Using Landscape Measures for the Online Tuning of Heterogeneous Distributed GAs

Carolina Salto
Facultad de Ingeniería
Universidad Nacional de la Pampa
General Pico, Argentina
saltoc@ing.unlpam.edu.ar

Enrique Alba
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Málaga, Spain
eat@lcc.uma.es

Francisco Luna
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Málaga, Spain
flv@lcc.uma.es

ABSTRACT

Tuning distributed genetic algorithms (dGAs) increases even more the task of finding an appropriate parameterization, since the migration operator adds, at least, five additional values that have to be set up. This work is a preliminary approach on using a landscape measure (the Fitness Distance Correlation) to dynamically adjust one of these five parameters, in particular, the migration period. The results have shown that, by using this information, the quality of the solutions is competitive with those obtained by the algorithms with the pre-tuned migration period, but with a saving of more than 100 hours of preliminary experimentation.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Theory

Keywords

Landscape measures, tuning, distributed genetic algorithms

1. INTRODUCTION

When using distributed genetic algorithms (dGAs) [1], it is very common in the literature to experimentally set the migration parameters for each island (or semi-isolated subpopulation), which are responsible for the structure of the algorithm and the intercommunication scheme (migration policy). Tuning all these parameters for the best possible performance is a computationally demanding task because there is a large number of possible combinations to evaluate (over many instances most times). Tuning algorithm performance is a topic that has been studied in the literature. Several approahces are offline (i.e., applied prior to the actual utilization of the algorithm) such as Design of Experiments [13] or F-RACE [3]. In this work, our approach for establishing such a parameter setting, known as parameter control [4], is based on starting the algorithm with an initial configuration that changes during the execution of the algorithm (i.e., online adaptation). Here, the migration parameter under study is the migration period at which the individuals are exchanged among subpopulations. The resulting algorithm is a heterogeneous dGA (HdGA) [11] not needing the traditional and costly ad-hoc pre-tuning of this migration parameter.

Our contribution here is a preliminary approach aimed at guiding the change in the migration period by using information from an empirical landscape measure, the Fitness Distance Correlation (FDC) [8], that is able to provide the algorithm with an estimation of the difficulty of the search space that each island is exploring. Other landscape measures could have been used [14, 15], but FDC is easy to understand and to compute as well as it does not cause an unaffordable computing overload.

In order to evaluate the performance of the proposed approach, the Max-Cut problem has been used. This is a well-known NP-hard problem [10], and, besides its theoretical importance, it has applications in several fields. That the Max-Cut problem deserves further study is evidenced by the continuing active research in this and related areas. By using 12 well-known instances from the specialized literature, we have compared HdGA to five homogeneous dGAs with five different pre-tuned migration periods. The results have shown that our approach reaches competitive quality solutions and, at the same time, saving more than 100 hours of pre-tuning computational time.

The outline of the paper is as follows. Section 2 presents the background to our parallel implementation of dGAs and the characteristics of the proposed heterogeneous algorithm. Section 3 formally defines the optimization problem and gives details of the main components of the algorithms. Section 4 presents experimentation performed to evaluate our proposal. Finally, we summarize the conclusions and discuss several lines for future research in Section 5.

2. HETEROGENEOUS DGAS: A CONTROL PARAMETER APPROACH

In this work we focus on distributed GAs (dGAs) [1], where the population is structured into smaller subpopulations (islands) relatively isolated one each other (see Algorithm 1 for the structure of a canonical GA (dGA_i)). Copies

GECCO'11, July 12–16, 2011, Dublin, Ireland.

Algorithm 1 Elementary dGA (dGA$_i$)

```
t = 0; {current generation}
initialize(P_i(t));
evaluate(P_i(t));
while (t < max_generations) do
   P'_i(t) = evolve(P_i(t)); {recombination and mutation}
   P'_i(t) = improve(P'_i(t)); {local search}
   evaluate (P'_i(t));
   P'_i(t) = send/receive individuals from dGA_j; {neighbor dGA}
   P_i(t+1) = select new population from P_i(t) ∪ P'_i(t);
   t = t + 1;
end while
```

of individuals within a particular subpopulation P_i (where i is the identifier of an island) can occasionally migrate to another one. A migration policy defines the island topology, when migration occurs, which individuals are being exchanged, the synchronization among the subpopulations, and the kind of integration of the exchanged individuals within the target subpopulations. Consequently, five additional parameters for controlling the migration policy are needed. This in general adds an extra set-up time, that in fact authors rarely report in their studies, but that represents a considerable effort. Also, since GAs are stochastic algorithms, each parameter configuration has to be evaluated at least 30 times in order to provide the results with statistical significance.

In this preliminary work, our approach is to dynamically adjust only one out of the five migration parameters (keeping the others fixed): the migration period. As a consequence, each island behaves differently so we are actually engineering a heterogeneous dGA, which has been called HdGA. The criterion used to modify the migration period (*mig_period*) is based on the fitness distance correlation (FDC) [8], which is a measure of how difficult a problem can be for GAs. Even though FDC is used to estimate the difficulty of the whole search space, our goal here is to use it as a measure of the difficulty of the portion of the search space which is being explored by each island. Formally, given a set $F = \{f_1, f_2, \ldots, f_n\}$ of n individual fitnesses and a corresponding set $D = \{d_1, d_2, \ldots, d_n\}$ of the n distances to the nearest global maximum, the FDC is defined as:

$$fdc = \frac{\frac{1}{n}\sum_{i=1}^{n}(f_i - \bar{f})(d_i - \bar{d})}{\sigma_F \sigma_D} \quad (1)$$

where $\bar{f}$ and $\bar{d}$ are the means of the fitness values and the distances to the nearest global maximum, respectively, while σ_F and σ_D denote the standard deviations. FDC determines how closely F and D are correlated. For a maximization problem like the Max-Cut, if the fitness increases when the distance to the optimum becomes smaller, then the landscape is expected to be easily explored by the GA, and the fdc should be close to -1.0. On the contrary, when such a correlation does not exist, i.e., fdc is close to zero, a GA can find difficulties. In each island, a local FDC value is computed with its local subpopulation and the distance to its best-known solution (values for F and D).

The way in which the FDC information is included in HdGA is as follows. When the fdc value is close to -1.0 value in an island, that island begins to diminish the interaction with other islands, i.e., it sends individuals to its neighboring subpopulation with less frequency (and so receiving fewer inmigrants). We are assuming that the island is in a region of high quality solutions so therefore the search has to be further intensified (increase exploitation with less frequent migrations). On the other hand, the strategy tries to promote the exploration (more frequent migrations) when fdc is far from -1.0. In particular, we have considered the FDC values not only at the current epoch, but in two consecutive epochs, so that the tendency of the value is taken into consideration. That is, during two consecutive migration steps, the average $\bar{fdc}$ is computed. Then, the HdGA changes the period in the island to its closer larger power of two if $\bar{fdc} < -0.75$. On the contrary, the period changes to its closer smaller value which is power of two. Of course, migration periods assume discrete values in the range $[mig\_period_{min}, mig\_period_{max}]$. This criterion is inexpensive to compute, since it checks simple conditions based on information already available in any standard GA, like the fitness values and the Hamming distances, and can be used to adjust additional parameters.

The execution is initialized by randomly choosing a power of two *mig_period* in the range [1,512] for each island. The bounding cases are the total communication (migration period equals to one) or almost independent execution (migration period equals to 512). When the current value is equal to one and the algorithm needs to change to the next lower one, the algorithm does not make any change in the migration period at all. The same action is triggered when the current value is 512 and the algorithm needs to increase the value to the next one.

3. HDGA FOR THE MAX-CUT PROBLEM

The Max-Cut problem can be formally defined as follows. Given an undirected graph with edge weights, this problem consists in partitioning the set of nodes of a weighted graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the set of nodes and $E = \{(i, j) : i, j \in V\}$ the set of edges, into two disjoint subsets S and $\overline{S} = V - S$, such that the sum of the weights of the edges from E that have one endpoint in S and the other in $\overline{S}$, is maximized. Let $w_{i,j}$ be the weight associated with edge $(i, j) \in E$. Then, the next function has to be maximized:

$$cut(S, \overline{S}) = \sum_{i \in S, j \in \overline{S}} w_{ij} \quad (2)$$

In order for HdGA to address the Max-Cut problem, we have used the following solution encoding and search operators:

Representation. A solution to this problem can be represented as a binary vector x=$(x_1, x_2, \ldots, x_n)$ of length n, where x_i corresponds to a node. Each vector encodes a partition of the nodes, with x_i=1 meaning $x_i \in S$ and x_i=0 meaning $x_i \in \overline{S}$.

Initialization. The algorithm creates several initial solutions using the constructive heuristic proposed by Kahruman *et al.* [9], called SG3. This method is fast and generates solutions with considerable quality and diversity. SG3 was also chosen in [2] to be the constructive heuristic for their SA and TS and seems to be a key component in all modern well-performing algorithms for the Max-Cut.

Genetic operators. HdGA uses the well-known uniform crossover (UX) and bit-flip mutation operators.

Local Search Heuristic. After genetic operators (crossover and mutation), the newly created individuals undergo a local optimization procedure with a certain probability. When applied on a solution x, the procedure starts by creating a set

Algorithm 2 Local Search

```
T ← V;
while ( T ≠ ∅) do
   i ← random vertex in T;
   if i ∈ S and σ_S(i) − σ_S̄(i) > 0 then
      S ← S \ {i};
      S̄ ← S̄ ∪ {i};
   end if
   if i ∈ S̄ and σ_S̄(i) − σ_S(i) > 0 then
      S̄ ← S̄ \ {i};
      S ← S ∪ {i};
   end if
   T ← T \ {i};
end while
```

T of all nodes i (for $i = 1, \ldots, n$). The iterative procedure randomly selects a node i from the set T and then changes that node from one subset to the other in x according to the following rules: i) **if** $x_i = 0$ and $\sigma_S(i) - \sigma_{\overline{S}}(i) > 0$ **then** $x_i = 1$, and ii) **if** $x_i = 1$ and $\sigma_{\overline{S}}(i) - \sigma_S(i) > 0$ **then** $x_i = 0$, where, for each node $i : 1, \ldots, n, \sigma_S(i) = \sum_{j \in S} w_{ij}$ and $\sigma_{\overline{S}}(i) = \sum_{j \in \overline{S}} w_{ij}$ [6], $\sigma_S(i) - \sigma_{\overline{S}}(i)$ represents the change in the objective function associated with moving a node i from one subset of the cut to the other. All possible moves of a solution are investigated by making a single pass through all the nodes. The current solution is replaced by the improved one. The pseudo-code of the local search procedure is given in Algorithm 2. This algorithm is an efficient variation of the local search phase proposed in [6] which has $O(n^2)$ complexity. The difference is that our local search procedure makes one single pass through all the nodes, i.e., so its complexity is $O(n)$.

4. EXPERIMENTATION

In this section, the experimentation conducted to assess the performance of HdGA on the Max-Cut problem is presented. We detail the algorithmic parameterizations, the instances used, and, finally, the analysis of the results.

4.1 Parameterization

The global population of all evaluated models is composed of 512 individuals and there are 16 islands with 32 individuals each (μ). The maximum number of generations is fixed to 6500. Each parent is selected by a binary tournament. In every island, the number λ of created offspring is 32 at each iteration. The UX crossover is applied with a probability of 0.65, the bit-flip mutation is applied to all new generated individuals with a rate of $1/|V|$ per bit, and the local search procedure is applied with a probability of 0.2. The next population is built up from the $(\mu + \lambda)$ individuals using fitness proportional selection. The distribution scheme of islands is based on a unidirectional ring topology with asynchronous communication. The individuals are integrated into the population whenever they arrive. A copy of the best individual is sent to the neighboring subpopulation, while the target island selects the worst individual to be replaced with the incoming one (only if it has better or equal fitness).

The algorithms are implemented in MALLBA [5], a C++ software library. Our computing system is a cluster of 8 machines with AMD Phenom8450 Triple-core Processor at 2GHz with 2 GB of RAM, linked by Gigabit, under Linux with 2.6.27-4GB kernel version. Each island is physically run on a separate processor.

Table 1: Max-Cut instance description.

Instances	|V|	Density (%)
G1 - G2 - G3	800	6.12
G11 - G12 - G13	800	0.63
G14 - G15 - G16	800	1.58
G22 - G23 - G24	2000	1.05

4.2 Instances

For our experiments we used the set of instances generated by Helmberg and Rendl [7][1]. In particular, we use a set of widely used graphs. They consist of toroidal, planar and randomly generated graphs of varying sizes and densities, with weights taking values 1, 0, or -1. The size of the graphs (|V|) varies from 800 to 3000 nodes meanwhile the density fluctuates from 0.17 % to 6.12 % (see Table 1 for more details). Recent works such as [2], [6], or [12], all used these graphs in their experiments, so it is a convenient election for comparison purposes.

4.3 Results

In this section we will compare the results produced by the proposed HdGA and a parallel homogeneous GA, called HomdGA (which performs the same kind of search on different sets of greedy generated individuals) in order to show the performance of the HdGA as an competitive search method. As we are interested in evaluating the self-adapted, FDC-based migration schedule, in the case of HomdGA we report a set of values for the migration period ranging from 1 (maximum coupling among islands) to 512 (fairly isolated islands), in order to characterize the effect of the migration period in the quality of the solutions obtained. Consequently, HomdGAi means a homogeneous approach with period i. All the results presented are averaged over 30 independent runs.

Figure 1 shows the relative error with respect to the best-known solution in the literature for each instance. A first conclusion is that HdGA is very competitive in terms of solution quality. The figure shows that the HdGA column (the darker one on the right of each group) has been able to reach solutions with the same (or even lower) error rates than the HomdGAs. Averaging all the error rates of HomdGAs over all the instances, a value of 0.27% is obtained, that is, on average, HomdGAs have reached solutions which are 0.27% far from the best-known solution. The same value for HdGA is 0.28%, what makes the difference very tight, but with the benefits of the pre-tuning time reduction. Indeed, HdGA needs 26 hours of execution to obtain the results from the 30 independent executions for all the graph set tackled in this work (see Table 1). For the HomdGA, five values has been considered for the migration period, so therefore $26 \times 5 = 130$ hours of computation have been required. This means that HdGA has allowed us to avoid running 4 configurations, thus saving 104 hours of pre-tuning.

We also want to report that HdGA has been not only able to hit the best-known solution for instances G1 and G3, but also to improve the best-known solution for instance G16. This is a remarkable result, since it is a long lasting very studied problem and because our true goal was to propose an algorithm that reduces pre-tuning. Hence HdGA is not just a simple heterogeneous version, but a cutting edge algorithm.

Figure 2 shows the mean number of generations that dGAs need to computer their best solution. The results of HdGA

[1] publicly available at `http://www.stanford.edu/yyye/yyye/Gset/`

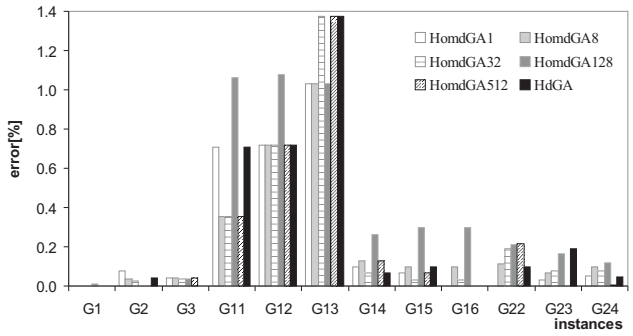

Figure 1: Relative error rates of HdGA and HomdGA.

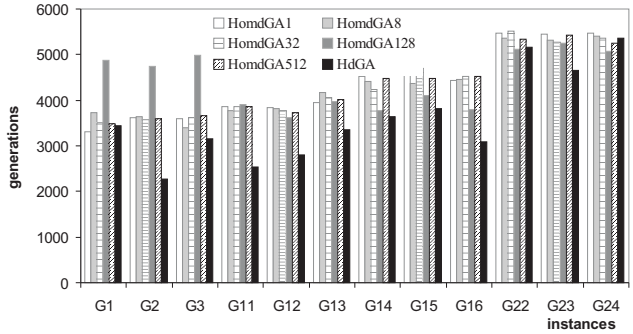

Figure 2: Number of generations to converge for HdGA and HomdGA.

are very promising. Indeed, in 9 out of the 12 instances studied, HdGA required a lower number of generations to converge towards its solution with the maximum (best) fitness value than any of the HomdGA*i* (all but G1, G22, and G24). The differences are remarkable for G2 and G11, in which the reductions is around 40% of the HomdGAs. It is important to note that this is an additional source of time savings (not just the pre-tuning), since high quality solutions for the problem instances are computed faster with HdGA.

5. CONCLUSIONS AND FURTHER WORK

This work has presented a preliminary approach on using fitness landscapes for dynamically adjusting the migration period of HdGA (Heterogeneous dGA). The goal of our proposal is to reduce the pre-tuning of such migration parameter, and therefore, to achieve a reduction in the computational times. We have evaluated HdGA over 12 instances of the Max-Cut problem and compared its suitability to homogeneous algorithms considering five different pre-tuned configurations. The results have shown that HdGA has been able to compute competitive solutions in terms of quality (the averaged error rate with respect to the best-known solution is 0.28% against the 0.27% of the pre-tuned algorithms), but saving more than 100 hours of computation. A second interesting finding is that HdGA has been the fastest algorithm to converge towards high quality solutions. As future work, we plan to extend this work with additional landscape measures such as the negative slope coefficient, and to further evaluate this approach over different optimization problems.

Acknowledgments

Carolina Salto acknowledges continuous support from CONICET, Universidad Nacional de La Pampa and ANPCYT in Argentina. The work of Enrique Alba and Francisco Luna has been partially funded by the "Consejería de Innovación, Ciencia y Empresa", Junta de Andalucía under contract P07-TIC-03044, and the Spanish Ministry of Science and Innovation and FEDER under contract TIN2008-06491-C04-01.

6. REFERENCES

[1] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443 – 462, 2002.

[2] E. Arráiz and O. Olivo. Competitive simulated annealing and tabu search algorithms for the Max-Cut problem. In *GECCO 2009*, pages 1797 – 1798, 2009.

[3] M. Birattari, T. Stutzle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *GECCO 2002*, pages 11 – 19, 2002.

[4] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In *Studies in Computational Intelligence*, volume 54/2007, pages 19–46. Springer, 2007.

[5] E. Alba et al. *MALLBA: A Library of Skeletons for Combinatorial Optimisation*, volume 2400 of *LNCS*, pages 927–932. Springer, 2002.

[6] P. Festa, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Randomized heuristics for the MAX-CUT problem. *Optimization Methods & Software*, 7:1033–1058, 2002.

[7] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal of Optimization*, 10(3):673–696, 2000.

[8] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *6th International Conference on Genetic Algorithms*, pages 184 – 192, 1995.

[9] S. Kahruman, E. Kolotoglu, S. Butenko, and I.V. Hicks. On greedy construction heuristics for the max_cut problem. *Int. J. Comput. Sci. Eng.*, 3(3):211–218, 2007.

[10] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85 – 103. 1972.

[11] F. Luna, E. Alba, and A.J. Nebro. Parallel heterogeneous metaheuristics. In E. Alba, editor, *Parallel metaheuristics*, pages 395–422. Wiley, 2005.

[12] R. Martí, A. Duarte, and M. Laguna. Advanced scatter search for the Max-Cut problem. *INFORMS Journal on Computing*, 21(1):26–38, 2009.

[13] E. Ridge and D. Kudenko. Tuning an algorithm using design of experiments. In *Experimental Methods for the Analysis of Optimization Algorithms*, pages 265 – 286. Springer, 2010.

[14] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. Fitness landscapes and evolvability. *Evolutionary Computation*, 10(1):1 – 34, 2002.

[15] V. Vassilev, T. Fogarty, and J. Miller. Information characteristics and the structure of landscapes. *Evolutionary Computation*, 8(1):31 – 60, 2000.