Assigning cells to switches in cellular mobile networks: a comparative study

Alejandro Quintero, Samuel Pierre*

Mobile Computing and Networking Research Laboratory (LARIM), Department of Computer Engineering, Ecole Polytechnique of Montreal, P.O. Box 6079, Station Centre-ville, Montréal, Que., Canada H3C 3A7

Received 13 February 2002; revised 24 July 2002; accepted 27 September 2002

Abstract

Assigning cells to switches in cellular mobile networks is an NP-hard problem which, for real-size mobile networks, could not be solved by using exact methods. In this context, heuristic approaches like genetic algorithms, tabu search, and simulated annealing can be used. This paper proposes a comparative study of three heuristics—tabu search, simulated annealing, and Parallel Genetic Algorithm with Migrations (PGAM)—used to solve this problem. The implementation of these algorithms has been tested in order to measure the quality of solutions. The results obtained confirm the efficiency of the heuristics to provide good solutions for medium- and large-sized cellular mobile networks, in comparison with the sequential genetic algorithm (SGA) and with other heuristic methods well known in the literature. The evaluation cost provided by tabu search, simulated annealing, and PGAM are very similar. Those three algorithms provide better results than standard genetic algorithm and always find feasible solutions. In terms of CPU times, tabu search is the fastest method. Finally, the results have been compared with a lower bound on the global optimum; they confirm the effectiveness of the tabu search, simulated annealing, and PGAM to solve this problem.

© 2003 Elsevier Science B.V. All rights reserved.

Keywords: Cellular networks; Cell assignment; Optimization; Heuristics; Genetic algorithms; Tabu search; Simulated annealing

1. Introduction

During the last decades, telecommunications field has known an extraordinary development. The need to transfer information between users, anywhere at anytime, led to the cellular mobile networks. In a typical cellular network, the area of coverage is often geographically divided into hexagonal cells. The cell is the basic unit of a cellular system (Fig. 1). Each cell contains a base station covering a small geographic area. Each base station uses an antenna for communications among users with pre-assigned frequencies. A number of cells are chosen to install switches that communicate with each other and serve as relays for communication between cells and provider backbone. Because of users’ mobility, switches serving as relays for a given user could change if the latter moves from its current cell to another during a call. As adjacent areas do not use the same radio channels, a call must either be dropped or transferred from one radio channel to another when a user crosses the line between adjacent cells. This operation is called a handoff and it occurs when the mobile network automatically transfers a call in progress from one cell, using a frequency pair, to another adjacent cell using a different frequency pair with an uninterruptible call. When a handoff occurs between two cells linked to the same switch, it is called a simple handoff, because there are few necessary updates. A complex handoff refers to a handoff between two cells related to different switches; in this case, the update procedures consume more resources than in the case of simple handoff.

This paper presents a comparative study of three heuristics—tabu search, simulated annealing, and Parallel Genetic Algorithm with Migrations (PGAM)—proposed to efficiently solve the problem of assigning cells to switches in cellular mobile networks. Section 2 presents the background and related work. Section 3 describes the parallel genetic algorithm and tabu search approaches and presents some
adaptation and implementation details. Section 4 analyzes results.

2. Background and related work

This assignment problem consists of determining a cell assignment pattern which minimizes a certain cost function while respecting certain constraints, especially those related to limited switch’s capacity. An assignment of cells can be carried out according to a single or a double cell’s homing. A single homing of cells corresponds to the situation where a cell can only be assigned to a single switch. When a cell is related to two switches, that refers to a double homing. In this paper, only single homing is considered.

Let \( n \) be the number of cells to be assigned to \( m \) switches. The location of cells and switches is fixed and known. Let \( H_{ij} \) be the cost per unit of time for a simple handoff between cells \( i \) and \( j \) involving only one switch, and \( H'_{ij} \) the cost per time unit for a complex handoff between cells \( i \) and \( j \) involving two different switches. \( H_{ij} \) and \( H'_{ij} \) are proportional to the handoff frequency between cells \( i \) and \( j \). Let \( c_{ik} \) be the amortization cost of the link between cell \( i \) and switch \( k (i = 1, ..., n; k = 1, ..., m) \).

Let \( x_{ik} \) be a binary variable, equal to 1 if cell \( i \) is related to switch \( k \), otherwise \( x_{ik} \) is equal 0.

The assignment of cells to switches is subject to a number of constraints. Actually, each cell must be assigned to only one switch, which can be expressed as follows:

\[
\sum_{k=1}^{m} x_{ik} = 1 \quad \text{for} \quad i = 1, ..., n. \tag{1}
\]

Let \( z_{ijk} \) and \( y_{ij} \) be defined as:

\[
z_{ijk} = x_{ik} x_{jk} \quad \text{for} \quad i, j = 1, ..., n \quad \text{and} \quad k = 1, ..., m, \quad \text{with} \quad i \neq j.
\]

\[
y_{ij} = \sum_{k=1}^{m} \quad \text{for} \quad i, j = 1, ..., n, \quad \text{and} \quad i \neq j.
\]

\( z_{ijk} \) is equal to 1 if cells \( i \) and \( j \), with \( ij \), are both connected to the same switch \( k \), otherwise \( z_{ijk} \) is equal to 0. Thus \( y_{ij} \) takes the value 1 if cells \( i \) and \( j \) are both connected to the same switches and the value 0 if cells \( i \) and \( j \) are connected to different switches.

The cost per time unit \( f \) of the assignment is expressed as follows:

\[
f = \sum_{i=1}^{n} \sum_{k=1}^{m} c_{ik} x_{ik} + \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} H_{ij}(1 - y_{ij}) + \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} H'_{ij} y_{ij} \tag{2}
\]

The first term of the equation represents the link or cabling cost. The second term takes into account the complex handoffs cost and the third, the cost of simple handoffs. We should keep in mind that the cost function is quadratic in \( x_{ik} \), because \( y_{ij} \) is a quadratic function of \( x_{ik} \).

Let’s mention that an eventual weighting could be taken into account directly in the link and handoff costs definitions.

The capacity of a switch \( k \) is denoted \( M_k \). If \( \lambda_i \) denotes the number of calls per time unit destined to \( i \), the limited capacity of switches imposes the following constraint:

\[
\sum_{i=1}^{n} \lambda_i x_{ik} \leq M_k \quad \text{for} \quad k = 1, ..., m \tag{3}
\]

according to which the total load of all cells which are assigned to the switch \( k \) is less than the capacity \( M_k \) of the switch. Finally, the constraints of the problem are completed by:

\[
x_{ik} = 0 \quad \text{or} \quad 1 \quad \text{for} \quad i = 1, ..., n \quad \text{and} \quad k = 1, ..., m. \tag{4}
\]

\[
z_{ijk} = x_{ik} x_{jk} \quad \text{and} \quad i, j = 1, ..., n \quad \text{and} \quad k = 1, ..., m. \tag{5}
\]
Eqs. (1), (3) and (4) are constraints of transport problems. In fact, each cell \( i \) could be assimilated to a factory which produces a call volume \( A_i \). The switches are then considered as warehouses of capacity \( M_k \) where the cells production could be stored. Therefore, the problem is to minimize Eq. (2) under Eqs. (1), and (3)–(6). When the problem is formulated in this way, it could not be solved with a standard method such as linear programming because constraint (5) is not linear. Merchant and Sengupta [20,21] replaced it by the following equivalent set of constraints:

\[
\begin{align*}
 z_{ijk} &\leq x_{ik} \quad (7) \\
 z_{ijk} &\leq x_{jk} \quad (8) \\
 z_{ijk} &\geq x_{ik} + x_{jk} - 1 \quad (9) \\
 z_{ijk} &\geq 0 \quad (10)
\end{align*}
\]

Thus, the problem could be reformulated as follows: minimizing Eq. (2) under constraints (1), (3), (4) and (6)–(10). We can further simplify the problem by defining:

\[
 h_{ij} = H'_{ij} - H_{ij},
\]

\( h_{ij} \) refers to the reduced cost per unit of a complex handoff between cells \( i \) and \( j \). Relation (2) is then re-written as follows:

\[
f = \sum_{i=1}^{n} \sum_{k=1}^{m} c_{ik}x_{ik} + \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} h_{ij}(1 - y_{ij}) + \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} H_{ij} \quad \text{subject to: Eqs. (1), (3), (4) and (7)–(10).}
\]

The assignment problem takes then the following form:

\[
 f = \sum_{i=1}^{n} \sum_{k=1}^{m} c_{ik}x_{ik} + \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} h_{ij}(1 - y_{ij}) \quad \text{subject to: Eqs. (1), (3), (4) and (7)–(10).}
\]

In this form, the assignment problem could be solved by usual programming methods.

The assignment problem may be reduced to a number of well-known operational research problems, such as graph partitioning or p-fixed hubs location problem. Our analysis is focus on the single homing assignment problem and the p-fixed hubs location problem. For further details on the graph partitioning, see [20] and [21]. The formulation of the p-fixed hubs location problem, which corresponds the best to the assignment problem, is known as the p-fixed hubs location problem and was introduced by Skorin-Kapov et al. [32] and Sohn and Park [33].

In this paper, the total cost includes two types of cost, namely cost of handoff between two adjacent cells, and cost of cabling between cells and switches. The design is to be optimized subject to the constraint that the call volume of each switch must not exceed its call handling capacity. This kind of problem is NP-hard, so enumerative searches are practically inappropriate for moderate- and large-sized cellular mobile networks [3,21,27]. Thus, heuristic approaches have been developed for this kind of problem [1–3,10,21,26,27].

### 3. Tabu search, simulated annealing and genetic algorithm approaches

The geographical relationships between cells and switches are considered in the value of the cost of cabling, so that the base station of a cell is generally assigned to a neighboring switch and not to far switches [36]. In Ref. [34], an engineering cost model has been proposed to estimate the cost of providing personal communications services in a new residential development. The cost model estimated the costs of building and operating a new PCS using existing infrastructure such as the telephone, cable television and cellular networks. In Ref. [9], economic aspects of configuring cellular networks are presented. Major components of costs and revenues and the major stakeholders were identified and a model was developed to determine the system configuration (e.g. cell size, number of channels, etc.). For example, in a large cellular network, it is impossible for a cell \( i \) located in east America to be assigned to a switch \( k \) located in west America. In this case, the variable \( c_{ik} \) is \( \infty \). Therefore, the design is to be optimized subject to the constraint that the call volume of each switch must not exceed its call handling capacity.

#### 3.1. Genetic algorithms approach

Genetic Algorithms (GA) are based on the Darwin’s concept of natural selection. They essentially consist of creating a population of candidates and applying probabilistic rules to simulate the evolution of the population. GAs are robust search techniques based on natural selection and genetic mechanisms [12,15,31].

##### 3.1.1. Basic principles of classical genetic algorithms

Genetic algorithms (GA) are robust search techniques based on natural selection and genetic production mechanisms [29]. GAs perform a search by evolving a population of candidate solutions through non-deterministic operators and by incrementally improving the individual solutions forming the population using mechanisms inspired from natural genetics and heredity (e.g. selection, crossover and mutation). In many cases, especially with problems characterized by many local optima (graph coloring, travelling salesman, network design problems, etc.), traditional optimization techniques fail to find high quality solutions. GAs can be considered as an efficient and interesting option [15].
GAs [12] are composed of three phases: a phase of creation of an initial population, a phase of alteration of this population by applying various genetic operators on its elements, and finally a phase of evaluation of this population during a certain number of generations. Each generation is supposed to provide new elements better than those of the preceding generation. Intuitively, the more larger is the number of generations, the more refined is the solution. It is hoped that the last generation will contain a good solution, but this solution is not necessarily the optimum [8].

We have introduced a simple notation to represent cells and switches, and to encode chromosomes and genes. We opted for a non-binary representation of the chromosomes [13]. In this representation, the genes (squares) represent the cells, and the integers they contain represent the switch to which the cell of row $i$ (gene of the $i$th position) is assigned. Our chromosomes have therefore a length equal to the number of cells in the network $n$, and the maximal value that a gene can take is equal to the maximal number of switches $m$. A chromosome represents the set of cells in the cellular mobile network, and the length is the number of cells. Fig. 2 shows an example of a chromosome representing an assignment plan of $n$ cells to $m$ switches. We can notice the adopted coding satisfies a constraint: the unique assignment of cells to switches, because a gene of a chromosome cannot take simultaneously more than one value.

**Crossover** is a process by which two chosen string genes are interchanged. The crossover of a string pair of length $l$ is performed as follows: a position $i$ is chosen uniformly between 1 and $(l - 1)$, then two new strings are created by exchanging all values between positions $(i + 1)$ and $l$ of each string of the pair considered (Fig. 3b).

**Mutation** is the process by which a randomly chosen gene in a chromosome is changed (Fig. 3a). It is employed to introduce new information into the population and also to prevent the population from becoming saturated with similar chromosomes.

The next generation of chromosomes is generated from present population by selection and reproduction. The selection process is based on the fitness of the present population, such that the fitter chromosome contributes more to the reproductive pool; typically this is also done probabilistically.

### 3.1.2. The parallel approach of genetic algorithms

Classical genetic algorithms are powerful and perform well on a broad class of problems. However, part of the biological and cultural analogies used to motivate a genetic algorithm search are inherently parallels.

One approach is the partitioning of the population into several subpopulations (multi-population approach) [35]. The evolution of each subpopulation is handled independently from each other and help maintain genetic diversity. Diversity is the term used to describe the relative uniqueness of each individual in the population. From time to time, there is however some interchange of genetic material between different subpopulations. This exchange of individuals is called *migration* [24].

The migration algorithm is controlled by many parameters that affect its efficiency and accuracy. Among other things, one must decide the number and the size of the populations, the rate of the migration, the migration interval and the destination of the migrants. The migration interval is the number of generations between each migration, and the migration rate is the number of individuals selected for migration. For further details on the parameters associated with the migration algorithm, see [4, 35]. Fig. 4 shows the parallel genetic algorithm proposed.

### 3.2. Tabu search

A tabu search method is an adaptive technique used in combinatorial optimization to solve difficult problems [11, 26, 27]. tabu search can indeed be applied to different problems and different instances of problems, but mainly the local search neighborhood and the way the tabu list is built and exploited are subject to many variations, which

---

**Initialize population(ges)**
- `evaluation population(ges) while not terminated do`
- `apply crossover and mutation to population(ges) giving children(C)`
- `evaluation population(children(C))`
- `population(ges+1) = select from(children(C) \cup population(ges))`

**begin migration**
- `if migration appropriate`
- `Choose emigrants(population(ges+1))`
- `Send emigrants`
- `endif`
- `if immigrants available`
- `Receive immigrants(I)`
- `endif`
- `end migration`
- `population(ges+1) = select from(immigrants(I) \cup population(ges+1))`
- `gen = gen + 1`  

**end while**

---
gives to this method its meta-heuristic nature. The tabu list is not always a list of solutions, but can be a list of forbidden moves/perturbations [10].

Tabu search is a hill-climber endowed with a tabu list (list of solutions or moves). Let $X_i$ denote the current point; let $N(X_i)$ denote all admissible neighbors of $X_i$, where $Y$ is an admissible neighbor of $X_i$ if $Y$ is obtained from $X_i$ through a single move not in the tabu list, and $Y$ does not belong to the tabu list; replace $X_i$ with the best point in $N(X_i)$; stop after $nbmax$ steps or if $N(X_i)$ is empty.

Other mechanisms of tabu search are intensification and diversification: by the intensification mechanism, the algorithm does a more comprehensive exploration of attractive regions which may lead to a local optimal point; by the diversification mechanism, on the other hand, the search is moved to previously unvisited regions, something that is important in order to avoid getting trapped into local minimum points [10].

3.3. Simulated annealing approach

Simulated annealing (SA) was introduced by Metropolis et al. [23] and is used to approximate the solution of very large combinatorial optimization problems [17]. Besides the traditional greedy local search techniques, the stochastic properties of the SA algorithm prevent it to get stuck to local minima. On the other hand, in traditional greedy local search, the quality of the final result heavily depends on the initial solution. In contrast, the idea behind SA is to adequately explore the whole solution space early on so that the final solution is insensitive to the starting state [17].

Conversely to a local search algorithm, SA allows for a given optimization problem to accept solutions which deteriorate the cost, even if later, these solutions will be abandoned if they generate no improvements. SA uses randomness to decide whether to reject or accept a solution which deteriorates the cost.

The algorithm starts with an initial feasible solution, which is set as the current solution. Randomly, a neighboring solution from the solution space is chosen, and its cost is compared to that of the current solution. If the cost is improved, this neighbor solution is kept and set as the current solution. Otherwise, this solution is accepted with a probability that is calculated according to the stage the algorithm is in (we designate this stage via a variable called "temperature") [6].

4. Numerical results

In order to compare the performance of heuristics, two types of experiments were performed: a set of experiments to evaluate the quality of the solutions in terms of their costs, and another set of experiments to evaluate the performance in terms of CPU times. All the tests runs described in this section were performed in a networked workstation environment operating at 100 Mbps with SPCs (Pentium 500 MHz).

4.1. Parameters and numerical results

The experiences were executed by supposing that the cells are arranged on an hexagonal grid of almost equal length and width. The antennas are located at the center of cells and distributed evenly on the grid. However, when two or several antennas are too close to each other, the antenna arrangement is rejected and a new arrangement is chosen. The cost of cabling between a cell and a switch is proportional to the distance separating both. We took a proportionality coefficient equal to the unit. The call rate $\gamma_i$ of a cell $i$ follows a gamma law of average and variance equal to the unit. The call duration inside the cells are distributed according to an exponential law of parameter equal to 1. This is justified by the fact that we are considering a simple model. For more realistic call duration distributions and models, the reader is referred to [7]. If a cell $j$ has $k$ neighbors, the [0,1] interval is divided into $k + 1$ sub-intervals by choosing $k$ random numbers distributed evenly between 0 and 1. At the end of the service period in cell $j$, the call could be either transferred to the $i$th neighbour ($i = 1, \ldots, k$) with a handoff probability $h_{ij}$ equal to the length of $i$th interval, or ended with a probability equal to the length of the $k + 1$th interval. To find the call volumes and the rates of coherent handoff, the cells are considered as $M/M/1$ queues forming a Jacksonnet work [18]. The incoming rates $\alpha_i$ in cells are obtained by solving the following system:

$$\alpha_i = \sum_{j=1}^{n} \alpha_i r_{ij} = \gamma_i \text{avec } i = 1, \ldots, n$$

If the incoming rate $\alpha_i$ is greater than the service rate, the distribution is rejected and chosen again. The handoff rate $h_{ij}$ is defined by:

$$h_{ij} = \lambda_i r_{ij}$$

All the switches have the same capacity $M$ calculated as follows:

$$M = \frac{1}{m} \left(1 + \frac{K}{100}\right) \sum_{i=1}^{n} \lambda_i$$

where $K$ is uniformly chosen between 10 and 50, which insures a global excess of 10 to 50% of the switches’ capacity compared to the cells’ volume of call.

4.1.1. PGAM test generation

To calculate the parameters associated with PGAM, it was executed over a set of 600 test cases with 3 mobile networks in series of 20 tests for each assignment pattern. To define the number of populations PGAM was executed with a number of populations varying between 1 and 8. This experience shows that PGAM converges to provide good
solutions with a number of populations varying between 6 and 8 (Fig. 5).

To define the population size, PGAM was executed over a set of 150 test cases with 3 mobile networks in series of 20 tests for each assignment pattern with 8 populations. This experience shows that PGMA converges to provide good solutions with a population size varying between 80 and 140 (Fig. 6).

The values used by parallel and sequential genetic algorithms are: the number of generations is 200; the population size is 100; the number of populations is 8 for PGAM and 1 for SGA; the crossover probability is 0.9; the mutation probability is 0.08; the migration interval \( (P_m) \) is 0.1; the migration rate \( (S_m) \) is 0.4 and the emigrants accepted \( (P_r) \) is 0.2. A larger migration interval is normally used in connection with larger migration rate (Fig. 7).

Whereas the migration algorithm seeks to improve the normalized cost and reliability of the SGA, it is important also to ensure that unacceptable time overhead is not introduced by migration (Table 1). In order to analyze the performance of the parallel genetic algorithm without migration (PGA) independent of the migration algorithm (PGAM), migration is turned off \( (P_m = 0.0) \). Turning off migration for analysing the parallel genetic algorithm ensures that timing measurements indicate the effects of
parallel algorithm, rather than the effects of migration. In order to compare the performance of PGAM and PGA, a set of experiments were performed to evaluate the quality of the solutions in terms of their costs and to evaluate the performance in terms of CPU times. In these experiments, the results obtained by PGA are compared directly with PGAM. The two algorithms always provide the feasible solutions. However, PGAM yields an improvement in CPU times and in evaluation cost in comparison with PGA, because PGMA converges faster than PGA to good solutions with a small number of generations.

4.1.2. Comparison with tabu search method

To analyze the results provide by the different tabu’s memories, tabu search was executed on a large number of test cases inspired from Ref. [19]. The middle and long term memories improve the results provides by the short term memory.

Experimental results from solving different mobile networks topologies show that PGAM and tabu search approaches [27] provide very good results. In terms of CPU times to find a feasible solution, tabu search is faster than PGAM. The results generated in terms of cost by tabu search and PGAM are very similar, but a little better for tabu search. The two heuristics always find feasible solutions. An advantage of PGAM, in comparison with tabu search, is the number of different feasible solutions provided for each instance of problem. PGAM always provides a different solution and tabu search always provides the same feasible solution (Fig. 8). Some economic aspects of cellular network configuration are not considered in the model used in this paper, therefore, it is important to PCS provider to consider different feasible solutions and to choose the best, not only in terms of switch’s capacity and the cost of cabling, but in terms of other aspects like the revenues, the mayor stakeholders, etc.

4.2. Comparison with standard approach of genetic algorithm

In this section, we present the comparison results, the solutions cost and the CPU times, between PGAM, tabu search, simulated annealing and SGA. For the experiments, each approach was executed over a set of 450 test cases with 3 mobile networks in series of 50 tests for each assignment pattern.

Measuring the solution cost and the CPU time. In the first set of experiments, the results obtained by SGA are compared directly with PGAM, SA and tabu search results.

<table>
<thead>
<tr>
<th>Element</th>
<th>CPU time percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate an initial population</td>
<td>1</td>
</tr>
<tr>
<td>Evaluation of the chromosomes</td>
<td>60</td>
</tr>
<tr>
<td>Crossover and mutation operators</td>
<td>5</td>
</tr>
<tr>
<td>Immigration procedure</td>
<td>1</td>
</tr>
<tr>
<td>Selection procedure</td>
<td>25</td>
</tr>
<tr>
<td>Emigration procedure</td>
<td>1</td>
</tr>
<tr>
<td>Others</td>
<td>7</td>
</tr>
</tbody>
</table>

Fig. 7. Number of immigrants and PGAM convergence.
The four algorithms always find the feasible solutions. In each of the three considered series of test, PGAM, SA and tabu search yields an improvement in CPU times and in evaluation cost in comparison with SGA. In terms of CPU times, tabu search is the fastest method and PGAM is between 10 and 20 faster than SGA to find a feasible solution; in terms of costs, the results generated by tabu search, SA and PGAM are very similar; PGAM, SA and tabu search provide better results than SGA. Experimental results from solving different mobile networks show that our approach, with migration, provides better results than a standard approach of genetic algorithm.

Fig. 8. Comparison between tabu search's memories and PGAM.

Fig. 9. Comparison results between PGAM, SA, tabu search and SGA.
4.3. Comparison with simulated annealing approach

In this section, we compare PGAM and tabu search with the simulated annealing (SA) approach. For the experiments, tabu search, PGAM, sequential genetic algorithm and SA are executed over a set of 900 test cases with a number of cells varying between 100 and 200, and a number of switches varying between 5 and 7, that means the search space size is between $5^{100}$ and $7^{200}$. We did not consider the small topologies with small search spaces, because in these cases it is possible to use enumerative searches for solutions. Fig. 9 shows the results obtained for the 3 different mobile networks used in the tests, which the evaluation costs represents the average over all tests of each algorithm. In all the tests, PGAM, tabu search and SA provide feasible solutions with a very similar cost, and SGA provides feasible solutions with a bigger cost (Fig. 9a). In all the tests, PGAM and tabu search are faster than SA; SGA is the slowest method and tabu search is the fastest method (Fig. 9b).

4.4. Comparison with hybrid genetic algorithm approach

In problems characterized by many local optima, traditional optimization techniques fail to find high quality solutions. In these cases, GA can be considered as an efficient and interesting option. However, SGAs can suffer from excessively slow convergence before finding an accurate solution because of their characteristics of using a priori minimal knowledge and failure to exploit local information [5,25,28,30]. This may prevent them from being really of practical interest for a lot of large-scale constrained applications. Genetic algorithms promise convergence but not optimality. This implies that the choice of when to stop a genetic algorithm is not well defined. Since there is no guarantee of optimality, successive runs of the GA will provide different chromosomes with varying fitness measures. If we run a GA several times, it will converge each time, possibly at different optimal chromosomes.

Hybrid genetic algorithms can be viewed as two-stage systems. The first stage, executes the genetic algorithm to produce different chromosome solutions, which are used as input to the second stage.

Kado et al. [16] compare different implementations of hybrid genetic algorithms. The hybrid genetic approach has been used in other kind of problems, generally NP-hard [14, 22,28,30,35]. We have hybridized PGAM with SA and tabu search to combine the strengths of both by providing global and local exploitation aspects to the problem of assigning cells to switches in cellular mobile networks. In this section, we present the comparison results, the solutions cost and the CPU times for tabu search, PGAM and hybrid algorithms.

We have tested four hybrid genetic algorithms, two based on SA and another two based on tabu search: two tests in combination with SGA and the other two tests with PGAM. For these experiments, SGA is executed over a set of 300 test cases with 3 instances of problem in series of 50 tests for each assignment pattern. PGAM is executed over a set of 600 test cases with 3 instances of problem in series of 50 tests for each assignment pattern, with 2 and 8 parallel populations.

The results of this experiment show that the tabu search and SA improved the solution provided by SGA and PGAM with 2 parallel populations, but in the case of PGAM with parallel 8 populations, the situation is very different (Fig. 10). In this case, the solutions provided by the tabu search and SA are very similar to the solution of PGAM. In the case of SGA, tabu search and SA improved the solution and the average improvement rate for tabu search is 40.24% and for SA is 39.69%; in PGAM with 2 populations, tabu search improved the solution and the average improvement rate is 16.5%. Finally, in the case of PGAM with 8 populations, tabu search and SA improved the solution and the average improvement rate for tabu search is only 0.31% and for SA is 1.22%.

![Fig. 10. Comparison between PGAM, tabu search and hybrid genetic algorithms.](image)
4.5. Quality of the solutions

An SA-based, TS-based or PGAM solution does not necessarily correspond to a global optimum. For lack of knowing the global optimum, we will define a lower bound allowing to evaluate the quality of solution provided by this approach. An intuitive lower bound for the problem is:

\[ LB = \sum_{i=1}^{n} \min_k (c_{ik}) \]

which is the cabling cost of the solution obtained by assigning each cell \( i \) to the nearest switch \( k \). This lower bound does not take into account handoff cost. In fact, we suppose that capacity constraint is being relaxed and that all cells could be assigned to a single switch. Thus, we have a lower bound whatever the values of \( M_k \) and \( \lambda_i \).

Table 2 shows the relative distances between the three heuristics solutions and the lower bound tabu search, simulated annealing and PGAM methods give solutions ‘close’ to the lowest bound (and thereby to the global optimum). The lowest bound does not include handoff costs and therefore, no solution could equal the lowest bound.

<table>
<thead>
<tr>
<th></th>
<th>5 switches and 100 cells</th>
<th>6 switches and 150 cells</th>
<th>7 switches and 200 cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGAM</td>
<td>10.5%</td>
<td>9.87%</td>
<td>8.87%</td>
</tr>
<tr>
<td>tabu search</td>
<td>9%</td>
<td>9%</td>
<td>8%</td>
</tr>
<tr>
<td>Simulated annealing</td>
<td>10.4%</td>
<td>10.6%</td>
<td>8.9%</td>
</tr>
</tbody>
</table>

5. Conclusions

In this paper, we presented a comparative study of three heuristics, tabu search, simulated annealing and a parallel genetic algorithm with migrations (PGAM), proposed for assigning cells to switches in cellular mobile networks. Experimental results from solving different mobile networks topologies show that the three approaches provide very good results, and better than a sequential genetic algorithm. In each of the series of tests considered, the tabu search, simulated annealing and PGAM yield an improvement in CPU times and in costs in comparison with SGA. In terms of CPU times to find a feasible solution, tabu search is the fastest method, and PGAM is faster than SA and SGA. The results generated in terms of cost by tabu search, simulated annealing and PGAM are very similar. They are always better than those generated by SGA. The three heuristics always find feasible solutions. An advantage of PGAM, in comparison with tabu search and simulated annealing, is the number of different feasible solutions provided for each instance of problem. PGAM always provides a different solution and tabu search always provides the same feasible solution. In terms of CPU times, tabu search is the fastest method. Finally, the results have been compared with a lower bound on the global optimum; they confirm the effectiveness of the tabu search, simulated annealing and PGAM to solve this problem. In summary, the numerical results have shown that tabu search, simulated annealing and PGAM can solve this NP-hard problem and they provide good solutions for medium- and large-sized cellular mobile networks (from 5 switches and 100 cells).

References