Problem of Assignment Cells to Switches in a Cellular Mobile Network via Beam Search Method

Cassilda Maria Ribeiro
Faculdade de Engenharia de Guaratinguetá - DMA
UNESP - São Paulo State University
Av. Ariberto Pereira da Cunha 333. Pedregulho – Guaratinguetá – SP – CEP: 12516-410
BRAZIL
cassilda@feg.unesp.br http://www.feg.unesp.br/~cassilda

Aníbal Tavares Azevedo
Faculdade de Engenharia de Guaratinguetá - DMA
UNESP - São Paulo State University
Av. Ariberto Pereira da Cunha 333. Pedregulho - Guaratinguetá - SP - CEP: 12516-410
BRAZIL
anibal@feg.unesp.br http://www.feg.unesp.br/~anibal

Rodolfo Florence Teixeira Jr.
Faculdade de Engenharia de Bauru - DEP
UNESP - São Paulo State University
Av. Eng. Luiz Edmundo C. Coube 14-01 - Bauru – SP - CEP: 17033-360
BRAZIL
rodolfo@feb.unesp.br

Abstract: - The problem of assigning cells to switches in a cellular mobile network is an NP-hard optimization problem. So, real size mobile networks could not be solved by using exact methods. The alternative is the use of the heuristic methods, because they allow us to find a good quality solution in a quite satisfactory computational time. This paper proposes a Beam Search method to solve the problem of assignment cell in cellular mobile networks. Some modifications in this algorithm are also presented, which allows its parallel application. Computational results obtained from several tests confirm the effectiveness of this approach to provide good solutions for medium- and large-sized cellular mobile network

Key-Words: - Combinatorial Optimization. Assignment problem. Beam Search Method. Cellular Network.

1. Introduction

The prodigious growing of telecommunications in recent times makes it more present in modern society. The popularization of one of the most fantastic equipments of all times, the mobile phone is a key factor for the competition between the operators, leading to a search for a more efficient cellular mobile network, and with capacity to attend the increasingly demand with more and more quality. In a cellular mobile network, services are offered by regions. These regions are called coverage areas and are divided into small geographical areas called cells. Each cell is responsible for covering (provide) a number of subscribers. To do so, on each cell there is an antenna (base station) that communicates between subscribers of the same cell. In addition, each cell is connected to special units called switches, which are located in mobile switching center (MSC), which is responsible for making connections between subscribers to two different cells. The cells have, due to computational reasons, the hexagonal format, which is similar to a honeycomb structure. Each mobile switching center (MSC) is able to switch a number of subscribers. So, multiple cells can be connected to a single MSC by respecting the maximum capacity of the same. However, one cell cannot be connected to more than one MSC at the same time. Figure 1 shows an example where the cells A and B are connected to the MSC 1 and the cells C and D are connected to the MSC 2. The base station installed in each cell use radio channels to make the assignments between the cells and to avoid interference; two adjacent cells use a group of different radio channels.

When the communication user moves from one cell to another, the base station of the new cell has the responsibility to maintain the communication of this user; therefore it has to allocate a new radio channel.

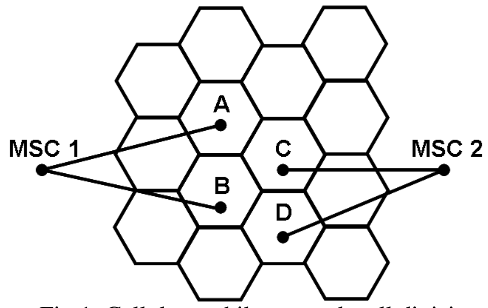

Fig.1: Cellular mobile network cell division.

The communication transfer from a cell to another is called handoff. The mechanism of user transfer between two base stations (cells) occurs when the signal level received by the user is too low. There are two types of handoff. In the example of figure 1, when the user moves from cell A to B, the handoff is called *soft handoff* because the two cells are connected to the same mobile switching center, and the induced cost for the transfer from A to B is negligible. On the other hand, when the user moves from the cell A to the cell C, the *handoff* is called *complex*. The induced cost for this transfer must be considered, because the two switching center (1 and 2) should be actives during the handoff process and the database containing information about the assignment should be updated.

In this paper is presented a Beam Search algorithm to solve the Problem of Assigning Cell to Switches in Cellular Mobile Network. Beam Search is a heuristic method for solving optimization problems. It is an adaptation of the branch and bound method in which only some nodes are evaluated in the search tree. In sections 2 and 3 we present the problem and its mathematical formulation. In the section 4 we present the algorithm developed. Sections 5 e 6 shows respectively the results obtained and the conclusions.

2. The Assigning Cell Problem

One of the main tasks about planning cellular mobile networks is optimally assigning cells to switches, i.e., find an assignment of cells to switches centers that respect some constraints and minimize the total cost of operation. To minimize the total cost of the operation, two factors must be considered. The first one is the cost of cabling, i.e. the link cost. This cost depends on the distance between the cell and it switching center. The second factor is the *handoff* cost between cells.

In this paper, we understand as assignment of cell to mobile switching center, the determination of which switching center will be responsible for the assignment communication of a cell with the other cells of the mobile network. As this communication uses the antenna of each cell, many times it is known as assignment of antenna to the mobile switching center instead of cell assignment to the switching centers.

The problem of assignment **n** cells to **m** switches in a cellular mobile network is a NP-hard problem [1] however there is no exact method able to solve this problem in polynomial time. Only the exhaustive search would guarantee the optimal solution for this problem. But, to make this, it is necessary examining all the possible solutions, which is really not viable, because the number of possible solution is enormous. For example, for a network with 150 cells and 4 mobile switching centers, considering a medium size network, would be necessary to examine (4^{150}) of possible solutions.

Using a computer able to examined one solution each 1ns, would be necessary $6{,}46\times10^{73}$ years. Due to this difficulty, the methods used to solve this problem are the heuristics. In the literature there are many types of heuristics and the ones called meta-heuristics is the more common, see for example: [1][2][3][4]. Other authors developed Implicit Enumeration heuristics: Merchant and Sengupta [5], Saha et al [6]. Menon and Gupta [7] present a mix heuristic that use Linear Programming and Simulated Annealing; and Salomão [8] solved this problem using Linear Programming techniques with Columns Generation.

3. Mathematical Formulation

Before presenting the formulation problem, we need to clarify that in this work it was considered that the operator would offer the costs of cabling and handoff. A more thorough description about the handoff calculation can be seen in Alonso [9]. The formulation of the problem as a problem of quadratic integer programming, as given below: Being n the number of cells to be assigned to m switches. It is assumed that the cells and the switching centers localization are set and known. Being H_{ij} and H'_{ij} according costs per unit of time for the *soft handoff* and for *complex handoffs* that occur between the cells i e j $(i,j=1,\cdots,n)$. As it was given above, it is assumed that the handoff costs are known and proportional to the frequency of handoffs that occur between cells i and j. Being C_{ik} the cabling costs per unit of time between the cell i and the switch k $(i=1,\cdots,n\,;k=1,\cdots,m)$. Being λ_i the number of calls per unit of time destined to cell i, and M_k the capacity of a switch *k*. This problem consists of assigning cells to switches in a way that minimize the cost function. The cost function integrates the handoff cost per time and the cabling cost between cells and switches. The optimization of this problem must not violate the maximum capacity of each switch and must be taken into account that each cell could be assigned to a unique switch.

The mathematical formulation of the described problem above was made using integers variables, and non-linear constraints. Therefore, the following variables were defined.

$$x_{ik} = \begin{cases} 1, \text{if the cell } i \text{ is assigned to switch } k \\ 0, \text{otherwise} \end{cases}$$

Considering that each cell can only be assigned to a unique center, there is the following constraint:

$$\sum_{k=1}^{m} x_{ik} = 1 \quad \text{for} \quad i = 1, \cdots, n \qquad (1)$$

The switches capacity constraints are given by:

$$\sum_{i=1}^{n} \lambda_i x_{ik} \le M_k \quad \text{for } k = 1, \cdots, m \qquad (2)$$

The total cabling cost is given by: $\sum_{i=1}^{n} \sum_{k=1}^{m} c_{ik} x_{ik}$.

The following additional variables were created to represent the costs of handoff:

$$z_{ijk} = x_{ik} x_{jk} \text{ for } i, j = 1, \cdots, n \text{ e } k = 1, \cdots, m. \qquad (3)$$

Observe in the constraint (3) above that $z_{ijk} = 1$ when the cells *i* and *j* are connected in the same switch k and $z_{ijk} = 0$, otherwise. Considering all the switches, we have $\sum_{k=1}^{m} z_{ijk} = \sum_{k=1}^{m} x_{ik} x_{jk} = 1$ if the cells *i* and *j* are connected to the same switch, and $\sum_{k=1}^{m} z_{ijk} = \sum_{k=1}^{m} x_{ik} x_{jk} = 0$ if they are connected to different switches.

The total network cost is given by the sum of the cabling costs plus the handoff costs, and can be written as:

$$\sum_{i=1}^{n}\sum_{k=1}^{m} C_{ik} x_{ik} + \sum_{i=1}^{n}\sum_{j=1}^{n} H_{ij} \sum_{k=1}^{m} x_{ik} x_{jk} + \sum_{i=1}^{n}\sum_{j=1}^{n} H'_{ij} (1 - \sum_{k=1}^{m} x_{ik} x_{jk}) \quad (4)$$

According to Hedible [10], the *soft* handoff costs H_{ij} can be considered irrelevant when compared to the *complex* handoff costs H'_{ij}. So, we have $h_{ij} = H'_{ij} - H_{ij}$, replacing in (4) and not taking the constant part into account, the objective function to be minimized is:

$$\sum_{i=1}^{n}\sum_{k=1}^{m} C_{ik} x_{ik} + \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij} (1 - \sum_{k=1}^{m} x_{ik} x_{jk}) \qquad (5)$$

Therefore the problem to be solved is the following:

$$\text{Min} \sum_{i=1}^{n}\sum_{k=1}^{m} C_{ik} x_{ik} + \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} h_{ij} x_{ik} x_{jk}$$

$$\text{s.a} \begin{cases} \sum_{k=1}^{m} x_{ik} = 1 \text{ for } i = 1, \cdots n \\ \sum_{i=1}^{n} \lambda_i x_{ik} \le M_k \text{ for k} = 1, \cdots, m \\ x_{ik} = 0 \text{ or } 1 \text{ for } i = 1, \cdots, n \text{ and } k = 1, \cdots, m \end{cases} \qquad (6)$$

The problem (6) could be represented by a bipartite graph as showed in Figure 2.

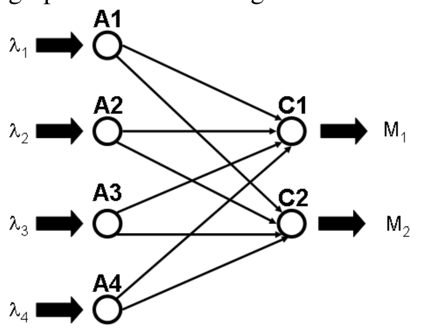

Fig.2: Bipartite graph formulation.

This representation is only in order to facilitate the visualization of the problems, because the problem presented here is not the classic assignment problem since in this, the second constraints turns to into an equality constraints as the following equation:

$$\sum_{i=1}^{n} \lambda_i x_{ik} = M_k \quad \text{for } k = 1, \cdots, m, \quad \text{where } \lambda_i = 1 \text{ and } M_k = 1$$

It must be observed that the problem of assigning cells to switches, as stated in (6), is a nonlinear one (due to the complex handoff costs) with integer variables (the decision do not allow attribution of a cell for more than one switch center).

Many authors considered a transformed version of this problem by replacing the constraints (3) for a set of linear constraints which leads to a linear model with integers variables, see: Menon [7], Merchand and Sengupta [5], Salomão [8]. The algorithm proposed here does not utilize this transformation since it proposes a partial enumeration heuristic that could handle nonlinear costs and also integer variables.

4. The Beam Search Algorithm

The Beam Search is an implicit enumeration method for solving combinatorial optimization problems. It could be said that it is an adaptation of the Branch and Bound method where only a predetermined number of best partial solutions (nodes) are evaluated in each level of the search tree while the others are discarded permanently. As a big part of the tree search nodes is discarded it means, only a few nodes are kept for further branching and the others are pruned off permanently, the method execution time is polynomial in the size of the problem. Resuming it can be said that the Beam Search is a tree search technique that in each level of the tree is analyzed a fixed number of nodes and thus a fixed number of solutions. The nodes number analyzed in each level is called beam width and is denoted by β.

The Beam Search was first used by the community of Artificial Intelligence to treat problems of speech recognition [11]. The literature gives a lot of applications of this method in scheduling problems see: [12] [13] [14] [15][16].

4.1 The Tree with all the Possible Solutions

Before presenting the proposed algorithm, the search tree structure (assignment) is described for a small example, and all possible solutions are listed properly. Let the problem of figure 2 where n = 4 is the number of cells that should be assigned to m = 2 switches. Each cell A_i is able to receive a fixed number of calls λ_i per unit of time. Thus, the number of calls covered by cell A_1 is λ_1 and so on. The switch C_k routs these calls if cell A_i is connected to the switch C_k. Each switch has a capacity M_k. Each cell can only be connected to one switch. Being $x_{ik} = 1$, if cell ***i*** is connected to switch k and $x_{ik} = 0$, otherwise. The tables 1, 2, 3 and 4 showed below presents the information about the cabling cost, handoff cost, capacity of each cell and switch, for the example presented.

Cell	Switch 1	Switch 2
1	0	0
2	4	4
3	4	4
4	8	8

Table 1: Cabling cost C_{ik} between cells and switches

Cell/cell	1	2	3	4
1	0	3	2	2
2	3	0	1	51
3	2	1	0	4
4	2	5	4	0

Table 2: handoff cost $\boldsymbol{h_{ij}}$ between cells.

Cell	Number of call
1	4
2	4
3	4
4	4

Table 3: Number of call $\boldsymbol{\lambda_i}$ for each cell ***i***.

Switch	Capacity
1	10
2	10

Table 4:switches capacity M_k.

Before building the search tree is necessary to establish the following definitions:

(D.1) The tree is constructed by level and in each level ***i*** is made the assignment of the ***i-esime*** cell to each one of switches.

(D.2) The nodes that are in the level 1 of the tree are called seed nodes because each one of them will generate a sub-tree of decisions.

(D.3) In each level ***i,*** when making the assignment of ***i*-esime cell** to a **switch *k***, two costs are considered: the cabling cost and the handoff cost considering all the (***i*-1**) attribution mentioned before.

Considering that the tree has **n** levels **(D.1)**, where **n** is the number of cells, a complete problem solution, with attribution of all **n** cells to all the ***m*** switches is only be obtained when defining the assignments until the level **n**.

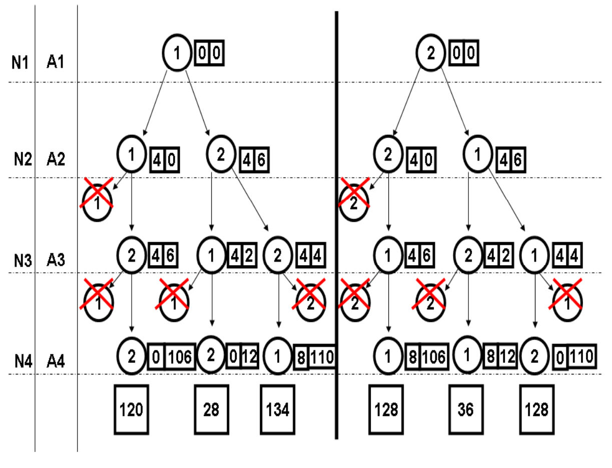

Fig.3: Complete Search Tree considering the seed node C1 in in level N1 (left sub-tree) and C2 in level N1 (right sub-tree).

The Figure 3 shows how the tree with all the possible feasible solutions for this problem is. The number of the switch is represented in the nodes, the cell number is represented by the letter A, so, A1 represents the cell 1, and the levels are represented by N, for example N1 represents the level 1, N2 the level 2 and so on. The nodes of each level of the tree represent an assignment. It begins with the tree fixing to a switch, for example, to the switch C_1 and attributing to this a cell, for example, A_1. Then, the node 1, of level N1, means that the cell A_1 was assigned to the switch C_1, i.e. $x_{11} = 1$. This first node is called the seed node. Once it was done the assignment on the node seed, follows making the other attributions, always considering the assignments that were already made in the first nodes. Note that the number of seed nodes is the same as the switches ***m***. Moreover each seed node will originate one sub-tree. The complete tree will have then ***m*** sub-trees, which is one to each switch.

In Figure 3, the nodes marked with a cross give an unfeasible assignment and for this reason they are automatically eliminated of the process. It is important give special attention for the total number of solution which is $\mathbf{m^n = 2^4 = 16}$, but among them only 6 are feasible, justifying the application of a search tree procedure.

As in each level of the tree is made the allocation of a cell for all the switches, thus in the level N1 is made, for example, the allocation of the cell A_1 to each one of the switches (seed nodes). In the level N2 is made the allocation of the cell A_2 for each one of the switches; in the level N3 is made the allocation of to the cell A3 and in the level N4 is made the allocation of the cell A4. Then, in the Figure 3, the level N2 has 4 nodes came from the allocation of the cell A2 to the switch 1 and 2, in the left sub-tree and to the switch 2 and 1 in the second right sub-tree. In the level N3 we have 6 nodes that came from the attribution of the cell A3 respectively to the switch 2, 1 and 2 in the left sub-tree and the switch 1, 2 and 1 in the right sub-tree.

Beside each node, of Figure 3, also are presented, inside the small rectangle, respectively the cabling costs and the partial costs of handoff related to the allocation made in the node. For example, in the level N3, we have the attribution of the cell A3 to the switch 2. In this case the cabling cost between A3 and C2 is 4 and the partial cost of *handoff* is 6 because there is a *handoff* between A3A1, A1A3, A3A2 and A2A3.

In each level are calculated only the partial costs of the solution that is being obtained because as it has been said the solution will only be complete when the ultimate level of the tree nodes is generated. Thus, in the nodes of level N1 there is only the cabling cost (linear cost), in the N2 level it is already the linear costs plus the handoff costs of the seed node until the nodes of the level N2, and therefore in the ultimate level, there are linear costs plus the handoff costs of the seed nodes until the nodes of level N_n.

As we go down in the tree, the cost is being completed by the decision making. Therefore, the branches of the tree containing the nodes A_1C_1, A_2C_1, A_3C_2, and A_4C_2 indicates that the following allocation has been done: $x_{11} = 1$, $x_{21} = 1$, $x_{32} = 1$, $x_{42} = 1$.

4.2. The Tree Generated by Beam Search

To avoid the exponential growth of the tree, the proposed algorithm does not create all the nodes of the tree, as it has been shown in figure 3. The nodes are created according to some rules. Theses rules also aim to prevent the creation of unfeasible solutions. In each node of the tree are stored the following information:

(1) Switch Identification;

(2) Cell Identification;

(3) Fixed cabling costs and partial handoff costs related to the allocation made in the node;

(4) Remaining switch capacity that it is representing. The remaining switch capacity $M_k^{(i)}$ is computed by decreasing the switch initial capacity M_k, the number of calls λ_i referring to the cell demands that were already allocated to this switch.

(5) Partial cost of the solution, i.e., cabling cost plus handoff of the seed node until the node were mentioned.

During the construction process of the solution tree, the first criterion for deciding whether to create or not a node in the level ***i***, is to check the remaining capacity $M_k^{(i)}$ of a switch. One node will only be created (assignment cell **Ai**) in the level *i* if $\lambda_i \leq M_k^{(i)}$, i.e., if the call number $\boldsymbol{\lambda_i}$ covered by the cell $\mathbf{A_i}$ is less or equal to the remaining capacity $M_k^{(i)}$ of the switch $\boldsymbol{C_k}$. If $M_k^{(i)} < \lambda_i$, all the branches that will be originated from this node are not going to exist, because the cell ***i*** cannot be attributed to the center ***k***.

After going through the test of the switch capacity, the node will only remain in the solution tree if it passes in the search width criterion **β**. For example, if the search width chosen by the user is **β =2**, in each level, only the nodes that generate the two minors solutions, computed by the technique of the greedy algorithm, remain in the tree.

Thus, for **β =2**, each node will only generate two branches, and they will be those that the greedy solution computation produces the least cost. The algorithm used for constructing the tree is the following algorithm:

```
Begin
level = 0
While (level < number of cells n) do:
  1º Step:
  Do level = level + 1
     S.1.1. Create the nodes of this new level
       according to the switch capacity rule.
     S.1.2. For each node created in the level i,
       follows:
       • Store the identification of the cell and the
       switch;
       • Calculate and store the node cost, i.e., the
       fixed cabling costs and the partial handoff costs
       related to the attribution made in the node.
       • Calculate and store the solution partial
       cost, i.e., the cabling cost plus the handoff cost
       of the seed node until the created node;
       • Calculate the remaining capacity $M_k^{(i)}$ of
       all the switches k;
  If (level < number of switch)
  Then
  2º Steps.
   For each node created in the 1º. Step (S1.1) and
   considering the calculated costs until then, find a
   greedy solution for the problem, through a Greedy
   Best-First Search (GBFS)[17][18] in the tree,
   starting from this node.
  3º. Steps
   Keep in the tree, in this level, the nodes that
   generated the best β greedy solutions. Discard the
   rest of the nodes.
End While
```

Initially the algorithm is in the level zero of solution, because the allocations will start to be made now. Then do level=1 and start creating nodes **(S1.1)** in this level. The nodes of level 1 are the seed nodes and are created one seed node for each switch. In **(S1.2)**, for each node created in **(S1.1)** is assigned a cell of the vector of cells considering the capacity of the switch. Follows it is calculated the cabling and handoff costs of each one of these nodes.

The greedy best-first search performed in the ***2º Step***, aims to choose, in the ***3º. Step***, which nodes, of the level ***i***, should remain in the tree of feasible solutions, to respect the width of search **β**. For example if **β=2**, in each level will remain only two nodes.

The creation process of the greedy solution is similar to the process of tree of solutions creation presented in Figure 3, except that here in each step it chosen the allocation that generated the best partial solution of the node, i.e., will be part of the solution the nodes with less costs. The greedy best-first search is made as described in subsection 4.2.1.

4.2.1 Greedy Best-First Search

For each node ***j*** created in the ***1º. Step*** **(S1.1)** makes a deep tree, from this node, going down to the last level, i.e.:

- Let node ***j*** created in the level ***i*** with a partial solution cost ***Sj***, calculated until the level ***i***.
- Choose to make part of this solution the node ***p*** from level ***i+1*** which is feasible and which generate the lowest partial solution **S*p***, where **S*p*= *Sj* +** fixed cabling cost + partial handoff costs related to the attribution made in the node ***p***.
- Repeat this process starting from the node ***p***, going down to the last level. Only after down the last branch of the node is that we have the total cost of a greedy solution.

As mentioned previously the greedy solutions are useful to choose among the nodes, of level ***i*** that was created in the ***1º. step***, those who will remain in the tree. This choice is made as follows: Order the nodes created at level i, in ascending order, according to the results of the cost of greedy solutions that each one of them generated. Between the nodes of level ***i*** choose for remaining in the tree the **β** nodes which generated the greedy solutions with less cost, cut the other nodes in the level ***i***, i.e.: if the search width is **β=2**, it will only continue to be part of the tree the two nodes that generated the two greedy solutions of less cost.

The value of the minor greedy solution calculated in the ***2º. Step***, is also used as upper bound (cutting) for generating or not the other nodes of the tree, in the ***1º. Step***. This upper bound should be updated when the minor greedy solutions are found. With this we have two cutting criterion for creating or not the nodes. The first of them is the switch capacity and the second is the cost of the minor greedy solution, i.e., if the partial cost of the solution in the node that has been just created is bigger than the best greedy solution cost this node should be excluded from the solution.

After finishing the greedy best-first search it should be verified if the level of the search tree is equal to the number of cells. If it is the case, the algorithm finishes, because the end of the tree is reached. If not, the algorithm goes back to the ***1º. Step*** to generate the nodes of the next level.

The Figures 4 (a)-(f), presented the detailed solution of the Example of the Section 3.1 by the Beam Search algorithm. In the example is considered the width equal to β =2. The steps to be followed for solving the example are:

(***i***) Construct the nodes of level **N1** (one node for each switch). The cabling and handoff costs, at this level are the same as zero (see the costs in Table 1 and 2).

(***ii***) In level **N2** initially are created 4 nodes, because from the assignment made in the level **N1**, there are 4 possible attributions for the cell 2 as follows: A_2C_1, A_2C_2, in the sub-tree of the left and A_2C_2, A_2C_1 in the sub-tree of the right. This situation is described in Figure 4(a) and corresponds to Beam Search ***1º. Step***. As the search width is two, only two of these nodes will remain in the tree.

(***iii***) For choosing which nodes of level **N2** will remain in the tree, a greedy best-first search is made, until the last level, for each one of the 4 attribution made in level **N2** (Figure 4(b)).

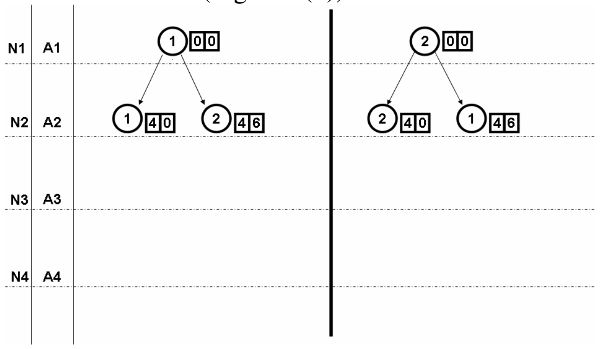

Fig. 4a Steps of Beam Search for the Example of section 3

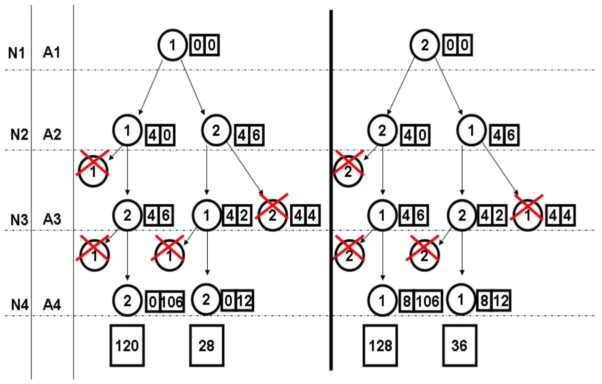

Fig. 4b Steps of Beam Search for the Example of section 3

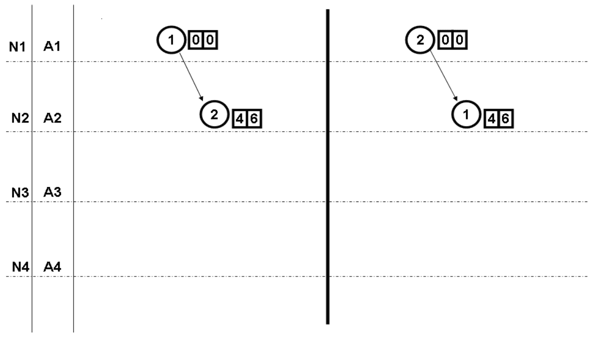

Fig. 4c Steps of Beam Search for the Example of section 3

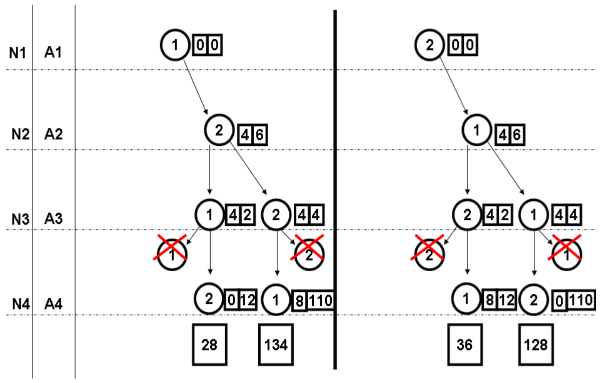

Fig. 4d Steps of Beam Search for the Example of section 3

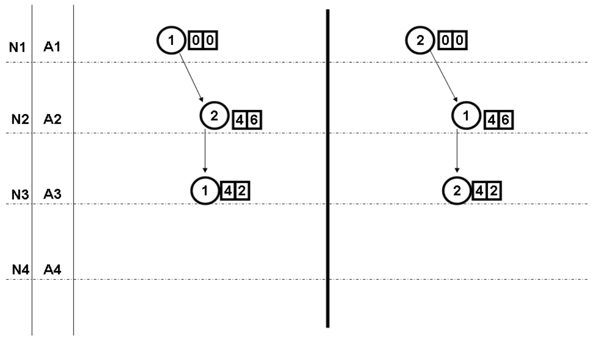

Fig. 4e - Steps of Beam Search for the Example of section

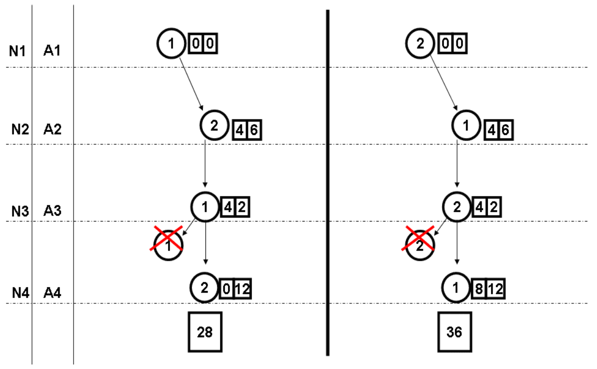

Fig. 4f - Steps of Beam Search for the Example of section

Suppose for example the node 2 of the level **N2**, on the left sub-tree, of Figure 4(b). The assignments made until this node are: A1C1 in level **N1** and A2C2 in level **N2** with cabling cost of 4 and partial handoff

6. Now, should be made the assignment of the cells A3 in level N3 and A4 in level N4. The cell A3 can be assigned to the switch C1 with cabling and handoff costs respectively of 4 e 2 (node 1 of level N3), and assigned to the switch C2 with cabling costs and handoff, respectively of 4 and 4 (node 2 of level N3). As it regards of a greedy algorithm, it is chosen to assign the cell A3 the switch C1 because the partial cost of this solution is less. It can be discard then the node 2 of level N3. Continuing the attributions from the node 1 of level N3, due to a capacity constraint, the cell A4 can only be attributed to the switch C2 and the cost of this attribution is zero for cabling and 12 for handoff. Applying this procedure for all the 4 nodes of level N2, will be found 4 greedy solutions, where will be choose the two with less cost. This situation corresponds to the ***2º Step of*** Beam Search.

(iv) The Figure 4c shows that the nodes choose to stay in the tree, level N2, (***3º Step***) were: node 2, in the left sub-tree and node 1 in the right sub-tree. Below are created the nodes of level N3 (***1º Step***), and later a greedy search from these nodes (Figure 4d), for choosing which nodes of level N3 will remain in the tree. The Figure 4e shows the nodes choose of level N3 (***3º. Step***).

(v) In the end, it is obtained two solutions: A_1C_1, A_2C_2, A_3C_1, A_4C_2 with cost of 28 and A_1C_2, A_2C_1, A_3C_2, A_4C_1 with cost of 36. It is then adopted the solution with less cost between the two. The Figure 4f shows these solutions.

5. Changes in the Beam Search

It is possible to improve the performance of Beam Search making some changes, such as establishing an order for making the attribution of the cells to switches, according to a heuristic criterion.

The example presented in section 4.1 followed for the numeric order of the cells. The descending order of the estimated cost of handoff and of cabling was chosen as a criterion to make the attribution of the cells, i.e., for each cell $\mathbf{A}_i$ was calculated the estimated cost of handoff and cabling CEHC given by Eq. (7).

$$CEHC_i = \sum_{j=1}^{n} h_{ij} + \sum_{k=1}^{m} c_{ik} \quad (7)$$

The Table 5 shows these calculations for the example of Section 4.1.

Antennan	**A1**	**A2**	**A3**	**A4**
CEHC	7	63	15	73
Order	4	2	3	1

Table 5: Allocation order of cells.

Another modification consists in applying the Beam Search algorithm into each sub-tree separately. The advantage is that instead of solving a big problem, we solve **m** smaller sub problems and so it is possible to decrease the risk of eliminating a complete sub-tree, when using the width search β, eliminating the region where the optimal solution is found. To better illustrate, imagine a more complex problem, than the one presented in Section 4.1, with 4 switches. Remember that each switch corresponds to a seed node, i.e, a complete tree. If we take the same width search used in Section 4.1 example, $\beta = 2$, then only two seed nodes will remain in level N1. Thus, two entire sub-trees will be cut without any Greedy Best-First Search Computation.

To avoid this problem we enforce the solution computation for each sub-tree by imposing the Beam Search application for each seed node. This procedure has two advantages:

(i) avoid the elimination of an entire tree, without any node evaluation of this tree;

(ii) could be implemented in parallel and each for tree a Beam Search is computed independently by one processor.

In this way, if the problem has four switches, it would have 4 processors and each one of these processors would execute one Beam-Search in one sub-tree. This implementation permits to obtain a significant reduction of computational time, because in each sub-tree we have a search problem totally independent, which dismisses the communication between the processors providing a speed-up [19] and an elevated efficiency.

6. Computational Results

Below are presented the computational tests made with the Beam Search algorithm applying the cabling and handoff costs to order the switch attribution and using two versions of the algorithm as follows: complete problem and sub-problems.

It is also run a Beam Search version (BS) applied to the entire problem and that make the attribution of the switches using a numeric criterion, as in the example of section 4.1. A number of 38 instances were run with varied dimensions. It has started with considered minor problems (45 cells and 2 switches), until problems considered large-scale (200 cells and 4 switches). The instances are the same of Merchant and Sengupta,(1995). The algorithms were implemented in Matlab 7.0, a machine with processor Intel 2.0 GHz, 1G of RAM.

Instance	Cells/Switch	O. F BSHC	O. F BSHCSP	O. F BS
1	45/2	975.88	975.88	**960.33**
2	45/3	555.80	555.80	555.80
3	45/4	**524.85**	**524.85**	529.01
4	45/5	441.95	441.95	**427.95**
5	60/2	921.91	921.91	921.91
6	60/3	**711.88**	**711.88**	767.25
7	60/4	**737.73**	**737.73**	739.89
8	60/5	725.41	725.41	**696.06**
9	75/2	1537.67	1537.67	1537.67
10	75/3	995.41	995.41	995.41
11	75/4	**1427.84**	**1427.84**	1444.14
12	75/5	**967.09**	**967.09**	973.093
13	100/2	1736.03	1736.03	1736.03
14	100/3	**1383.43**	**1383.43**	1401.56
15	100/4	**1441.01**	**1441.01**	1448.91
16	100/5	1489.90	1489.90	**1466.66**
17	125/2	2914.48	2914.48	2914.48
18	125/3	2217.93	2217.93	**2213.78**
19	125/4	2965.73	2965.73	**2959.14**
20	125/5	**1943.16**	**1943.15**	2041.40
21	150/2	4007.00	4007.00	4007.00
22	150/3	3393.07	3393.07	**3247.26**
23	150/4	2787.55	2787.55	**2744.26**
24	150/5	**3122.70**	**3122.70**	3177.71
25	175/2	5122.59	5122.59	5122.59
26	175/3	5566.67	5564.08	**4994.93**
27	175/4	**3229.75**	**3229.74**	3365.60
28	175/5	**2523.86**	**2523.86**	2532.85
29	200/2	5782.84	5782.84	5782.84
30	200/3	4371.26	4371.26	4371.26
31	200/4	4393.75	4393.75	4393.75
32	200/5	**3768.43**	**3768.43**	3792.84
33	225/2	**6771.95**	**6771.95**	6892.52
34	225/3	4960.16	4960.16	**4956.89**
35	225/4	**5384.48**	**5384.48**	**5375.05**
36	225/5	5630.72	5624.69	**5398.15**
37	250/2	8075.77	8075.77	8075.77
38	250/3	**8624.23**	**8624.23**	8714.41
39	250/4	6101.35	**6088.19**	6199.61

Table 6 – Results obtained with the Beam Search methods

The Table 6 shows, for each one of the instances, the values of objective function obtained with the three versions of Beam Search: BS, BSHC and BSHCSP. In the first column is the instance number, then number of cells and switches of the instances, and then, the objective function. In every versions it was used the width search equal to the number of switches. Table 6 shows that the heuristics that used the sort order to the cost provided better results than the single BS in most instances. Comparing the single BS with BSHC and BSHCSP, the BS single showed better results in 14 cases, in 17 cases the heuristics BSHC and BSHCSP performed better results and in other cases the results were similar. Comparing the heuristics to sort order by cost, the BSHCSP showed better results than the BSHC in 4 cases. In the others, they were equal.

To better present the geographical arrangement implicit in each different objective function value obtained with each version of Beam Search Method, the instances 8 (60/5) and 20 (125/5) had been selected and each solution is showed in Figures 5 and 6, respectively. In these figures the cells with a small antenna, denote the locations of the MSC.

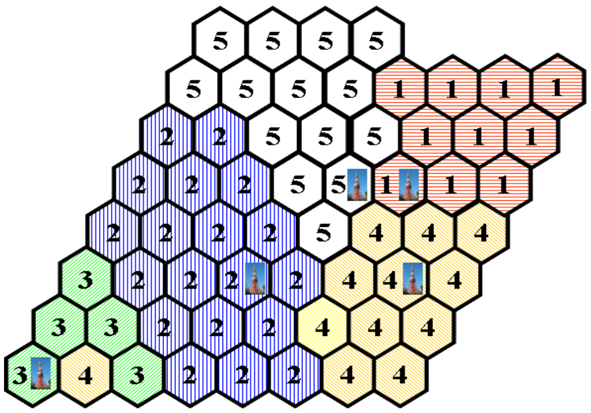

Fig. 5a Solution provided by BS for instance 8 (60/5).

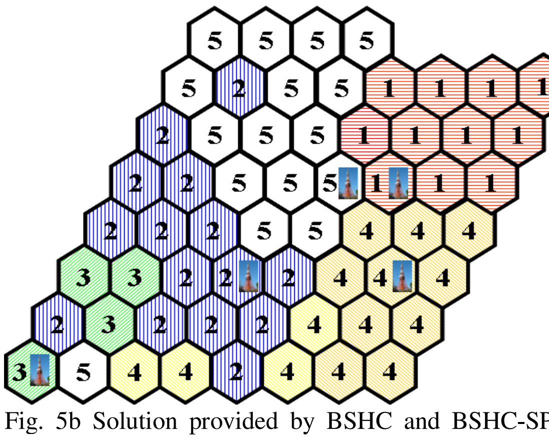

Fig. 5b Solution provided by BSHC and BSHC-SP for instance 8 (60/5).

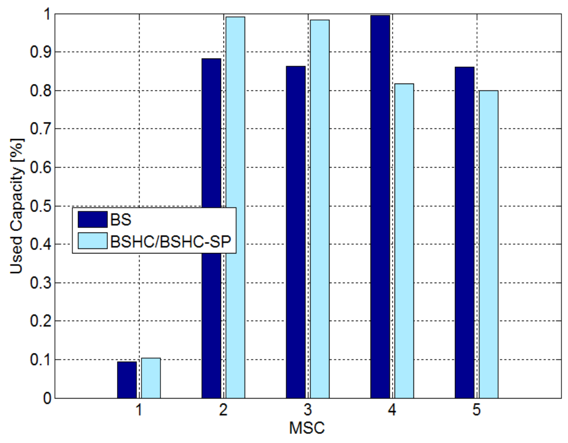

Fig. 5c Capacity usage in percentage for each MSC in the solution provided by BSHC and BSHC-SP for instance 60/5.

For instance 8, the Table 6 shows that the method BS produced a better solution than the methods BSHC and BSHC-SP. The consequence of 4.21% difference on the objective function value could be better visualized in the Figure 5a and 5b. The better objective function value of BS (value 696.06, Fig. 5a) provides a solution which better cluster the cells to each MSC than the solution proposed by BSHC/BSHC-SP methods (value 725.41, Fig. 5b). The Figure 5c presents the capacity usage in percentage for each MSC and for each method and clearly points out that the difference on the solutions mainly depends on the attributions made for MSC 2, 3 and 4.

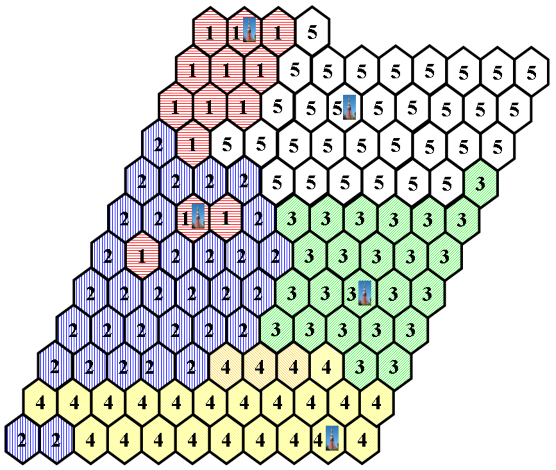

Fig. 6a Solution provided by BS for instance 20 (125/5).

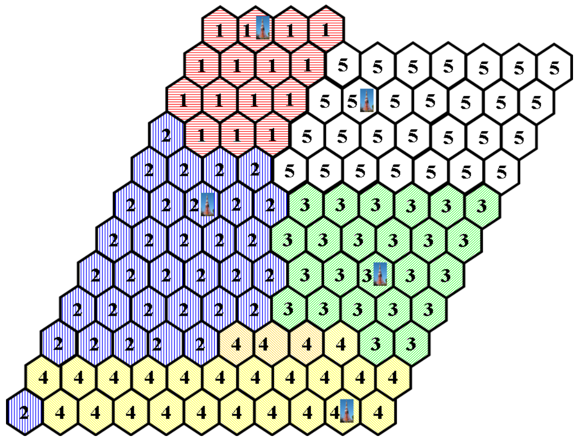

Fig. 6b Solution provided by BSHC and BSHC-SP for instance 20 (125/5).

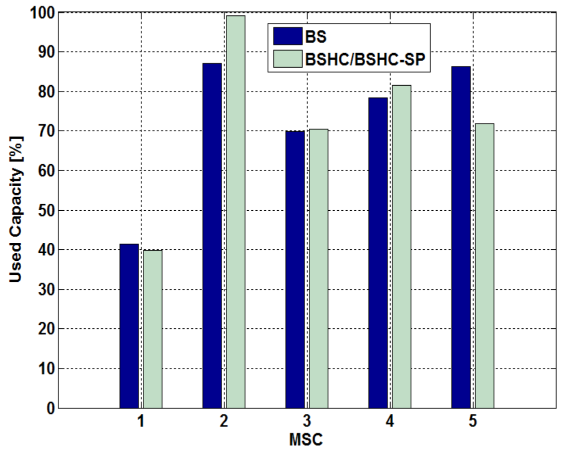

Fig. 6c Capacity usage in percentage for each MSC in the solution provided by BSHC and BSHC-SP for instance 20 (125/5).

For the results presented for instance 20, in Figure 6, the table 6 shows that the methods BSHC and BSHC-SP (1943.15) produced a better solution than the method BS (2041.40). As a consequence BSHC and BSHC-SP produces a solution, which better cluster the cells to each MSC (Fig. 6b) than the solution proposed by BS (Fig. 6a). The Figure 6c presents the capacity usage in percentage for each MSC and for each method and clearly points out that the difference on the solutions mainly depends on the attributions made for MSC 2 and 5.

7. Conclusion

Heuristics type Beam Search does not require an initial solution and always find a feasible solution. One of the advantages of theses heuristics is that they do not generate all branches of the tree; their execution time is polynomial. The depth-first search allows finding good solutions, because it considers the cost of allocation and the switches capacity. The result obtained with the 3 heuristics showed that they are relatively equivalents, concerning the value of the objective function, with a small gaining for those that used the order of allocation considering the handoff and cabling costs. Furthermore, the Beam Search that works in the sub-problems (BSHCSP) could be easily implemented in parallel and this can provide a large reduction of computational time. Therefore, it is needed to put a processor to work on each sub-tree. Each processor will then search the best solution of each one of these sub-trees.

The efficiency of a parallel algorithm depends on a balance distribution of the tasks that are performed on each processor and the volume of communications made between the processors. In the BSHCSP the sub-trees have the same size and are completely independent, thus the tasks to be executed by each processor are balanced and require no communication between them.

Then is possible to obtain a reduction of computational time, which is of about 80% of the time spent in the sequential algorithm. For example, in the case of 5 processors the execution time in parallel would be approximately the sequential execution time divided by the number.

References:

[1] AbuAmara, M. H.; Sait, S. M.; Subhan, A., A Heuristics Based Approach for Cellular Mobile Network Planning, *The International Wireless Communications and Mobile Computing Conference (IWCMC06)*, July, 2006,Vancouver, Canada.

[2] Dianati, M.; Naik, S.; Shen, X.; Karray, F., Genetic Algorithm Approach for Cell to Switch Assignment in Cellular Mobile Networks, *2003 Canadian Workshop on Infor. Theory*, pp. 159-162, May 18-21, 2003, Waterloo, Ontario, Canada.

[3]Houeto, F.; Pierre, S., Assigning cells to switches in cellular mobile networks using taboo search, *IEEE Transactions on Systems, Man and Cybernetics, Part B,* vol. 32 n. 3, 2002, pp. 351 –356.

[4]Quintero, A.; Pierre, S., Assigning cells to switches in cellular mobile networks: a comparative study. *Computer Communications*, v. 26, n. 9,2003, p. 950-960.

[5]Merchant,A.; Sengupta, B., Assignment of cells to switches in PCS networks, *IEEE/ACM Trans. Networking*, vol. 3, Oct, 1995, pp. 521-526.

[6]Saha, D.; Mukherjee, A.; Battacharya, P.S, A simple heuristic for assignment of cells to switches in a PCS network, *Wireless Personal Communications* 12 (3), 2000, pp.209-224.

[7]Menon, S.; Gupta, R., Assigning cells to switches in cellular networks by incorporating a pricing mechanism into simulated annealing, *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, n. 1, 2004, pp. 558-565.

[8]Salomão, N.A., *Métodos de Geração de Colunas para Problemas de Atribuição,* (in Portuguese) PhD. thesis in Applied Computer, INPE, São José dos Campos, SP, Brazil, 2005.

[9]Alonso, E., Meier-Hellstern, S., Pollini, G.P., Influence of Cell Geometry on Handover and Registration Rates in Cellular and Universal Personal Telecommunications Networks, *8th ITC Specialist Seminar on universal Communication,* Genova, Italy, 1992.

[10] Hedible, C.; Pierre, S., Genetic algorithm for the assignment of cells to switches in personal communication networks, *Electrical and Computer Engineering, Canadian Conf. on, vol. 2*, 2000, pp. 1077-1081.

[11] Lowerre, B. T., *The HARPY Speech Recognition System*, PhD. thesis, Carnegie-Mellon University, USA, 1976.

[12] Sabuncuoglu, I.; Bayiz, M., Job Shop Scheduling with Beam Search, *European Journal of Operational Research,* vol. 118, 1999, pp. 390-412.

[13] Della Croce, F.; T'kindt, V., A Recovering Beam Search Algorithm for the One-Machine Dynamic Total Completion Time Scheduling Problem, *Journal of the Operational Research Society*, vol 54, 2002, pp. 1275-1280.

[14] Fox, M.S., Constraint-Directed Search: A case Study of Job-Shop Scheduling, *PhD. thesis*, Carnegie-Mellon University, USA, 1983.

[15] Ow, P.S; Morton T.E., Filtered Beam Search in Scheduling, *International Journal of Production Research,* vol. 26, 1988, pp. 35-62.

[16] Valente, J. M. S; Alves, R. A. F. S., Filtered and Recovering Beam Search Algorithm for the Early/Tardy Scheduling Problem with No Idle Time, *Computers & Industrial Engineering*, vol. 48, 2005, pp. 363-375.

[17] Vidal, V. A lookahead strategy for heuristic search planning, In Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-04), pp. 150-160, 2004.

[18] Russell, S. Norvig, P. *Artificial Intelligence: A Modern Approach.*, pp.94-95, Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, 2nd edition, 2003.

[19] Schendel, V., *Introduction to Numerical Methods for Parallel Computers*, Ellis Horwood Series, John Wiley and Sons N. Y., 1984.