Sequential and multi-population memetic algorithms for assigning cells to switches in mobile networks

Alejandro Quintero *, Samuel Pierre

Mobile Computing and Networking Research Laboratory (LARIM), Department of Computer Engineering, École Polytechnique de Montréal, C.P. 6079, succ. Centre-Ville, Montréal, Qué., Canada H3C 3A7

Received 17 July 2002; received in revised form 21 February 2003; accepted 8 March 2003

Responsible Editor: I.F. Akyildiz

Abstract

The design of mobile networks is a complex process, which requires solving simultaneously many difficult combinatorial optimization problems. Assigning cells to switches in cellular mobile networks is an NP-hard problem, which can not be practically solved using exact methods. In this context, heuristic approaches like memetic algorithms (MAs) are recommended. This paper proposes different MAs to solve this problem. The implementation of these algorithms has been subject to extensive tests. The results obtained confirm the efficiency and the effectiveness of MA to provide good solutions for moderate- and large-sized cellular mobile networks, in comparison with standard genetic algorithm and with other heuristic methods well known in the literature.

Keywords: Cellular networks; Cell assignment; Memetic algorithms; Genetic algorithms; Tabu search; Simulated annealing; Migration; Multi-population algorithm

1. Introduction

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 and using 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 to 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.

Assigning cells to switches in cellular mobile networks being an NP-hard problem, enumerative search methods are practically inappropriate to solve large-sized instances of this problem [31]. Because they exhaustively examine the entire search space in order to find the optimal solution, they are only efficient for small search spaces corresponding to small-sized instances of the problem. For example, for a network with \( m \) switches and \( n \) cells, \( m^n \) solutions should be examined.

Merchant and Sengupta [31] have proposed the first heuristic to solve this problem. Their algorithm starts from an initial solution, which they attempt to improve through a series of greedy moves, while avoiding to be blocked in a local minimum. The moves used to escape a local minimum explore a very limited set of options. These moves depend on the initial solution and do not necessarily lead to a good final solution. Others heuristic approaches have been developed for this kind of problem [3,4,42,43,50].

In the real world, many mobiles operators dedicate an important proportion of their budget to the costs facilities that carry traffic from cell sites to switches. These facilities are often leased from local exchange carriers. The pressure to reduce costs adds new urgency to the search for optimized networks designs which can minimize the number and cost of required facilities. Typically, the design of mobiles networks requires [43]:

(a) the analysis of radio-wave propagation and the field topology to identify a set of possible base stations locations;
(b) the selection of a least-cost subset of locations as hubs where the traffic is to be aggregated and switched;
(c) the assignment of each cell to a switch while taking into account a certain number of constraints including capacity constraints, routing diversity to assure reliability, handoff frequencies, etc.;
(d) the selection of the type of links between the nodes and the switches.

In the most general terms, evolution can be described as a two-step iterative process, consisting of random variation followed by selection [12,14]. In the real world, an evolutionary approach to solving engineering problems offers considerable advantages. One such advantage is adaptability to changing situations [12,14]. Evolutionary algorithms have been applied successfully in various domains of search, optimization, and artificial intelligence [5,9,22,25,34,41,48,51,54,56].
Genetic algorithms (GAs) are robust search techniques based on Darwin’s concepts of natural selection and genetic mechanisms [21,26,38]. They consist of creating a population of candidate solutions and applying probabilistic rules to simulate the evolution of the population [21]. They are used to solve extremely complex search and optimization problems which are difficult to handle using analytic or simple enumeration methods, by combining the space exploration of solutions with an adequate selection of the best results.

Moscato and Norman [39] have introduced the term memetic algorithm (MA) to denote evolutionary algorithms in which local search plays a significant role. MAs are inspired by Dawkins’ notion of a meme defined as a unit of information that reproduces itself while people exchange ideas. In contrast to genes, memes are typically adapted by the people who transmit them before they are passed on to the next generation. Radcliffe and Surry [44] formalise Moscato’s MAs and provide a unified framework for considering both memetic and genetic algorithms. MAs have been applied with success to several other combinatorial optimization problems [34–36].

This paper proposes two multi-population memetic algorithms with migrations (MPM) and two sequential memetic algorithms (SMA) to efficiently solve the problem of assigning cells to switches in cellular mobile networks. Section 2 presents background and related work. Section 3 first describes the genetic and memetic algorithms, then presents a multi-population approach. Section 4 presents some adaptation and implementations details of memetic and local search strategies. Finally, Section 5 presents an analysis of results and compares them to other methods well studied in the literature.

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 unit time for a complex handoff between cells \( i \) and \( j \) \((i,j = 1,\ldots,n \text{ with } i \neq j)\) 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 \( k \) \((i = 1,\ldots,n; k = 1,\ldots,m)\).

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 by the follows:

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

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 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 \( x_{ik} \).

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

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

$$x_{ik} = 0 \text{ or } 1 \quad \text{for } i = 1, \ldots, n \text{ and } k = 1, \ldots, m,$$

(4)

$$z_{ijk} = x_{ij}x_{ik} \text{ and } i, j = 1, \ldots, n \text{ and } k = 1, \ldots, m,$$

(5)

$$y_{ij} = \sum_{k=1}^{m} z_{ijk} \text{ for } i, j = 1, \ldots, n.$$

(6)

$z_{ijk}$ is equal to 1 if cells $i$ and $j$, with $i \neq j$, 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.

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 $\lambda_i$. The switches are then considered as warehouses of capacity $M_k$ where the cells production could be stored. Therefore, the problem is to minimize (2) under (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 [31,32] replaced it by the following equivalent set of constraints:

$$z_{ijk} \leq x_{ik},$$

(7)

$$z_{ijk} \leq x_{jk},$$

(8)

$$z_{ijk} \geq x_{ik} + x_{jk} - 1,$$

(9)

$$z_{ijk} \geq 0.$$  

(10)

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

$$h_{ij} = H'_{ij} - H_{ij},$$

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

$$f = \sum_{i=1}^{n} \sum_{k=1}^{m} c_{ik}x_{ik} + \sum_{i=1}^{n} \sum_{j=1,j \neq i}^{n} h_{ij}(1 - y_{ij})$$

$$+ \sum_{i=1}^{n} \sum_{j=1,j \neq i}^{n} H_{ij}.$$

The assignment problem takes then the following form:

Minimize

$$f = \sum_{i=1}^{n} \sum_{k=1}^{m} c_{ik}x_{ik} + \sum_{i=1}^{n} \sum_{j=1,j \neq i}^{n} h_{ij}(1 - y_{ij})$$

subject to (1), (3), (4) and (7)–(10). In this form, the assignment problem could be solved by usual programming methods. The total cost includes two types of cost, namely cost of handoff between two adjacent cells, and cost of link 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 [31,43].

Merchant and Sengupta [31,32] studied this assignment problem. Their algorithm starts from an initial solution, which they attempt to improve through a series of greedy moves, while avoiding to be stranded in a local minimum. The moves used to escape a local minimum explore only a very limited set of options. These moves depend on the initial solution and do not necessarily lead to a good final solution.

The geographical relationships between cells and switches are considered in the value of the cost of linking, so that the base station of a cell is generally assigned to a neighboring switch and not to far switches [57]. In [47], 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 tele-
phone, cable television and cellular networks. In [17], economic aspects of configuring cellular networks are presented. Major components of costs and revenues as well as the major stakeholders were identified and a model was developed to determine the system configuration (e.g., cell size, number of channels, link cost, etc.). For example, in a large cellular network, it is impossible for a cell located in east America to be assigned to a switch located in west America. In this case, the variable link cost is $\infty$. In [18], different methods have been proposed to estimate the handoff rate in PCS and the economic impacts of mobility on system configuration decisions (e.g., annual maintenance and operations, channel cost, etc.).

3. Memetic approach

In the field of combinatorial optimization, it has been shown that combining evolutionary algorithms with problem-specific heuristics can lead to highly effective approaches [1,2,13,49].

GAs are robust search techniques based on natural selection and genetic production mechanisms. GAs perform a search by evolving a population of candidate solutions through nondeterministic 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 [26,53].

GAs [21] 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 [15].

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

Mutation is the process by which a randomly chosen bit in a chromosome is flipped. It is employed to introduce new information into the population and also to prevent the population from becoming saturated with similar chromosomes (premature convergence). Large mutation rates increase the probability that good schemata be destroyed, but increase population diversity. A schema is a subset of chromosomes which are identical in certain fixed positions [21,26].

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 contribute more to the reproductive pool; typically this is also done probabilistically. For a selection, two essential ingredients are required: (1) inheritance, offspring must retain at least some of the features that made their parents fitter than average; (2) variability, at any given time, individuals of varying fitness must coexist in the population [21].

3.1. Basic principles of memetic algorithms

In problems characterized by many local optima, standard genetic algorithm (SGA) 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 [7,16,33,41,45,54]. This may prevent them from being really of practical interest for a lot of large-scale constrained applications.

MA are population-based heuristic search approaches for combinatorial optimization problems based on cultural evolution [39,48]. The memetic
approach takes the concept of evolution as employed in GAs and combines this with an element of local search. It can be seen that a GA where one variation operator is a local search operator, either providing the local minimum closest to the starting point or a point on the path leading to this closest local optimum. This local search in the neighbourhood is applied to each newly created offspring before its insertion into the new population.

In the context of evolutionary computation, a hybrid evolutionary algorithm is called memetic if the individuals representing solutions to a given problem are improved by a local search or another improvement technique [36]. Kado et al. [29] compare different implementations of hybrid GAs.

In this paper, we propose MAs with local refinement strategies 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. The local refinement strategies used with our MAs are tabu search and simulated annealing.

A tabu search method is an adaptive technique used in combinatorial optimization to solve difficult problems [19,42,43]. 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 gives to Tabu its meta-heuristic nature. The tabu list is not always a list of solutions, but can be a list of forbidden moves/perturbations [20,46].

Tabu search is a hill-climber endowed with a tabu list (list of solutions or moves). The hill-climber further explores the neighborhood of their current point as to get away from the repousoir [52]. 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 [20].

Simulated annealing (SA) was introduced by Metropolis et al. [37] and is used to approximate the solution of very large combinatorial optimization problems [27,30]. 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 [30].

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”) [10].

3.2. Multi-population approach

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

One approach is the partitioning of the population into several subpopulations (multi-population approach) [55]. 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 [40]. Sometimes a topology is introduced on the population, so that individuals can only interact with nearby chromosomes in their neighborhood [23,24].

The parallel implementation of the migration model shows not only a speedup in computation time, but it also needs less objective function evaluations when compared to a single population algorithm. So, even for a single processor computer, implementing the parallel algorithm in a serial manner (pseudo-parallel) delivers better results (the algorithm finds the global optimum more often or with less function evaluations) [2]. Cohoon et al. [8] present results in which parallel algorithms with migration found better solutions than a sequential GA for optimization problems, and Lienig [28] indicates that parallel GAs in isolated evolving subpopulations with migrations may offer advantages over sequential approaches.

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. For further details on the parameters associated with the migration algorithm, see [6,55].

4. Implementation details

We tested the MAs with local refinement strategies to a series of tests in order to determine the efficiency and sensitivity to different parameters. Thus, we will present a results, that we compare to those provided by other known heuristics.

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 $c_i$ 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 [11].

4.1. Local search strategies

This section presents the implementations details of the local refinement strategies used to improve the individuals representing solutions provided by GAs: tabu search and simulated annealing.

To solve the assignment problem with tabu search, we have chosen a search domain free from capacity constraints on the switches, but respecting the constraints of unique assignment of cells to switches. The feasibility of the final solution is therefore not guaranteed, but as we explore a large number of possibilities, we increase the chances of obtaining good solutions. We associate with each solution two values: the first one is the intrinsic cost of the solution, which is calculated from the objective function; the second is the evaluation of the solution, which takes into account the cost and the penalty for not respecting the capacity constraints. At each step, the solution that has the best evaluation is chosen. Once an initial solution built from the problem data, the short term memory component attempts to improve it, while avoiding cycles. The middle-term memory component seeks to intensify the search in specified neighbourhoods, while the long-term memory aims at diversifying the exploration area.

The short-term memory moves iteratively from one solution to another, by applying moves, while prohibiting a return to the $k$ latest visited solutions. It starts with an initial solution, obtained simply by assigning each cell to the closest switch, according to an Euclidean distance metric. The objective of this memory component is to improve the current solution, either by diminishing its cost or by diminishing the penalties.

The middle-term memory component tries to intensify the search in promising regions. It is introduced after the end of the short-term memory component and allows a return to solutions we may have omitted. It mainly consists in defining
the regions of intensified search, and then choosing the types of move to be applied.

To diversify the search, we use a long-term memory structure in order to guide the search towards regions that have not been explored. This is often done by generating new initial solutions.

In simulated annealing, the parameters are the temperature $\theta$, the annealing factor $\alpha$ ($0 < \alpha < 1$), the current solution $S_1$, the solution after perturbations $S_2$, and the stopping criterion. The steps in simulated annealing are:

**Step 1.** Create the initial solution $S_1$ (at random, each cell being allocated to a switch in an unpredictable way, we create a solution free from capacity constraints on the switches, but respecting the constraint of unique assignment of cells to switches), $\theta$ is set to a relatively high value.

**Step 2.** Select a new solution $S_2$ that can be obtained by carrying out a small perturbation over $S_1$.

**Step 3.** If the solution $S_2$ is better than $S_1$, then this topology is saved as current one. However, it happens that, further to a disturbance, the obtained nearby topology is kept as current solution, even if it is not better than the current one, provided that it respects a certain probability of acceptance (the fact of accepting a loss of quality or fitness allows to avoid being trapped too early into a local optimum).

**Step 4.** $\theta$ is reduced by a multiplication by an annealing factor $\alpha(\theta_{k+1} = \theta_k \times \alpha)$, where $\alpha = 0.95$.

**Step 5.** If the stopping criterion will not have been reached, go to Step 2.

### 4.2. Memetic algorithm implementation

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 [22]. 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. A particular value of the string is called a *gene* and the possible values are called *alleles*, taken from the alphabet $V = \{1, 2, \ldots, m\}$.

The first element of the initial population is the one obtained when all cells are assigned to the nearest switch. This first chromosome is created therefore in a deterministic way. The creation of other chromosomes of the population is probabilistic and follows the strategy of population without doubles, that means, we test equality between individuals and remove doubles. This strategy permits to ensure the diversity of the population and a good cover of the search space. All chromosomes of the population verify the unique assignment constraint, but not necessarily the one of the switches’ capacity.

The choice of the candidates is based on the evaluation function given by (11). In our adaptation, every chromosome is evaluated according to the criterion of cost in a first time. The sort by ascending order of the objective value of those chromosomes permits to have the best potential chromosomes as the first elements of population. The second stage of evaluation consists of verifying the chromosomes in relation to the capacity constraint on the switches and to determine the best chromosome that verifies this constraint.

To select the elements of the new generation, we select the best chromosome of the present population and for the others we used the *roulette wheel* method. Because the problem we have to solve is a minimization problem, we applied the caster in order to invert the objective values of chromosomes. Then, we recover both in the new selected population, chromosomes that verify the switches capacity’s constraint and those that violate it. The number of generations is fixed at the beginning of the execution.

For the migration algorithm used in this paper, subpopulations are arranged in fully meshed topology. Here, individuals may migrate from any
subpopulation to another. For each subpopulation, a pool of potential emigrants is constructed from the other subpopulations. Fig. 2 shows the MPM proposed.

Migration intervals are typically specified as a fixed number of generations, known as an epoch. The problem with using a fixed epoch value is that migration is globally synchronized across all subpopulations. Using a random interval allows the subpopulations to evolve asynchronously [40].

4.3. Some computational experiments

In the first step, we generate an initial population of size 100 chromosomes. In the second step, we estimate each chromosome by the objective function, what allows to deduct its value of capacity. Finally, in the last step, the cycle of generations of the populations begins then, each new population replacing the previous one.

To determine the number of subpopulations in parallel, MPM was executed over a set of 600 test cases with three instances of problem in series of 20 tests for each assignment pattern, with a number of populations varying between 1 and 10. This experience shows that MPM converges to good solutions with a number of populations varying between 7 and 10.

To define the population size, MPM was executed over a set of 150 test cases with three mobile networks in series of 20 tests for each assignment pattern with eight populations. This experience shows that MPM converges to provide good solutions with a population size varying between 80 and 140.

The values used by MPM are: the stopping criteria used in our experiences is ‘if the solution cannot be improved in the last \( n \) iterations (\( n \) is 10) then the algorithm is terminated’. This criteria is applied only after minimum 400 generations; the population size is 100; the number of populations is eight for MPM; the crossover probability is 0.9; the mutation probability is 0.08; the migration interval \( (\text{P}_m) \) is 0.1; the migration rate \( (\text{S}_m) \) is 0.4 and the emigrants accepted \( (\text{P}_r) \) is 0.2. A larger migration interval is normally used in connection with larger migration rate.

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 multi-population memetic algorithm without migration (MPW) migration is turned off (\( \text{pm} = 0.0 \)). Turning off migration for analysis of the multi-population memetic

<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>45</td>
</tr>
<tr>
<td>Crossover, mutation, local-search</td>
<td>20</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>Communication time</td>
<td>7</td>
</tr>
</tbody>
</table>
algorithm ensures that evaluation costs measurements indicate the effects of parallel algorithm, rather than the effects of migration. In order to compare the performance of MPM and MPW, a set of experiments were performed to evaluate the quality of the solutions in terms of their costs.

In these experiments, the results obtained by MPW are compared directly with MPM. The two algorithms always provide the feasible solutions. However, MPM yields an improvement in evaluation cost in comparison with MPW, because MPM converges faster than MPW to good solutions with a small number of generations.

5. Performance evaluation and numerical results

In order to compare the performance of MA and genetic algorithm (SGA) with that of the other 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 to evaluate the performance of MA and SGA in terms of CPU times. All the test runs described in this section were performed in a networked workstation environment operating at 100 Mbps with 10 PCs (Pentium 500 MHz).

Fig. 3. Comparison between MPM and SGA: (a) CPU times average generated by 50 tests of each algorithm and (b) evaluation cost average.
5.1. Comparison with standard genetic algorithm approach

In this section, we present the comparative results, solution costs and CPU times, between MA, and SGA. We have tested four MAs, two memetic sequential algorithms (SGA in combination with simulated annealing and tabu search) and the other two MPM (multi-population algorithm in combination with simulated annealing and tabu search). For the experiments, MPM, SMA and SGA are executed over a set of 900 test cases with three topologies in series of 50 tests for each assignment pattern. MPMs was executed with eight parallel populations. The population size in the case of MA is 100, and in the case of SGA is 800. The other values (e.g. crossover probability, mutation probability, etc.) are the same.

The results of this experiment show that the simulated annealing and tabu search improved the individuals representing solutions provided by sequential genetic algorithm and multi-population algorithm. In the case of simulated annealing, the average improvement rate for sequential algorithm is 32.27% and the average improvement rate for multi-population algorithm is 20.22%. In the case of tabu search, the average improvement rate for sequential algorithm is 34.05% and the average improvement rate for multi-population algorithm is 22.04%.

A comparison between MPM and SGA is undertaken in order to measure the evolution cost and the execution time. In the first set of experiments, the results obtained by SGA are compared directly with those provided by MPM. MPM and SGA always find the feasible solutions. In each of the three considered series of tests, MPM yields an improvement in terms of both CPU times and evaluation cost, in comparison with SGA. In terms of CPU times, SMA2 and MPM2 are between 10 and 20 faster than MPM1 and SMA1 to find a feasible solution (Fig. 3a). SGA is the slowest method. In terms of costs, MPMs provide always feasible solutions with a very similar cost, with better results than SMA and SGA. Fig. 3b shows the comparison between MAs and SGA in which each simulation represents the average over 50 tests of each algorithm. In conclusion, experimental results from solving different instances of assignment problem show that MA approach provides better results than an SGA.

5.2. Comparison with other heuristics

Merchant and Sengupta [31] have designed a heuristic, which we call H, for solving the cell assignment problem. Pierre and Houéto [43] have been used tabu search (TS) for solving the same problem.

We compare TS and heuristics H with MA. For the experiments, these three heuristics were executed over a set of 960 test cases with a number of cells varying between 100 and 200, and a number of switches varying between 3 and 7, that means the search space size is between $3^{100}$ and $7^{200}$. Varying the switches’ geographical location by maintaining the number of cells and the switches permanent (or fixed), we obtain a different configuration. For each one of the three topologies we have analyzed 16 different configurations.

The three heuristics always find feasible solutions. However, these results inform only on the feasibility of obtained results without demonstrating whether these solutions are among the best. Fig. 4 shows the results obtained for the three different instances of problem used in the tests whose the evaluation costs represent the average over 25 tests of each algorithm.

The three heuristics always find feasible solutions with objective values close to the optimum solution. In each of the all considered series of tests, MA yields an improvement in the cost function in comparison with the other two heuristics. In terms of evaluation fitness, MA provides better results than tabu search and heuristics H. Table 2 summarizes the results. Nevertheless, given the initial link, the handoff and the annual maintenance costs for large-sized cellular mobile networks (in the order of hundred of millions of dollars) this small improvement represents a large reduction in costs over a 10-years period in the order of millions of dollars. For example, in a cellular network composed by 300 cells, with initial link and handoff cost of $350,000 for each cell, an improvement of 2% in the cost function represents an approximate saving of $2M over 10 years.
In terms of CPU times, for a large number of cells, TS is a bit faster than heuristic H. Conversely, for problems of smaller size, TS is a bit slower. MA is slower than heuristics H and TS. However, this is not an important fact because this heuristic is used in designing and planning phase of cellular mobile networks.
6. Conclusions

In this paper, we proposed a memetic genetic algorithm (MA) approach to solve the problem of assigning cells to switches in cellular mobile networks. We have developed two MPM and two SMAs. The local refinement strategies used with our MAs are tabu search and simulated annealing.

Experiments have been conducted to measure the quality of solutions provided by these algorithms. The results of this experiment show that the sequential and multi-population memetic algorithms improved the individuals representing solutions provided by SGA.

These results confirm the efficiency and the effectiveness of MA to provide good solutions for large-sized cellular mobile networks. In general, they are better than those generated by tabu search and Sengupta’s heuristics. The improvement obtained represents significant savings in maintenance and operations costs over a 10-years period.

Table 2

<table>
<thead>
<tr>
<th>MA average improvement rates</th>
<th>Three switches, 100 cells (%)</th>
<th>Five switches, 150 cells (%)</th>
<th>Seven switches, 200 cells (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tabu search</td>
<td>1.15</td>
<td>1.62</td>
<td>2.01</td>
</tr>
<tr>
<td>Heuristics H</td>
<td>1.52</td>
<td>1.78</td>
<td>3.87</td>
</tr>
</tbody>
</table>

References


Alejandro Quintero received the engineer degree in computer engineering from Los Andes, Colombia, in 1983. In June 1989, and in 1993, he received the diploma of advanced studies, and the Ph.D. degree in computer engineering, respectively, from the Joseph Fourier University, Grenoble, France. He is currently an assistant professor at the Department of Computer Engineering of Ecole Polytechnique de Montreal, Montreal, Canada. His main research interests include mobile computing and third generation wireless infrastructures. He is the co-author of one book, as well as over 20 other technical publications including journal and proceedings papers.

Samuel Pierre received the B.Eng. degree in civil engineering in 1981 from Ecole Polytechnique de Montreal, the B.Sc. and M.Sc. degrees both in mathematics and computer science in 1984 and 1985, respectively, from the UQAM, Montréal, the M.Sc. degree in economics in 1987 from the Université de Montréal, and the Ph.D. degree in Electrical Engineering in 1991 from Ecole Polytechnique de Montréal. From 1987 to 1998, he was a Professor at the Université du Québec à Trois-Rivières prior to joining the Télé-Université of Québec, an Adjunct Professor at Université Laval, Ste-Foy, Québec, an Invited Professor at the Swiss Federal Institute of Technology, Lausanne, Switzerland, then the Université Paris 7, France. He is currently a Professor of Computer Engineering at Ecole Polytechnique de Montréal where he is Director of the Mobile Computing and Networking Research Laboratory (LARIM) and Industrial Research Chair NSERC/ Ericsson in Next-Generation Mobile Networking Systems. He is the author of four books, co-author of two books and five book chapters, as well as over 200 other technical publications including journal and proceedings papers. He received the best paper award of the Ninth International Workshop on Expert Systems and their Applications, held in France in 1989. In 1994, one of these co-authored books, Telecommunications et Transmission de données (Eyrolles, 1992), received special mention from Telecoms Magazine (France). He is a Fellow of the Engineering Institute of Canada. His research interests include wireline and wireless networks, mobile computing, performance evaluation, artificial intelligence, and electronic learning (e-learning). He is a senior member of IEEE and a member of ACM. He is an Associate Editor of IEEE Communications Letters and IEEE Canadian Review. He also serves on the editorial board of Telematics and Informatics published by Elsevier.