g(\mathbf{x}_M))h(f_1, f_2, \dots, F_{M-1}, g), \\
\text{where} & g(\mathbf{x}_M) = 1 + \frac{9}{|\mathbf{x}_M|} \sum_{x_i \in \mathbf{x}_M} xi, \\
 & h = M - \sum_{i=1}^{M-1} \left[\frac{f_i}{1+g} (1 + \sin(3\pi f_i)) \right], \\
\text{subject to} & 0 \le x_i \le, \text{ for } i = 1, 2.., n,
\end{array}
\tag{4}
$$

This test function has 2^{M-1} disconnected Pareto-optimal regions in the search space. *Deb et al.* [7] propose to use 22 variables to make it more challenging. That is precisely the number of decision variables that were adopted for our experiments. Results for the first test function are summarized in Table 1. Figures 2 and 3 show the average behavior of each algorithm with respect to the generational distance metric. From the results shown in Table 1, it can be seen that in the first test function, the μGA^2 had the best performance with respect to generational distance and spacing and it placed second (after the NSGA-II) with respect to the error ratio.

Table 1. Results obtained in the first test function (DTLZ6) by the μGA^2, the NSGA-II, PAES and the original micro-GA. We show in **boldface** the best values for each metric.

		μ**GA**^{2}		
Perf. measure	**Iterations**	**GD**	**ER**	**SP**
Average	20382	**0.003561016**	0.171	**0.07382801**
Best	16954	0.00304658	0.1	0.0598198
Worst	24394	0.00440405	0.25	0.0886338
Std. Dev.	2019.793840	0.000372	0.04290	0.007385
		PAES		
Average	20382	0.0161938745	0.49492855	0.125067925
Best	20382	0.00260934	0.2	0.0770419
Worst	20382	0.109795	0.75	0.258494
Std. Dev.	0	0.023217	0.1603101	0.049333
		NSGA-II		
Average	20100	0.003606146	**0.115**	0.077738055
Best	20100	0.00281355	0.07	0.039322
Worst	20100	0.0052915	0.16	0.0940669
Std. Dev.	0	0.000634	0.030174	0.012038
		micro-GA		
Average	20376	0.8760464	1.0425015	0.97022395
Best	20376	0.381188	1.025	0.232188
Worst	20376	1.66206	1.07143	3.4051
Std. Dev.	0	0.3524874	0.01302171	1.0298174

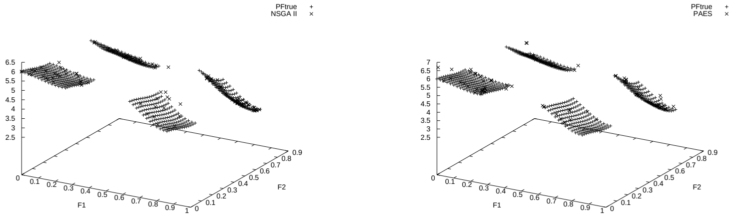

Fig. 2. Pareto fronts produced by the NSGA II (left) and PAES (right) for the first test function.

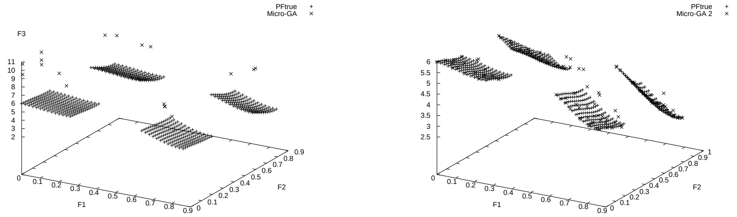

Fig. 3. Pareto fronts produced by the micro-GA (left) and the μGA^2 (right) for the first test function.

5.2 Test Function 2

Our second example is an n-objective, n-variable test function proposed by *Deb et al.* [7]:

$$\begin{array}{l} \text{Minimize } f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \ldots \cos(x_{M-1}\pi/2), \\ \text{Minimize } f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \ldots \sin(x_{M-1}\pi/2), \\ \quad\vdots \qquad \vdots \\ \text{Minimize } f_M(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \sin(x_1\pi/2), \\ \text{subject to } 0 \leq x_i \leq, \ \text{for } i = 1, 2.., n, \\ \quad \text{where} \quad g(\mathbf{x}_M) = \sum_{xi \in \mathbf{x}_M} (x_i - 0.5)^2. \\ \text{subject to } 0 \leq xi \leq 1, \ for \ i = 1, 2, \ldots, n. \end{array} \qquad (5)$$

Results for the second test function are summarized in Table 2. Figures 4 and 5 show the average behavior of each algorithm with respect to the generational distance metric. It is interesting to observe in Table 2, that for the second test function, PAES had the best performance with respect to generational distance and spacing, and that the original micro-GA produced better results than the μGA^2 for all of the metrics considered. In fact, the μGA^2 exhibited the worst performance with respect to spacing although it was not the worst of all with respect to error ratio and generational distance. This poor performance is mainly due to the fact that the μGA^2 could not get rid of some points that did not lie on the true Pareto front of the problem.

Table 2. Results obtained in the second test function (DTLZ2) by the μGA^2, the NSGA-II, PAES and the original micro-GA. We show in **boldface** the best values for each metric.

		$\mathbf{\mu GA^2}$		
Perf. measure	**Iterations**	**GD**	**ER**	**SP**
Average	28035.6	0.06768955	0.4782749	0.12874763
Best	22662	0.0466557	0.07	0.0864405
Worst	38626	0.0808061	0.87	0.258394
Std. Dev.	3935.705252	0.009093	0.307199	0.036003
		PAES		
Average	28035	**0.00188515945**	0.661	**0.07808585**
Best	28035	0.000960599	0.49	0.0645913
Worst	28035	0.00372801	0.83	0.0969163
Std. Dev.	0	0.000729	0.099994	0.007436
		NSGA-II		
Average	28100	0.071364695	0.595	0.086525355
Best	28100	0.0012885	0.37	0.0600044
Worst	28100	0.146352	0.88	0.151025
Std. Dev.	0	0.039513	0.12279	0.026498
		micro-GA		
Average	28032	0.0079807725	**0.43616665**	0.07967385
Best	28032	0.00196516	0.25	0.0469776
Worst	28032	0.0139376	0.82	0.1319
Std. Dev.	0	0.0035755	0.1511332	0.0271390

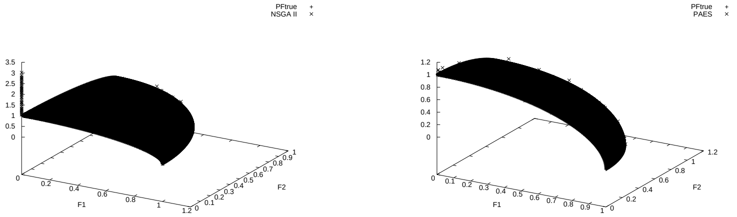

Fig. 4. Pareto fronts produced by the NSGA II (left) and PAES (right) for the second test function.

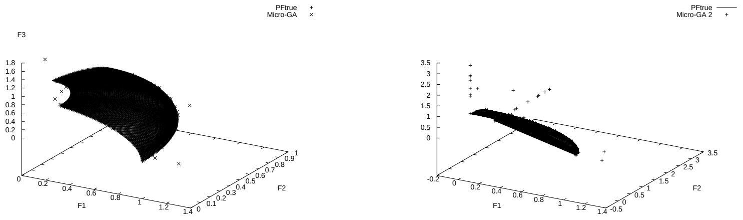

Fig. 5. Pareto fronts produced by the micro-GA (left) and the μGA^2 (right) for the second test function.

5.3 Test Function 3

The third example is a bi-objective optimization problem proposed by Deb [5]:

$$\begin{array}{l}\text{Minimize } f_1(x_1,x_2)=x_1\\ \text{Minimize } f_2(x_1,x_2)=g(x_1,x_2)\cdot h(x_1,x_2)\end{array} \tag{6}$$

where:

$$g(x_1,x_2)=11+x_2^2-10\cdot\cos(2\pi x_2) \tag{7}$$

$$h(x_1,x_2)=\begin{cases}1-\sqrt{\frac{f_1(x_1,x_2}{g(x_1,x_2}} & \text{if } f_1(x_1,x_2)\leq g(x_1,x_2)\\ 0 & \text{in other case}\end{cases} \tag{8}$$

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

Results for the third test function are summarized in Table 3. Figures 6 and 7 show the average behavior of each algorithm with respect to the generational distance metric. Table 3 shows that in the third test function, the μGA^2 produced the best results for both the generational distance and the error ratio metrics. With respect to spacing, it placed second (after the NSGA-II).

Table 3. Results obtained in the third test function (Deb) by the μGA^2, the NSGA-II, PAES and the original micro-GA. We show in **boldface** the best values for each metric.

		μGA^2		
Perf. measure	**Iterations**	**GD**	**ER**	**SP**
Average	9171.8	**0.00016127085**	**0.115**	0.0088751215
Best	6186	0.000102157	0	0.00721023
Worst	13826	0.000218467	0.32	0.0100066
Std. Dev.	0.081917	1956.912487	4.252223-05	0.000822
		PAES		
Average	9171	0.4651514	0.70408545	5.46232964
Best	9171	0.242424	0.0252054	0.0829736
Worst	9171	1	7.97044	64.8108
Std. Dev.	0	0.180424	2.012568	16.406210
		NSGA-II		
Average	9100	0.0002118179	0.2105	**0.0079981215**
Best	9100	0.000155758	0.01	0.00646298
Worst	9100	0.000282185	0.74	0.0089998
Std. Dev.	0	3.577123-05	0.224252	0.000594
		micro-GA		
Average	91068	0.0556739552	0.162	0.281928729
Best	91068	0.000159071	0.05	0.00637886
Worst	91068	0.465348	0.31	1.22778
Std. Dev.	0	0.1079727	0.0796439	0.3647516

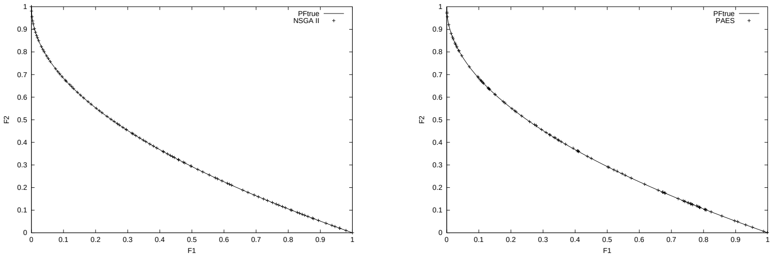

Fig. 6. Pareto fronts produced by the NSGA II (left) and PAES (right) for the third test function.

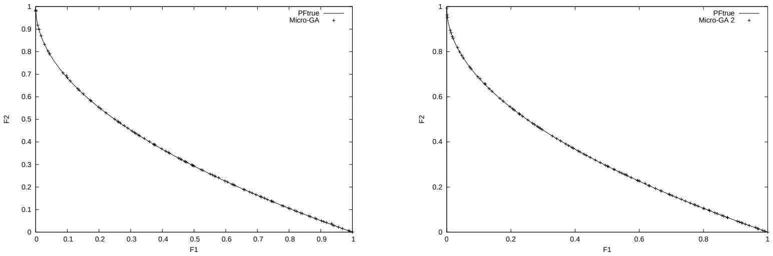

Fig. 7. Pareto fronts produced by the micro-GA (left) and the μGA^2 (right) for the third test function.

5.4 Test Function 4

Our fourth example is a bi-objective test function proposed by *Kursawe* [10]:

$$\text{Minimize } f_1(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left(-10 \exp \left(-0.2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right) \tag{9}$$

$$\text{Minimize } f_2(\boldsymbol{x}) = \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5 \sin(x_i)^3 \right) \tag{10}$$

where:

$$-5 \leq x_1, x_2, x_3 \leq 5 \tag{11}$$

Our fourth test function has a discontinuous Pareto optimal set and a discontinuous Pareto front. The main difficulty of this test function resides in the fact that it has a fairly large search space.

Results for our fourth test function are summarized in Table 4. Figures 8 and 9 show the average behavior of each algorithm with respect to the generational distance metric. Table 4 shows that in the fourth test function, the μGA^2 produced better results than our original micro-GA for all the metrics, but it had poorer values than the NSGA-II, which beats all of the other approaches in this problem with respect to all the metrics considered. Note, however, that the NSGA-II does not completely cover the true Pareto Front (see Figure 8, left graph). In contrast, the μGA^2 has a wider spread of nondominated solutions, but with a poorer distribution than the NSGA-II.

Table 4. Results obtained in the fourth test function (Kursawe) by the μGA^2, the NSGA-II, PAES and the original micro-GA. We show in **boldface** the best values for each metric.

		$\mu\mathbf{GA^2}$		
Perf. measure	**Iterations**	**GD**	**ER**	**SP**
Average	12521.2	0.005006659	0.3505	0.11070785
Best	9350	0.00326891	0.2	0.103748
Worst	16262	0.0176805	0.51	0.131396
Std. Dev.	2075.483192	0.003133	0.080031	0.005867
		PAES		
Average	12521	0.1963858	1.01001	0.30378948
Best	12521	0.139281	1.01	0.00477854
Worst	12521	0.383218	1.0102	1.24913
Std. Dev.	0	0.061842	4.4721-05	0.355380
		NSGA-II		
Average	12100	**0.0036544415**	**0.2765**	**0.06236726**
Best	12100	0.00311988	0.21	0.0536878
Worst	12100	0.00541468	0.44	0.0815969
Std. Dev.	0	0.000506	0.052342	0.007113
		micro-GA		
Average	12520	0.0065217035	0.59144695	0.130839305
Best	12520	0.00357395	0.39	0.0636691
Worst	12520	0.00969116	0.73	0.180439
Std. Dev.	0	0.00171691	0.08471597	0.02968868

6 Conclusions and Future Work

We have proposed an on-line adaptation scheme that allows the use of a micro-GA designed for multiobjective optimization without requiring the definition of any non-intuitive parameters. The proposed approach obtains information from the evolutionary process to guide the search efficiently. Among other things, our scheme decides on the most appropriate crossover operator and it switches between an exploration and an exploitation stage by changing the importance of the crossover and the mutation operators. We also define a criterion that allows to stop the execution of the algorithm when the search seems to be fruitless and therefore, the definition of a maximum number of generations is no longer necessary.

The approach has been validated with several test functions from which four were included in this paper. We compared our results with respect to our original micro-GA and with respect to PAES and the NSGA-II using three metrics: generational distance, error ratio and spacing. Our preliminary results indicate that although our approach does

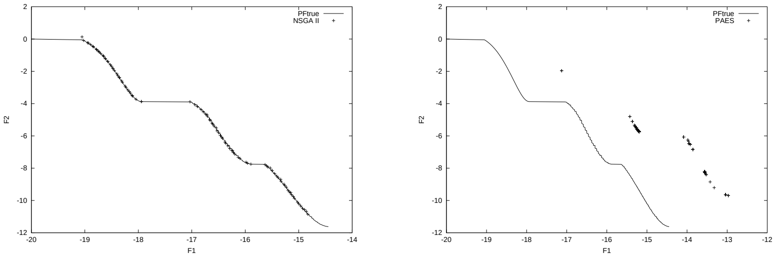

Fig. 8. Pareto fronts produced by the NSGA-II (left) and PAES (right) for the fourth test function.

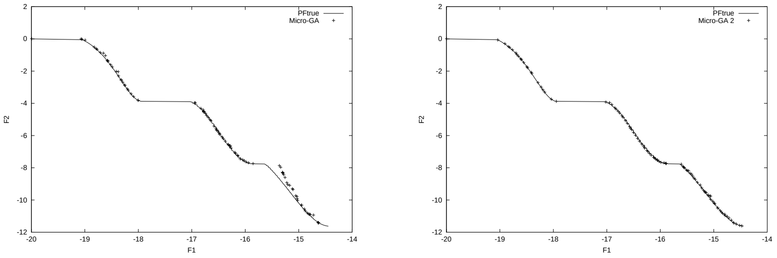

Fig. 9. Pareto fronts produced by the micro-GA (left) and the μGA^2 (right) for the fourth test function.

not always beat the other approaches, it remains competitive, and it normally improves on the results generated by the micro-GA without the need of defining any parameters.

As part of our future work, we want to experiment with spatial data structures to make more efficient the storage and retrieval of nondominated vectors from the external memory. We are also working on the development of a mechanism that allows to reduce the number of fitness function evaluations required to approximate the true Pareto front of a problem.

7 Acknowledgments

This paper is representative of the research performed by the Evolutionary Computation Group at CINVESTAV-IPN (EVOCINV). The first author acknowledges support from the mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through project number 34201-A. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Section of the Electrical Engineering Department at CINVESTAV-IPN.

References

1. Hussein A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 831–836, Piscataway, New Jersey, May 2002. IEEE Service Center.

2. Dirk Büche, Gianfranco Guidati, Peter Stoll, and Petros Kourmoursakos. Self-Organizing Maps for Pareto Optimization of Airfoils. In Juan Julián Merelo Guervós et al., editor, *Parallel Problem Solving from Nature—PPSN VII*, pages 122–131, Granada, Spain, September 2002. Springer-Verlag. Lecture Notes in Computer Science No. 2439.
3. Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. In Lee Spector et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.
4. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.
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, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.
7. Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 825–830, Piscataway, New Jersey, May 2002. IEEE Service Center.
8. Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
9. Teuvo Kohonen, T.S. Huang, and M.R. Schroeder, editors. *Self-Organizing Maps*. Springer-Verlag, 2001.
10. 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.
11. Marco Laumanns, Günter Rudolph, and Hans-Paul Schwefel. Mutation Control and Convergence in Evolutionary Multi-Objective Optimization. In *Proceedings of the 7th International Mendel Conference on Soft Computing (MENDEL 2001)*, Brno, Czech Republic, June 2001.
12. Jason R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.
13. K.C. Tan, T.H. Lee, and E.F. Khor. Evolutionary Algorithms with Dynamic Population Size and Local Exploration for Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 5(6):565–588, December 2001.
14. David A. Van Veldhuizen. *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, May 1999.
15. David A. Van Veldhuizen and Gary B. Lamont. 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, 1998.
16. Zhong-Yao Zhu and Kwong-Sak Leung. Asynchronous Self-Adjustable Island Genetic Algorithm for Multi-Objective Optimization Problems. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 837–842, Piscataway, New Jersey, May 2002. IEEE Service Center.