MRMOGA: Parallel Evolutionary Multiobjective Optimization using Multiple Resolutions

Antonio López Jaimes and Carlos A. Coello Coello
Centro de Investigaciones y de Estudios Avanzados del IPN
Departamento de Ingeniería Eléctrica, Sección de Computación
Av. IPN núm. 2508, col. Sn. Pedro Zacatenco, México, D.F. 07300, MÉXICO
alopez@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

Abstract- Whereas Multiobjetive Evolutionary Algorithms have reached certain effectiveness in solving many real-world problems, efficiency still remains as an open problem. One choice to reduce the execution time of the Multiobjetive Evolutionary Algorithms is their parallelization. This paper introduces a novel parallel MOEA which is based on the *island model* with heterogeneous nodes. This algorithm is characterized by encoding the solutions using a different resolution for each island. In this way, the search space is divided into well-defined overlapped regions in decision variable space.

1 Introduction

Nowadays Multiobjetive Evolutionary Algorithms (MOEAs) have solved with acceptable effectiveness plenty of real-world problems originated in engineering, scientific and industrial areas [5]. Nonetheless, in some of these problems is required to evaluate a large number of objective functions or a large population size in order to reach a desired effectiveness. Furthermore, often the evaluation of the objective functions is very time-consuming which makes impractical the running time required to find an acceptable solution.

In order to reduce the execution time to solve these problems there are mainly two approaches: (1) to improve the MOEA so that the number of objective functions evaluations is reduced and (2) to adopt a parallel or distributed execution of the MOEAs. Besides the time reduction, the parallel MOEAs (pMOEAs) are attractive for many other reasons such as the possibility to use larger populations, the improvement of population diversity, and the ease to cooperate in parallel with another search technique.

During the last three decades, parallel Evolutionary Algorithms used for global optimization (pEAs) have been widely studied [2, 4, 15]. However, due to the peculiarities of multiobjective optimization, there are some issues that need to be tackled with a different approach. For instance, the fact that the evaluation of each solution implies the evaluation of k ($k \geq 2$) objective functions, which gives rise to more ways for evaluating the solutions in parallel. Additionally, real-world multiobjective optimization problems tend to have high-dimensionality (i.e., a large number of decision variables) which also requires of a large computational effort. Also, we have to solve how to parallelize some features of pMOEAs such as archiving and niching techniques.

This paper presents a new scheme to split up decision variable space in order to distribute it among different processors. The proposed algorithm, called Multiple Resolution Multi-Objective Genetic Algorithm (MRMOGA), is based on the island paradigm with heterogenous nodes. This algorithm encodes the solutions using a different resolution in each island. In this way, the variable decision space is divided into hierarchical levels with well-defined overlaps. The performance assessment of the proposed approach considers both effectiveness and efficiency. To evaluate the effectiveness we adopt well-known metrics traditionally used to evaluate serial MOEAs, namely: success counting, inverted generational distance, spacing, and two set coverage. On the other hand, efficiency was evaluated using the following well-known metrics from parallel computing: speedup, efficiency and serial fraction. The results of the proposed algorithm were compared against those of a parallel version of the NSGA-II [6].

The remainder of this paper is organized as follows: The next section briefly describes the island model and also presents an overview of the most relevant related work. Section 3 is devoted to describe the proposed approach. Then, an assessment of our approach performance is discussed in section 4. Finally, section 5 contains the conclusions about the viability of the suggested scheme and some research paths for future work.

2 Related Work

2.1 Parallelization of Multiobjective Evolutionary Algorithms

The parallelization approaches of MOEAs are derived from the models designed for single-objective optimization: master-slave, island and diffusion models. Nevertheless, there is no current standard way to extend these models to the multiobjective field. Since our approach is based on the island model, we shall only describe this model.

Island model. In an island pMOEA, the population is divided into several small subpopulations, called *islands* or *demes*, which evolve independently of each other. In each of these islands a serial pMOEA is executed for a number of generations called an *epoch*. At the end of each epoch, individuals migrate between neighboring islands. The neighbors are given by the *migration topology*, which determines the migration paths along which individuals can move to other islands.

The model allows each island to have their own parameter setting. Depending on this homogeneity we can recognize two variants of the island model:

- **Island pMOEA with homogeneous nodes.** Every MOEA performed in each island has the same parameter values (e.g., population size, mutation, crossover and migration rate).

0-7803-9363-5/05/$20.00 ©2005 IEEE.

- **Island pMOEA with heterogeneous nodes**. Each island applies a MOEA which has a different parameter setting, or uses its own evolutionary operators and solution encoding technique. Additionally, each island may be explicitly instructed to explore a particular region of decision or objective space.

2.2 Overview of Selected Parallel Multiobjective Evolutionary Algorithms

The Divided Range Multi-Objetive Genetic Algorithm (DRMOGA) was proposed by Hiroyasu et al. [8]. Here, the global population is sorted according to an alternating objective function. Then the population is divided in equally-sized subpopulations. Each of these subpopulations is allocated to a different processor in which a serial MOEA is applied. At the end of a specific number of generations, the subpopulations are gathered and the process is repeated. The main goal of this approach is to focus the search effort of the population on different regions of the objective space. However, in this approach we cannot guarantee that the subpopulations will remain in their assigned region.

Zhu and Leung proposed the Asynchronous Self-Adjustable Island Genetic Algorithm (aSAIGA) [16]. In aSAIGA rather than migrating a set of individuals, the islands exchange information related with their current explored region (i.e., an hypercube containing most of the individuals in the archive). Based on the information coming from other islands, a "self-adjusting" operation modifies the fitness of the individuals in the island to prevent two islands to explore the same search region. In a similar way as DRMOGA, this approach cannot guarantee that the subpopulations move tightly together throughout the search space, hence the information about the explored region may be meaningless.

Other island pMOEA was introduced by Deb et al. [7]. Although all processors search on the entire decision space, this approach assigns to each processor a different search region of the Pareto-optimal front. In order to steer the search towards the assigned region, this approach uses a guided domination concept based on a concept defined elsewhere [3]. The weakness of this approach is that we must have *a priori* knowledge of the shape or continuity of the Pareto-optimal front in order to define accurately the search directions. Furthermore, this technique can only deal with convex Pareto fronts.

A recent approach was proposed by Streichert et al. [14]. This approach partitions the overall population using a clustering algorithm as to specialize each island on different areas of the Pareto front. Each certain number of generations the islands are gathered, clustered and redistributed onto the available processors. The individuals are kept within their region using zone constraints, that is, any individual generated outside its constrained region is considered "invalid". The main drawback of the approach is that the repeated gathering of all subpopulations produces a high communication overhead, which is increased with the number of processors.

3 Multiple Resolution Scheme

In most of the current pMOEA approaches that use a "divide and conquer" strategy, the division has been made in the objective function space [7, 8, 16]. To the best of our knowledge, only Streichert et al. [14] carried out such division in decision variable space. Nonetheless, none of these approaches guarantees that the generated individuals belong to their assigned region. On the other hand, the secondary population (external archive), if any, does not interact in the search process in any of these approaches.

Taking care of these observations, we designed a model called Multiple Resolution Multi-Objective Genetic Algorithm (MRMOGA). Our approach is based on the island model with heterogeneous nodes and is characterized by the following:

- Each island encodes the solutions with a different resolution (i.e., different number of precision digits after the decimal point). This implies a division of the decision space in such a way that every new individual belongs to their assigned region.
- It uses a migration strategy that considers both the primary population and the external archive.
- It uses a strategy to detect the nominal convergence of the islands in order to increase their initial resolution.

In a single-objective optimization context, Lin et al. [12] suggested an island model named Injection Island GA (iiGA). In this approach there is a number of subpopulations that encode the problem using different resolutions. The known implementations of iiGA were designed for combinatorial optimization, so the binary strings, represent the problem's solution. On the contrary, the MRMOGA is designed for numerical optimization and, therefore, it embodies a strategy to convert the binary strings from a resolution into another one.

The MRMOGA scheme is described in Algorithm 1. Each island executes a serial MOEA which is initialized as usual, but each of them uses a different resolution. During migration, each processor takes some individuals from its archive ($PF_{known}\binom{i}{t}$) to send them to the primary population of its neighbors. At the end of the search process the root process combines the non-dominated front of each processor ($PF_{known}\binom{i}{}$). Figure 1 shows a schematic view of the algorithm. All of the components of the approach are described in detail in the following sections.

3.1 The Principle of Multiple Resolutions

The idea behind the proposed approach is based on the following assumption: the Pareto optimal solutions are found in fewer iterations using low resolution representations than using higher resolutions representations. In effect, the search space is smaller as the resolution decreases, and hence we need to explore fewer solutions to determine the known Pareto front (PF_{known}).

Taking advantage of this idea, we can design an island model in which each island encodes the solutions with a different resolution. The low resolution islands have the pur-

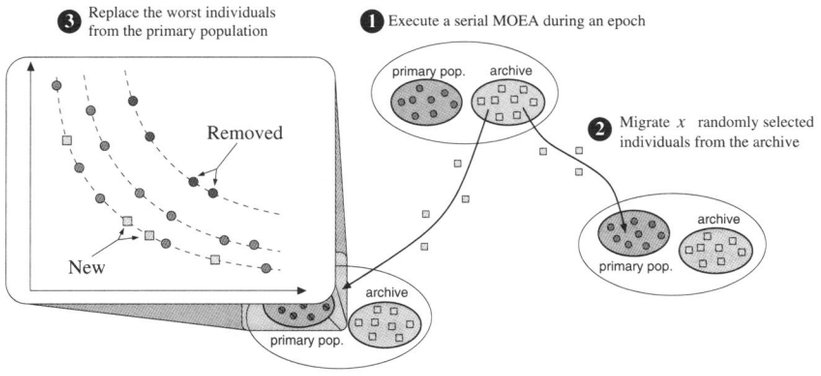


Figure 1: Schematic view of MRMOGA.

E: Maximum number of epochs.
G: Generations per epoch.
Generate (randomly) the population $\mathcal{P}_i$.
for $e \leftarrow 1$ until E **do**
  for $g \leftarrow 1$ until G **do**
    SERIALMOEA($\mathcal{P}_i$)
  end for
  if $e \neq E$ **then** ▷ Do not migrate in last epoch
    ▷ According to migration topology
    for all outgoing neighbor j of processor i **do**
      MIGRATE($PF_{known}\binom{i}{g}$, $\mathcal{P}_j$)
    end for
    for all incoming neighbor j of processor i **do**
      REPLACE($\mathcal{P}_i$, $PF_{known}\binom{j}{g}$)
    end for
  end if
end for
Combine all $PF_{known}\binom{i}{}$ in processor 0.
Filtering using the grid and show PF_{known} to the user.

Algorithm 1: MRMOGA pseudocode.

pose of approaching quickly towards the true Pareto front (PF_{true}), thus finding individuals with high fitness. Afterwards, these individuals are migrated into higher resolution nodes for exploiting the region nearby these highly fitted individuals. This approach can be considered as a division of the variable decision space. In such a partition there are well-defined overlapped solutions. That is, the search space of an island is contained in the search space of the higher resolution islands.

3.2 Need for Conversion in Migration

Since in the migration process of our approach we exchange chromosomes (binary strings) with different lengths, we need to adapt a given string into a longer length string. This conversion is achieved by the following expression[1]:

$$d_i^* = \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}(2^{\ell_i} - 1),$$

[1]Derived from the well-known expression that maps a chromosome into a real value

where ℓ_i is the new length of the string that encodes the i-th variable, and $x_i \in [x_i^{min},\ x_i^{max}]$ is the decoded variable produced from a chromosome of length ℓ_i' ($\ell_i' < \ell_i$). Now, we round up d_i^* to the nearest integer. Finally, we encode the rounded value of d_i^* to obtain the new string with length ℓ.

3.3 Topology and Migration/Replacement Scheme

Since the search space of an island is contained within the search space of any island with higher resolution, but not the converse, the migration topology has a hierarchical structure. In this paper only the so-called *strict topology* is presented. In this topology each island migrates solutions only to the island with a higher resolution. In the migration/replacement scheme are involved both, the primary population and the external archive.

Migration scheme. Migrate x randomly selected members of the archive (i.e., the known Pareto front). If the archive size is less than x, then the missing individuals are randomly selected from the primary population.

Replacement scheme. Initially, the members of the primary population are ranked in non-dominated levels. Later, x individuals are replaced from the last ranked Pareto front (the "worst" individuals). If the size of the last front is less than x, a sufficient number of individuals are taken from the next ranked fronts until the x individuals are completed. Lastly, the chromosome of the incoming individuals are converted to the new length.

3.3.1 Technique for maintaining diversity

In order to retain the non-dominated solutions and to distribute them evenly along the Pareto front, MRMOGA uses an adaptive grid similar to the one proposed in [10]. The adaptive grid is invoked only if the archive has reached its maximum capacity.

3.3.2 Constraint-handling strategy

Our approach uses a rather simple constraint-handling strategy. The dominance between two solutions is determined comparing the number of constraints violated by each solution. The solution with smaller number of violated con-

straints will dominate the other. In the case that both individuals violate the same number of constraints, Pareto dominance is applied as usual.

3.3.3 Increase of the island resolution.

MRMOGA has the ability to generate solutions only within the decision subspace that each island was devoted for. Unfortunately, this approach has some drawbacks since the regions are assigned statically:

- The search space of the low resolution islands is proportionally smaller, therefore, the true Pareto front is found in fewer iterations than in higher resolution islands. Thus, once the low resolution islands have converged, they spend the remainder of time oscillating around the Pareto front without any real contribution to discover other regions of the front.
- For some MOPs, the PF_{known} obtained with certain resolutions (particularly the low ones) do not belong to PF_{true}, thus, the islands with those resolutions do not contribute at all to find portions of PF_{true}.

In order to cope with these difficulties, we propose to increase the resolution of each island once it has reached nominal convergence. In this way, at every moment the processors' utilization is kept high, and the islands that have already converged will keep contributing to find other regions of the true Pareto front.

3.3.4 Criterion to Detect Nonimal Convergence

The detection of the nominal converge in the islands is achieved by monitoring the movements in the external archive. The population of an island is near to the nominal convergence if at each generation only a very few individuals of the archive are dominated by the primary population. To detect convergence, we keep a record of the number of dominated individuals of $PF_{known}(i)$ at generation i ($dominated_i$).

Depending on $dominated_i$ is decided whether an island has reached the convergence as to increase the resolution of the population. Every generation is checked to see if in the k last generations the condition $\sum_{i=1}^{k} \frac{dominated_i}{k} \leq \epsilon$ is satisfied, where ϵ is the average of dominated individuals of PF_{known} in the last k generations. The constant ϵ was experimentally set to 0.05. If the above condition is met, then the island resolution is increased to the next resolution. The resolution change is accomplished in the following way: one half of the population is randomly reinitialized taking into account the new resolution. The other half is a random sample taken from the archive.

4 Results and Discussion

4.1 Methodology to Evaluate the Effectiveness

As a reference point to evaluate the performance of our proposal we used an island model implementation of the NSGA-II (PNSGA-II) with a ring topology. The test problems used in this paper are described in Table 2. In order to evaluate the effectiveness of our approach we used four metrics: the *spacing* metric is intended for measuring the distribution; for evaluating the closeness to the Pareto-optimal front we used the *inverted generational distance*[2], the *two set coverage*[3], and the *success counting*[4]. Finally to evaluate the spread we also used the *inverted generational distance*. Besides showing the viability of the proposed approach, we are interested in analyzing the gain in convergence with increasing number of processors. For this purpose, we set the subpopulation size of each processor to the minimum so that both algorithms still achieve acceptable effectiveness (30 individuals in this study). Thus, we could expect that the effectiveness increases with the number of processors. In these experiments both algorithms are run 30 times varying the number of processors (1, 4, 8, 12 and 16). For the sake of a fair comparison, in all problems we used the same parameter setting. The number of evaluations is fixed to 18000 and the size of PF_{known} is 100. The simulation runs were carried out in a cluster of 16 dual nodes whose characteristics are described in Table 1.

Characteristic	Description
CPU	Intel Xeon; 2.45MHz
Number of nodes	16 (2 processors per node)
Memory	2 GB per node
Operating System	Red Hat Linux 3.2.2-5
Communications network	FastEthernet
Communications library	MPICH 1.12

Table 1: Characteristics of the cluster used in the experiments.

4.2 Test Suite ZDT

The size of the search space in the problems adopted is $2^{510} \approx 3.352 \times 10^{153}$ (using 5 digits of precision).

¿From the SC graphs (Figs. 6a and 6b) is clear that MRMOGA outperforms PNSGA-II when more than one processor is used on problems ZDT1 and ZDT2. The IGD values (Figs. 6e– 6g) of both algorithms are very similar for the three problems, which indicates that the algorithms converged very closely to the true Pareto front. With one processor, MRMOGA achieves a poor performance in both problems as reflected by the C metric presented in Figure 4. On problems ZDT1 and ZDT2, the nondominated fronts reported by MRMOGA-1 are 100% covered by PNSGA-II-1, while the former covers 0% of the PNSGA-II-1 outcomes. However, when more processors are used, is remarkable the improvement of MRMOGA in all metrics, especially in convergence since from 1 to 4 processors SC is improved about 100 times in ZDT1 and around 1000 times in ZDT2. It is important to note that when MRMOGA uses one processor

[2]This metric measures the progress of PF_{known} towards PF_{true} and the spread of PF_{known} onto PF_{true}.

[3]This metric calculates the fraction of vectors in B which are weakly dominated by A: $\mathcal{C}(A, B) = |\{b \in B \mid \exists\, a \in A : a \preceq b\}|/|B|$.

[4]This metric counts the number of vectors in PF_{known} that belong to PF_{true}.

there is no division of the search space. Using more than one processor, MRMOGA substantially outperforms PNSGA-II both in ZDT1 and ZDT2. Comparing the best coverage results achieved by each algorithm on ZDT1, PNSGA-II-16 weakly dominates less than 6% of the MRMOGA-16 outcomes. Whereas the nondominated fronts produced by the latter cover about 65% of the PNSGA-II-16 outcomes. A similar behavior is observed using 4-12 processors. In ZDT2 the difference in convergence is more noticeable.

With reference to ZDT3, the SC metric shows that MRMOGA has better convergence than PNSGA-II when 8 processors are used (Figure 6c). Considering the SC metric and the $\mathcal{C}$ comparison, it can be noted that only when MRMOGA uses 1 or 4 processors, PNSGA-II performs better. When both algorithms use 8 or more processors, MRMOGA weakly dominates a larger fraction of the generated fronts of PNSGA-II. Again, it is interesting the improvement in the convergence of MRMOGA when adopting more than one processor (Figures 6c and 6g).

Concerning the distribution using SP, PNSGA-II performs better than MRMOGA in the three problems. However, as can be seen in the plots of Figure 2, the distribution of MRMOGA is very competitive. Also, it is noted that both algorithms cover all regions of the true Pareto front.

4.3 Test problem OSY

In the plots of Figure 3 is shown that MRMOGA is able to maintain vectors near to all regions composing the Pareto-optimal front of OSY. On the contrary, PNSGA-II, for some runs, does not converge to certain regions of the front, especially the segment DE. Additionally, based on SC, MRMOGA outperforms PNSGA-II for every number of processors since the former achieves a larger number of vectors belonging to PF_{true} (Figure 6d). Nevertheless, it is interesting to note that PNSGA-II outperforms MRMOGA in the $\mathcal{C}$ comparison (see Fig. 4). For instance, the fronts of MRMOGA-n ($n = 4, ..., 16$) weakly dominate around 50% of the PNSGA-II-n outcomes, whereas PNSGA-II-n, covers at most 84% of the nondominated fronts of MRMOGA-n. A possible reason for this discrepancy is that the vectors of PF_{known} reported by MRMOGA that do not belong to PF_{true} (and maybe some that do) are weakly dominated by the PF_{known} set generated by PNSGA-II. This conjecture is reinforced by the fact that the values of IGD obtained by PNSGA-II are marginally better than those achieved by MRMOGA (Fig. 6h).

It is worth mentioning that using one processor (that is, without exploring using multiple resolutions), MRMOGA was not able to find any vector that belongs to PF_{true}. Conversely, with more than one processor, MRMOGA reported some solutions that belong to PF_{true}. This is an indication that searching through different resolutions allows to yield solutions that would not be possible (or easy) to find without this scheme.

With respect to the distribution, PNSGA-II obtains the best results in SP (Figure 6(l)), although MRMOGA is competitive as can be seen in the graphs of Figure 3.

4.4 Methodology to Evaluate the Efficiency

In the experiments presented in this section the metrics were calculated based on the orthodox speedup suggested by Alba [1]. This speedup allows a fair comparison since both the serial and the pMOEA are stopped when they have reached the same target solution "quality". In order to calculate the speedup, in the experiments of this section the pMOEA is stopped when it reaches a certain value of the hypervolumen metric (HV), which is defined in advance.[5] In our experiments we assume that the set $PF_{known}\binom{p}{t}$ in the higher resolution island reflects closely to PF_{known}, therefore only for this island we calculated HV. Unfortunately, for PNSGA-II we cannot proceed in a similar way as all islands contribute equally to form the known Pareto front. As a consequence, we do not include an efficiency comparison between PNSGA-II and MRMOGA. Additionally to the speedup, we use the *efficiency* metric and the *serial fraction* [9] which measure the performance of a parallel algorithm on a fixed-size problem.

In this paper only the results for the ZDT1 problem using the strict topology are shown. The orthodox speedup of MRMOGA is calculated in the following way: the serial algorithm is MRMOGA-16 (16 islands) running on a single processor. On the other hand, the parallel algorithm is MRMOGA-16 on an increasing number of processors (from 2 to 16). The population size of each island is set to 25 ($16 \times 25 = 400$ individuals) for all runs. Note that is possible that some processors have unequal workload[6] depending on the number of processors (e.g., with 6 and 10–14).

4.4.1 Evaluation Using the Orthodox Speedup

As can be seen in Figure 5a, with any number of processors MRMOGA achieves a sublinear speedup, but it is near-linear for some number of processors. From 2 to 8 processors the speedup tends to be superlinear as reveals the F metric, since this decreases with the number of processors (Fig. 5c). However, between 10 and 14 processors the speedup begins to decay, and for 16 processors the speedup is again near to the linear speedup. This behavior is due to the load imbalance when 10–14 processors are used. In a balanced situation the low resolution islands are faster than the high resolution islands, but the imbalance causes that the low resolution islands turn into the slower ones (since there are two islands in the same processor). Therefore, the high resolution islands stay idle waiting for the immigrants coming from the low resolution islands, which causes the speedup drops.

If we consider a balanced environment (e.g., excluding 10–14 processors) the efficiency seems to get better whenever a processor is added as shows the graph in Figure 5b. In a similar manner, F decreases with an increasing number of processors (Fig. 5c). This behavior might be caused by the addition of cache memory. Whichever this factor is, we

[5] Due to time constraints, we did not use hypervolumen in the effectiveness assessment.

[6] The unequal workload is due to the fact that MPI assigns cyclicly the processes (islands) to the available processors

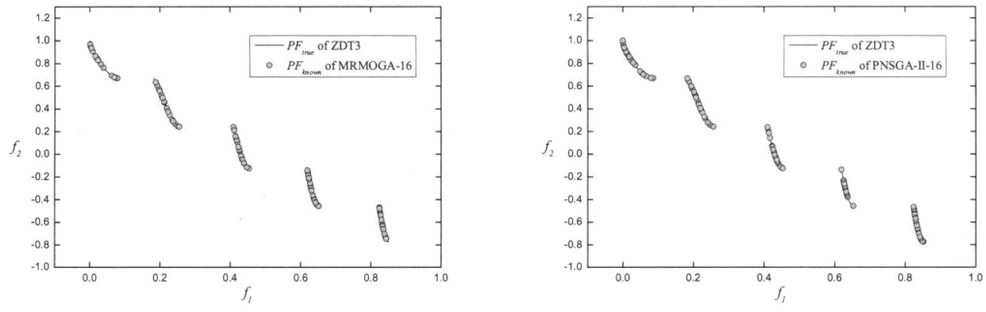


Figure 2: PF_{known} obtained by MRMOGA (left) and PNSGA-II (right) using 16 processors on ZDT3.

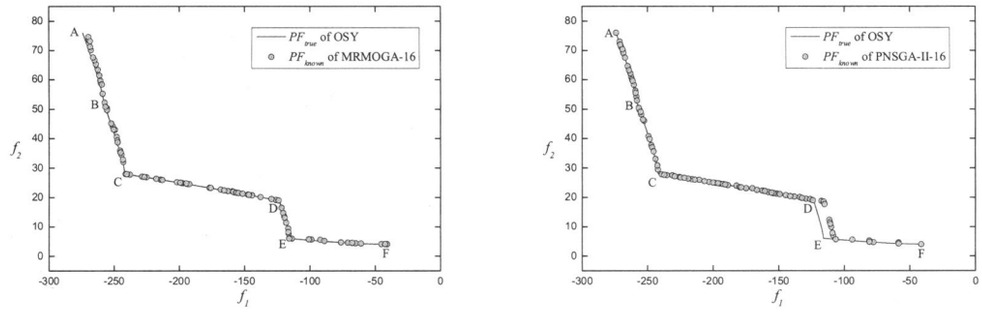


Figure 3: PF_{known} obtained by MRMOGA (left) and PNSGA-II (right) using 16 processors on OSY.

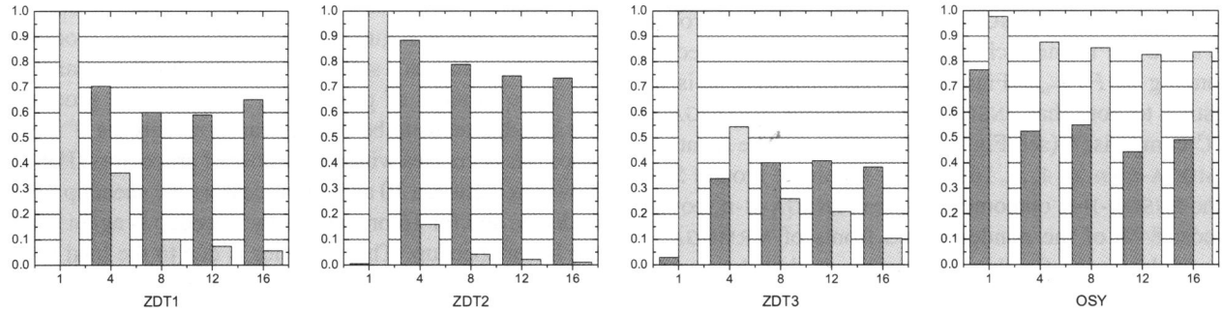
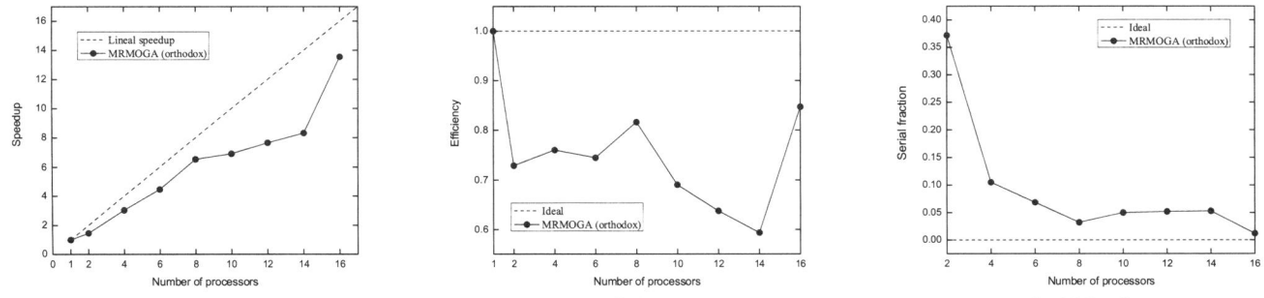


Figure 4: Comparison between MRMOGA and PNSGA-II using the $\mathcal{C}$ metric. Each pair of bars represents the $\mathcal{C}$ value for each number of processor. The left bar indicates $\mathcal{C}$(MRMOGA,PNSGA-II), while the right one $\mathcal{C}$(PNSGA-II,MRMOGA).



(a) Orthodox speedup (b) Effi ciency (c) Serial fraction

Figure 5: Results of the efficiency of MRMOGA according to the orthodox speedup, efficiency, and serial fraction on ZDT1.

can say that MRMOGA has great scalability as reveals the decrease of F. Additionally, the overhead due to communications and synchronization increases very slowly as more processors are used as shows E in Figure 5b.

5 Conclusions and Future Work

The main goal of this paper was to propose a novel parallel multiobjective evolutionary algorithm. Our approach is based on the island model and is characterized by dividing the decision space through an encoding of the solutions with a different resolution for each subpopulation.

¿From the comparative study we can conclude that the proposed scheme to divide the decision space improves noticeably the convergence of a pMOEA. In all problems considered[7] PNSGA-II performs better than MRMOGA when using one processor[8]. However, using more than one processor, MRMOGA surpasses PNSGA-II. Furthermore, we observed that the scheme of MRMOGA has more merit in problems with large search spaces such as the ZDT problems. Using multiple resolutions not only improves convergence, but also yields solutions that would be difficult to find otherwise (as in problem OSY). The main weakness of our approach is the poor distribution at the "fine-grain level". That is, although its PF_{known} covers all the extent of PF_{true}, the vectors are not equidistantly spaced. Also, it is interesting to realize that in some problems the convergence (considering the SC metric) of an algorithm reached its maximum with a certain number of processors and then dropped. This means that it makes no sense using more processors because the convergence will not increase. This implies that the algorithm is not able to take advantage of an increasing number of processors or, in other words, the algorithm has poor scalability. This behavior suggests to consider both the gains in convergence and the speedup to decide whether or not a pMOEA has good scalability. Since the distribution is the weakest point of our approach, as part of our future work we want to integrate a new mechanism to achieve a better distribution. An option to cope with this problem is the use of ϵ-dominance [11]. This technique could be integrated to the optimizer of each island or applied at the final of the search process when the root processor combines all the non-dominated fronts.

Acknowledgments

The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies in Computer Science at the Sección de Computación of the Electrical Engineering Department at CINVESTAV-IPN. The second author acknowledges support from CONACyT through project No. 42435-Y.

[7] Alongside the four problems of this paper, five more problems were considered in the experiments but not included here.

[8] By using only one processor each serial optimizer is isolated, so that MRMOGA do not divide the search space and PNSGA-II is reduced to the original NSGA-II.

Bibliography

[1] E. Alba Torres. Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters, Elsevier*, 82(1):7–13, April 2002.

[2] E. Alba Torres and J. M. Troya Linero. A survey of parallel distributed genetic algorithms. *Complexity*, 4(4):31–51, 1999.

[3] J. Branke, T. Kaußler, and H. Schmeck. Guidance in Evolutionary Multi-Objective Optimization. *Advances in Engineering Software*, 32:499–507, 2001.

[4] E. Cantú Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Boston: Kluwer Academic Publishers, 2002.

[5] C. A. Coello Coello, D. A. V. Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Nueva York: Kluwer Academic Publishers, May 2002.

[6] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In M. S. et al., editor, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[7] K. Deb, P. Zope, and A. Jain. Distributed Computing of Pareto-Optimal Solutions with Evolutionary Algorithms. In C. M. F. et al., editor, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 534–549, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[8] T. Hiroyasu, M. Miki, and S. Watanabe. The New Model of Parallel Genetic Algorithm in Multi-Objective Optimization Problems–Divided Range Multi-Objective Genetic Algorithm–. In *2000 Congress on Evolutionary Computation*, volume 1, pages 333–340, Piscataway, New Jersey, July 2000. IEEE Service Center.

[9] A. H. Karp and H. P. Flatt. Measuring parallel processor performance. *Communications of the ACM*, 33(5):539–543, 1990.

[10] J. D. Knowles and D. W. Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Washington, D.C., July 1999. IEEE Service Center.

[11] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, Fall 2002.

[12] S.-C. Lin, W. F. Punch III, and E. D. Goodman. Coarse-grain genetic algorithms, categorization and new approaches. In *Sixth IEEE Symposium on Parallel and Distributed Processing*, pages 28–37, Dallas, Texas, USA, October 1994. IEEE Computer Society Press.

[13] A. Osyczka and S. Kundu. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, 10:94–99, 1995.

[14] F. Streichert, H. Ulmer, and A. Zell. Parallelization of Multi-objective Evolutionary Algorithms Using Clustering Algorithms. In C. A. Coello Coello and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Fourth International Conference, EMO 2005*, pages 92–107, Guanajuato, México, 2005. Springer-Verlag. Lecture Notes in Computer Science. Volume 3410.

[15] M. Tomassini. Parallel and distributed evolutionary algorithms: A review. In K. Miettinen et al., editor, *Evolutionary Algorithms in Engineering and Computer Science*, pages 113–133. John Wiley and Sons, 1999.

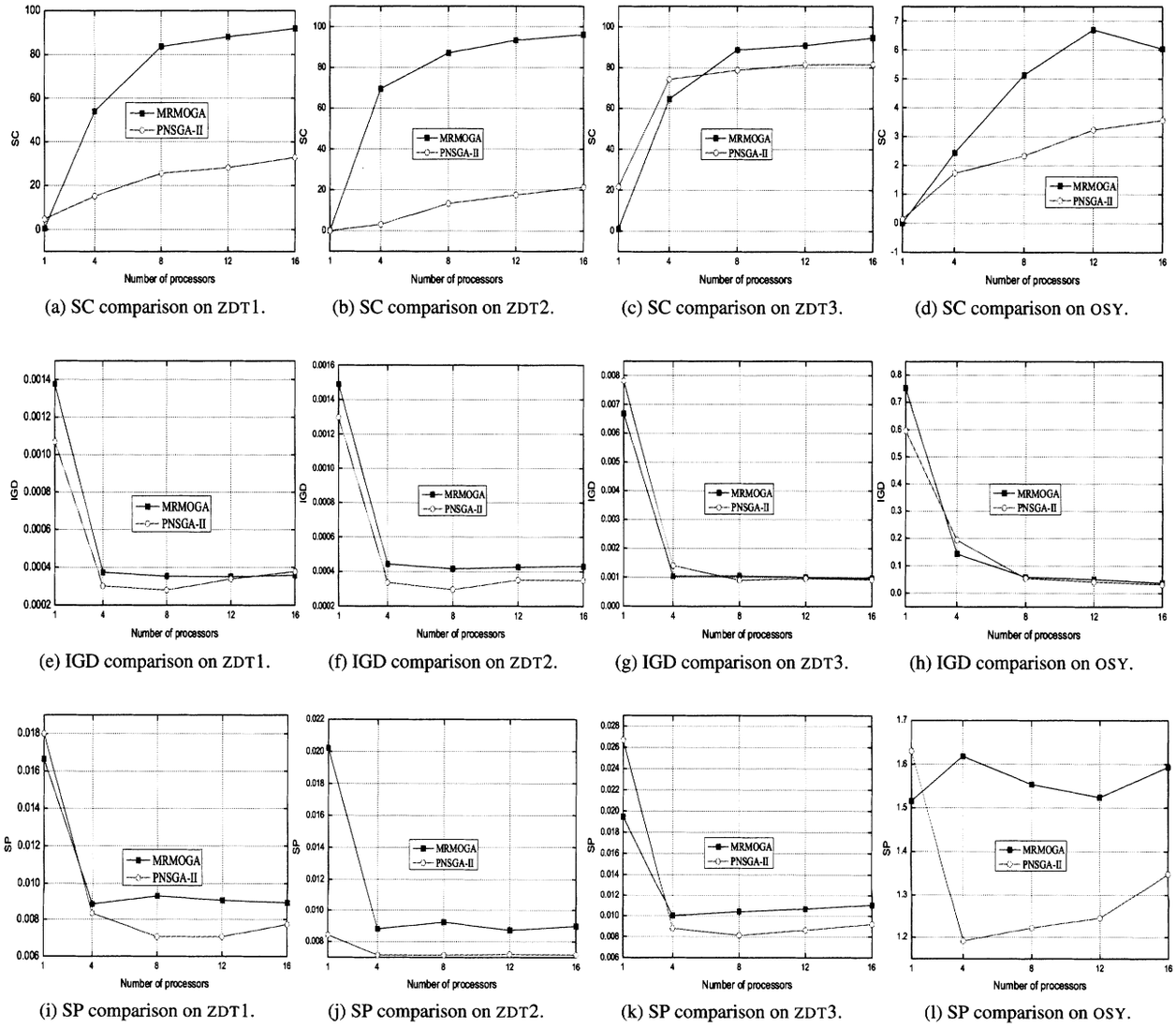
[16] Z.-Y. Zhu and K.-S. Leung. Asynchronous Self-Adjustable Island Genetic Algorithm for Multi-Objective Optimization Problems. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 837–842, Piscataway, Nueva Jersey, May 2002. IEEE Service Center.

[17] E. Zitzler, K. Deb, and L. 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.

MOP	n	Constraints	Objective functions*	Characteristics
ZDT1 [17]	30	$0 \leq x_i \leq 1 \quad i = 1, \ldots, n$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (f_1/g(\mathbf{x}))^{1/2}\right]$ $g(\mathbf{x}) = 1 + 9 \times \sum_{i=2}^{m} x_i/(m-1)$	convex
ZDT2 [17]	30	$0 \leq x_i \leq 1 \quad i = 1, \ldots, n$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (f_1/g(\mathbf{x}))^{2}\right]$ $g(\mathbf{x}) = 1 + 9 \times \sum_{i=2}^{m} x_i/(m-1)$	non-convex
ZDT3 [17]	30	$0 \leq x_i \leq 1 \quad i = 1, \ldots, n$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (\frac{f_1}{g(\mathbf{x})})^{1/2} - \frac{f_1}{g(\mathbf{x})}\sin(10\pi f_1)\right]$ $g(\mathbf{x}) = 1 + 9 \times \sum_{i=2}^{m} x_i/(m-1)$	convex, discontinuous
OSY [13]	6	$0 \leq x_1, x_2, x_6 \leq 10$ $1 \leq x_3, x_5 \leq 5, \quad 0 \leq x_4 \leq 6$ $0 \leq x_1 + x_2 - 2$ $0 \leq 6 - x_1 - x_2$ $0 \leq 2 - x_2 + x_1$ $0 \leq 2 - x_1 + 3x_2$ $0 \leq 4 - (x_3 - 3)^2 - x_4$ $0 \leq (x_5 - 3)^2 + x_6 - 4$	$f_1(\mathbf{x}) = -(25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2)$ $f_2(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$	Both P_{true} and PF_{true} are disconnected

*All objective functions are intended to be minimized

Table 2: Test problems used in this paper.



(a) SC comparison on ZDT1. (b) SC comparison on ZDT2. (c) SC comparison on ZDT3. (d) SC comparison on OSY.

(e) IGD comparison on ZDT1. (f) IGD comparison on ZDT2. (g) IGD comparison on ZDT3. (h) IGD comparison on OSY.

(i) SP comparison on ZDT1. (j) SP comparison on ZDT2. (k) SP comparison on ZDT3. (l) SP comparison on OSY.

Figure 6: Effectiveness of MRMOGA and PNSGA-II according to the number of processors on all problems.