From Theory to Practice: Efficient Self-tuning of Parallel Genetic Algorithms

Karel Osorio[1], Gabriel Luque[2], and Enrique Alba[2]

[1] Universidad de las Ciencias Informáticas (Cuba)
[2] Universidad de Málaga (Spain)

Abstract. In this paper we develop a new self-adaptive method using a new general methodology which indicates some simple steps to include theoretical models as additional information in our algorithm. In concrete, we design a new distributed genetic algorithms, which is able to self-adapt the value of one of the most important parameter in this kind of parallel techniques using the information of some theoretical models. We also study different alternative ways to use the mathematical results in our method. We test our technique on the wide set of instances of the well-known MAX-SAT problem. Experiments show that our proposal is able to obtain similar, or even better, results when it is compared to traditional algorithms which setting is made by hand. We also show the benefits in terms of saving time and complexity of migration policy settings for distributed genetic algorithms without affecting its efficiency.

Keywords: Parallel Genetic Algorithm, Self-* Techniques, Growth Curves, Takeover Time, Migration Period

1 Introduction

The development of self-* algorithms which adapt their behaviour to the specific characteristics of the problem or to a predefined expected behaviour of an algorithm is currently a very hot topic [1] in Computer Science. One of the goals of these self-* methods is to ease the utilization of the proposed algorithms by unspecialized users by no requiring additional knowledge due to the ability of the technique to self-adjust its parameters.

In this paper we use a general methodology which allows to use several mathematical models to build new self-* techniques. In concrete, we use the results of some theoretical studies about the convergence of genetic algorithms to design a self-adaptive distributed algorithm. However, these mathematical models make several assumptions that cannot be met in real scenarios where the method is to be applied. Therefore, our first challenge consists in adjusting the mathematical models to work for real cases. We will need to propose and evaluate different approaches before we can build new self-* methods. Some initial results about this topic can be found in [2], but here we will refine that basic approach to obtain broader impact.

Once we have accurate approaches to tackle that theory-to-practice step, our idea is the utilization of the information provided by the mathematical model during the execution of the method, helping to self-adapt its search decisions using the learned information. This used methodology to incorporate the mathematical information in our method is showed in Fig. 1, where we can notice that the mathematical model predicts an expected behaviour but still needs corrections compared to the actual behavior of the algorithm (as it is often the case). When a difference between the theoretical and the actual search behavior of the algorithm is detected, the algorithm adapts its parameters. This change can be automatically made using the theoretical model, since it relates the expected behaviour with the parameters of the algorithm.

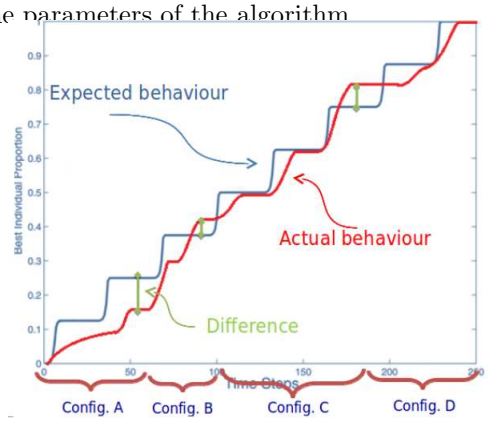


Fig. 1. Theoretical convergence model used to self-adapt during the search of solutions

Therefore, the goals of this paper are manifold: first, we use this methodology to design a new self-* method (a self-adaptive distributed genetic algorithm); second, we propose different approaches to tackle with the problems of using a mathematical model in real scenarios, specifically the problem of not knowing what is the optimum for a problem or not having it in the present population. Then, we test the proposed method to solve a large set of instances of the well-known MAX-SAT problem. Another important goal of this work is that the overhead provoked by the new mechanism needed to obtain this self-adaptive scheme should be low, being our final objective to develop a very efficient and accurate algorithm. We do not believe in self-* approaches that incur in high overheads in time or having high computational complexities, ... and of course, this means reporting the overhead, what it is not always found in many works on self-working techniques.

The rest of this paper is organized as follows. Section 2 gives a very brief background about some concept needed to understand our proposal. Section 3 describes our proposal to adjust the migration period, and various strategies for estimating the growth curve of the best solution. In Section 4 we show the experimental design used to verify the effectiveness of the adaptive method. Later, in Section 5, we present and analyze the results obtained in the experiments. Finally, in Section 6, we discuss the conclusions and make proposals for future work.

2 Background Information

In this section we describe several concepts which are needed to understand our proposal. In concrete, we briefly present the main features of distributed genetic algorithms, and some mathematical models about the convergence for these methods.

2.1 Distributed Genetic Algorithms

A genetic algorithm (GA) is a population based technique. A fitness function assigns a value to each individual. This method applies stochastic operators such as selection, crossover and mutation in a population in order to find a satisfactory optimal solution. In distributed GAs (dGAs), the population is structured into smaller subpopulations relatively isolated one from the others. The principal aspect of this kind of algorithm is that copies of individuals within a particular sub-population (or island) can occasionally migrate to another one [3].

The dGA model requires the identification of a suitable migration policy, and this is often done at hand based in the experience of the researcher and running a set of preliminary experiments whose overhead (time, effort) and the knowledge gained in it) is often not reported in scientific articles.. The main parameters of the migration policy include the following ones: *migration period* (determines how many iterations occur between consecutive migrations), *migration size* (defines the number of solutions that migrate), *selection/replacement* (decides how to select emigrant solutions, and which solutions have to be replaced by the immigrants), and *topology* (defines the neighbor of each island, i.e., the islands that a concrete subpopulation can send individuals to, or receive from).

In general, as we said before, the choices of migration policy are made by experimental studies. The idea of this work is the utilization of the results of some theoretical studies (see next subsection) which characterize the parameters to self-configure some of them. In particular, in this paper we focus on the migration period, which is considered the most important one.

2.2 Theoretical Background

A common analytical approach to study the selection pressure of an GA is to characterize its takeover time [4], i.e., the number of generations it takes for the best individual in the initial population to fill the entire population under selection only. The growth curves are another important issue to analyze the dynamics of the distributed GA. The growth curves are functions that relate the number of generations of the algorithm to the proportion of the best individual in the whole population.

Several works have studied the takeover time and growth curves for other structured EAs in the past [5]. In [6], Alba and Luque made a similar contribution and propose several models for a $(\mu + \mu)$-dEA, where μ is the total population

size. The proposed equation for the takeover time calculation is:

$$t* = period \cdot d(\triangle) - \frac{1}{b} \cdot ln\left(\frac{1}{a} \cdot \frac{\varepsilon}{N - d(\triangle) - \varepsilon \cdot N}\right), \quad (1)$$

where t^* is the takeover time value, period is the migration period, N is the number of islands, $\triangle$ represents the topology, $d(\triangle)$ is the length of the longest path between any two islands (diameter of the topology), ε is the the expected level of accuracy (a small value near to zero) and a and b are adjustable parameters.

Takeover seems a too general and theoretical value far from practice, but it is our goal here to create a way to use this for actual problem solving. This will allow also future new ideas in doing the same for other theoretical models. In fact, in the next section, we will show how to use the model of Eq. 1 to generate our proposal.

3 Migration Period Tuning

As we comment in the first section, our idea is to use the mathematical models presented in the previous section to design new enhanced methods that allow automatic tuning of the migration period. The proposed method is based on models for growth curve and takeover time developed in [6], and consists in changing the migration period so that the algorithm converges to the optimum towards the end of execution and thus improve their performance. The proposed method to self-adjust the migration period follows the next equation:

$$period = \frac{t_{remaining} + K}{d(\triangle) - \left(\frac{P(t)}{1/N}\right)}, \text{ where } K = \frac{1}{b} \cdot \ln\left(\frac{1}{a} \cdot \frac{\varepsilon}{N - d(\triangle) - \varepsilon \cdot N}\right), \quad (2)$$

and $P(t)$ is the proportion of the best individual at the generation t, $t_{remaining}$ is the number of remaining iterations for the end of execution, N is the number of islands, $d(\triangle)$ is the length of the longest path between any two islands (diameter of the topology), a is equal to the size of a sub-population hosted by an island (μ/N), $b = 0.4$ and the tolerance parameter $\varepsilon = 0.1$. This mathematical model is derived from the takeover time which was proposed and validated in real cases in [6], and here we will use it as an oracle telling us what the best migration period is to be used for the next future k steps of the distributed algorithm.

However, the model makes two assumptions that cannot be met in real scenarios where dGA is to be applied. The first one is that the optimum is already present in the initial population. This is a very strong assumption and is extremely unlikely to happen in practice, of course: otherwise running the algorithm would be useless since we already have the optimal solution. The second assumption is that only selection operators are employed; thus no mutation and no crossover are modeled. These two assumptions need for some adjustments on the proposal for it to overcome them. To consider these two assumptions we need to continuously adjust the model during the search, and the new expression

that we propose for the migration period is as follows ($period^*$):

$$period^* = \frac{f_{bestfound}}{f_{objective}} \cdot \frac{period}{N}, \tag{3}$$

where $f_{bestfound}$ is the best found fitness value at the iteration t, and $f_{objective}$ is the objective fitness value, i.e. the fitness value that a researcher considers as a good target value for the algorithm. In a real problem, this task can be completed by defining the minimal features required in the solution, or setting it to the best known value.

The implementation of this technique is performed synchronously, i.e. after each migration period, the processes send all current fitness values of their population to a master process, and they are blocked waiting for a response. The master process collects all fitness values, calculates the new migration period to be used in every island, sends the new value to the islands, and continue execution.

3.1 Growth Curve Calculation

An important aspect to get the new migration period (Eq. 3) is the calculation of the growth curve ($P(t)$, number of optimal solutions at generation t). In real problems, the optimal solution is not present in the initial population, and in many cases this never happens during the execution of the algorithm, so the value of $P(t)$ is equal to zero for almost the entire run time. In order to use $P(t)$ in an actual algorithm we propose alternative definitions of this concept.

The first proposal to obtain a value of $P(t)$ is to calculate the proportion of all values greater than $\alpha \cdot f_{optimum}$ in generation t:

$$P(t) = \frac{\sum_{i=1}^{N} \sum_{f \geq f'} n_f^i(t)}{\mu}, \tag{4}$$

where $f^{'} = \alpha \cdot f_{optimum}$ for some $\alpha \in (0..1]$, and the $n_f^i(t)$ indicates the number of solutions with fitness value equal or higher than f in the island i at generation t. Clearly, for $\alpha = 1$ we obtain the original expression of $P(t)$. We shall refer to this model as the αOpt.

The second proposal, which we call *bestSF*, is based on the best solution found in the current generation ($f_{bestfound}$). This proposal consists in calculating the proportion of individuals with this fitness value in generation t:

$$P(t) = \frac{\sum_{i=1}^{N} n^i_{f_{bestfound}}(t)}{\mu}. \tag{5}$$

The third proposal is based on the diversity of the population. To do this we consider the standard deviation of the fitness values at generation t:

$$\sigma(t) = \sqrt{\frac{1}{\mu} \cdot \sum_{i=1}^{\mu} \left(\bar{f}(t) - f_i(t)\right)^2}, \tag{6}$$

where $f_i(t)$, con $i = 1 \ldots \mu$, are all the fitness values of μ individuals at generation t and $\bar{f}(t)$ is the mean of these values. We can calculate the growth curve as:

$$P(t) = \max\left(1 - \frac{\sigma(t)}{\sigma(0)}, 0\right), \tag{7}$$

where $\sigma(0)$ is the standard deviation of de initial population fitness distribution. We shall refer to this model as the *stdDev*.

The last proposal we will explore is one that was used in [2] and is also based on diversity. In this case the diversity is calculated doing some modifications to the expression of the standard deviation (Equation 6). This is done introducing a threshold fitness value (f^{ϕ}), so that all fitness values under f^{ϕ} are considered equal to it. Besides, the standard deviation calculation will be made with respect to the objective fitness value instead of the mean:

$$\sigma_{f_{objective},f^{\phi}} = \sqrt{\frac{1}{\mu} \cdot \sum_{i=1}^{\mu} \left(f_{objective} - \max(f_i, f^{\phi})\right)^2}, \tag{8}$$

then the expression of $P(t)$ is thus:

$$P(t) = \left(1 - \frac{\sigma_{f_{objective},f^{\phi}}}{f_{objective} - f^{\phi}}\right). \tag{9}$$

We shall refer to this model as the *stdDevObj*, for more details consult [2].

4 Experimental Design

In this section we discuss the design of a set of experiments in order to observe the performance of the models proposed in the previous section and compare them against the traditional distributed approach (constant migration periods).

To test the different models we chose the well-known problem MAX-SAT problem [7]. This problem consists in finding an assignment for a set of variables which maximizes the number of clauses satisfied on a given Boolean formula.

In the experiments we used nine different instances of this problem with 50, 75, 100, 125, 150, 175, 200, 225, and 250 variables. These instances are in the phase transition (difficult ones, although we are not seeking at solving them, but at to show that our self-* approach works in practice), where the ratio between the number of clauses and the number of variables is approximately 4.3 [8].

There are many factors that influence the intensity and diversity of the search, one of them being the selection operators. In this sense, two different configurations for the selection parameters have been tried. The first one emphasizes diversity, and combines a random selection of the parents and a binary-tournament selection of the next generation. We will refer to this configuration as the *Random* selection. The second one has a special stress on intensity and it combines binary-tournament selection for the parents, and elitist selection of the next generation. We will refer to this configuration as *Elitist* selection.

In these experiments we used a total population of 400 individuals ($\mu = 400$), distributed in 8 islands ($N = 8$) each of which hosted a population of 50 individuals. We use a directed ring topology, so that $d(\triangle) = N - 1$. Our migration policy only sends a single solution in each exchange, and the selections for migration will be elitist: the best solution of the source island is transmitted, the worst solution of the receiving island is always replaced. We test a wide set of values for the migration period ranging from 1 (constant communication among islands) to 5000 (complete isolation). Each result was obtained from conducting of 50 independent executions, in order to obtain reliable statistical results. The Table 1 shows the values of configuration parameters for the distributed algorithm.

Table 1. Set of configuration parameters for the dGA

Parameter	*Value*
number of iterations	5000
population size	400
mutation type	*bit–flip*
mutation probability	$1/l$
crossover type	*two–points*
crossover probability	0.6
selection	*random/binary tournament*
replacement	*binary tournament/elitist*

The executions have been physically run in parallel; each island was hosted in a independent process. We used a single host with a Intel Core i7 Q720 processor at 1.6 Ghz, with 4 GB of RAM, and GNU/Linux Ubuntu 11.04 operative system.

5 Analysis of the Results

In this section we present and discuss the results of the different approaches. We analyze the effectiveness of adaptive models, seen in Section 3.1, by means of a comparison with the results produced by the distributed algorithms with constant migration period.

In order to analyze whether the results are statistically reliable, we applied the Wilcoxon test [9], which allows us to make pairwise comparisons between algorithms to know about the significance of the obtained data. A confidence level of 95% have been used in all the cases. The Table 2 shows the results of applying the Wilcoxon test confronting each of the models against the 180 different configurations of the traditional dGA (9 instances $\times$ 10 standard migration periods

$\times$ 2 selection procedures). The column ▲ indicates the number of times that the model produced statistically better results, column $-$ indicates the number of times that there were no statistically significant differences, and column $\triangledown$ indicates the number of times that the model produced significantly worse results compared against the executions with constant migration period. The columns tagged with %> and %$\geq$ contain the percentage of times that our self-* model improved and improved or equaled the results with permanent migration periods, respectively. Finally, the %< indicates the percentage in which a fixed value of the migration period parameter is better than our proposed method.

Table 2. Results of applying the Wilcoxon test

	▲	$-$	$\triangledown$	%>	%$\geq$	%<
$\alpha_1 Opt$	49	130	1	27.22	**99.44**	**0.66**
$\alpha_{0.985} Opt$	38	135	7	21.11	96.11	3.88
$\alpha_{0.95} Opt$	23	123	34	12.78	81.11	18.88
$bestSF$	47	132	1	26.11	**99.44**	**0.66**
$stdDev$	55	123	2	**30.56**	98.89	1.11
$stdDevObj$	42	129	9	23.33	95.00	5.00

Several conclusions can be obtained from the Table 2. First, $\alpha_1 Opt$ and $bestSF$ models are the best ones. They obtain very accurate results: they allow to improve the results of handmade tuning done by a researcher in 55 scenarios of the 180 testing ones, and they are only worse in a single one. These two models have a similar behaviour, since they are very strict about what solutions are considered in the calculation of $P(t)$, which is very beneficial for this problem. In fact, when the number of solutions considered as "good" ones by the model is increased (i.e., it increases the value of $P(t)$ quite fast), the result of the model is worse. This can be observed when we change α value in αOpt model: the lower the α value is (and then $P(t)$ grows faster), the worse the results are.

The models based on a diversity metric, $stdDev$ and $stdDevObj$, also obtain quite accurate results but their accuracy is sightly worse than $\alpha_1 Opt$ and $bestSF$ models. In any case, all the models (with the exception $\alpha_{0.95} Opt$) achieved better results than the traditional dGA (the values of the column %> is greater than the ones of the column %<).

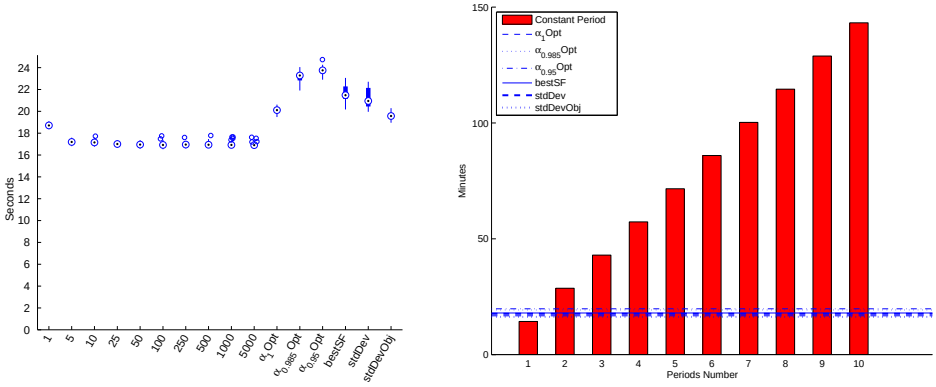
In the next section, we focus on analyzing the overhead provoked by the calculations needed by the different models to their self-adaptation.

5.1 Runtime and Parameters Tuning Cost

As the first point in this section, we will analyze the execution time of each studied model. The use of self-tuning technique means an increase in the computation time of the algorithm. This is because the overhead of sending all fitness values, after each migration to the master process for the calculation of the new migration period.

The Fig. 2a shows a boxplot graph with runtimes of all executions carried out to the 250 variables instance using Elitist selection. As can be easily seen, the executions with constant migration periods usually take about 17 seconds, while

using our adaptive migration takes between 20 and 24 seconds. This represents an increase of between 3 and 7 seconds (depending of the model used) in the computation time for each run in case of using the adaptive technique. Although this sounds a bit negative at a first glance, this increase can be offset by the global time saved in finding an appropriate configuration for the migration period.



(a) (b)

Fig. 2. Execution time: (a) Boxplot with the runtimes of the dGAs with constant and adaptive migration periods, and (b) the global computation time using different numbers of migration periods.

The benefit of using the adaptive technique is showed in the Fig. 2b. In that figure, we can see that the time required by the different proposed models against the time required to test different values for constant periods. If we only test one constant period, it can be observed that our models provoke an small overhead, but this situation is not realistic, since we need to analyze several periods to get the best one, as usually done by researchers. When two or more values are analyzed, our proposal allows to reduce the computation time significantly. In fact, this reduction is around 86% when 10 different values are tested. This means that using our algorithms is not only interesting because they are theoretically grounded, but that the overall time that a researcher needs to employ for his/her scientific analysis is globally reduced in a meaningful manner. This is good from different points of views: larger instances solved in the same time, wider analysis included in future papers, or even higher productivity in publication in conferences and journals since results are got in a reduced time.

6 Conclusions

In this paper, we design a new self-adaptive distributed genetic algorithm and we explored different strategies for the growth curve calculation as the base for the theoretical-to-practice step to address problems where the optimal solution is not present in the initial population.

Results confirm that the utilization of this technique allows to decrease up to 86% the time required to manually configure dGAs, without affecting the perfor-

mance of the algorithm. In fact, the results of most of the proposed approaches outperform the traditional dGA.

In future works, we plan to analyze the performance of these models in other problems with different characteristics (we already have results confirming this good behavior). We are also interested in studying the influence of other parameters as the number of generations, the topology, or the migration size.

We believe that self-tuning, to a given degree, can be brought to several important parameters, and we bet for a line of research that is theoretically ground and that considers at the same time overheads in the analysis, to create a fair and clear picture for potential users. This is also an example of theory-to-practice cross-fertilization that we are making with other ideas and algorithms.

Acknowledgments

Authors acknowledge funds from the Spanish Ministry of Sciences and Innovation European FEDER under contract TIN2008-06491-C04-01 (M* project, available in URL `http://mstar.lcc.uma.es`), TIN2011-28194 (RoadME project available in URL `http://roadme.lcc.uma.es`, TIN2011-28194, CICE Junta de Andalucía under contract P07-TIC-03044 (DIRICOM project, available in URL `http://diricom.lcc.uma.es`). and AUIP as sponsors of the Scholarship Program Academic Mobility.

References

1. Smith, J.: Self-adaptation in evolutionary algorithms for combinatorial optimisation. Adaptive and Multilevel Metaheuristics (2008) 31–57
2. Osorio, K., Luque, G., Alba, E.: Distributed Evolutionary Algorithms with Adaptive Migration Period. In: Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on. (nov. 2011) 259 –264
3. Alba, E., Tomassini, M.: Parallelism and Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation **6** (October 2002) 443–462
4. Goldberg, D.E., Deb, K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In Rawlins, G.J., ed.: Foundations of Genetic Algorithms. Morgan Kaufmann, San Mateo, CA (1991) 69–93
5. Sprave, J.: A Unified Model of Non-Panmictic Population Structures in Evolutionary Algorithms. In Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A., eds.: Proceedings of the Congress of Evolutionary Computation. Volume 2., Mayflower Hotel, Washington D.C., USA, IEEE Press (July 1999) 1384–1391
6. Alba, E., Luque, G.: Theoretical Models of Selection Pressure for dEAs: Topology Influence. Evolutionary Computation, 2005. The 2005 IEEE Congress on **1** (2005) 214–221
7. Garey, M., Johnson, D.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)
8. Cheeseman, P., Kanefsky, B., Taylor, W.: Where the really hard problems are, Morgan Kaufmann (1991) 331–337
9. Demsar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research **5** (2006) 1–30