Machine Learning Methods to Analyze Migration Parameters in Parallel Genetic Algorithms

S.Muelas, J.M.Peña, V.Robles, A. LaTorre, P. de Miguel

CeSViMa, Computer Architecture Department, Universidad Politécnica de Madrid
smuelas@cesvima.upm.es, {jmpena,vrobles,atorre,pmiguel}@fi.upm.es

Abstract. Parallel genetic algorithms (PGA) are a powerful tool to deal with complex optimization problems. Nevertheless, additional parameters are required to configure properly their performance. The task to select these parameters accurately is an optimization problem by itself. Any additional help or hints to adjust the configuration parameters will lead both towards a more efficient PGA application and to a better comprehension on how these parameters affect optimization behavior and performance. This contribution offers a preliminary analysis on some of the PGA parameters, like migration frequency, topology, connectivity and number of islands and population sizes. The study has been carried out after an intensive set of experiments to collect PGA performance on several representative problems. The results have been analyzed using machine learning methods to identify behavioral patterns that are labeled as "good" PGA configurations. This study is a first approach to generalize extracted patterns from different problems into a configuration hints and suggested parameters.

Keywords: parallel genetic algorithms, migration parameters, migration topologies, optimization problem hardness.

1. Introduction

Parallel genetic algorithms (PGA) are powerful optimizations tools for complex real-world scenarios. Besides the straightforward parallel implementation of genetic algorithms (master/slave models), PGA have been a subject of research by themselves. Superlineal speed-ups reached by the most sophisticated PGA models (cellular and island models) have opened interesting questions about their particular behavior. Theoretical analysis [1] has been carried out to explain this behavior, but these studies consider simplified models that are not used in practice when their application is proposed for a complex optimization problem.

Genetic algorithm application requires many parameters to be configured; the number of these parameters is greater in the case of PGA. This work presents a preliminary study on the relationship between these parameters and the algorithm performance. This study does not come from the theoretical point of view rather than a more analytical perspective from the results generated by many experimental runs.

The methodology proposed on this contribution is the following: (i) identify several PGA parameters to be analyzed, (ii) select a significant number of

optimization problems, (iii) execute several times each experiment for every combination of the selected parameters, (iv) compare the results obtained on each optimization problem and label "good" performing configurations, and (v) analyze each problem data separately to identify efficient configuration patterns and, then try to combine the experimental data to extract general patterns common for different optimization problems.

The work performed on this study answers interesting questions but opens even more new questions to be analyzed in further detail.

The outline of this paper is as follows. Section 2 presents an introduction of PGA taxonomy. In section 3, the experimental scenario is described in detail. Finally, section 4, presents the obtained results together with the discussion and the most relevant facts extracted from this analysis.

2. Parallel Genetic Algorithms

When tackling big optimization problems through genetic algorithms, the first alternative that arises is the parallelization of the algorithms. This idea has been the subject of many pages of literature about the so-called parallel genetic algorithms [1] (or PGA).

There are different ways of carrying out this parallelization task, and there are all included in taxonomy of different parallel approaches [2]:

- Master-slave models (also called centralized) are a more intuitive action line. It consists on the parallel execution of the evaluation stage of the objective functions. A central node executes the evolutionary algorithm, per se, including the necessary operators, but subsequently divides the work of each evaluation to slave nodes.
- Island models (also called coarse-grain); do represent a differentiated algorithmic structure. In this case, each node, with its own population, executes a different evolutionary algorithm. The evolving independent populations interact with each other through migration strategies that make the genetic information exchange among populations possible. These strategies are subject to certain frequencies and figures that represent the migration rates, and are also subject to a migration topology that indicates which population (island) are individuals migrating from, and to which other.
- Cellular models (also called fine-grain), can be considered as an extreme variant of the previous ones: nodes populations have only one individual. In these cases, in order to create new individuals genetic recombination need to select (from the nearest nodes) the best individual to perform mating.

2.1 Migration in Island-Model Parallel Genetic Algorithms

An important parameter on the performance of PGA (specifically in multipopulation models based on islands) is the migration strategy. This is configured through different parameters [1]:

1. Migration frequency: How often (generations) are individuals sent?

2. Migration rate: How many individuals migrate each time?
3. Immigrants' selection: How is it decided the individuals which migrate?
4. Replacement policy: How is the combination among immigrants and the original population made?
5. Migration topology: Where do individuals migrate to, which island sends individuals to which other?

Upon studying these parameters [3] has proved that, even though subpopulations with a smaller size risk of falling into local optimum, an appropriate migration strategy can stop a suboptimum solution from dominating all populations. This appropriate strategy must be adjusted in between the limits of a low interaction (which would practically imply the execution of N independent algorithms) and an excessive interaction (that would lead to the predominance of only one solution). A correct configuration can help us obtain better results with fewer evaluations [4].

2.2 Connection Topologies

For this study, several dynamic topologies have been selected. These topologies are a sample of the possible connection schemas in PGA, but in general covers both static and dynamic approaches.

1. **Static Hypercube Topology:** In static topologies, the neighbors for each island are fixed and cannot change during the generations. Among the different alternatives for static configurations, the hypercube configuration has been selected. Hypercubes can be modified according to the desired dimensionality, allowing multiple connectivity (2, 3, … n connections per node). Ring, double ring, mesh and other connection topologies are equivalent, at least in terms of structural connectivity, although hypercube keeps minimal diameter.
2. **Dynamic Random Topology:** Dynamic topologies reconfigure their neighborhood relationships at each migration step. The simplest strategy to figure out the appropriate neighborhood is to select connection nodes based on randomness. This topology has been selected as a reference point to compare other dynamic topologies.
3. **Dynamic Nearest Neighbor Topology**: Before the migration of individuals, each island determines the medoid of its population and broadcasts it to all other islands. In order to determine the neighbor relationships, each island uses the medoids and Gabriel relationship criterium. Grabriel neighborship is proposed in [5] and in contrast with Delaunay neighbors, with high-dimension problems, the neighbors can be computed efficiently. Two points i,j are Gabriel neighbors if there is no other point in the hypersphere with diameter:

(**1**)

Each island selects then the nearest Gabriel neighbors until the topology degree is reached. In case there were not enough Gabriel neighbors to reach the degree, the rest of the neighbors are completed with the nearest (not neighbors) islands.

4. **Dynamic Furthest Neighbor Topology**: This topology is similar to the previous one. Each island calculates its medoid and broadcasts it to the rest of the system so that each one can compute its Gabriel neighbors. The difference in this case, is that

the selected neighbors from the set of all Gabriel neighbors, are the furthest ones. Also, when not enough Gabriel neighbors are available, the completion of neighbors is based on the furthest not-neighbors ones.

5. **Hybrid Topology:** This topology is composed from a static hypercube topology and a dynamic nearest neighbor topology both with half of the degree of the hybrid one. If the topology is not a multiple of two, the integer part of the division by two is assigned to the dynamic part and the rest to the static part. For example with a degree of 3, the hybrid topology will have a static topology of degree two and a dynamic nearest neighbor topology of degree one. This implementation will allow us to analyze the beneficial effects of combining both static and dynamic topological approaches.

3. Experimental Scenario

Studies on the correct configuration of PGA have been made by different authors. In [6], the performance of synchronous and asynchronous PGAs is studied, for the case of given problem, studying the results for the algorithm model (steady-state/generational), fine/coarse grain and the number of processors. Nevertheless, this study was carried out with for single problem.

[7] presents an interesting analysis of the migration frequency, considering also different migration sizes (number of individuals), but keeping the same topology and connectivity. The experiments have been repeated several times for statistical significance. The authors used specially created functions as well as standard test functions (Rosenbrock, Schwefel, Rastrigin, and Griewank).

Relevant parameters have been selected, based on [2] their reported influence in PGA performance, and we have run several tests on a set of problems varying only these parameters between a representative collection of values:

1. **Topology**: Static Hypercube (dimensions=2, 3, and 4), Dynamic Random, Dynamic Nearest Neighbor, Dynamic Furthest Neighbor, and Hybrid Topology.
2. **Number of islands:** 4, 8 and 16 (keeping the same overall population).
3. **Migration frequency:** Migrations happen on every 5 or 10 generations.

3.1 Algorithm

The algorithm tested in this study is based on the canonical PGA [1] and for the running of these tests we have fixed some of its parameters:

1. **Initialization**: Random uniform initialization of the population.
2. **Selection**: Tournament-based.
3. **Crossover probability**: 1.0 / **Mutation probability**: 0.01
4. **Full elitism**: The best set of individuals of the next generation is composed from both the parents and the children of the actual generation.
5. **Migration rate**: Top 10% / **Replacement policy:** Replace the worst individuals of the host population only if the immigrants have better scores.
6. **Convergence**: Fixed number of generations, estimated as 80% of the required generations in an average configuration combination.

3.2 Problems

Problems have been selected with the idea to represent different optimization difficulties in both real-valued and binary representations. The problems range from simple scenarios to multimodal, epistatic and deceptive features. For this study, the following problems have been proposed: Griewank, Rastrigin, Sphere, Deceptive, Two-peaks, Holland's Royal-Road, Maxbit and Travelling Salesman Problem (TSP).

1. **Griewank**: It is a classical minimization multimodal problem with a local optimum at (0…0) and several local optima that makes traditional search algorithms converge in the wrong direction.
2. **Rastrigin**: Similar to the problem above, this function is a minimization multimodal problem with a local optimum at (0…0) and several concentrated local optima. This function is a fairly difficult problem due to its large search space and its large number of local minima.
3. **Sphere**: The simplest real problem analyzed, the sphere problem, is a relative easy optimization case, but it is useful to develop performance comparisons.
4. **Deceptive**: This deceptive maximization problem [8] has $2^{n/2\ -1}$ local optima and only 1 global optimum. One important feature that makes deceptive functions hard to solve is that the attractor around the global optimum is small. Deceptive functions cannot be efficiently solved by mutation only; however, an arbitrary recombination operator will not work in this case either. Here it is necessary that we learn the linkage between pairs of variables that contribute to the fitness through the same basis function.
5. **Two-peaks**: The n-dimensional two-peaks function [8] is also a maximization problem which haves one global optimum at (0.1,…,0.1) and several 2^n local optima for an n-dimensional function. Two-peak functions are practically unsolvable by mutation only, because the chances of hitting the attractor around the global optimum decrease exponentially with the size of the problem. On the other hand, recombination allows fast and reliable solution of two-peak functions by exploiting their decomposition and processing many partial solutions simultaneously**.**
6. **Holland's Royal-Road**: The Royal Road functions were designed by Holland and coworkers to highlight the building block approach the GA was thought to take in problem solving. We have used the Holland's 1993 ICGA version of the Royal Road Problem [9].
7. **Maxbit problem**: It is a classical easy binary problem for GA where the objective function tries to maximize the number of 1's.
8. **Travelling Salesman Problem (TSP)**: it is typical combinatorial problem used in the optimization literature. Given a finite number of 'cities', the objective is to find the shortest way of visiting all the cities in a Hamiltonian graph. There are different representations to deal with optimization problems using GAs [10], we have selected order representation (also called ranking representation. This coding assigns one (real) value to each of the cities. The cities are then sorted by this value (from lowest to highest value). This ordered list of cities defines the tour. The main advantage of this representation is that traditional crossover and mutation operators can be used, in our case we restrict the analysis to real-valued and binary representations.

In order to do the tests in the same conditions, we have fixed the following parameters. The objective of this study is not to optimize these parameters, they have been proposed as feasible ones, considering only the influence of the abovementioned parameters.

Table 1. Dimension, overall population size and problem-specific parameters.

Problem	Dimension	Generations	Pop. Size	Cross. Op.	Mut. Op.
Griewank	100	400	512	Blend	Gaussian
Rastrigin	100	400	512	Blend	Gaussian
Sphere	100	400	512	Blend	Gaussian
Deceptive	15	400	512	Blend	Gaussian
Two-Peaks	25	400	512	Blend	Gaussian
Royal-Road	209	400	512	Two Points	Bit Flip
Maxbit	700	400	512	Two Points	Bit Flip
TSP	48	400	512	Blend	Gaussian

3.3 Experiments Procedure

For each of the proposed problems the following experimental procedure is performed: (i) all the combinations of the input parameters are considered (60 different combinations), (ii) for each combination, 50 independent executions have been ran and the we record the fitness obtained when the maximum number of iterations is reached, (iii) the fitness results obtained for each one of the combinations of parameters is compared pair wise, using a Wilcoxon non parametric t-test with $p<0.001$. If one combination of parameters happens to be better than another (according to the t-test) the winning combination is granted with *+1 wins* and the losing combination penalized with *-1 wins*. As all the combinations are compared against each other, they are ranked (depending on how many other combinations are better/worse that it is).

Once all the combination of parameters are ranked, we label them as "**high performance configurations**" if $wins>=10$ and as "**low performance configurations**" if $wins<=-10$.

4. Analyzing the Results

Once experimental results have been labeled, each individual problem has been analyzed using a C5.0 rule-induction algorithm (ten fold cross-validation). The results on table 2 show the rules that classify high and low performance configurations, including rule support and accuracy. Most of the rules are based on migration frequency and the number of islands (it should be mentioned that as the global population is fixed, the higher the number of islands is the smaller the population per island becomes). Topology and connectivity paremeters are not considered in most of the rules and they only appear in low support rules.

Another relevant aspect comes from the existence of contradictory rules that identify either HIGH or LOW performance configurations depending on the problem (E.g

Royal Road and TSP High performance rule #1 vs. Griewank low performance rule #2).

Table 2. Induced rules from individual problems (support, accuracy)

Problems	High Performance	Low Performance
Griewank	1. #islands=4 (27%,1.0) 2. frequency=5 and #islands<=8 (32%,0.842)	1. #islands=16 (37%,0.818) 2. frequency=10 and #islands>4 (37%,0.72)
Rastrigin	1. frequency=5 and connectivity>2 (32%,1.0) 2. frequency=5 and #islands<8 (32%, 0.947)	1.frequency=10 (50%,1.0)
Sphere	1. frequency=5 (50%,0.80)	1. #islands=16 and frequency=10 (18%,1.0) 2. connectivity=2 and frequency=10 (18%, 1.0) 3. #islands>4 and frequency=10 and connectivity<=3 (23%, 0.929)
Deceptive	1. #islands=16 (37%,1.0)	1.#islands=4 (27%,0.875) 2.#islands<=8 and topology=random (17%, 0.8)
Two-peaks	1. frequency=5 (50%, 0.73)	1.frequency=10 (50%, 0.767)
Royal Road	1. frequency=10 and #islands>4 (37%,0.72)	1. frequency=5 and #islands<=8 (32%,0.73)
Maxbit	1. frequency=10 and #islands<=8 (32%,1.0) 2. #islands=4 and connectivity=2 (17%, 0.9)	1. frequency=5 and connectivity>2 (32%,1.0) 2. #islands>8 (37%, 0.818)
TSP	1. frequency=10 and #islands>4 (37%,0.636)	1. #islands=4 (27%,1.0) 2. frequency=5 (50%,0.667)

4.1 Problem Characterization

PGAs show different behavior patterns on each different problem. The results extracted before are a clear example of difficulties to figure out a general rule on the problem characterization.

The proposal that we are tabling is the following: with measures traditionally used to evaluate genetic algorithm hardness, it is now possible to label problems (with their parameters) to have extra information extend the inducted analysis performed before.

Studies carried out by [11] on the difficulty of different optimization problems, measure the complexity of the problem as the correlation between fitness values and the distance in the solutions space (fitness-distance correlation, or FDC). In the case of real-value problems Manhattan distance has been used, and binary encodings are measured using Hamming.

FDC requires information about the global optimum and its value. [12] has proved that the correlation metric between distance and fitness can also be applied on the individuals of a population without knowing the global optimum. This approach uses the best fitness value in the population as a reference. This approach is called local FDC (or LFDC). This approximation has been considered as an evaluation tool for the initialization processes quality. There is a lot of literature on algorithms that calculate the difficulty of a problem for an evolutionary algorithm. One interesting

example of this is [13]'s study; as it uses the FDC and other metrics (e.g. epistasis variance) for the purpose of generating problem equivalence classes. This work presents the measures used to characterize problem difficulty types.

Recently, [14] compares predictive and posterior measures of problem difficulty. The results show that there are cases in which predictive measures are limited in their accuracy, but they are a helpful tool to guess problem hardness.

On this contribution, we propose an additional difficulty measure. We have named this measure **fitness increment**. It scales the fitness of a given population from *0* to *1*. Then, the difference between the fitness of the best children minus the fitness of the best parent is calculated. The fitness of the best children is also scaled according to the same mechanism used with its parents (it could make scaled fitness values greater than *1* if the offspring is better than the overall best value of its parents' generation). This measure does not consider landscape difficulties, directly. It takes into account selection, crossover and mutation operators as well as their respective parameters.

4.2 General Discussion

In order to include difficulty measures, to characterize each of the problems, a random uniform population of 3000 individuals has been created. With all these individuals, FDC, LFDC and fitness_increment has been computed.

Table 3. Difficulty measures obtained for a random sample of 3000 individuals.

	FDC	LFDC	Fitness increment
Royal road	-0.133	0.006	-0.00376
Maxbit	1	0.18	0.00469
TSP	0.002	-0.009	-0.027
Griewank	0.96	0.46	0.15
Sphere	1.0	0.44	0.148
Rastrigin	0.72	0.32	0.08
Deceptive	-0.42	-0.01	-0.0042
Two-Peaks	0.19	0.21	0.007

A correlation analysis has been performed between the three metrics, resulting a higher correlation rate that validates Kallel's hypothesis [12] (FDC and LFDC), unexpectedly the only exception is Maxbit, but LFDC using 700-bit maxbit problem with uniform initialization would generate initial individuals with an average of 700/2 1's (really poor performance after initialization). There exists also a higher correlation result between FDC and fitness_increment. This is considerably interesting because FDC measures fitness landscape properties, while fitness_increment deals with the combination of the selection, crossover and mutation operators.

Then all the 60 parameters combinations taken from each problem are appended on a single table, including the columns that represent the 3 difficulty measures (FDC, LFDC, fitness increment).

This new data table is analyzed using also C5.0 (10 fold cross-validation). The following induced rules are extracted:

Table 4. General rules with support and accuracy (Sup./Acc.).

Conditions	Perf.	Sup./Acc.
fitness_increment>0.00468 and frequency=5 and #islands<=8	High	16%/0.934
fitness_increment in (0.00468,0.148] and frequency=5 and connectivity>2	High	12%/0.877
fitness_increment<=0.00468 and frequency=10	High	25%/0.55
fitness_increment in (0.00468,0.148] and frequency=10	Low	19%/0.844
fitness_ increment>0.00468 and frequency=10 and #islands>4	Low	18%/0.818
fitness_increment<=0.00468 and frequency=5	Low	25%/0.542

Fitness incremented is selected as the best measure to characterize the problem. According to this measure, Royal Road, TSP and Deceptive are considered difficult problems (fitness_increment<=0.00468), and the rest of problems are easy problems (or, at least, their genetic operators are more appropriate to improve the results). For the easier problems, good configurations should have frequent migration rates and either not too many islands or good connectivity. Difficult problems, instead, require less frequent migrations (although, this rule is not very accurate).

The presence of the number of islands in the extracted rules should be interpreted under two possible perspectives, the obvious number of different populations (dwelling the islands), and also the population size in each of them. The experiments performed in this study have always used a fixed population size divided among the islands. In this sense fewer islands means also more individuals for each of them. Complementarily, another analysis has been performed using neural networks (1 hidden layer with 20 neurons, alpha 0.9, eta range [0.01, 0.1]). This analysis shows that the parameters with the highest relative significance are: frequency, #islands and the three difficulty measures (leaded by fitness_increment).

It is remarkable to realize that topologies have really minimal influence in the algorithm performance. This validates the results presented by Cantú-Paz [1], in which mention than topology could be generalized as the connectivity and diameter of the proposed island connections. Nevertheless, connectivity has only a marginal influence in the performance, compared with the number of islands and the migration frequency.

The general results presented by [7] also indicate that moderate migration intervals with a small migration size are better than large number of individuals migrated after more generations. In our work we proof that this rule is not valid in general, and only with simple problems it would effective. The two problems in common in both [7] and our study, Griewank and Rastrigin, have been considered “easy” by the fitness_increment measure and the intervals induced by the machine learning algorithm.

The conclusions derived from this preliminary study have shown relevant patterns that identify better configurations in PGAs. Although this early results are quite promising further analysis should be carried out including more parameters (migration rate, overall population size, more topologies and selection and replacement policies). Additionally, more values from the already considered parameters could be included. The sample of the selected problems could be also extended with more real world and synthetic problems. The results achieved by this study have required 24000 executions ran on Magerit System belonging to the CeSViMa Supercomputing Center; future work would require much more executions and also considerable computational requirements.

References

1. Cantú-Paz, E. (1999). Designing Efficient and Accurate Parallel Genetic Algorithms. PhD thesis. University of Illinois at Urbana-Champaign.
2. M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. In L. C. Jain, editor, Proceedings of the Third International conference on knowledge-based intelligent information engineering systems (KES'99), pages 88-92, Adelaide, August 1999
3. Chrisila C. Petty and Michael R. Leuze. A theoretical investigation of a parallel genetic algorithm. In Proceedings of the third international conference on Genetic algorithms, pages 398-405, 1989
4. Whitley, D., Rana, S., Heckendorn, R.B.: The island model genetic algorithm: On separability, population size and convergence. Journal of Computing and Information Technology 7 33-47. 1999
5. Jaromczyk, J.W. and Toussaint, G.T: Relative Neighborhood Graphs And Their Relatives. Proceedings IEEE Vol 80, pages 1502-1517, 1992
6. E. Alba, J.M. Troya, An Analysis of Synchronous and Asynchronous Parallel Distributed Genetic Algorithms with Structured and Panmictic Islands, Parallel and Distributed Processing, J. Rolim et al. (eds.), Lecture Notes in Computer Science 1586, pp. 248-256. Springer-Verlag, 1999
7. Zbigniew Skolicki and Kenneth De Jong. The influence of migration sizes and intervals on island models. In GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, pages 1295–1302, 2005.
8. Pelikan, M., Goldberg, D. E., and Tsutsui, S. 2003. Getting the best of both worlds: discrete and continuous genetic and evolutionary algorithms in concert. Inf. Sci. 156, 3-4 (Nov. 2003), 147-171.
9. J.H. Holland. Royal Road Functions. Internet Genetic Algorithms Digest, 7(22), 1993.
10. P. Larrañaga, C. Kuijpers, R. Murga, I. Inza, S. Dizdarevich. Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review 13, pages 129-170. 1999
11. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Larry Eshelman, editor, Proceedings of the Sixth International Conference on Genetic Algorithms, pages 184–192, San Francisco, CA, 1995. Morgan Kaufmann.
12. Leila Kallel, Marc Schoenauer. Alternative Random Initialization in Genetic Algorithms. ICGA 1997: 268-275
13. Bart Naudts, Leila Kallel. Comparison of Summary Statistics of Fitness Landscapes. IEEE Trans. Evol. Comp. V.4.1:1--15, 2000
14. J. He, C. Reeves, and X. Yao. A Discussion on posterior and prior measures of problem difficulties. In PPSN IX Workshop on Evolutionary Algorithms - Bridging Theory and Practice, 2006.