PSFGA: A Parallel Genetic Algorithm for Multiobjective Optimization

Francisco de Toro
University of Huelva (Spain)
ftoro@uhu.es

Julio Ortega; Javier Fernández, Antonio Díaz
University of Granada (Spain)
{julio,javier,afdiaz}@atc.ugr.es

Abstract

This paper presents the Parallel Single Front Genetic Algorithm (PSFGA), a parallel Pareto-based algorithm for multiobjective optimization problems based on an evolutionary procedure. In this procedure, a population of solutions is sorted with respect to the values of the objective functions and partitioned into subpopulations which are distributed among the processors. Each processor applies a sequential multiobjective genetic algorithm that we have devised (called Single Front Genetic Algorithm, SFGA) to its subpopulation. Experimental results are provided comparing PSFGA with previously proposed multiobjective evolutionary algorithms.

1. Introduction

Most real-world engineering optimization problems are multiobjective in nature, since they normally have several (usually conflicting) objectives that must be satisfied at the same time. These problems are known as MOP (*Multiobjective Optimization Problems*) **[Coello98]** in contrast with SOP (*Single-objective optimization problems*). The notion of *optimum* has to be re-defined in this context and instead of aiming to find a single solution, a procedure for solving MOP should determine a set of good compromises or *trade-off* solutions, generally known as *Pareto optimal solutions* from which the decision maker will select one. These solutions are optimal in the wider sense that no other solution in the search space is superior when all objectives are considered. Pareto optimal solutions form *the Pareto frontier* in a k-dimensional objective space, where k is the number of the objectives in the optimization problem.

Evolutionary Algorithms (EAs) have the potential to finding multiple Pareto optimal solutions in a single run and have been widely used in this area **[Ishibuchi96] [Cunha97] [Valenzuela97] [Fonseca98b] [Parks98].** A good Multiobjective Optimization Evolutionary Algorithm (MOEA) should achieve the following goals:

1. The distance between the set of solutions found by the MOEA and the Pareto frontier should be minimized.

2. The solutions found by the MOEA should present a distribution that provides a good description of the Pareto frontier.

3. The spread of the solutions obtained should be maximized (for each objective a wide range of values should be covered by the set of solutions found).

Parallel computing has been widely applied to EAs **[Cantu97]**. In the case of MOEAs, parallelism has been successfully applied to reduce the time needed to evaluate the objective functions in sequential EAs, but little insight has been gained into the performance of a parallel MOEA where the population is distributed among the processors. In that sense, the work of some authors i.e. **[Quagliarella00]** shows that sometimes there are no advantages with respect to the single population model. One of the reasons for this is that Pareto-dominance comparisons should be made considering the whole population , thus reducing the efficiency of the parallel procedure.

In this paper, Section 2 introduces the MOPs, while Section 3 and Section 4 present some ideass about the application of evolutionary techniques, and their corresponding parallel strategies previously proposed for MOPs. Section 5 presents the EAs proposed in this paper to solve MOPs, namely: the multiobjective evolutionary algorithms SFGA and PSFGA. Finally, experimental results and concluding remarks are summarized in Sections 6 and 7 respectively.

2. Statement of the problem

A multiobjective optimization problem (MOP) can be defined **[Coello98]** as one of finding a vector of *decision variables* which satisfies a set of constraints and optimizes a vector function whose elements represent the objectives. These functions form a mathematical description of performance criteria, and are usually in conflict with each other.

The problem can be formally stated as finding the vector $\mathbf{x^*} = [x_1^*, x_2^*, \ldots, x_n^*]$ which satisfies the *m* inequality constraints

$$g_i(\mathbf{x}) \geq 0 \qquad i=1,2,\ldots,m \qquad (1)$$

the *p* equality constraints

$$h_i(x) = 0 \qquad i=1,2,\ldots,p \qquad (2)$$

and optimizes the vector function

Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (EUROMICRO-PDP'02)
1066-6192/02 $17.00 © 2002 **IEEE**

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_k(\boldsymbol{x})]^T \qquad (3)$$

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

The constraints given by (1) and (2) define the *feasible region* Φ: any point **x** in Φ is a *feasible solution.*

The vector function f (**x**) is one that maps the set Φ in the set χ of all possible values of the objective functions. The k components of the vector **f** (**x**) represent the *non-commensurable criteria* to be considered. The constraints g_i (**x**) and h_i (x) represent the restriction imposed on the decision variables. The vector **x*** will be reserved to denote the optimal solutions (normally there will be more than one).

The meaning of *optimum* is not well defined in this context, since in these problems it is difficult to have an **x*** where all the components of f_i (**x**) have a minimum in Φ. Thus, a point $\mathbf{x}^* \in \Phi$ is defined as *Pareto Optimal* if $\forall \mathbf{x} \in \Phi, \forall i := 1..k$:

$$(f_i(\mathbf{x}) = f_i(\mathbf{x}^*)) \quad or \quad f_i(\mathbf{x}^*) < f_i(\mathbf{x}) \qquad (4)$$

The inequality equation in (4) must be fulfilled by at least one component i.

This means that **x*** is Pareto optimal if there exists no feasible vector **x** which would decrease one criterion without causing a simultaneous increase in at least one of the others. The notion of Pareto optimum almost always gives, not a single solution, but rather a set of solutions called non-inferior or *non-dominated* solutions (Figure1). This set of non-dominated solutions is known as the *Pareto front*. As, in general, it is not easy to find an analytical expression for the *Pareto front*, the usual procedure is to determine a set of *Pareto optimal* points that provide a good approximate description of the *Pareto front* .

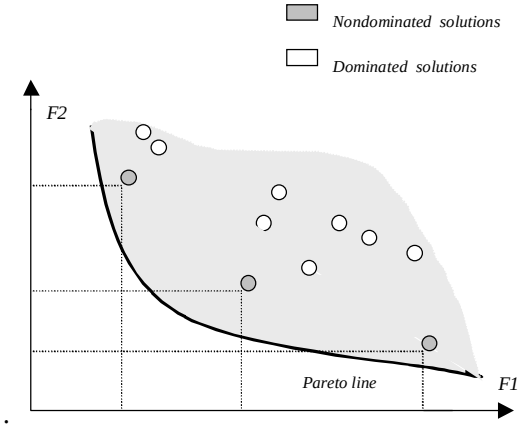

Figure 1. Two-objective space

3. Evolutionary multiobjective optimization

An evolutionary algorithm (EA) is a stochastic optimization algorithm that simulates the process of natural evolution **[Bäck97].** Thus, an EA operates on a set of candidate solutions, which are subsequently modified by simplified implementations of the two basic principles of evolution: selection and variation. Selection represents the competition for resources among living beings. Some are better than others and more likely to survive and transmit their genetic information. A stochastic selection process simulates natural selection. Each solution is given a chance to reproduce a certain number of times, dependent on their quality, which is assessed by assigning a fitness value to each individual. The other principle, variation, imitates the natural capability of creating *new* living beings by means of recombination and mutation (see Figure 2). Recombination and mutation are typically the variation operators.

In a SOP (Figure2), *Task2* obtains a scalar value (fitness) for each individual (candidate solution) by evaluating a single function. In MOPs (Figure2) *Task2* has two subtasks: Task2a obtains a vector of scalar values (vector fitness) for each individual by evaluating the set of objective functions: the dimension of vector fitness is equal to the size of the set of objective functions. *Task2b* converts the vector fitness for each individual into an scalar value (fitness) using some specific technique (different for each MOEA). Task2b usually incorporates a niching technique.

Task1: Initiate population
Repeat *t=1,2,...*
Task2: Evaluate solutions in the population:
a) for each individual obtain a scalar fitness
Task3: Perform competitive selection in population
Task4: Apply variation operators to the population
Until *convergence criterion is satisfied*

Task 1: Initiate population
Repeat *t=1,2...*
Task 2: Evaluate solutions in the population:
a) for each individual obtain a vector fitness
b) for each individual convert vector fitness into a scalar fitness
Task3: Perform competitive selection in the population
Task4: Apply variation operators to the population
Until *convergence criterion is satisfied*

Figure2. Pseudo-code of an EA for single-objective optimization (top) and multiobjective optimization (bottom).

Evolutionary algorithms seem to be especially suited to multiobjective optimization because they are able to capture multiple Pareto-optimal solutions in a single run, and may exploit similarities of solutions by recombination. Indeed, some research suggests that

multiobjective optimization might be an area where EAs perform better than other search strategies. The considerable amount of research related to MOEAs currently reported in the literature[1] is evidence of present interest in this subject.

In MOPs, it is necessary to find not one but several solutions, in order to determine the entire Pareto front. Nevertheless, due to stochastic errors associated with the evolutionary operators, EAs can converge to a single solution **[Goldberg89].** There exist literature several methods, called *niching techniques* **[Sareni98]**, to preserve diversity in the population, in order to converge to different solutions. These techniques can also be applied to MOP.

Pareto-based fitness assignment in a genetic algorithm (GA) was first proposed by Goldberg **[Goldberg89]**. The basic idea is to find a set of Pareto non-dominated individuals in the population. These individuals are then assigned the highest rank and eliminated from further competition. Then, another set of Pareto non-dominated individuals are determined from the remaining population and are assigned the next highest rank. This process continues until the whole population is suitably ranked. Goldberg also suggested the use of a *niching technique* to keep the GA from converging to a single point on the front**.**

The *Non-dominated Sorting Genetic Algorithm (NSGA)* **[Srinivas93]** uses several layers of ranked individuals. Before selection is performed, the population is ranked on the basis of non-domination: all non-dominated individuals are classified into one category (with a dummy fitness value, which is proportional to the population size, to provide an equal reproductive potential for these individuals). To maintain the diversity of the population, those so classified are shared with their dummy fitness values. Then this group of classified individuals is ignored and another layer of non-dominated individuals is considered. The process continues until all individuals in the population have been classified. Then a stochastic remainder proportionate selection is used, followed by the usual cross and mutation operators.

Fonseca and Fleming **[Fonseca93]** have proposed an algorithm called *Multiple Objective Genetic Algorithm (MOGA)* where the rank of each individual is obtained from the number of individuals in the current population that dominate it. Thus, if at generation t, an individual x_i is dominated by p_i (t) individuals, its current rank can be given by:

$$\text{Rank}\ (x_i, t)=1+p_i\ (t)$$

All non-dominated individuals are assigned rank 1, while dominated ones are penalized according to the population density of the corresponding region of the trade-off surface. In this way, the fitness assignment is performed by the following steps:

1. - Sort population according to the rank of the individuals.
2. - Assign fitness to individuals by interpolating from the best (rank 1) to the worst (rank n<=N), according to a function (not necessarily linear)
3. - Average the fitness of individuals with the same rank, so that all of them will be sampled at the same rate. Sharing on the objective function values is carried out to distribute population over the Pareto-optimal region.

In their *Niched Pareto Genetic Algorithm (NPGA),* Horn and Nafpliotis **[Horn93]** proposed a tournament selection scheme based on Pareto dominance. Instead of limiting the comparison to two individuals, a number of other individuals (usually about 10) in the population are used to help determine dominance. Whether the competitors are dominated or non-dominated, the result is decided through fitness sharing.

In **[Zitzler99]**, it was clearly shown that elitism helps to achieve better convergence in MOEAs. Zitzler and Thiele **[Zitzler98]** suggested an elitist multi-criterion EA with the concept of non-domination in their *Strength Pareto Evolutionary Algorithm (SPEA).* They suggested maintaining an external population at every generation storing all non-dominated solutions discovered so far beginning from the initial population. This external population participates in genetic operations. At each generation, a combined population with the external and the current population is first constructed. All non-dominated solutions in the combined population are assigned a fitness based on the number of solutions they dominate, and dominated solutions are assigned fitness worse than the worst fitness of any non-dominated solution. This fitness assignment assures that the search is directed towards the non-dominated solutions. A deterministic clustering technique is also used to maintain diversity among non-dominated solutions. Although the implementation suggested is O (mN^3), with appropriate book-keeping the complexity of SPEA can be reduced to O (mN^2). An improved version of SPEA known as SPEA2 **[Zitzler01]** has recently been proposed. This incorporates additionally fine-grained fitness assignment strategy, a density estimation technique, and an enhanced archive truncation method.

Knowles and Corne **[Corne00]** suggested a simple MOEA using an evolutionary strategy (ES). In their *Pareto-Archived ES (PAES),* a parent and a child are compared. If the child dominates the parent, the child is accepted as the next parent and the iteration continues.

[1] C.Coello maintains a repository on MOEAs at: http://ww.lania.mx/~ccoello/EMOO/

On the other hand, if the parent dominates the child, the child is discarded and a new mutated solution (a new child) is found. However, if the child and the parent do not dominate each other, the choice between the child and the parent considers the second objective of maintaining diversity among obtained solutions. To achieve this diversity, an archive of non-dominated solutions is created. The child is compared with the archive to determine whether it dominates any member of the archive. If so, the child is accepted as the new parent and the dominated solution is eliminated from the archive. If the child does not dominate any member of the archive, both parent and child are examinated for their proximity to the solutions of the archive. If the child resides in an uncrowded region in the parameter space among the members of the archive, it is accepted as a parent and a copy is added to the archive. Knowles and Corne later, suggested a *multiparent PAES* with similar principles to the above. The authors have calculated the worst case complexity of PAES for N evaluations as O(amN) where a is the archive length. Since the archive size is usually chosen proportional to the population size N, the overall complexity of the algorithm is $O(mN^2)$.

Finally, Deb **[Deb00]** has also proposed an improved parameter-less version of NSGA, called NSGA-II.

4.Parallel genetic multiobjective optimization

Parallel Genetic Algorithms (PGAs) are naturally prone to parallelism since the genetic operations on the individuals of the population can be easily undertaken in parallel (not so easy for the selection method, since it usually uses the full population). A survey of parallelism strategies applied to GAs is provided in **[Cantu97]**. Figure 3 shows different models of PGAs, in order to present the terms we use bellow.

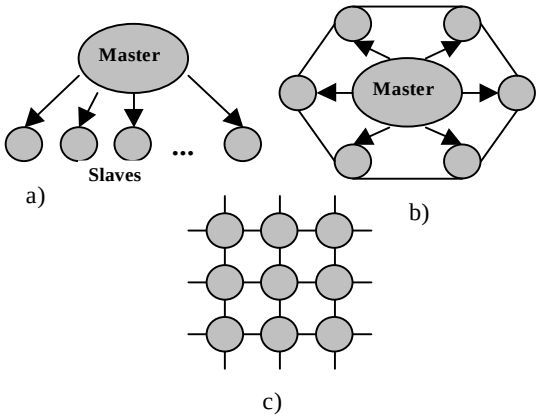

Figure 3. Different models of PGA: (a) global parallelization, (b) coarse grain, and (c) fine grain.

In global parallelization (Figure 3.a), explicit parallelization of the genetic operators and/or evaluations of individuals are performed. The algorithm proceeds in the same way as a sequential GA, but in a faster manner. However, only for problems with a time-consuming function evaluation do they represent a viable choice; otherwise the communication overhead is higher than the benefits of their parallel execution. The rest of the PGA models fit into two classes, depending on their computation/communication ratio, called *coarse* (Figure 3.b), or *fine-grain (*Figure 3.c) *parallel GAs*. Coarse grain PGAs are also known as distributed (dGA) or island GAs, and fine grain PGAs are known as cellular (cGA), diffusion, or massively-parallel GAs. A dGA has a bigger subpopulation size than a cGA but fewer of subpopulations and less coupled.

Although much work is reported about PGA models (and implementations on different parallel architectures), involving SOPs, PGA models could also be applied for MOPs, where *global parallelization* (Figure 3.a) is the most used strategy. In **[Mäkinen97]** a modified NSGA, with tournament selection instead of roulette wheel selection **[Srinivas93]**, is implemented by using a global parallelization scheme based on the master-slave prototype (Figure 3). The algorithm is applied to the multiobjective optimization of two-dimensional airfoil designs and it is executed in an IBM SP2.

In **[Rogers00]** a PGA using *global parallelization* is applied to optimal actuator selection, which means finding the minimum number of actuators and their localization to provide uncoupled pitch, roll and yaw control for a simplified wing model. The algorithm allocates three processors (*working processors*) to evaluate the three objective functions. One processor is devoted, as a master processor, to sending data (an array containing the actuator locations), and to receiving data (pitch, roll, and yaw moments) to and from the working processors. Recombination, mutation and selection are performed in the master processor.

In **[Stanley95]** a real-valued MOEA called *GAIN (Genetic Algorithm running on the INternet)* is applied to design cache memories. The algorithm is executed on a *Network of Computers (NOW)*, with 80 to 120 workstations. GAIN uses a *generation process* that performs the selection and applies the recombination and mutation operators; several *evaluation processes* simultaneously evaluate the different objective functions of the problem. In **[Jones98]** a global paralleled MOEA is applied to aerodynamic and acoustic optimization of airfoils. A master process dynamically balances the work to be done by the other processors, whose sole purpose is to evaluate fitnesses. In **[Obayashi00],** a parallel implementation of MOGA **[Fonseca93]** is used to

optimize the design of supersonic wing shapes. This problem presents 66 design variables and 3 objective functions. Fitnesses are evaluated on 32 processing elements of an NEC SX-4 computer. **[Meunier00]** also uses a parallel version of MOGA (implemented in a NOW with 24 workstations) for Radio Network Optimization.

Parallelization should be used not only for obtaining a speedup to the sequential version of the genetic algorithm by reducing evaluation execution time. Parallelization strategies may also help to maintain diversity in the population **[DeToro01]** working as a niching technique to obtain a well-distributed non-dominated set of solutions. This is a very important issue in MOP, which has also been addressed in some works by using a structured population (cGA model) **[Rowe96] [Murata00]**.

5. Parallel Single Front Genetic Algorithm

The MOEA presented in this paper is based on a genetic algorithm (see Figure 4), called *Single Front Genetic Algorithm (SFGA)*, that implements an elitist procedure in which:

- Only the non-dominated (and well-diversified) individuals in the current population are copied to the mating pool for recombination purposes, and
- All non-dominated individuals in the current population are copied to the next population.

Parameters:		
	N	(population size)
	T	(maximum number of generations)
	Pm	(mutation rate)
	Pc	(crossover probability)
Output:	**A**	(non-dominated set)
Step 1:	**Initialization:** **P**t=**P**0 (initial population)	
Step 2:	**Fitness assignment:** Determine the objective vector **f(x)** for each individual in **P**t. Determine **A**, (the non-dominated set in **P**t).	
Step 3:	**Selection:** If S (**A**) = *N* apply a **filter** function to **A** producing **A'** (a well diversified set of non-dominated individuals in **P**t). Set **P'= A'**	
Step 4:	**Recombination:** Chose two individuals **i**, **j** in **P'** (with replacement). Recombine **i** and **j** with probability *Pc* (by using a one-point crossover function) producing individual **k.** Mutate **k** with mutation rate *Pm* producing **k'**. Set **P'=P'**U{**k'**}. Repeat Step 4 until size of **P'** is equal to *N*.	
Step 5:	**Termination:** Set *t=t+1* and **P**t+1= **P'**. If *t>T* then **A** is the non-dominated set of **P**t+1 else goto Step 2.	

Figure 4.: Description of SFGA

The rest of the individuals required to complete the population are obtained by recombination and mutation of the aforementioned non-dominated individuals.

The preservation of diversity in the population is ensured by means of a filtering function, which prevents the crowding of individuals by removing individuals according to a given *grid* in the *objective* space. The filtering function uses the distance evaluated in the objective space. On the other hand, the sharing function used in NSGA is based on distance in the decision space.

In this way, PSFGA (*"Parallel Single Front Genetic Algorithm"*) is an elitist Pareto-based algorithm for multiobjective optimization based on a structured-population dGA model **[DeToro01]**. In PSFGA, the population is sorted with respect to the values of the objective function and divided into subpopulations. In each subpopulation, the SFGA is executed, and after some generations, all individuals are gathered, sorted and compared according to the Pareto dominance criteria. After that, the individuals are again divided among the different processors.

PSFGA presents a two-level mechanism to maintain diversity in the population: at high-level PSFGA presents a structured-population cGA model; while at low-level, the filtering function implemented in SFGA is used. Both mechanisms are applied in the objective space.

Moreover, PSFGA is based on a master-slave prototype. In the Master process, an initial random population of size N is created, and then individuals are sorted according to the values of the objective functions (f_i). Then, a set of N/m are selected according to the value of the considered objective function f_i, in order to define m subpopulations (Figure 5). Each subpopulation is assigned to a given working process that runs a number of generations of SFGA on its allocated subpopulation. Periodically, the population is gathered in the master process, sorted again according to a different objective function, and comparisons according to global Pareto dominance are performed.

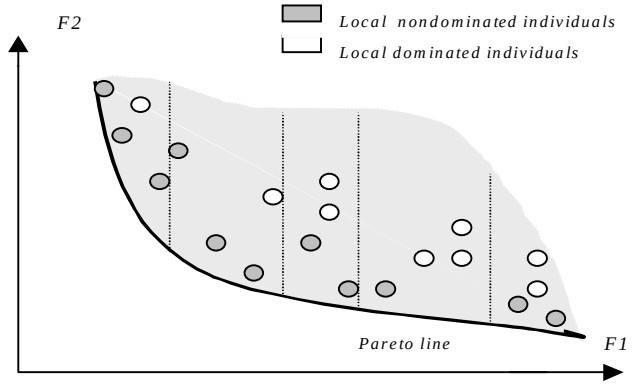

Figure 5. Five subpopulations in PSFGA, for a two-objective problem.

6. Experimental Results

To evaluate PSFGA and compare it with other proposed algorithms we have used the benchmark functions used by Zitzler **[Zitzler99]**. These functions are selected by taking into account a wide range of features that may cause difficulties for an MOEA **[Deb99]** such as convexity (function ZT1), non-convexity (ZT2), discreteness (ZT3), multimodality (ZT4), and non-uniformity (ZT6). The deceptive function ZT5 is not used because in SFGA the individuals in the population are coded as real numbers.

For performance comparison, we used the *hypervolume metric* **[Zitzler99]** for minimization problems. For comparison of two non-dominated solution sets, *Ai* and *Aj*, the following measures are computed: *S(Ai)* is the volume of the space that is non-dominated by the set *Ai*. *D(Ai,Aj)* is the volume of the space that is non-dominated by the first solution set *Ai* but dominated by the second set *Aj*. The smaller *S(Ai)* and *D(Ai,Aj)* related to *S(Aj)* and *D(Aj,Ai)* respectively, the better *Ai* related to *Aj*. In all experiments *Pm*=0.01, *Pc*=1 and population size is 100 as in **[Zitzler99]**. The filter parameter, ft, is set to 0.01.

The SFGA and PSFGA procedures are compared with NSGA, which is the procedure that provides the best results **[Zitzler99]**. The selection mechanism used in NSGA is binary tournament selection with continuous updating sharing, with sharing radius set to 0.4889. The distances in sharing function are calculated on decision space.

Table 2 shows the speedup of SFGA with respect to NSGA (*TCRatio*), and the size of the set of non-dominated solutions of SFGA with respect to NSGA (SRatio). Table 3 compares SFGA and NSGA by using the previously described *hypervolume metric*. In addition, graphical representations of the final non-dominated sets for both NSGA (diamonds) and SFGA (crosses) are provided in Figures 6-8 for ZT1, ZT3, and ZT6 (as examples).

Table 2: Performance of SFGA related to NSGA

Function	Generation	TCRatio	SRatio
ZT1	750	1.22	3.7
ZT2	500	2.20	2.6
ZT3	500	1.83	1.0
ZT4	1000	1.31	2.0
ZT6	750	1.70	1.4

Table 3: Performance by Zitzler's metric

Function	S(sfga)	S(nsga)	D(sf,ns)	D(ns,sf)
ZT1	0.371978	0.405349	0.008159	0.041528
ZT2	0.770345	0.796941	0.014282	0.040878
ZT3	0.009983	0.014344	0.031004	0.035364
ZT4	1.679213	1.828798	0.084241	0.233825
ZT6	2.807990	3.518460	0.000000	0.710470

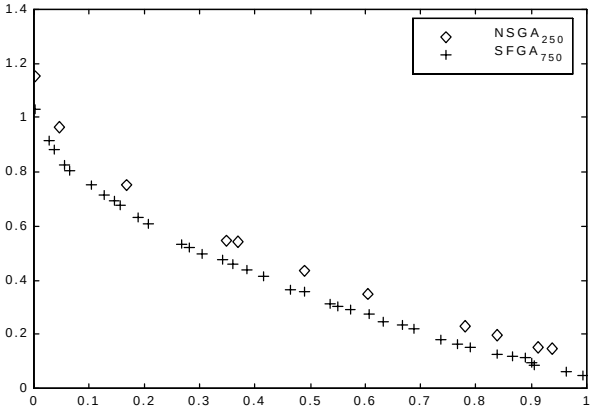

Figure 6. Solutions for ZT1

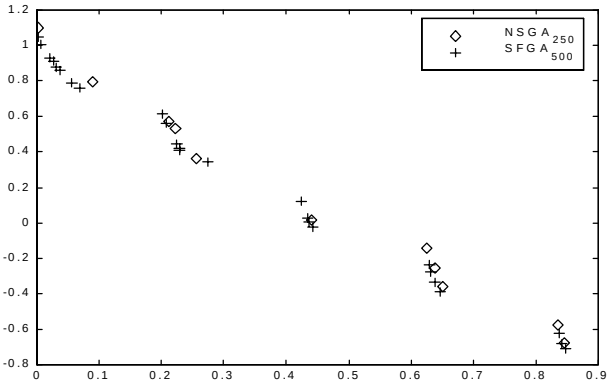

Figure 7. Solutions for ZT3

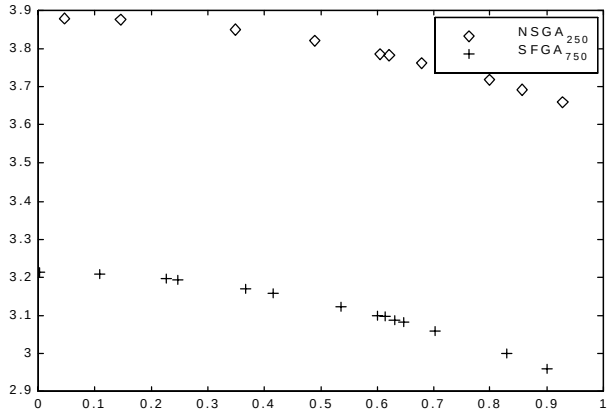

Figure 8. Solutions for ZT6

Evidently, SFGA outperforms NSGA. Bellow, we provide the results of PSFGA compared with those of SFGA. The total number generations of performed by PSFGA are:

MaxGen := genpar + (Comm-1)(genpar+genser)*

Thus, *genpar* iterations are performed in each subpopulation, while *genser* iterations are performed in the *panmictic subpopulation*; The parameter *comm* control the frequency of master-slave communications. The proposed PSFGA has been implemented using a cluster of 8 PCs connected by Fast Ethernet.

Results are available for the parallel configurations shown in Table 4 (in all cases *comm*=63):

Table 4: Parallel configurations

Configuration	*Gen_ser*	*Gen_par*
A	8	8
B	10	6
C	6	10
D	13	3
E	3	13

The speedups in Figure 9 have been obtained for 2,4,6 and 8 processors and different parallel configurations. Executions by using 4 ($\mathbf{s_4}$) and 6 processors ($\mathbf{s_6}$) are shown in table 5. As_4 stands for configuration A executed on 4 subpopulations. Each row in Table 5 shows some of the results for the corresponding benchmark function, i.e. the values of the metric S for SFGA (1000 iterations), and different configurations of PSFGA (1000 iterations). As can be seen, the parallel algorithm is able to speed up the optimization procedure, although in most cases, it produces a reduction in the quality of the solutions found. Figure 10 shows some results corresponding to the solutions for ZT4 obtained by NSGA, SFGA and two configuration of PSFGA.

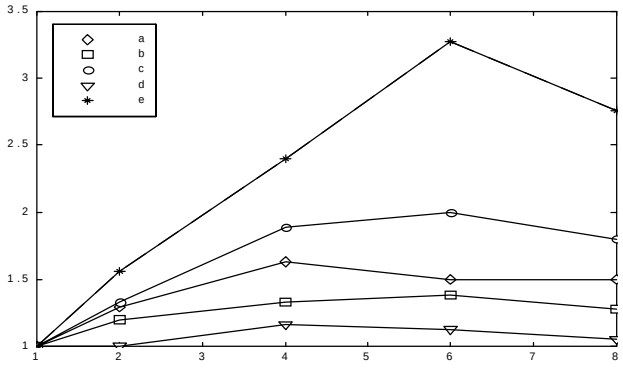

Figure 9. Speedup of PSFGA

Table 5. Performance Comparison for PSFGA by S metric

Func	SFGA	As_4	As_6	Bs_4	Bs_6
ZT1	0.3612	0.3806	0.3941	0.3820	0.3856
ZT2	0.7268	0.7462	0.7912	0.7801	0.7640
ZT3	0.0059	0.0034	0.0035	0.0031	0.0043
ZT4	1.6792	1.5470	1.7941	1.0972	1.5586
ZT6	2.9659	3.2051	3.0743	2.9872	3.1957

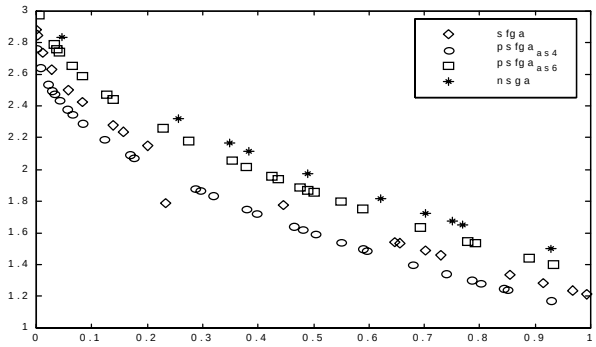

Figure 10. Solutions for ZT4

7. Concluding remarks

This paper presents the *Parallel Single Front Genetic Algorithm (PSFGA)*. PSFGA is a parallel implementation (using a dGA model) of the *Single Front Genetic Algorithm (SFGA)*, a new elitist MOEA with a fitness assignment based just on the first front of non-dominated individuals of the population, in contrast with the use of multiple fronts used in other algorithms. The performance of PSFGA has been analyzed in comparison with SFGA and the previously proposed NSGA. The experimental results show that SFGA clearly outperforms NSGA for the whole benchmark set. The aggressive convergence pressure that SFGA implements based on a super-elitist selection policy allows a fast convergence without premature convergence effects. PSFGA reduces the time execution of SFGA. Moreover, a better convergence is sometimes observed in PSFGA for some functions of the benchmark set.

Acknowledgements. This paper has been supported by the Spanish *Ministerio de Ciencia y Tecnología* under grant TIC2000-1348.

8. References

[Bäck97] Bäck, T.; Hammel, U.; Schwefel, H.-P.: "*Evolutionary Computation: comments on the history and current state*". IEEE Trans. on Evolutionary Computation, Vol.1, No.1, pp.3-17. Abril, 1997.

[Cantú97] E.Cantú-Paz, *"A Survey of Parallel Genetic Algorithms"*, Technical Report. Illlinois Genetic Algorithms Laboratory, 1997.

[Coello98] Carlos A. Coello Coello. An Updated Survey of GA-Based Multiobjective Optimization Techniques, *Technical Report Lania-RD-98-08, Laboratorio Nacional de Informática Avanzada (LANIA), 1998.*

[Corne00] Corne, D.W., Knowles, J.D., and Oates, M.J. (2000). The Pareto envelope-based selection algorithm for multiobjective optimization. Proceedings of the Parallel Problem Solving from Nature VI Conference, pp. 839-848.

[Deb99]Kalyanmoy Deb. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems, *Evolutionary Computation*, 7(3):205-230, Fall 1999.

[DeToro01] F.deToro, A.F.Díaz, C.Gil, J.Ortega. *AGEMM: Optimización MultiModal Paralela usando Algoritmos Genéticos*. XII Jornadas de Paralelismo, Valencia 2001.

[Fonseca93] Carlos M. Fonseca and Peter J. Fleming. *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*, In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416-423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[Fonseca98] Fonseca, C.M. and P.J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part i: A unified formulation. IEEE Transactions on Systems, Man, and Cybernetics 28(1), 38-47.

[Goldberg89] D.E. Goldberg, "*Genetic Algorithms in Search, Optimization and Machine Learning*", New York: Addison Wesley, 1989.

[Jones98] B.R Jones ;W.A Crossley; A.S.Lyrintzis. *Aerodynamic and Aeroacoustic optimization of airfoils via a parallel genetic algorithm*. American Institute of Aeronautics and Astronautics. (AIAA-98-4811) .Pardue University, 1998

[Mäkinen97] Mäkinen, R.A.E., et al. *"Parallel Genetic Solution for Multiobjective MDO."* Parallel Computational Fluid Dynamics: Algorithms and Results Using Advanced Computers, edited by P. Schiano, et al., 352-359, Elsevier Science, 1997.

[Meunier00] Herve Meunier, El-Ghazali Talbi, and Philippe Reininger. *A Multiobjective Genetic Algorithm for Radio Network Optimization***,** In *2000 Congress on Evolutionary Computation*, volume 1, pages 317-324, Piscataway, New Jersey, July 2000. IEEE Service Center

[Murata00] Murata, T. And Gen, M.: *"Cellular genetic local search for multi-objective optimization"*. Proc. Of the Genetic and Evolutionary Computation Conference 2000 (2000) 307-314.

[Obayashi98] Obayashi, S., S. Takahashi, and Y. Takeguchi, *"Niching and elitist models for mogas"* In A. E. Eiben, T. Bäck, M. Schoenauer, and H.P.Schwefer(editors), 5th International Conference on Parallel Problem Solving from Nature (PPSN-V), Berlin, Germany, pp.260-269.Springer.

[Obayashi00] Shigeru Obayashi, Daisuke Sasaki, Yukihiro Takeguchi, and Naoki Hirose. *"Multiobjetive Evolutionary Computation for Supersonic Wing-Shape Optimization"*. IEEE Transactions on Evolutionary Computation. Vol 4. No. 2. July 2000.

[Quagliarella00]D.Quagliarella, A. Vicini. "Sub-population policies for a parallel multiobjective genetic algorithm with applications to wing design". C.I.R.A., Centro Italiano Ricerche Aerospaziali, Via Maiorise-81043 Capua (Italy).

[Srinivas93] N.Srinivas and Kalyanmoy Deb, "*Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms"*. Evolutionary Computation, Vol. 2, No. 3, pages 221-248

[Stanley95] Stanley. **"***A parallel genetic algorithm for multiobjective microprocessor design*". In L.J. Eshelman, editor, 6th Int. Conf. On Genetic Algorithms, pages 597-604. Morgan-Kaufmann, 1995.

[Zitzler98] Zitzler, E.; Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength Pareto approach, Technical Report No. 43 (May 1998), Zürich: Computer Engineering and Networks Laboratory, Switzerland.

[Zitzler99] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*, Technical Report 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, December 1999.

[Zitzler01] Eckart Zitzler, M. Laumannas; L.Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm, TIK – Report 103, May 2001.