ALTERNATIVE STRATEGIES FOR ASYNCHRONOUS MIGRATION-CONTROLLED SCHEMES IN PARALLEL GENETIC ALGORITHMS

OCHOA C. GALLARD R.
Proyecto UNSL-338403[1]
Departamento de Informática
Universidad Nacional de San Luis (UNSL)
Ejército de los Andes 950 - Local 106
5700 - San Luis, Argentina.
E-mail: {cochoa, rgallard}@inter2.unsl.edu.ar
Phone: + 54 652 20823
Fax : +54 652 30224

ABSTRACT

Migration of individuals allows a fruitful interaction between subpopulations in the *island model*, a well known distributed approach for evolutionary computing, where separate subpopulations evolve in parallel. This model is well suited for a distributed environment running a Single Program Multiple Data (SPMD) scheme. Here, the same Genetic Algorithm (GA) is replicated in many processors and attempting better convergence, through an expected improvement on genetic diversity, selected individuals are exchanged periodically. For exchanging, an individual is selected from a source subpopulation and then exported towards a target subpopulation. Usually, the imported string is accepted on arrival and then inserted into the target subpopulation. Our earlier experiments on controlled migration showed an improvement on results when contrasted against those obtained by conventional migration approaches.

This paper describes extended implementations of alternative strategies to oversee migration in asynchronous schemes for an *island model* and enlarges a previous work on three processors with a set of softer testing functions [9]. All of them try to decrease the risk of premature convergence. A first strategy attempts to prevent unbalanced propagation of genotypes by applying an acceptance threshold parameter to each incoming string. A second one permits independent evolution of subpopulations and acts only when a possible stagnation is detected. In such condition an attempt to evade falling towards a local optimum is done by inserting an expected dissimilar individual to improve genetic diversity. A third alternative strategy combines both previous mentioned strategies. The results presented are those obtained on the functions that showed to be more difficult for the *island model* using a replication of a simple GA. A description of the corresponding system architecture supporting the PGA implementation is described and results for the parallel distributed approach among 3, 6 and 12 processors is discussed.

KEYWORDS: Parallel genetic algorithms, island model, migration schemes, acceptance threshold, dynamic arbiter.

[1] The Research Group is supported by the Universidad Nacional de San Luis and the ANPCYT (National Agency to Promote Science and Technology).

1. INTRODUCTION

Parallel genetic algorithms (PGAs), attempt to exploit the intrinsically parallel nature of genetic algorithms. In his first work Holland [5] recognised the parallel nature of the reproductive paradigm and the intrinsic efficiency of parallel processing.
A particular subclass of the Levine's [7] taxonomy of parallel genetic algorithms are the Coarse Grained Parallel Genetic Algorithm (CGPGA). Here, the entire population is compounded of a number of subpopulations distributed among multiple processors. Each processor runs a sequential GA on their own subpopulation and interacts exchanging chromosomes. In this manner, each subpopulation explores a different area of the searching space, maintains its highest fitness individual (elitism) and carries out its migration to other subpopulations.
CGPGAs can be further classified according to the migration scheme, interconnection topology and homogeneity of processing nodes (Shyh-Chang Lin [10]).
Chromosome migration is a key characteristic of CGPGA, which helps to maintain genetic diversity by inserting strings arriving from separately evolved subpopulations. A migration scheme determines how frequently and under which time constraints string exchange is made.
Asynchronous migration schemes are well suited for a network of workstations where dissimilar computer architectures and workloads cause different evolution speeds. In this case migration is allowed at any moment independently of the evolution state of subpopulations. This asynchronous behaviour reflects the kind of migration that, in fact, happens in nature where diverse populations have distinct evolution paces.
Interconnectivity of processing nodes determines which nodes are considered as the neighbours of a particular one, for string exchanging. Interconnection schemes can be subdivided into two main groups: *static* and *dynamic*.
In a *static interconnection scheme,* node connections are defined at the beginning of the run and once for all. In a *dynamic interconnection scheme* the initial topology may be modified during execution.
Homogeneity in parallel genetic algorithms refers to the similitude of the GAs running in different nodes. In an *homogeneous CGPGA* model the GA executed in each processor have the same parameters set, genetic operators and objective function while in an *heterogeneous CGPGA* model any of this features can be different.
Many researchers have extensively investigated the island model, initially Tanese [11] and Cohoon et al. [3], and more recently Whitley [12] and Belding [2]. Another work [8] on PGAs implementation, corroborated that, usually, asynchronous schemes behave better than synchronous schemes in a distributed system of workstations running popular test functions.
The present work shows the outcomes of asynchronous migration schemes when migration is controlled, in an attempt to prevent premature convergence. The control is carried on by an acceptance threshold parameter, a dynamic arbiter which decides at migration time according to the progress of evolution or a combination of both previous strategies. Some details on implementation are also discussed. Results in network of 3, 6,10,12 and 16 nodes are discussed.

2. THE SYSTEM SUPPORTING PGA EXECUTION

Architectures corresponding to virtual and real nodes are displayed in figure 1. To support parallel execution of GAs, a Single Program Multiple Data approach [4] resulted appropriate.
Two parallel schemes were supported. The first one with real processors, as depicted in the upper part of figure 1, limited the number of islands to the number of available workstations. The second one combining real and virtual processors, as depicted in the lower part of figure 1, allowed to extend considerably the number of islands.

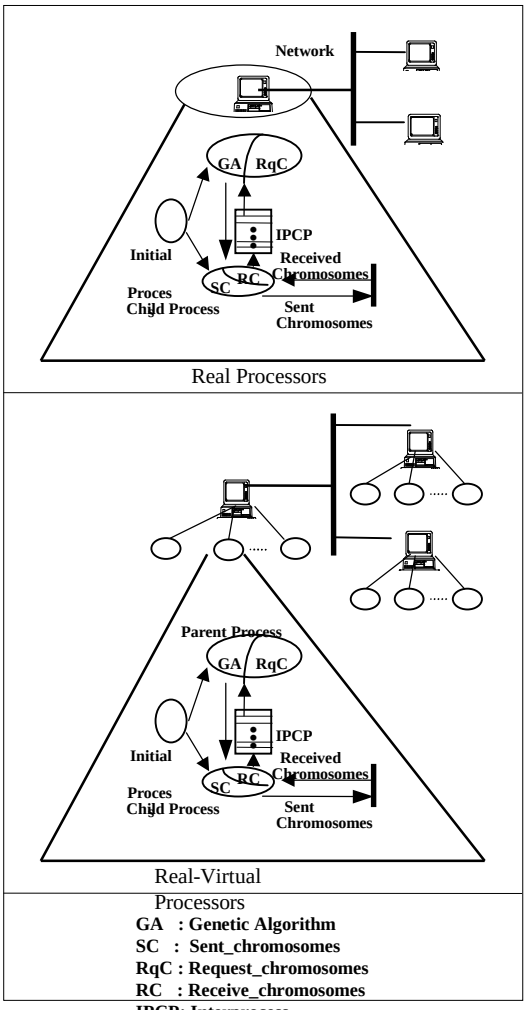


Fig. 1. System Architecture to Support Migration

As expected, the whole process was slowed down under this later architecture but processing time was not a main consideration. After specifying a number of input parameters to begin execution of a PGA, the initial process in each processor forked once. The parent process was responsible of GA execution and of reques-ting of migration services. The child process was in charge of managing arrival of chromosomes from remote processors, forwarding them to the parent process (importing), and sending local migrating chromosomes throughout the network to other subpopulations (exporting).

Both parallel approaches were implemented by using Interprocess Communications Primitives (IPCPs) and Application Program Interfaces (APIs).

Two primary processes, *send_chromosomes* and *receive_chromosomes* were the main components performing the interaction with other processes of the distributed system. Because the local area network is highly reliable and small amount of data was transmitted each time, these routines used a socket interface and a connectionless protocol (UDP) to minimise communication overhead [1].

Sending and receiving chromosomes needed synchronisation. A class of non blocking IPCP was used in order to allow progressing the GA even if chromosomes did not yet arrived from remote processors.

3. ALTERNATIVE STRATEGIES FOR MIGRATION

To favor genetic exchange between subpopulations the asynchronous scheme, under these strategies, allowed the same prefixed number of chromosome migrations (exporting) from each processor. Therefore, different migration intervals were defined for each workstation.

To prevent loss of genetic diversity, when choosing a *victim* for replacement, we used a policy consisting in a random selection of two candidate strings which then were subjected to a probabilistic tournament for ultimate decision [7].

Maximum Gap Allowed Strategy (MGAS)

The rational for this strategy relies in some unexpected behaviour encountered during the tuning of the model under a *simple static scheme* which exported an individual to a predefined neighbour in a ring topology. The interaction of subpopulations was done via the migration of the best individual, which always was accepted on arrival. Incidentally, running the model on *f5* , it was detected that a high performer migrating string from workstation W_2 (slower) had a fitness value of 38.19 while the corresponding value for the best individual of the supposedly more evolved population on W_1 (faster) was 35.95. This was the appealing fact to think about an strategy for avoiding premature convergence by adding a new parameter θ in the model.

When we are dealing with asynchronous migration, the insertion of differently evolved (time-aged) strings is likely to favor genetic diversity. But also, because subpopulations evolve at different speeds at least two extreme consequences of migration can be expected:

- The incoming individual was originated in a low evolved population. Depending on the population size, evolution level and selection scheme we can expect a null or slight influence of the new arrived string on global fitness but also a beneficial contribution to maintain or increase genetic diversity. This helps to explore new searching areas.

- The incoming string arrives from a high evolved population and is close to a local optima. In this case if the gap between the fitness of the new individual and that of the best local individual is very large then a risk of premature convergence can arise.

In order to avoid premature convergence, by contagion, the MGAS strategy was devised. For this strategy a parameter θ, called *maximum gap allowed*, was defined as the maximum (percentile) difference accepted between the best local individual fitness and the incoming string fitness. So, the local GA executing in the destination node decides to accept or reject the imported string. Therefore, if the external string is superior than the best local individual beyond a certain threshold θ it will be rejected, otherwise it will be inserted into the subpopulation. In other words, if the following acceptance criteria (θ test) holds:

$$\text{fitness}_{\textbf{external}} - \text{fitness}_{\textbf{local}} \leq \theta \quad (0 \leq \theta \leq 1)$$

then accept insertion of the incoming string, otherwise reject string.
By using this strategy the influence of high evolved external strings was decreased and, consequently, the risk of falling into local optimum also declined.
The interconnectivity scheme used under this strategy was that of an *static logical ring* (the neighbour of $\text{node}_{\mathbf{i}}$ is $\text{node}_{(i+1) \bmod n}$, if the number of processors is n).

Dynamic Arbiter Strategy (DAS)

Under DAS a global arbiter resolves if a migrated chromosome should be inserted or not into some subpopulation. This decision is based on the knowledge the arbiter has about the evolutive progress of subpopulations, hence exerting a sort of dynamic convergence control.
At migration time, under this strategy, instead of exporting a single chromosome to its neighbour, the process managing the chromosome exchange in each node exports a packet to the arbiter containing the following data structure:

Subpop struct:
source node address,
best individual chromosome,
worst individual chromosome,
best individual fitness,
worst individual fitness,
subpopulation mean fitness

On its side, at each migration arrival, the arbiter updates information about the best and worst global individuals and subpopulation fitness. Also, information about the best individual of the first migration is maintained on hand. This is supported through the following internal data structures:

Arbiter struct I:
best global (fitness,chromosome,owner host)
worst global (fitness,chromosome,owner host)
best first migrated (fitness,chromosome,owner host)

Arbiter struct II:
$Subpop_1$ mean fitness
$Subpop_2$ mean fitness
...
$Subpop_n$ mean fitness

In more detail, when the arbiter receives a packet, from the source, the following actions take place:

- If it is the case of the first migration then updates its internal data structures.
- Otherwise, updates its internal data structures and to determine[2] the progress of the evolutive process, compares the current value of the mean fitness of source subpopulation with the last updated corresponding value and,
  - If they stay similar (possible search stagnation) a migration of an individual to the source subpopulation will take place.
  - Otherwise (search improves results) no action take place.

To determine which individual to migrate the following criteria was adopted:

if the best global individual does not reside in the
source subpopulation
then migrate the best global individual
else migrate the worst global individual

Giving to the arbiter the faculty to migrate, or not, a global individual (originated in any node) to the source node results in a *dynamic interconnection scheme*. See figure 2.

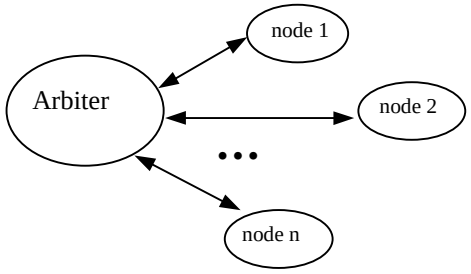


Fig. 2. Interconnectivity through Arbiter

Combined MGA-DA Strategy (CMGA-DAS)

Finally, the combined application of both previous strategies was also examined by simply adding to DAS the acceptance criteria imposed by θ. So, the migration criteria applied for this strategy was:

[2] Other criteria to establish population evolution progress are also being implemented.

if the best global individual resides in the source
subpopulation
then migrate the worst global individual
else if θ *test* holds for the best global individual
then migrate the best global individual
else if θ *test* holds for the best first
migrated individual
then migrate the best first
migrated individual[3]
else migrate the worst global
individual

4. EXPERIMENTS DESCRIPTION

A large set of runs were performed for our experiments. To achieve subpopulations interaction, with and without migration arbitration, sets of 3, 6, 10, 12 and 16 real and virtual processors were used. The *island model* was run on the set of test functions indicated below, solving optimisation problems. (See the appendix on fitness landscapes)

f1: Easom's Function

$$f(x_1, x_2) = -\cos(x_1)\cdot\cos(x_2)\cdot e^{-((x_1-\pi)^2+(x_2-\pi)^2)},$$

$$x_1, x_2 \in [-100, 100]$$
$$-1 \quad in \ (\pi, \pi)$$

$$f(x_1, x_2) = 0.5 - g(x_1, x_2) \quad h(x_1, x_2), \text{ where}$$

$$g(x_1, x_2) = \left[\cos(x_1)\cdot\cos(x_2)\cdot e^{-((x_1-\pi)^2+(x_2-\pi)^2)}\right]^{(2\cdot m)-1},$$

$$h(x_1, x_2) = \frac{0.5}{1.0 \quad 0.000\ 1((x1-\pi)^2+(x2-\pi)^2)^{2\cdot m}},$$

$$x_1, x_2 \in [-100, 100], \ m = 1..10$$
$$0 \quad in \ (\pi, \pi)$$

$$f(x_1, x_2) = 0.5 + \frac{sin^2\sqrt{x_1^2+x_2^2}-0.5}{1.0 \quad 0.000\ 1(x_1^2 \quad x_2^2)^2}, \ for;$$
$$-100 \le x_i \le 100, \ i = 1, 2$$
$$0$$

f4: Shubert's Function

$$f(x_1, x_2) = \sum_{i=1}^{5} i \cdot \cos[(i+1) \cdot x_1 + i] \cdot \sum_{i=1}^{5} i \cdot \cos[(i+1) \cdot x_2 + i],$$

$$-10 \leq x_i \leq 10, i = 1,2$$

minimum global value: −186.73

$$\cdot \quad \pi \cdot \quad \cdot \quad \pi \cdot$$

$$- \leq \quad \leq \quad \leq \quad \leq$$

$$f(x_1, x_2) = (x_1^2 + x_2^2)^{0.25} \cdot (sin^2(50 \cdot (x_1^2 + x_2^2)^{0.1}) + 1.0)$$

$$-100 \leq x_1, x_2 \leq 100$$

minimum global value: 0 *in* $(0,0)$

θ, a value of 0.05 was used.

5. RESULTS

In this section we will show some results obtained when using the first three above mentioned strategies for 1, 3, 6 and 12 processors, contrasting them against the *simple static scheme*, on the first five functions. Results for experiments no reported here are similar to those actually shown.

According to the number of processors the following number of generations were assigned ;

1	24,000 (the sequential GA)
3	8,000
6	4,000
12	2,000

[3] The best first migrated individual is a good intermediate value which contributes to genetic diversity.

The single processor case, Sequential Genetic Algorithm, (SGA) was included to be contrasted with those in parallel (PGA). In the latter case each node exported 10 individuals per run.
After the runs were completed, mean values for the following relevant performance variables were determined:

Optimal Hits = (# optimal hits / # runs)
It is the hit ratio to find the optimal solution, all over the total number of runs.

Ebest = (*opt_val* - best value / *opt_val*)100
It is the percentile error of the best found individual when compared with *opt_val*[4]. It gives us a measure of how far are we from that *opt_val*.

The following tables and graphs show a report of experimental results. All the values in the tables are mean values obtained from the multiple run series.

5.1 Unimodal Functions

f1: Easom's Function

S1: $P_{cross} = 0.50$, $P_{mut} = 0.005$

Ebest values

#nodos	Static	MGAS	DAS	CMGA-DAS
3	0.021795	0.014531	0.017298	0.018336
6	0.0089957	0.0076117	0.013147	0.011763
12	0.0051896	0.0017298	0.0096873	0.0031201
1	0.021450			

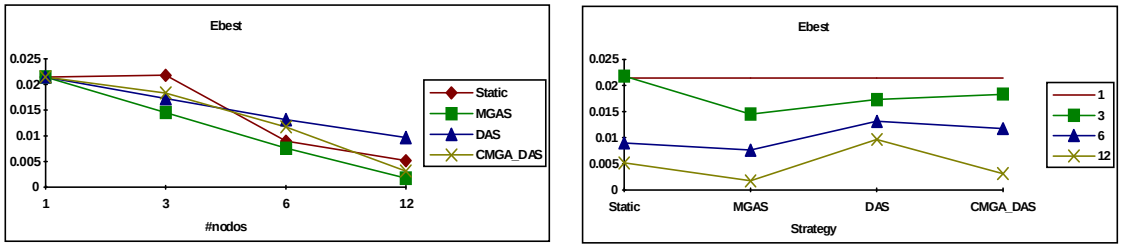


Fig. 3. Ebest values for function *f1*

Optimal Hits

#nodos	Static	MGAS	DAS	CMGA-DAS
3	0.28	0.46	0.36	0.3
6	0.56	0.66	0.41	0.55
12	0.75	0.90	0.56	0.83
1	0.20			

[4] *opt_val* is the known, or estimated, optimum value.

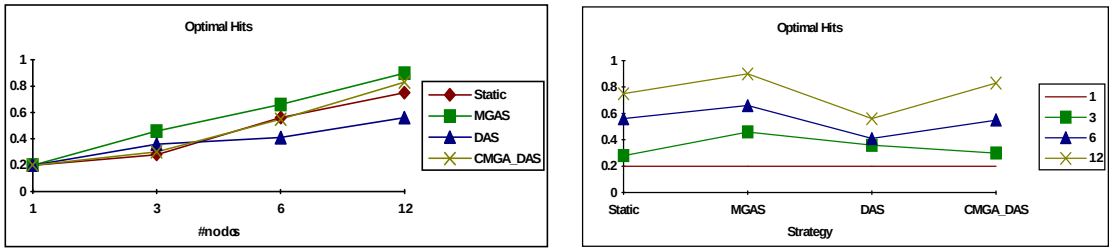


Fig. 4. Optimal Hits for function *f1*

In the above figures and tables it can be observed that *Ebest* (from 0.02% to 0.002%) and *Optimal Hits* (from 28% to 90%) attain better values as the number of processors are augmented. About strategies, those using the threshold parameter θ, MGAS and CMGA-DAS, show better performance.
Similar results were found with parameter set S3. However when S2 is used it cannot be observed a regular behaviour of the performance variables under any strategy. Consequently it can be concluded that for the Easom's function low mutation probabilities leads to poor convergence (eg;*Ebest* values range between 13% and 26% and *Optimal Hits* varied from 11% to 25%).

f2: Volcano function

Results are reported for S3 parameter set . Similar outcomes were obtained under S1. Using S2 the function was extremely hard and never a near optimal value was reached under any strategy.

S3: P_{cross} = 0.65 and P_{mut} = 0.005

Ebest values

#nodos	Static	MGAS	DAS	CMGA-DAS
3	1.6784	4.1743	7.5211	3.3396
6	16.688	1.6997	4.1702	3.3382
12	9.1748	9.1800	7.5117	4.0713
1	0.011072			

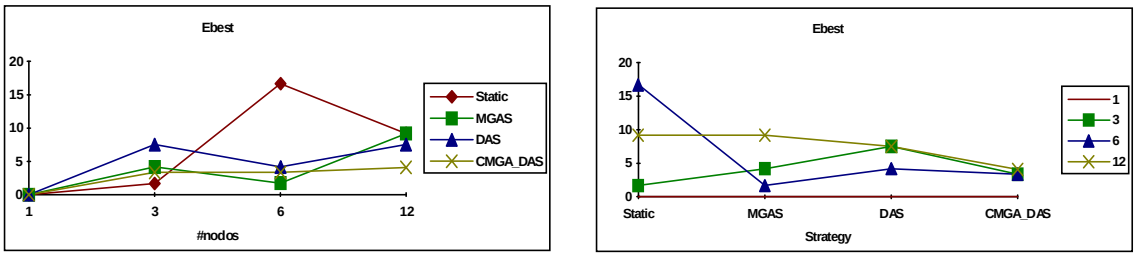


Fig. 5. Ebest values for function *f2*

Optimal Hits

#nodos	Static	MGAS	DAS	CMGA-DAS
3	0.68	0.73	0.71	0.71
6	0.41	0.65	0.83	0.78
12	0.46	0.46	0.53	0.63
1	0.73			

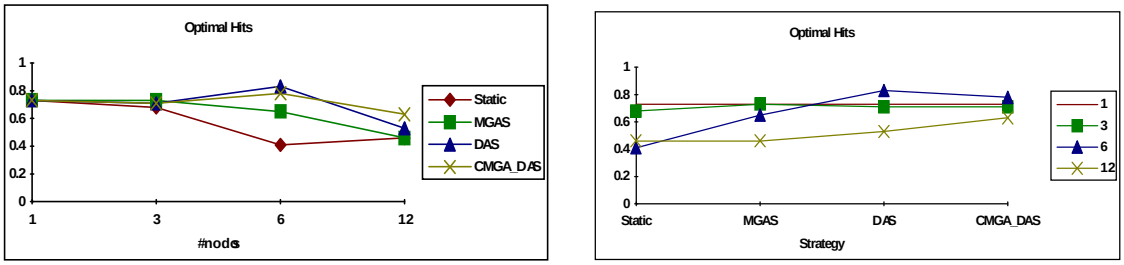


Fig. 6. Optimal Hits for function *f2*

High *Ebest* values come from those individuals who did not fall into the "hole". The Volcano function is hardest to optimise than the Easom's function and the best values are obtained with lesser number of processors, possibly, because in this case a greater number of generations are performed. *Optimal Hits* achieve a maximum value (83%) under DAS and 6 processors.

For this reason a set of new experiments were performed with the Volcano function allowing 24000 generations in all processors. A summary is following on:

S3: P_{cross} = 0.65 and P_{mut} = 0.005

Ebest values

#nodos	Static	MGAS	DAS	CMGA-DAS
3	1.6784	0.0048434	0.0076116	1.6715
6	0.84164	0.0	0.0069196	0.0
12	0.0027680	0.0	0.0	0.0
1	0.011072			

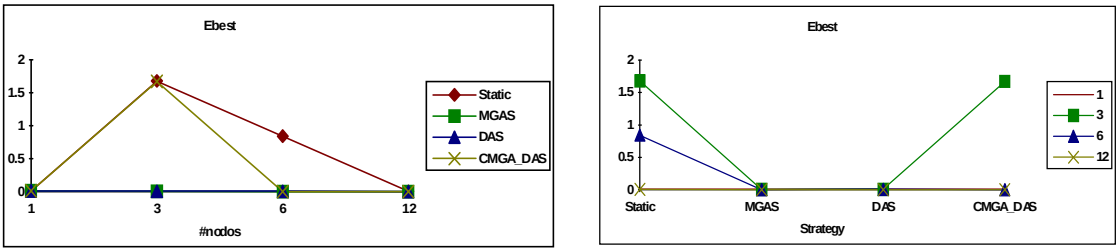


Fig. 7. Ebest values for function *f2* (24000 gen)

Optimal Hits

#nodos	Static	MGAS	DAS	CMGA-DAS
3	0.68	0.92	0.83	0.88
6	0.81	1.0	0.85	1.0
12	0.93	1.0	1.0	1.0
1	0.73			

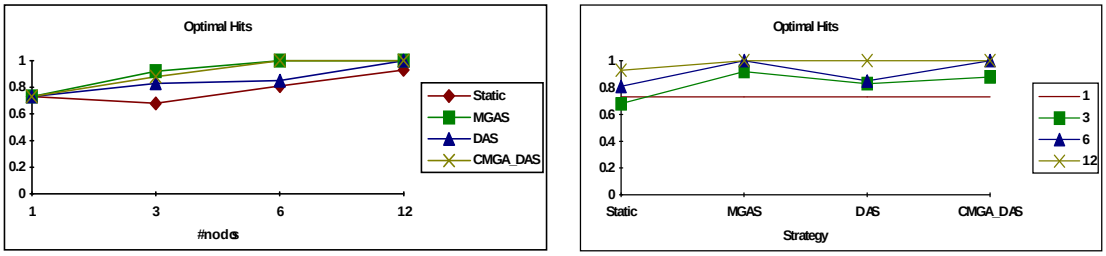


Fig. 8. Optimal Hits for function *f2* (24000 gen)

In this case the above figures and tables indicates that better performance is achieved for large number of processors and generations where the number of *Optimal Hits* reach a 100% (the optimum was found on each run). Again strategies MGAS and CMGA-DAS perform better for any number of processors.

5.2 MULTIMODAL FUNCTIONS

f3: Schaffer function F6

S1: $P_{cross} = 0.50$, $P_{mut} = 0.005$

Ebest values

#nodos	Static	MGAS	DAS	CMGA-DAS
3	0.065577	0.050912	0.062380	0.055839
6	0.049241	0.047628	0.042701	0.047599
12	0.021350	0.032847	0.032849	0.042672
1	0.073876			

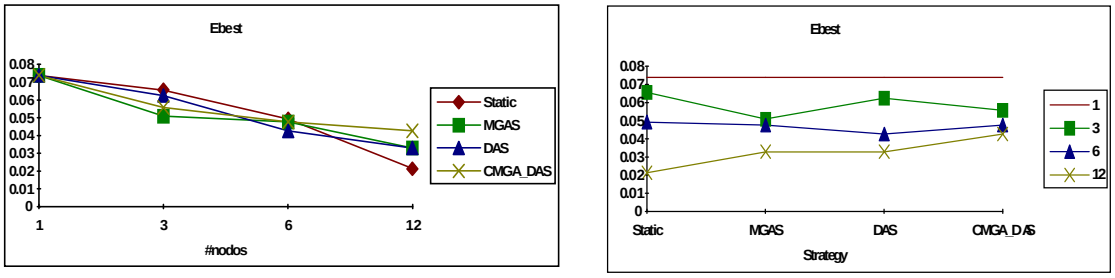


Fig. 9. Ebest values for function *f3*

Optimal Hits

#nodos	Static	MGAS	DAS	CMGA-DAS
3	0.53	0.48	0.41	0.43
6	0.55	0.51	0.56	0.56
12	0.78	0.66	0.65	0.60
1	0.30			

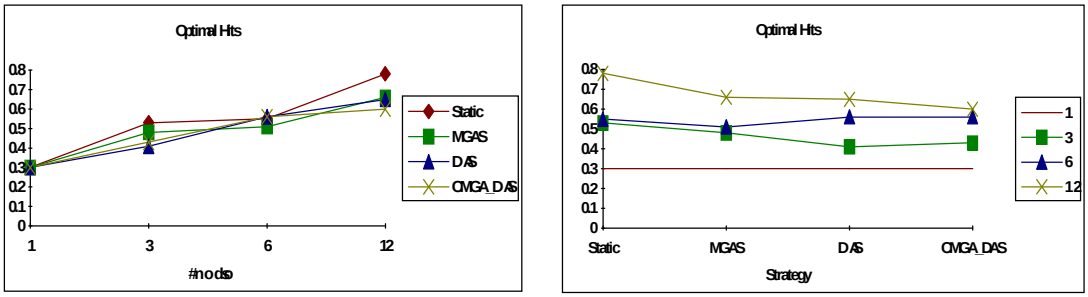


Fig. 10. Optimal Hits for function *f3*

For parameter sets S1 and S3 the behaviour showed by the PGA was similar. *Ebest* values under any of the three proposed strategies are better than those under the static (no controlled) strategy, except for 12 nodes were the latter was better. *Optimal Hits* values were in general better under the Static strategy. Both performance variables improve as the number of nodes incremented. This function showed to be one of the most difficult multimodal functions. Values

with the S2 set are not so good as with S1 or S3. Mutation plays also an important role on this landscape. Here the main hole must be found in the way to the optimum.

f4: Shubert's Function

S1: $P_{cross} = 0.50$, $P_{mut} = 0.005$

Ebest values

#nodos	Static	MGAS	DAS	CMGA-DAS
3	6.5785E-09	7.7443E-09	8.9894E-09	5.9236E-09
6	2.0808E-01	8.6381E-09	6.1309E-09	7.1009E-09
12	5.0545E-09	6.4626E-09	6.3455E-09	6.1927E-09
1	9.2404E-09			

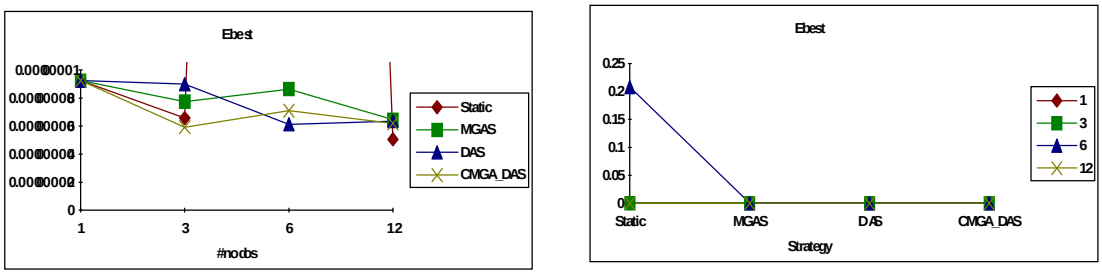


Fig. 11. Ebest values for function *f4*

Optimal Hits

#nodos	Static	MGAS	DAS	CMGA-DAS
3	0.10	0.15	0.08	0.20
6	0.16	0.15	0.20	0.22
12	0.18	0.14	0.15	0.20
1	0.13			

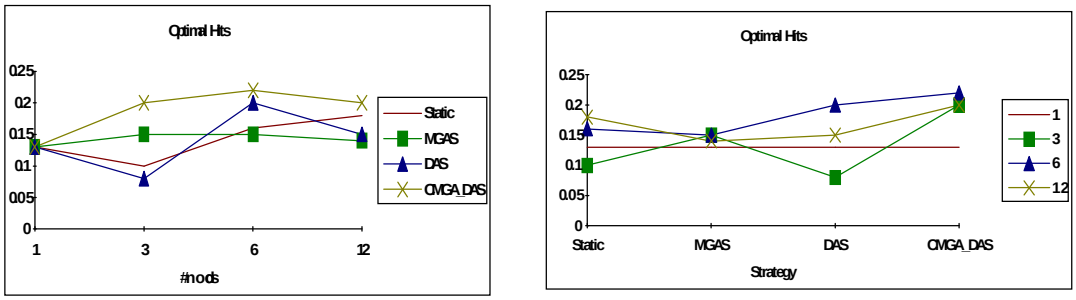


Fig. 12. Optimal Hits for function *f4*

Even if the number of *Optimal Hits* is low for function *f4 Ebest* values are quite good. This means that the fitness landscape is hard to prevent finding the optimum during a PGA run but on the other hand near optimal solutions are feasibly reached. Using the S2 set results are poorer (for both *Ebest* and *Optimal Hits*), than those obtained with the other two parameters sets.

f5: Highly multimodal Michalewickz's function

S3: P_{cross} = 0.65 and P_{mut} = 0.005

Ebest values

#nodos	Static	MGAS	DAS	CMGA-DAS
3	3.2840E-03	3.2840E-03	3.2840E-03	1.9349E-03
6	1.3683E-03	5.4734E-04	0.0	2.7367E-04
12	0.0	0.0	0.0	0.0
1	2.4630E-03			

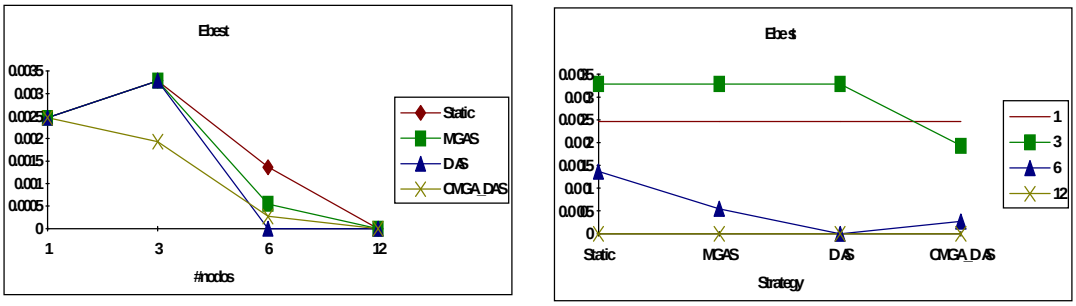


Fig. 13. Ebest values for function *f5*

Optimal Hits

#nodos	Static	MGAS	DAS	CMGA-DAS
3	0.80	0.80	0.83	0.86
6	0.92	0.97	1.0	0.98
12	1.0	1.0	1.0	1.0
1	0.85			

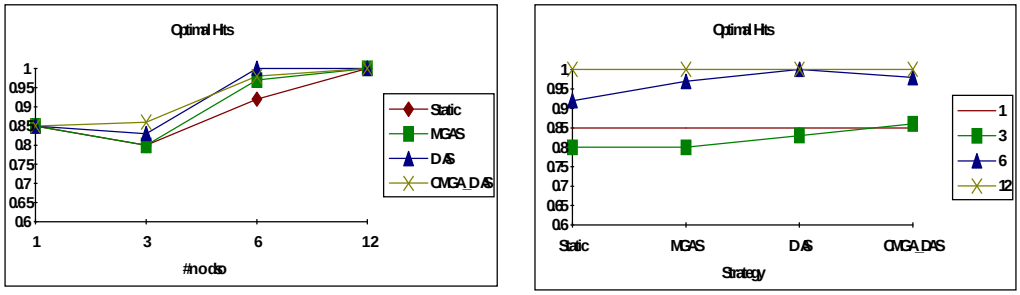


Fig. 14. Optimal Hits for function *f5*

For function f5 the best values were attained. For parameter sets S1 and S3 the proposed strategies outperformed the static alternative. The values are improving as long as the number of processors is increased arriving to a 100% of optimal values found when the number of processors is 12. Using S2, even though the values observed are not so good as with the other parameter sets, almost a 100% of optimum values are also found for the larger number of processors. Also here the new strategies outperformed the static one.

6. CONCLUSIONS

Here we discussed the effect of three new strategies to control migration in asynchronous Parallel Genetic Algorithms distributed in a network with different number of processors involved in PGA execution. Results show always the outstanding performance of any asynchronous parallel scheme when compared with the sequential one. It is worth remarking that the base for the evolutionary approach, upon which outcomes rely here, is the weakest one; a simple GA. Given the difficulty showed by the selected testing functions, the experiments were devised with three sets of parameters in order to alter mutation and crossover to find better results. Two kind of problems were addressed for optimisation: unimodal and multimodal.

The Easom's and the (hardest) Volcano functions are good representatives of the first class of problems; to find a needle in a haystack. For them, MGAS and CMGA-DAS were the strategies showing better performance. Even if DAS did not show better behaviour than the Static one we must remember that the determination of the convergence is made via a very simple means; mean population fitness. Another method is now being implemented to measure diverse degrees of convergence for future studies. In every case *Optimal Hits* increases accordingly with increments in the number of processor, arriving to 90% under MGAS for 12 nodes. The Volcano function definitively needs a much greater number of generations and when this is allowed very good results are obtained in either variable for 6 and 12 processors. As a conclusion, for hard unimodal functions, it is observed that those strategies using the parameter θ outperform the remaining strategies.

For the second class of problems, difficult highly multimodal functions of varied landscapes were chosen. Here there cannot be detected a clear preeminence of one strategy over the others but, in general, the proposed strategies show better behaviours than the Static strategy on either variable. Always the performance variables values were better with parameters set S1 and S3. This result reinforce the assumption that higher values of P_{mut} helps the searching process for hard functions.

Finally we want to remark that PGAs implementation are notably superior than SGA implementation in either aspect; quality of results and processing time. These results are even improved by using the strategies proposed in this paper.

In view of these promising results, which where obtained using the simplest GA model, new experiments for more complex and varied migration-controlled strategies are being devised.

Addressing to tune selective pressure and to increase the contribution to genetic diversity, in future work, these experiments will run under hybrid schemes combining diverse crossover and selection methods.

7. ACKNOWLEDGEMENTS

We acknowledge the cooperation of the project group for providing new ideas and constructive criticisms. Also to the UNSL and the CONICET from which we receive continuous support.

8. REFERENCES

[1] Arredondo D., Printista M., Gallard R. - Un Sistema Distribuido para el Procesamiento Paralelo de Algoritmos Genéticos - Proceedings of the Primer Congreso Argentino de Ciencias de la Computación, pp 242-252, Universidad Nacional del Sur, Bahia Blanca, October 1995.

[2] Belding T.C. - The Distributed Genetic Algorithm Revisited - Proceedings of the Sixth International Conference on Genetic Algorithms, pp 114-121, Morgan Kauffman, 1995.

[3] Cohoon J.P., Hedge S.U., Martin W.N., Richards D. - Puntuacted Equilibria: A Parallel Genetic Algorithm - Proceedings of the Second International Conference on Genetic Algorithms, pp 148-154, Lawrence Erlbaum Associates Publishers, 1987.

[4] Foster I. - Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering - ISBN 0-201-57594-9, Addison Wesley, 1994.

[5] Holland J.H. - Adaptation in Natural and Artificial Systems, Ann Arbor, The University of Michigan Press, 1975.

[6] Horn J., Goldberg D. - Genetic Algorithms Difficulty and the Modality of Fitness Landscapes - Fundations of Genetic Algorithms (FOGA-94), pp 243 -269, Morgan Kaufmann, 1995.

[7] Levine, D. - A Parallel Genetic Algorithm for the Set Partitioning Problem - Ph D Thesis, Illinois Institute of Technology and Argone National Lab. (ANL -94/23), 1994.

[8] Ochoa C., Gallard R. - Parallel Genetic Algorithms: A Feasible Distributed Implementation - Proceedings of the Segundo Congreso Argentino de Ciencias de la Computación, pp 189-199, Universidad Nacional de San Luis, San Luis, November 1996.

[9] Ochoa C., Gallard R. - Strategies for Migration Overseeing in Asynchronous Schemes of Parallel Genetic Algorithms - Accepted for it publication in the International ICSC Symposium on Soft Computing: SOCO'97.

[10] Shyh-Chang Lin, Punch W., Goodman E. - Coarse-Grain Parallel Genetic Algorithms: Categorizations and New A pproach. Parallel & Distributed Processing, Dallas TX, Oct. 1994.

[11] Tanese R. - Distributed Genetic Algorithms - Proceedings of the Third International Conference on Genetic Algorithms, pp 434-439, Morgan Kauffman, 1989.

[12] Whitley D. - An Executable Model of a Simple Genetic Algorithm - Fundations of Genetic Algorithms-2 (FOGA-92), pp 45-62, Morgan Kaufmann, 1993.

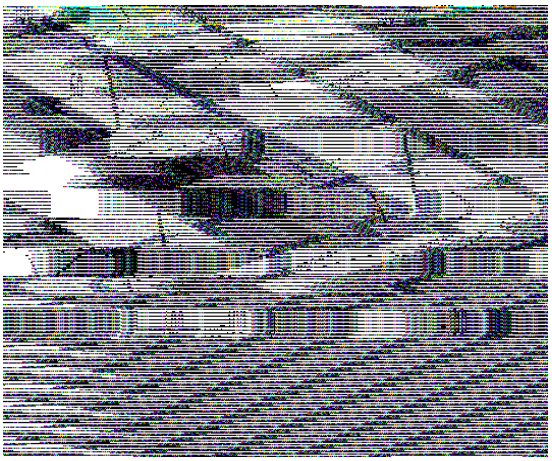
[13] Whitley D., Mathias K., Rana S., Dzubera J. - Building Better Test Functions - Proceedings of the Sixth International Conference on Genetic Algorithms, pp 239 -246, Morgan Kauffman, 1995.

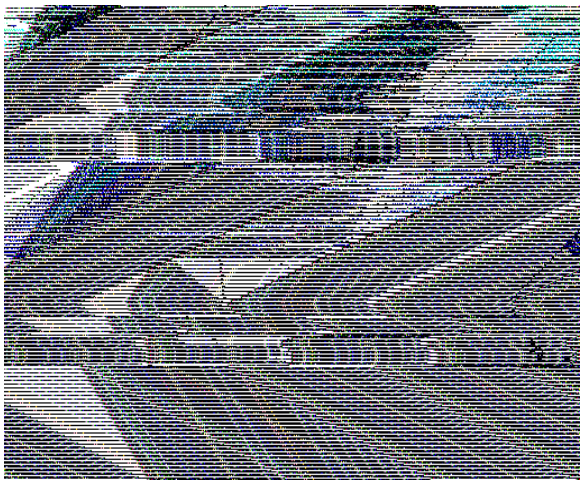
APPENDIX A: FITNESS LANDSCAPES

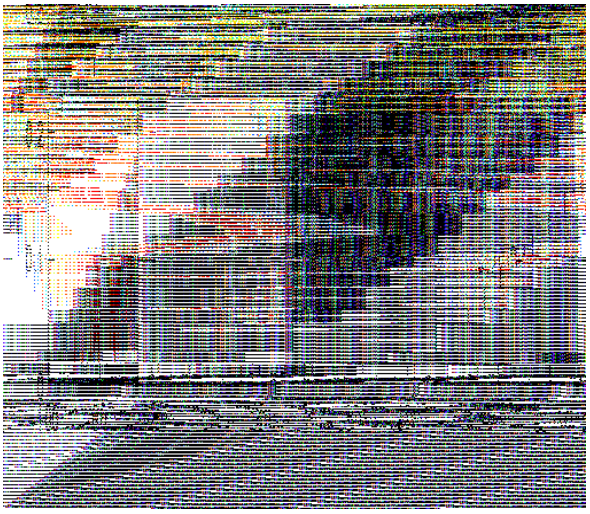
SCHAFFER F6 FUNCTION

$$(\quad _1, \quad _2) \quad 0.5 + \frac{^2\sqrt{\quad _1^2 \quad \quad _2^2} - 0.5}{[1.0 \quad 0.0001 (\quad _1^2 \quad \quad _2^2)]^2} ,$$

$$_1 , \quad _2 \in [-100 , 100]$$



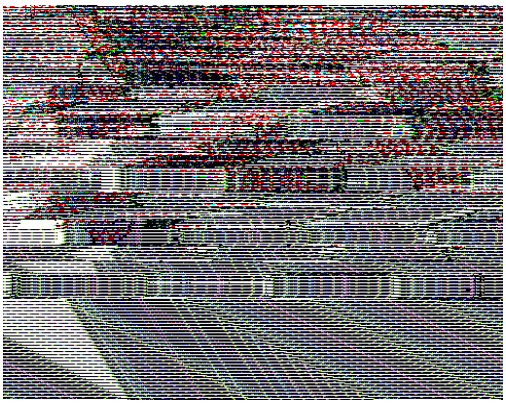


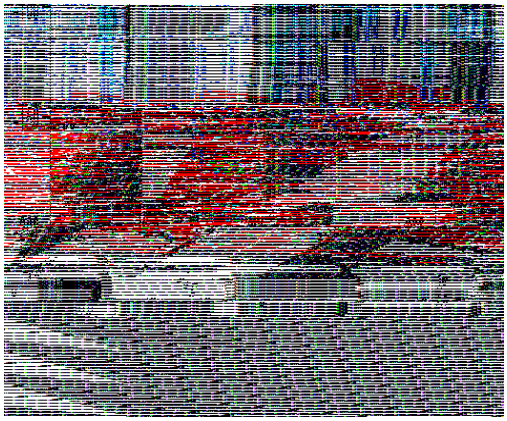


SHUBERT FUNCTION

$$(\ _1, \ _2) = \left[\ \sum^{5} \ _1 \cdot \cos[(\ \ 1) \cdot \ _1 + i] \right] \cdot \left[\sum_{i=1}^{5} i \cdot \cos[(\ \ 1) \cdot \ _2 \ \ 1] \ \right],$$

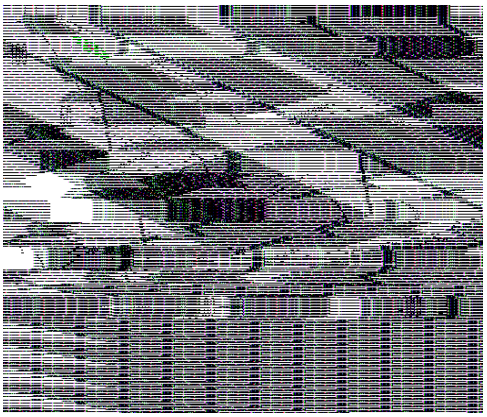
$$\in [-10,10]$$

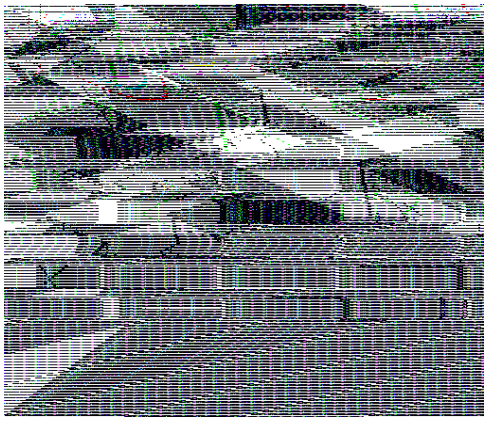




$$- \quad . \quad . - \quad -\pi \quad + \quad -\pi$$

$$\in -$$





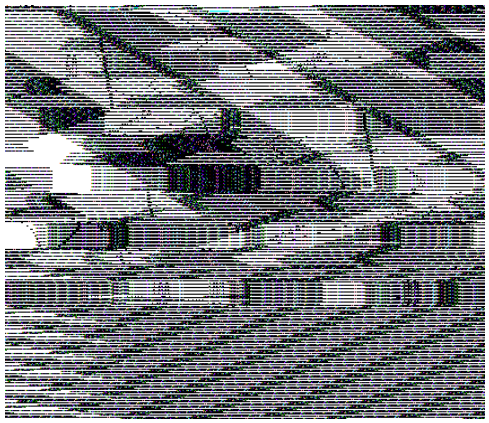
VOLCANO FUNCTION

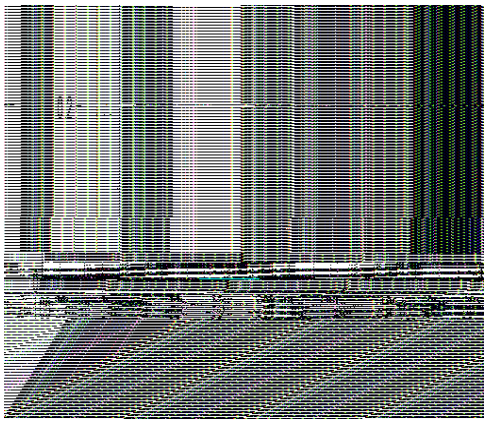
$$_1(\ _1,\ _2)\quad \left[\cos(\ _1)\cdot\cos(\ _2)\cdot e^{-((x_1-\pi)^2\ (\ _2-\pi))}\right]^{(2\cdot\ )-1},$$

$$_2(\ _1,\ _2)\quad \frac{0.5}{1.0\quad 0.0001\left[(\ _1-\pi)^2\quad(\ _2-\pi)^2\right]^{2\cdot}},$$

$$(\ _1,\ _2)\quad 0.5-\ _1(\ _1,\ _2)\quad\ _2(\ _1,\ _2),$$

$$\in[-100,100],\quad =1..10$$





$$\cdot\qquad \pi\qquad \cdot\qquad \pi$$

$$\in\ -\qquad\qquad \in$$