

Parallel Island-Based Genetic Algorithm for Radio Network Design

P. Calégari, F. Guidec, P. Kuonen, D. Kobler
École Polytechnique Fédérale de Lausanne (Switzerland)

PAPER SUBMITTED FOR PUBLICATION IN THE JOURNAL OF PARALLEL
AND DISTRIBUTED COMPUTING (JPDC)
SPECIAL ISSUE ON PARALLEL EVOLUTIONARY COMPUTING

ACCEPTED AND PUBLISHED IN NOVEMBER 1997, VOL. 47 N.1, PAGES 86-90



Parallel island-based genetic algorithm for radio network design

1 Introduction

The LEOPARD¹ project aims at studying the characteristics of evolutionary population-based algorithms, such as Genetic Algorithms (GA), in order to find out how these algorithms can be parallelized in the most efficient way. An original realistic combinatorial optimization problem is used as a test-bench to evaluate the efficiency of the parallel programs that are developed in the project. This problem is taken from the telecommunication world. One of the issues telecommunication companies must face when deploying a network of radio transmitters is the selection of a good set of sites for siting these transmitters. The problem comes down to guaranteeing the best service at a minimum cost in a given geographical

¹The LEOPARD project (parallel population-based methods for combinatorial optimization) is funded by the Swiss National Science Foundation.

area, that is, to covering a maximum surface of the area with a minimum number of transmitters. A geographic location is said to be covered when it can receive the signal broadcasted by a transmitter at any time. The coverage of an area is defined as the surface that is covered by a set of transmitters. Our goal is to select the best subset of sites capable of satisfying the coverage requirements.

The rest of this paper is organized as follows. Section 2 shows that this problem can be modeled as a variant of the well-known: the set covering problem. The first results obtained with a GA program that was developed and tested on artificial data are shown in Section 3. Section 4 introduces the concept of ‘islands’ and shows how our algorithm was parallelized, based on this concept. The performance observed with the resulting island-based parallel GA is discussed in Section 5. A real-life problem is considered in Section 6 and solved with our algorithm: the placement of fixed radio transmitters for a mobile phone network in the French north-eastern region “les Vosges”.

2 Problem modelling

The radio coverage problem introduced in Section 1 amounts to covering an area with a set of transmitters. The part of an area that is covered by a transmitter is called a *cell*. A cell is usually not connex. In the rest of this paper, we assume that the cells and the area considered are discretized, that is, they can be described as

a finite collection of geographical locations (taken from a geo-referenced grid, for example). The computation of cells may be based either on sophisticated wave propagation models, on measurements, or on draft estimations. In any case, we assume that cells can be computed and returned by an *ad hoc* function.

Let us consider the set L of all potentially covered locations, and the set M of all potential transmitter locations. Let G be the graph $(M \cup L, E)$, where E is a set of edges such that each transmitter location is linked to the locations it covers. As the geographical area needs to be discretized, the potentially covered locations are taken from a grid, as shown in Figure 1.

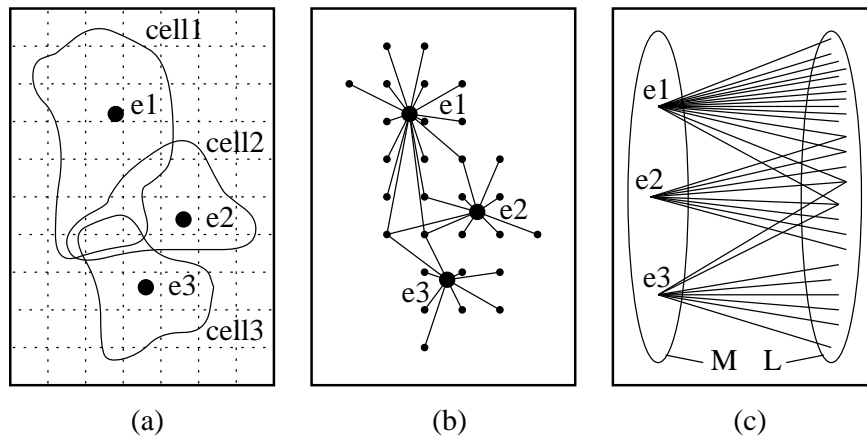


Fig. 1: Three potential transmitters e_1, e_2, e_3 and the associated cells (a) are discretized on a grid. A graph, whose edges link transmitters to the locations they cover, is then built (b). This graph can be represented as a bipartite graph (c) where M is the set of potential transmitter locations and L is the set of potentially covered locations.

Searching for the minimum subset of transmitters that covers a maximum surface of an area comes to searching for a subset $M' \subseteq M$ such that $|M'|$ is minimum

and such that $|Neighbours(M', E)|$ is maximum, where $Neighbours(M', E) = \{u \in L \mid \exists v \in M', (v, u) \in E\}$.

The problem we consider recalls the Unicast Set Covering Problem (USCP), that is known to be NP-hard [5]. The radio coverage problem differs however from the USCP in that the goal is to select a subset of transmitters that ensures a good coverage of the area, and not to ensure a total coverage. The difficulty of our problem arises from the fact that the goal is twofold, no part being secondary. If minimizing $|M'|$ was the primary goal, the solution would be trivial ($M' = \emptyset$). If maximizing the number of covered locations was the primary goal, then the problem would be the USCP. We therefore build an objective function f to combine the two goals:

$$f(\text{solution}) = \frac{\text{CoverRate}^\alpha}{\text{number of transmitters selected}}$$

where $CoverRate = 100 * \frac{|Neighbour(M')|}{|Neighbour(M)|}$. The parameter α can be tuned to favor the cover rate with respect to the number of transmitters. The goal is now to maximize f . So far, telecommunication companies have considered that the results obtained with $\alpha = 2$ are of best quality.

Because the likely NP-hardness of the problem, an exact algorithm was not investigated in our project. A suboptimal algorithm can, however, be used to find a satisfactory solution for this combinatorial optimization problem in a short time. There exist many different classes of algorithms for combinatorial optimization

problems. One of the most recent and promising techniques are represented by the so-called *evolutionary algorithm class*, that covers *genetic algorithms* (GA), *scatter search algorithms* [6], *ant systems* [3], etc. One of the major advantages of these algorithms is that, they are parallel in essence. In the remaining sections we shall focus on the GA approach.

3 First tests with a standard genetic algorithm

In a first step, we have implemented a classic generational replacement GA, as described in [4] and [7], to solve the radio coverage problem modeled in Section 2.

This algorithm works on a set, called a *population*, of candidate solutions of the problem. Each candidate solution, called an *individual*, is represented by a bit string that indicates for each entry if the corresponding transmitter location is selected or not. The objective function f defined in Section 2 serves as a *fitness function*. To allow fair comparisons, all executions of a GA terminate after a predefined number of generations. In the remaining sections, this algorithm will be referred to as the Standard Genetic Algorithm (SGA).

The SGA was first tested using artificial data to evaluate the quality of the results obtained with this algorithm, since we know the optimum solution *a priori* in this case. The results obtained with realistic data will be discussed in Section 6.

For our first study, a set of dummy potential transmitters was generated. On

a 287×287 point grid representing an open-air flat area, 49 primary transmitters were distributed regularly in order to form a 7×7 grid structure, and each transmitter was associated a 41×41 point cell. The coverage hence obtained is total. A hundred complementary transmitter locations were then randomly selected, associated to 41×41 point cells (fewer when clipped by the border of the area), and shuffled with the 49 primary ones. By construction, the best solution for this example is the one that covers 100% of the area with the 49 primary transmitters (giving a fitness value of about 204.08).

After extensive testing of the SGA with different population sizes on a Sparc-4 workstation, the highest fitness value (125.4) was obtained after running 250 generations (or iteration steps) of a population of 160 individuals. The execution took 9 hours. The solution is shown in Figure 2.a.

The SGA has two major shortcomings. First, it is rather slow, and therefore it cannot be used interactively as required by telecommunication operators. Second, the solutions obtained remain far from the optimum solution.

4 Toward a parallel island-based genetic algorithm

Two reasons can explain the slowness of the program: first, the huge bipartite graph that models coverage relationships must be run through every time the fitness value of an individual must be computed. We are currently considering

several techniques for reducing the size of this graph, but these are out of the scope of this paper. Second, a population must contain many individuals for the SGA to give good results, hence a large computation load. The quantity of independent processing (computation of fitness values, mutation operations, etc) required for each individual suggests that the algorithm is parallel in essence. The computation time could thus be decreased by distributing the individuals on several processors.

The low quality of the results is a consequence of the over rapid convergence of the SGA (the fitness value stops increasing too soon).

Many studies have been done to improve the quality of the results obtained with genetic algorithms [10]. One of them consists in distributing the population in subpopulations that evolve independently [9]. These subpopulations, called islands (or demes), increase the chances for the algorithm to find good solutions by having many SGAs competing, each of them operating on small populations. In order to let some islands benefit from the information found by others, some individuals are allowed to *migrate* from one island to another. This mechanism is called migration.

In our implementation [1], the population is distributed on a network of workstations² so that each workstation is in charge of one island.

²10 groups of 9 sparc-4 workstations (with 64Mb of RAM) are connected by FDDI, and the workstations are connected by one ethernet within each group.

The islands are virtually positioned on an oriented ring, and migrations are only allowed along that ring. This topology was chosen so as to minimize the amount of migrations, and thus to minimize the communication load due to migrations between remote islands. Every time a new generation is computed, a copy of the best individual (that with the greatest fitness value) ever met by each island is sent to the next island on the ring. Each island thus receives a new individual that replaces a randomly selected local individual. The number of iteration steps is fixed, and an implicit synchronisation is established by communications between islands. This Island-based Parallel Genetic Algorithm will be called IPGA in the remainder of the paper. The program is written in C++ and it uses the PVM library for communications.

5 Quality of the results and speedup

Figure 2.b shows the solutions obtained with 4 islands of 40 individuals each. This solution is better than that found with a single population of 160 individuals (see Figure 2.a). The solution obtained with 40 islands of 4 individuals each, which is shown on Figure 2.c, is even better. The fitness values are 125.3 for the solution obtained without islands, 177 with 4 islands, and 193.8 with 40 islands.

According to these results, it seems that the number of islands influences significantly the quality of the results returned by the IPGA. A possible explanation

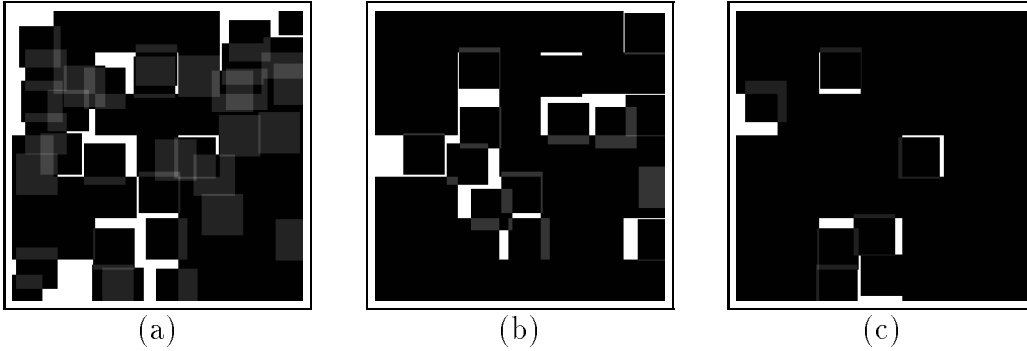


Fig. 2: Solutions obtained with the genetic algorithm with 160 individuals without islands (a), with 4 islands of 40 individuals each (b), and with 40 islands of 4 individuals each (c). White locations are not covered, black locations are covered once, grey locations are covered several times.

of this phenomenon is that, if the population of an island tends to converge too quickly towards a local optimum, this convergence can be slowed down by the arrival of an individual migrating from another island. In other words, a good individual arriving in a homogeneous population can lead to some interesting diversification.

Figure 3 shows the evolution of the best fitness value ever found during 250 generations over a population of 160 individuals. It shows that the fitness value increases quicker and more regularly when the population is distributed on 10 islands than when it is considered as a whole (as with the SGA).

Since the total population size is kept constant for the experiment, and since one individual migrates from each island, the migration rate

$$R_m = \frac{\text{number of individuals that migrate}}{\text{population size}} \text{ increases when the number of islands}$$

increases. As migrations imply diversification in the subpopulations, this could partially explain the change in the behavior of the algorithm with respect to the number of islands.

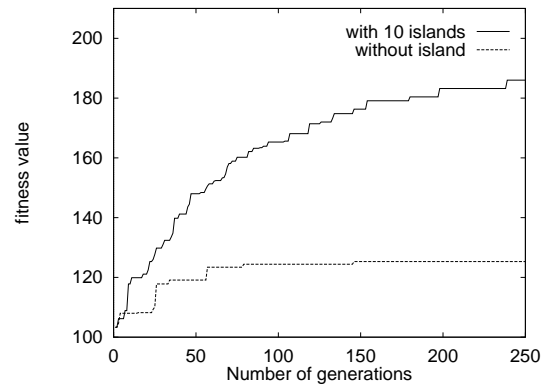


Fig. 3: Evolution of the best fitness value observed on a single population of 160 individuals, and on 10 islands of 16 individuals each.

Figure 4.a shows the execution speedup of IPGA when the individuals are distributed on workstations, while keeping the total size of the population constant and while having one island on each machine. It can be noticed that the speedup observed is surlinear. This is because IPGA behaves slightly differently for each number of workstations tested: the number of operations achieved during the selection step of the algorithm is indeed proportional to the square of the number of individuals per island.

In order to study the speedup of the program independently from the number of islands, a new version of IPGA was developed, allowing to keep this number constant. With this new version, each machine can be in charge of several islands.

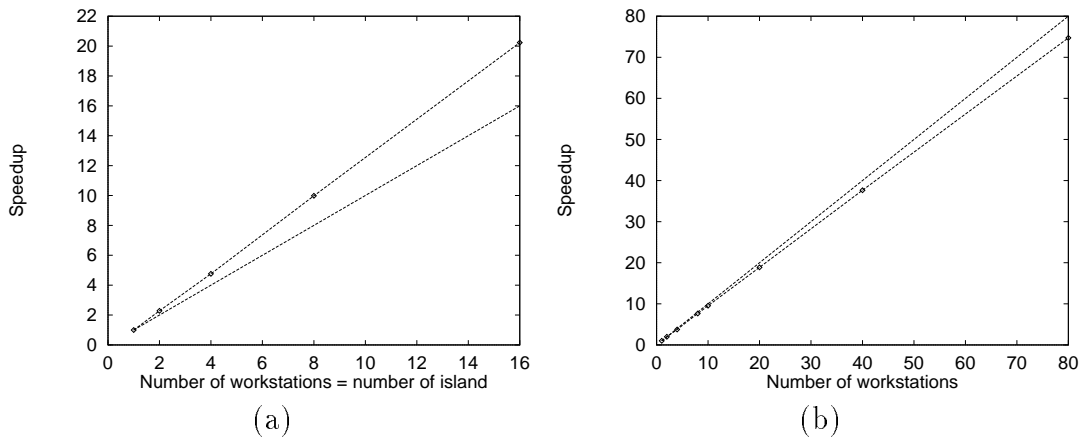


Fig. 4: (a) Speedup observed when distributing the 160 individuals of a population on a network of workstations, such that each machine is in charge of a single island. (b) Speedup of the parallel algorithm observed when running 250 iteration steps over 80 islands of 2 individuals each.

Figure 4.b shows the speedup obtained on a network of Sparc-4 workstations with 80 islands of 2 individuals each. It is almost linear and has an efficiency of 93% on 80 workstations. This good behavior of IPGA enables to obtain solutions in acceptable time. For example, the solution shown in Figure 2.c was found in 17 minutes on a network of 40 Sparc-4 workstations.

Island-based parallel genetic algorithms are often called ‘coarse grained parallel genetic algorithms’ in the literature [2]. Such naming is not convenient because it mixes up the concept of islands with the parallelization technique: as shown by the previous results, the two mechanisms are of different nature and should be clearly distinguished.

6 A real-life case: “les Vosges”

The results obtained on the set of artificial cells are most promising. It is however more interesting to observe the behavior of the algorithm on a real-life case: the placement of fixed radio transmitters for a mobile phone network. This problem is considered in the project STORMS³.

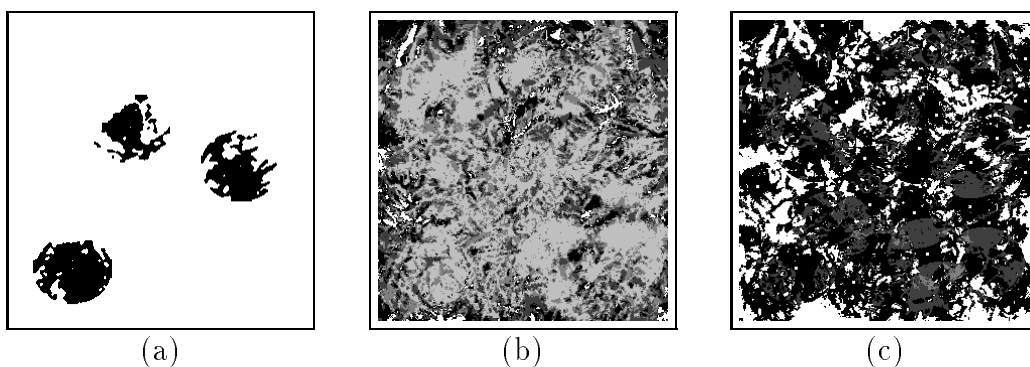


Fig. 5: Three cells (a). The initial set of simulated cells on the French region “les Vosges” (b) and a solution obtained with the parallel genetic algorithm using islands (c). White locations are not covered, black locations are covered once, grey locations are covered several times.

We used a $70km \times 70km$ digital terrain model of the French region “les Vosges”, discretized on a 300×300 point grid. Using 150 user-provided potential transmitter locations, 150 cells were computed by a wave propagation simulation tool⁴. Figure 5.a shows the coverage that could be obtained with 3 transmitters selected at random, and Figure 5.b that obtained with 150 user-provided trans-

³STORMS (Software Tool for the Optimization of Resources in Mobile Systems) is funded by EC and the Swiss Government

⁴The data and the tool [8] are provided by France Telecom-TDF.

mitters selected. Locations that are covered exactly once are colored in black, and those that are covered several times are colored in grey (the lighter the grey, the more transmitters covers them). It can be noticed that some locations (in white) cannot be covered by any of the user-provided potential transmitters. As a lot of points are covered many times, it is clear that some of the proposed transmitters are useless. Figure 5.c shows a solution that was obtained after the IPGA ran through 160 generations with a population of 4 individuals distributed on 40 islands. The result is a good compromise between the coverage and the number of transmitters: 80% of coverage is obtained with 40 transmitters.

Figure 6 confirms the idea, announced in Section 5 with artificial data, that the use of islands improves the quality of the results. The program was executed ten times with 160 individuals and for six different numbers of islands. The ten initial populations, generated randomly, are exactly the same for the six tests.

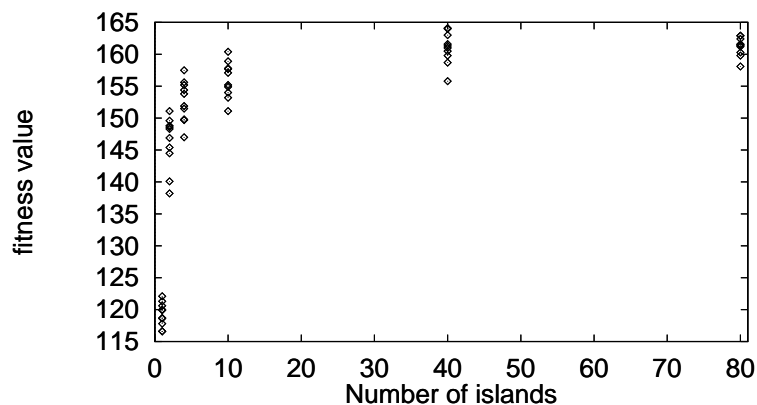


Fig. 6: Variation of the fitness value of the solutions found (after 250 generations) with respect to the number of islands on which the 160 individuals are distributed.

7 Conclusion

In this paper we presented our approach to solve a real life problem in the field of telecommunication. We first modeled the problem as a variant of the set covering problem. We then investigated the use of genetic algorithms to solve this problem. In order to be able to evaluate precisely the quality of the solution we tested this approach on a artificial set of data and on a real-life case.

Island model and parallelism were introduced to improve the quality of the results and to decrease the execution time. The parallel version of our island-based GA has shown a very good scalability and the quality of the results was considered as satisfying. Future works will aim at decreasing the size of the graph, and speeding up the computation of the fitness function.

References

- [1] CALÉGARI, P., KUONEN, P., GUIDEC, F., AND WAGNER, D. A Genetic Approach to Radio Network Optimization for Mobile Systems. In *Proceedings of the IEEE 47th Vehicular Technology Conference (VTC)* (May 1997), vol. 2, pp. 755–759.
- [2] CANTÛ-PAZ, E. A Summary of Research on Parallel Genetic Algorithms. Tech. rep., Illinois Genetic Algorithms Laboratory, 1995.

- hal-00346551, version 1 - 12 Dec 2008
- [3] COLORNI, A., DORIGO, M., AND MANIEZZO, V. Distributed Optimization by Ant Colonies. In *First European Conference on Artificial Life (1991)*, M. Press, Ed., Bradford Books, pp. 134–142.
 - [4] DAVIS, L. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.
 - [5] GAREY, M., AND JOHNSON, D. *Computers and Intractability: a Guide to the Theory of NP-completeness*. Freeman and Co., 1979.
 - [6] GLOVER, F. Genetic Algorithms and Scatter Search: unsuspected Potentials. *Statistics and Computing*, 4 (1994), 131–140.
 - [7] GOLDBERG, D. *Genetics algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Company, Inc., 1989.
 - [8] KUONEN, P., JOSSELIN, S., UBÉDA, S., AND WAGNER, D. Parallel Computing of Radio Coverage. In *Proceedings of the IEEE VTC 46th Vehicular Technology Conference (VTC) (Apr. 1996)*, vol. 3, pp. 1438–1442.
 - [9] TANESE, R. Distributed Genetic Algorithms. In *Second International Conference on Genetic Algorithms (1989)*, J. Schaffer and P. Laboratories, Eds., Morgan Kaufmann.
 - [10] WHITLEY, D. A genetic algorithm tutorial. Tech. Rep. CS-93-103, Colorado State University, 1993.