Parallel LAN/WAN Heuristics for Optimization

Enrique Alba Gabriel Luque
Departamento de Lenguajes y Ciencias de la Computación
E.T.S. Ingeniería Informática
Campus Teatinos, 29071 Málaga (España)
eat@lcc.uma.es gabriel@lcc.uma.es

Abstract

We present in this work a wide spectrum of results on analyzing the behavior of parallel heuristics for solving optimization problems. We focus on evolutionary algorithms as well as on simulated annealing. Our goal is to offer a first study on the possible changes in the search mechanics when shifting from a LAN distributed algorithm to a WAN environment. We will address six optimization tasks of considerable complexity. The results show that, despite the expected slower execution time, the WAN versions of our algorithms consistently solve the problems. We even report some interesting results in which WAN algorithms outperform LAN ones. We also extend the study to include hybrid versions to check the scope of our conclusions.

1 Introduction

Solving complex problems means designing new algorithms that improve in some manner the computational effort and the waiting time for an acceptable solution. Analyzing and designing parallel algorithms is a healthy activity since it pursuits some of the most promising objectives in optimization, namely reducing the computing time, allowing cooperation of different algorithms (hybridization), and even modifying the search pattern to yield new methods for complex problems.

In this work, we study how a parallel algorithm changes its behavior when executed in LAN with respect its execution in a WAN. Due to the growing park of available computers in Internet, our aim is to study the new scenario of WAN computing, in contrast of the more traditional LAN analysis.

The MALLBA project [2] is an effort of research in this direction. In the MALLBA project we intend to design and analyze exact, heuristic and hybrid techniques in sequential, LAN and WAN environments.

The contributions of this paper are manyfold. First, we test the efficiency of some algorithms developed in the MALLBA project, since such an ambitious goal must be validated in practice. Second, we want to put aside the expected and actual outcomes of computing in LAN and WAN. Third, we are interested in showing really useful results, and this is why we check the algorithms on six quite different problems, accounting for epistasis, multimodality, continuous, and discrete optimization. Last, but not least, our conclusions are somewhat expected and somewhat surprising at the same time, since we do validate in practice some theoretical thoughts on WAN overhead, but also report competitive performance in WAN.

The organization of the paper is as follows. Next section (Section 2) introduces the MALLBA project. Section 3 discusses the search model considered in our study. Section 4 presents the details on the problems being solved. We then turn in Section 5 to comparatively analyze the LAN/WAN behavior of the considered algorithms for all the problems. Finally, we provide in Section 6 some concluding remarks, and points out the future work we envision after our conclusions.

2 The MALLBA Project

The MALLBA project is an effort to develop a library of algorithms for optimization that can deal with parallelism (LAN and WAN) in a user-friendly and, at the same time, efficient manner. All the algorithms described in the next section are implemented as *software skeletons*. Skeletons are generic templates to be instantiated with the features of the problem, by the user. All the knowledge related to the resolution method (e.g. parallel considerations) and its interactions with the problem are implemented by the skeleton. Skeletons are implemented by a set of *required* and *provided* C++ classes that represent an abstraction of the entities participating in the resolution method:

- **Provided Classes**: They implement internal aspects of the skeleton in a problem-independent way. The most important *provided* classes are `Solver` (the algorithm) and `SetUpParams` (setup parameters).

- **Required Classes**: They specify information related to the problem. Each skeleton includes the `Problem` and `Solution` required classes, that encapsulate the problem-dependent entities needed by the resolution method. Depending on the skeleton, other classes may be required.

The infrastructure used in the MALLBA project is made of communication networks and clusters of computers located in Málaga, La Laguna and Barcelona. These nodes are interconnected by a chain of Fast Ethernet and ATM circuits. For the WAN experiments, we use one machine from each site.

The MALLBA library is publicly available at `http://www.lsi.upc.es/~mallba`.

3 Algorithms

In this paper we deal with several evolutionary algorithms, in particular with genetic algorithms (GAs), a CHC algorithm and an evolution strategy (ES); also local search methods are considered, like simulated annealing (SA) (see a global description in [10]). All methods have been parallelized for LAN and WAN platforms. Finally, we will include some hybrid algorithms in our LAN/WAN study.

Evolutionary algorithms (EAs) are stochastic search methods that have been successfully applied in many real applications of high complexity. An EA is an iterative technique that applies stochastic operators on a pool of individuals (tentative solutions). An evaluation function associates a value to every individual indicating its suitability to the problem. For this work we implemented three parallel distributed EAs, whose component sub-algorithm is a GA, an ES or a CHC.

GAs are a very popular class of EAs. Traditionally, GAs are associated to the use of a binary representation, but nowadays you can find GAs that use other types of representations. A GA usually applies a recombination operator on two solutions, plus a mutation operator that randomly modifies the individual contents to promote diversity.

A CHC [7] is a non-traditional GA which combines a conservative selection strategy (that always preserves the best individuals found so far) with a highly disruptive recombination (*HUX*). Certain highly disruptive crossover operator provide more effective search in many problems, which represents the core idea behind the CHC search method. This algorithm introduce a bias against mating individuals who are too similar. Mutation is not performed, instead, a *restart* process re-introduces diversity whenever convergence is detected.

The last EA we include in our study is an evolution strategy. This algorithm is suited for continuous values, usually with an elitist selection and a specific mutation (crossover is used rarely). In ES, the individual is made of the objective variables plus some other parameters guiding the search. Thus, an ES facilitates a kind of *self-adaption* by evolving the problem variables as well as the strategy parameters at the same time. Hence, the parameterization of an ES is highly customizable.

The simulated annealing algorithm (SA) was first proposed in 1983. SA is a stochastic relaxation technique that can be seen as a hill-climber with an internal mechanism to escape local optima. To allow escaping from a local optimum, moves that increase the energy function are accepted with a decreasing probability $\exp(-\delta/T)$, where T is a parameter called the "temperature". Here, we are using a proportional method for updating the temperature ($T_k = \alpha \cdot T_{k-1}$).

In its broadest sense, hybridization refers to the inclusion of problem-dependent knowledge in a general search algorithm [6] in one of two ways: *strong hybrids*, where problem-knowledge is included as problem-dependent representation and/or operators, and *weak hybrids*, where several algorithms are combined in some manner to yield the new hybrid algorithm.

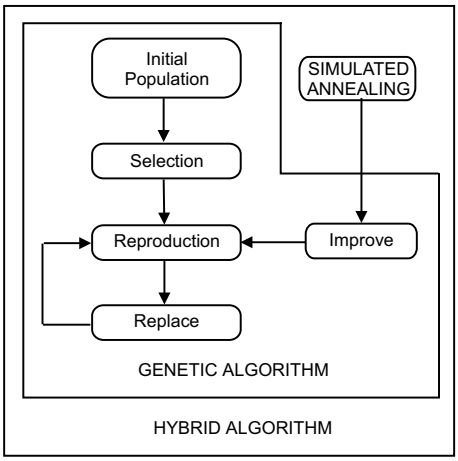

Figure 1. Hybrid schema 1 (GASA1).

We define two main classes of hybrids in this work: A first hybrid (GASA1/CHCES1) where a GA/CHC algorithm uses the other algorithm (SA/ES) as an evolutionary operator (see an example for GASA1 in Fig. 1). The rationale for this selection of algorithms is that, while the GA/CHC locates "good" regions of the search space (exploration), the SA/ES allows for exploitation in the best regions found by its partner. The second hybrid schema executes a

GA/CHC until the algorithm completely finishes. Then the hybrid selects some individuals from the final population and executes a SA/ES algorithm over them. Concretely, we analyze a first version (GASA2/CHCES2) that uses a tournament selection (model 2.1 of Fig. 2), and another version (GASA3/CHCES3) that uses a random choice of individuals (model 2.2 of Fig. 2).

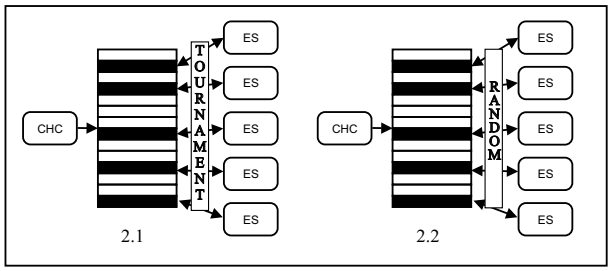

Figure 2. Hybrid schema 2 (CHCES2/3).

Since we want to conduct our research in LAN and WAN platforms it seems natural to explore the behavior of parallel heuristics. A parallel EA (PEA) is an algorithm having multiple components EAs, regardless of their population structure [3]. In this work, we have chosen a *distributed EA* (dEA) because of its popularity and because it can be readily implemented in clusters of machines. In distributed EAs there exists a small number of islands performing separate EAs, and periodically exchanging individuals after a number of isolated steps (*migration frequency*). As migration policy we use a static ring topology, select random migrants and include them in the target populations only if they are better than the worst-existing solutions.

4 Problems

In this section we discuss the optimization tasks that will be used to test our parallel heuristics. The first two problems were chosen because their continuous nature makes them adequate for testing the ES algorithm. The rest of problems represent a broad spectrum of challenging intractable tasks in the areas of scheduling, coding theory, graph theory and transportation.

The generalized Rastrigin function (Eq. 1) is a problem with a large search space and a very large number of local optima [11]. This function is a non-epistatic function representing a typical test for EAs. For the experiments, we have used a problem instance of 20 variables (optimum at $f^* = 0$).

$$Ras(x_i|_{i=1..n}) = 10 \cdot n + \sum_{i=1}^{n} x_i^2 - 10 \cdot \cos(2 \cdot \pi x_i) \quad x_i \in [-5.12, 5.12] \qquad (1)$$

The frequency modulation sounds [12] (or FMS) has been proposed as a hard real task consisting in evolving six parameters $\vec{x} = (a_1, w_1, a_2, w_2, a_3, w_3)$ in order $y(t)$ to fit the target $y_0(t)$. The goal function and the evolved and target models have the expressions shown in Eq. 2-4. For the experiments, we consider as an optimum any solution with fitness value below 0.12.

$$FMS(\vec{x}) = \sum_{i=0}^{N} (y(t) - y_0(t))^2 \qquad (2)$$

$$y(t) = a_1 \sin(w_1 t\theta + a_2 \sin(w_2 t\theta + a_2 \sin(w_3 t\theta))) \qquad (3)$$

$$y_0(t) = 1.0 \sin(5.0 t\theta + 1.5 \sin(4.8 t\theta + 2.0 \sin(4.9 t\theta))) \qquad (4)$$

$$\theta = 2\pi/100 \quad a_i, w_i \in [-6.4, 6.35]$$

The maximum cut problem [1] (MaxCut) consists in partitioning the set of vertices of a weighted graph into two disjoint subsets, such that the sum of the weights of edges with one endpoint in each subset is maximized. We use a binary string where each digit corresponds to a vertex. If a digit is 1 then its corresponding vertex is in set V_1, else the vertex is in set V_0. The function to be maximized is shown in Eq. 5. For the experiments, we use a scalable problem instance [1] with a graph of size $n = 100$, "cut100" (optimum at $f^* = 1077$).

$$F(\vec{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij}\big(x_i(1 - x_j) + x_j(1 - x_i)\big) \qquad (5)$$

The minimum tardy task problem is a task-scheduling problem [8]. Each task i from the set of tasks T has an associated length l_i (its execution time), a deadline d_i and a weight w_i, which is a penalty indicating the importance that a task remains unscheduled. Scheduling the tasks of a subset S of T consists in finding the starting time of each task in S, such that at most one task at a time is performed, and such that each task finishes before its deadline. The optimal solution is a feasible schedule S with the minimum tardy task weight W which is the sum of weights of unscheduled tasks (Eq. 6). We use a scalable problem instance [8] of size 100 tasks, "mttp100" (optimum at $f^* = 20$).

$$min\, W = \sum_{i \in T - S} w_i \qquad (6)$$

The error correcting code sign problem [1] (ECC) consists in assigning codewords to an alphabet that minimizes the length of transmitted messages, and that provides maximal correction of single uncorrelated bit errors. This problem can be formally represented by a three-tuple (n, M, d), where n is the length of each codeword, M is the number of codewords, and d is the minimum Hamming

distance between any pair of codewords. The function to be maximized is:

$$f(C) = \frac{1}{\sum_{i=1}^{M} \sum_{j=1, i \neq j}^{M} \frac{1}{d_{ij}^2}} \quad (7)$$

where d_{ij} represents the Hamming distance between codewords i and j. In this study, we consider a problem instance, where $s = 12$ and $M = 24$. The optimum solution is here set to have a fitness value of $f^* = 0.0674$.

The vehicle routing problem [9] (VRP) consists in determining minimum cost routes for a fleet of vehicles originating and terminating in a depot. The fleet of vehicles gives service to a set of customers with a known set of constrains. All customers must be assigned to vehicles such that each customer is serviced exactly once, taking account for the limited capacity of each vehicle. The goal is to minimize the sum of route costs given by:

$$min \sum_{k=0}^{K-1} \alpha(W_k) \quad (8)$$

$$W_k = \sum_{i=0}^{N} \sum_{j=0}^{N} c_{i,j} \cdot x_{i,j,k} \quad (9)$$

In these equations, $x_{i,j,k}$ is a binary variable that is 1 if route k has an arc between nodes i and j, $c_{i,j}$ is the cost to go from node i to node j, α is the function that calculate the route cost depending on W_k, N is the number of clients (0 is the depot) and K is the number of routes.

For the experiments, we use the two first instances of the classical benchmark after Christofides [5]. We consider as an optimum any solution with $f_1^* < 800$ for the first instance, and $f_2^* < 1200$ for the second one.

5 Analysis of the Results

In this section we report our results in studying our parallel heuristics in three phases. First, we analyze the behavior of a genetic algorithm both in LAN and WAN environments. Then, since we face the parallel GA against combinatorial optimization problems, we include a second phase of study that address the continuous optimization case. In this second phase we analyze a parallel evolutionary strategy on our two continuous problems (RAS and FMS). In our final third phase of analysis we try to validate our conclusions in the field of hybrid parallel algorithms, and go forth to discuss the results when using CHCES variants on RAS and GASA variants on MaxCut100.

The algorithm parameters used in the experiments are described in Table 1.

We always report average values of 30 independent runs for each of the parameters included in the tables of this section. We show values for the average best solution found (**opt**), average number of evaluations to get a solution (**#evals**), average time (**time**, in seconds) and the percentage of hits (**hits**) i.e. number of runs finding an optimal solution.

In Table 2 we include our results when using a parallel GA on the four combinatorial problems of our benchmark (for VRP we used two instances). We must notice that the two instances of VRP are very difficult to solve without including specific operations. If we compare LAN and WAN executions of the parallel GA we can detect a clear trend of the WAN execution to spend more time in finding a solution for the three first problems.

An interesting detail is that MTTP and ECC got an increased number of hits when executed in WAN, although, as expected, the wall-clock time for a solution is higher in WAN due to the communications overhead. Numerically speaking this means that the WAN environment retards the migrations, and this effect is somewhat perceptible in some problems. Since larger isolation is not always appropriate, we could think about a poorer WAN numerical results for other problems, as it effectively happens for the MaxCut problem, because its percentage of hits decays from 17% in LAN to a smaller 10% in WAN.

However, in the case of VRP, we can see that similar solutions are found during LAN and WAN executions in a slightly lower runtime for the two instances in WAN. We did check statistical significance for these results, what effectively make us to claim that the WAN execution is more efficient than the LAN one in this case. This is a promising result, that can be explained because of the considerably larger grain of the VRP problem with respect to the other problems, and due to the longer isolation provided by the WAN for the parallel GA islands (which results advantageous).

We now proceed to the second phase (continuous optimization) in which we analyze the behavior of a ring of evolution strategies running in parallel to solve RAS and FMS. The results are shown in Table 3. We can see that the conclusions we made for the parallel GA also hold in the case of the parallel ES. We can notice the higher execution times of the WAN version of the algorithm for the two problems, while the final fitness are approximately the same. Besides, we show that the numerical effort is almost the same in the LAN and WAN cases, what is a signal of correct implementation.

Finally, we want to check our findings in a new arena, i.e. on hybrid parallel algorithms. In Table 4 we include the results for RAS (three first rows) and MaxCut (three lower rows). We selected them as representatives of the class of continuous and combinatorial problems. This is why we use the CHCES hybrid for RAS and the GASA hybrid for MaxCut. After these results the above conclusions we got for PGA and parallel ES really hold. It seems that all the algorithms present a clear higher time when running in WAN

Problem	Popsize	C. prob.	M. prob.	Others
MaxCut, VRP (GA)	100	0.8	0.01	-
MTTP, ECC (GA)	200	0.8	0.02	-
RAS, FMS (ES)	100	0.3	0.8	-
MaxCut (GASA)	100	0.8	0.01	SA operator applied with prob 0.1, MarkovChainLen = 10, 100 iterations
RAS (CHCES)	100	0.8	-	35% population restart, (1+10)-ES operator (prob 0.01)

Table 1. Parameters of algorithms.

	LAN				WAN				p-value	
	opt	#evals	time	hits	opt	#evals	time	hits	eval	time
MaxCut	1031	23580	49.1	17%	1014	14369	89.1	10%	0.906	8.13e-9
MTTP	201	40002	5.2	97%	200	32546	25.7	100%	0.091	3.74e-10
ECC	0.0642	18279	9.1	7%	0.0657	26041	238.4	10%	0.428	1.13e-14
VRP 1	696.15	12843	78.9	100%	690.693	7168	68.5	100%	0.917	4.78e-4
VRP 2	1080.67	5873	391.1	100%	1077.83	8104	358.7	100%	0.725	4.99e-6

Table 2. Average results for a parallel GA.

	LAN				WAN				p-value	
	opt	#evals	time	hits	opt	#evals	time	hits	eval	time
RAS	0	10763	5.7	97%	0	11546	67.5	93%	0.171	2.20e-16
FMS	0.099	10904	4.7	100%	0.093	11372	13.0	100%	0.450	3.26e-5

Table 3. Overall best results for a parallel ES Algorithm.

but needing a similar number of iterations to yield the optimum. The LAN behavior can be assumed then to appear in the WAN environment from a numerical point of view.

We conclude this section with a summary of the results. We have shown that, whatever the basic search method is used (GA, ES or hybrids) an execution in a WAN environment presents an equivalent numerical result, while showing a natural larger time to achieve a solution with respect its LAN execution. Of course, in case of really high number of processes this could not hold; this question remains open, and has a wide relevance in the case of grid computing platforms, but not for a moderate or low number of sub-algorithms.

The surprising result appears when analyzing the VRP problem, because the WAN execution reduces the search time with respect to the LAN execution, what seems a counter-intuitive conclusion. We explain this scenario because of the considerable computation time needed by the fitness function (which makes more similar the LAN and WAN search), and also because of the larger isolation time induced in practice in the WAN platform that, for this problem, represents a more efficient parameterization.

We are conducting additional tests to validate the assumption that a given migration frequency in an asynchronous parallel algorithm in LAN produces, when used in WAN, an implicit lower frequency of migrations. We should be able of validating this by running a LAN version of the algorithm with such a lower frequency of migration in order to find whether the results are similar to the WAN *de facto* frequency.

6 Concluding Remarks and Future Work

In this work we have provided an important evidence to create a body of knowledge that will lead us from our understanding of LAN parallel optimization heuristics to their WAN execution. Our aim is to point out the important facts and to try to explain them. We have selected a non-trivial benchmark for which a WAN execution comes as a natural research scenario.

The conclusions of this work can be summarized attending to different criteria. With respect the numerical behavior, LAN and WAN results seem very similar, and this similarity seems to be readily stable. This is very interesting since this allow us to reduce our discussion to another criteria: real time execution. With this goal in mind, we have found that WAN versions are consistently slower than LAN

	LAN				WAN				p-value	
	opt	#evals	time	hits	opt	#evals	time	hits	eval	time
CHCES1	0	13048	3.7	100%	0	13681	10.0	100%	0.091	8.62e-6
CHCES2	0	8093	3.4	100%	0	8593	7.2	100%	0.849	3.97e-5
CHCES3	0	8182	3.4	100%	0	9635	7.9	100%	0.127	7.89e-11
GASA1	1038	40682	89.6	17%	1031	28956	298.5	10%	0.529	2.97e-5
GASA2	1031	19477	45.7	7%	1024	17412	123.5	8%	0.220	0.0029
GASA3	1038	21651	43.8	8%	1021	12162	202.9	7%	0.755	0.0344

Table 4. Overall best results of all experimental runs performed for Hybrid Algorithms.

ones, as one could expect. However, the harder the problem becomes the more similar LAN and WAN algorithms are.

In the limit (i.e., in our harder instances) we have found that WAN executions are even more efficient than LAN ones. This is by no means a question of faster execution, since WAN has always higher latencies and delays, but it is an effect of isolation. We hypothesize that running these algorithms in WAN is somewhat similar to running them in a LAN cluster but with higher isolation times among the sub-algorithm components. We confirmed this hypothesis on some problem instances. For some complex problems, large isolation times promote alternating phases of exploration and exploitation in a natural manner, which yield an algorithm of higher efficiency, since isolation time is a well-known influent parameter in distributed parallel algorithms (e.g. see [4] for dGAs).

We have also extended the whole study to hybrid algorithms. Hybridization is a natural way of enhancing the search algorithm itself.

As a future work we plan to quantify the relationship between the isolation time imposed by the WAN network and the isolation time in a LAN cluster, and to check this result on additional problems and algorithms.

Acknowledgments

This work has been partially funded by the Ministry of Science and Technology and FEDER under contract TIC2002-04498-C05-02 (the TRACER project).

References

[1] E. Alba and S. Khuri. Applying evolutionary algorithms to combinatorial optimization problems. In *Proceedings of the International Conference on Computational Science*, volume 2074 (Part II) of *LNCS*, pages 689–700. Springer-Verlag, 2001.

[2] E. Alba and the MALLBA Group. MALLBA: A library of skeletons for combinatorial optimisation. In R. F. B. Monien, editor, *Proceedings of the Euro-Par*, volume 2400 of *LNCS*, pages 927–932, Paderborn (GE), 2002. Springer-Verlag.

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

[4] E. Alba and J. M. Troya. Influence of the migration policy in parallel distributed GAs with structured and panmictic populations. *Applied Intelligence*, 12(3):163–181, 2000.

[5] N. Christofides, A. Mingozzi, and P. Toth. *Combinatorial optimization*, chapter 11. John Wiley, 1979.

[6] L. Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[7] L. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann, 1991.

[8] S. Khuri, T. Bäck, and J. Heitkötter. An evolutionary approach to combinatorial optimization problems. In *Proceedings of the 22nd ACM Computer Science Conference*, pages 66–73, Phoenix, Arizona, 1994. ACM Press.

[9] G. Laporte, M. G. J.-Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *Les Cahiers du GERAD*, 1999.

[10] Z. Michalewicz and D. B. Fogel, editors. *How to Solve It: Modern Hueristics*. Springer, 2000.

[11] A. Töorn and Ž. Antanas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 1989.

[12] S. Tsutsui, A. Ghosh, D. Corne, and Y. Fujimoto. A real coded genetic algorithm with an explorer and an exploiter populations. In T. B. et al., editor, *Proceeding of the Seventh International Conference on Genetic Algorithms*, pages 238–245. Morgan Kaufmann, 1997.