Journal of Computer Science 1 (2) : 175-181, 2005
ISSN 1549-3636
© Science Publications, 2005

A Variable Neighborhood Search Algorithm for Assigning Cells to Switches in Wireless Networks

Matthieu André, Gilles Pesant and Samuel Pierre
Department of Computer Engineering, École Polytechnique de Montréal
C.P. 6079, succ. Centre-ville, Montréal, Qué., Canada H3C 3A7

Abstract: The problem of assigning cells to switches in wireless networks consists of minimizing the total operating cost, that is, the cost of linking cells to switches and the cost of handover from one cell to another, by taking into account factors such as network topology, switch capacity and traffic load in the entire network. Such a problem is well known in the literature as NP-hard, such that exact enumerative approaches are not suitable for solving real-size instances of this problem. Thus, heuristics are recommended and have been used for finding good solutions in reasonable execution times. Tabu Search (TS) is one of the best heuristics used to solve this problem. This research proposes a hybrid heuristic approach for further improving the quality of solutions obtained from TS. This approach applies TS in combination with variable neighborhood search, a recent metaheuristic that is based on the principle of systematic change of neighborhood during the local search. A key element in the success of this approach is the use of several neighborhood structures that complement each other well and that remain within the feasible region of the search space.

Key words: Cell Assignment, Mobile Networks, Variable Neighborhood Search, Tabu Search, Heuristics

INTRODUCTION

Mobile communication services are provided on personal communications networks. The coverage area serviced by such networks is generally divided into limited small geographic areas, commonly known as cells and presented in hexagonal shapes, whose radius varies from a few hundreds meters to several kilometers. In each of these cells, there is a sub-radio system made of a base station also called Base Transceiver Station (BTS) transmitting radio signals on a cell. Signal channels integrated to the base station allow communication subscribers to communicate with the BTS and vice versa. Base stations, for their part, are linked to Base Station Controllers (BSC). This subsystem is actually the radio interface between each mobile terminal and the network. The network subsystem is composed of switches or Mobile service Switching Centers (MSC) located in some strategically chosen cells. The role of a switch is to manage the interconnection of different mobile network cells and ensure interconnection with other telecommunications networks.

To avoid interference, two contiguous cells should not use identical radio channels. The transmission must thus change channel every time the mobile passes from one cell to another. This automatic transfer process from a base station to another is called handover or handoff. The cellular system permanently controls the signal power between the mobile unit and the nearest base station. As soon as the power drops below a certain level, the system automatically attributes a new cell to the mobile unit. This transfer of cells could cause a change of switch. In this case, update operations are performed and this is considered to be a complex handoff; otherwise, it is a simple handoff that involves only one switch.

Taking into account the signal level determines the choice of a cell. Generally, a threshold is defined to establish the limit beyond which the power received could be considered so great that a cell could not take it into account. The information exchanged by the users is generated by the switch servicing the cell. When different contiguous cells receive signals greater than the established threshold, only the one that has a superior level could ensure the service to the user.

Figure 1 shows that several scenarios are possible during the movement of the mobile unit of cell C_i, linked to switch MSC_k, toward another cell C_j with a superior signal power:

The mobile unit is being serviced by cell C_2. In this case, C_1 and C_2 are controlled by the same switch MSC_2. Therefore, the handoff is simple and does not require the update of the user's database. In this operation, only switch MSC_2 is involved.

The mobile unit is serviced by cell C_0, which is controlled by a different switch MSC_3. In this case, a large amount of information is exchanged between the two switches to update the network's database (localizing the user, exchange of protocols, type of calls, etc.). Moreover, certain operations such as billing could be carried out by switch MSC_2. As a result, we will have a user connection to switch MSC_3, then to switch MSC_2 and finally to the network.

A complex handoff is an operation requiring several network resources and its cost has to be reduced as much as possible. Thus, it would be desirable to establish a handoff frequency among the different cells, in order to assemble those with frequent mutual handoff. These considerations are at the basis of the assignment problem that could be expressed as follows: Given a set of cells and switches of finite capacities expressed in terms of Erlang or BHCA (Busy Hour Call Attempts), the problem is to assign cells to switches that would minimize the total cost, that is, the cost of linking cells to switches and the cost of handoff from one cell to another, by taking into account factors such as network topology, switch capacity and traffic (volume of communications exchanged by time unit in each cell).

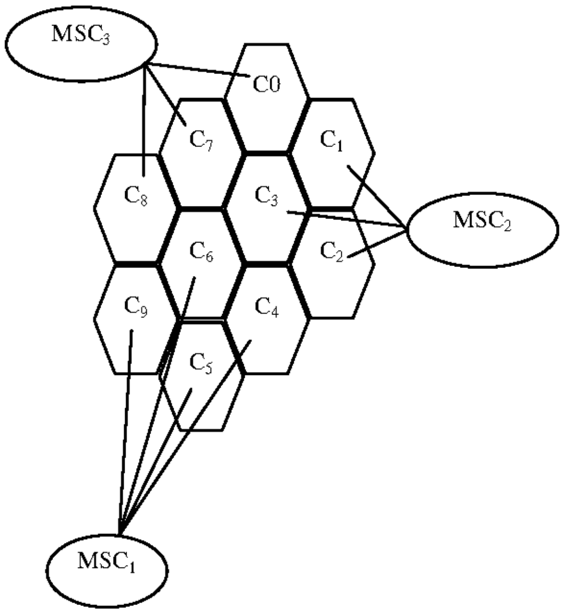

Fig. 1: Handover in a Cellular Network

Such a problem is well-known in the literature as NP-hard [4], such that exact enumerative approaches are not suitable for solving real-size instances of this problem. As a result, heuristics are recommended and have been used for searching good solutions in reasonable execution times.

One such heuristic approach is simulated annealing (SA). SA is a probabilistic algorithmic approach based on the analogous physical process of heating and then slowly cooling a substance to obtain a strong crystalline structure. Kirkpatrick *et al.*, [11] adapt it to solve combinatorial optimization problems. SA picks a neighbor at random and then accepts it if it improves the current cost, otherwise SA rejects it with probability depending on a cooling schedule and the amount of deterioration.

TS is one of the best heuristics used to solve this problem [6]. It is an improvement on the descent algorithm that avoids the trap of the local optimum [3]. When a local optimum is found by the descent algorithm, it accepts solutions that do not improve the current solution. To avoid any risk of cycling, a tabu list T is added that keeps some information about the last k solutions visited in terms of the reverse action used to go from a recent solution to the previous one. Then a solution is prohibited if it is obtained by applying a tabu action to the current solution. Usually an aspiration criterion may be used to temporarily remove the tabu status of an action that would lead to a new best solution.

To adapt the method to the assignment problem, Pierre and Houeto [6] start from a solution obtained by assigning each cell to the closest switch, ignoring constraints. A tabu list component allows the exploration of this search space without cycling. The search space is defined without switch capacity constraints while respecting the assignment constraint specific to cells. Each new solution achieved is evaluated according to two criteria. The first criterion is related to the cost calculated from the objective function, while the second takes into account a penalty introduced whenever a capacity constraint is violated. At each step, the best solution is chosen according to these criteria. Three memory structures avoid cycling around a local optimum and refine the search: a short-term memory which improves the current solution based on cost and the penalty criteria associated with each solution; a medium-term memory which leads back to promising regions to locally intensify the search; and a long-term memory which diversifies the regions explored [6].

Nevertheless, a comparison with constraint programming (CP) results [7] proves that the results provided by this TS algorithm are often non optimal and may be improved easily within a few seconds of execution. Moreover, [6] does not guarantee the feasibility of the final solution, because of the relaxation it uses and [7] CP needs too much execution time to find the optimum. Thus this study proposes another TS approach, combined with variable neighborhood search, for further improving the quality of solutions obtained from [6], given small improvements lead to big money savings.

PROBLEM FORMULATION

The problem of assigning cells to switches could be modeled using several approaches. In this study, we assume that the network has n cells and m switches whose locations are fixed and known. To each switch a set of cells must be assigned, according to the volume of calls that the switch is capable of managing. For each pair of cells i and j, we have the following data:

H_{ij} – Cost by time unit for a complex handoff between cells i and j;

C_{ik} – Cost of cabling between cell i and switch k;

λ_i – Volume of calls by time unit received by cell i;

M_k – Capacity of switch k.

Without loss of generality, we can neglect the cost of a simple handoff. The objective is to find an assignment that minimizes the total cabling and handoff cost and respects the capacity constraint of each MSC. To describe the problem, we introduce the following variables that express the unique assignment of each cell:

$S_i \in \{1,\dots,m\}$, S_i takes for value the switch to which all i is assigned.

The cost of each cell i is the sum of the cabling cost between cell i and assigned switch S_i and the cost of handoff:

$$\chi_i = C_{iS_i} + \sum_{\substack{j=1 \\ S_j \neq S_i}}^{n} H_{ij}$$

The total volume of calls generated by all cells linked to switch k must not be higher than the maximum capacity of the switch and can be expressed as follows:

$$\sum_{\substack{i=1 \\ S_i = k}}^{n} \lambda_i \leq M_k$$

Thus, the problem could be formulated as follows:

$$\text{Minimize} \sum_{i=1}^{n} \chi_i$$

subject to

$$\sum_{\substack{i=1 \\ S_i = k}}^{n} \lambda_i \leq M_k$$

$1 \leq k \leq m, \quad 1 \leq S_i \leq m, \quad 1 \leq i \leq n.$

VARIABLE NEIGHBORHOOD SEARCH

VNS is a recent metaheuristic that is based on the principle of systematic change of neighborhood during the local search. Following [12], consider an optimization problem formulated as follows: $\min\{f(x), x \in X, X \subset S\}$ where S is the solution space, X the feasible set, x a feasible solution and f a real valued function. Consider also a finite set of pre-selected neighborhood structures $\mathcal{N}$. With these neighborhoods, we can build many optimization algorithms. The simplest is called variable neighborhood descent (VND) where changes of neighborhood are made in a deterministic way, as described in Fig. 2.

```
Initialization: Select a subset of neighborhoods $\{N_1, N_2, \dots, N_{k_{max}}\} \subseteq N$ that will be used in the descent and an initial feasible solution $x$
Repeat the following sequence until no improvement is obtained
Step 1: Set $k$ to 1;
Step 2: Find the best neighbor $x'$ of $x$ in $N_k$;
Step 3: If $x'$ is better than $x$ set $x$ to $x'$ and go to Step 2, otherwise increment $k$;
Step 4: If $k \leq k_{\max}$ go to Step 2 else end.
```

Fig. 2: Variable Neighborhood Descent Algorithm

With $k_{\max} = 1$, we can note that this algorithm is the well-known descent algorithm. We can also note that the final solution is a local optimum with respect to all the neighborhoods used and so the chances of obtaining a global optimum are increased.

This deterministic algorithm may be combined with a stochastic one to give the algorithm called basic VNS, described in Fig. 3.

```
Initialization: Find an initial feasible solution $x$ and a stopping criterion
Repeat the following sequence until the stopping criterion is met:
Step 1: Set $k$ to 1;
Step 2: Generate a random neighbor $x'$ of $x$ in $N_k$;
Step 3: Apply a local search method with $x'$ as initial solution; denote $x''$ the so obtained local optimum;
Step 4: If $x''$ is better than $x$ set $x$ to $x''$ and go to Step 2 otherwise increment $k$;
Step 5: If $k \leq k_{\max}$ go to Step 2 else end.
```

Fig. 3: Basic Variable Neighborhood Search Algorithm

We can observe that point x' is generated at random in Step 2 in order to avoid cycling which might occur if any deterministic rule was used, as in the VND algorithm. The local search in Step 3 may also be replaced by VND in an approach that led to the most successful applications recently reported.

VNS may also be combined with Tabu Search. Basically there are two ways to realize the combination: use TS within VNS or use VNS within TS. An example of TS used as the local search (Step 3 in Fig. 3) within VNS can be found in [13]. The method we propose in the next section falls into the other category.

THE ALGORITHM

Given that Tabu Search (TS) and Constraint Programming (CP) both provide interesting results, the first idea is to combine them: a CP algorithm [7] is first used to construct a feasible initial solution to each instance and then a TS algorithm is applied to improve the result. We then try to improve TS with the addition of a variable neighborhood structure.

Reassignment: To be consistent with Constraint Programming, we first chose to ensure the feasibility of each intermediate solution of the TS algorithm, whereas [6] chose not to. Thus we have a different neighborhood from [6] to explore that may lead to better solutions. The feasibility of each intermediate solution also avoids the relaxation of the problem and thus ensures that the final solution is feasible. We first chose to use a basic local move structure that modifies the assignment of a single cell. Thus, a cell i assigned to switch k' becomes assigned to switch k. We call this neighborhood reassignment. We define Δ_{ik} for each cell i and each switch k, so that moving cell i from switch k' to switch k changes the cost Γ of the current solution to $\Gamma - \Delta_{ik}$. The table of Δ_{ik} values is updated after each move and a simple inspection of its values defines the following move.
Each move that was applied is stored in the tabu list. The latter is a LIFO queue of fixed length that forbids inverse moves from the present one. The aspiration criterion used is the traditional one: a move that improves the best solution found so far and that is tabu is performed anyway. Older tabu moves are dropped when the queue is full. The stop criterion is also classical and is defined by a maximal number of consecutive moves without an improvement of the best solution.

Redistribution: This simple implementation of TS does not always find the optimum for small instances for which the optimum is known.
In another adaptation of TS to the problem, traditional intensification and diversification techniques have been used to improve the algorithm [6]. We here propose to use another neighborhood, as in VNS, when there exists no move that can improve the current solution in the neighborhood. In this case, the classical algorithm would choose the best move even if it deteriorates the cost. Doing so, the hope is that we may find a better solution in a later iteration. This method may look too optimistic because there is very little chance that such a move would effectively lead to a better solution. We propose here to choose a move that will lead to good moves that are currently infeasible because of the constraints of the problem. Let D_k be the demand for switch k defined by:

$$D_k = \sum_{\substack{i=1 \\ \Delta_{ik} > 0 \\ \lambda_i > \mu_k}}^{n} \Delta_{ik}$$

where μ_k is the residual capacity of switch k given by:

$$\mu_k = M_k - \sum_{\substack{i=1 \\ S_i = k}}^{n} \lambda_i$$

The demand can be interpreted as the possible improvement on the cost if switch k would have an infinite capacity and can be understood as the overall desire of cells to be assigned to this switch. Then, switch k' with the greatest value of D_k is the switch most in demand and the algorithm tries to choose a local move that reassigns a cell i assigned to k', possibly with a large number of calls λ_i. The resulting new assignment will also be chosen with the lowest impact on the cost function: since $\Delta_{ik} < 0$, the algorithm will maximize Δ_{ik} / λ_i. Such a move, which frees call-handling capacity on switch k', may allow many good moves that were infeasible before.
Actually this method tends to reassign a cell that stops other cells from taking its place and thus greatly improves the solution, so in others words we can say that it redistributes switches between cells to satisfy a majority. We will call this neighborhood redistribution.
We can further improve the formulation of demand. The current formulation gives the same impact to a cell with a low number of calls (which has a great chance of having its demand satisfied) than to a cell with a huge number of calls (with very little chance of having it satisfied). Then we use the following formulation:

$$D_k = \sum_{\substack{i=1 \\ \Delta_{ik} > 0 \\ \lambda_i > \mu_k}}^{n} \frac{\Delta_{ik}}{\lambda_i - \mu_k}$$

Using this formulation, a cell that is very likely to be redistributed will have a more important weight in the demand.

Double Move: TS with simple reassignment only allows moves that satisfy the limited handling capacity of switches. As a result, there may be a lot of moves that would improve the current solution but which are infeasible. We use the redistribution neighborhood which gives a chance to bypass this problem when the solution cannot be improved with the first neighborhood. The following idea is that we may also try to bypass this problem by using a third neighborhood. That's why we decided to add a third type of local move that changes the assignment of two cells at a time. We will call this new local move double move. The equivalent result obtained by two consecutive local moves using the first neighborhood might not always be feasible because of capacity constraints.

Figure 4 shows two resulting interesting local moves. On the left, we can presume that, because of the capacity constraint, cells A and B cannot be assigned respectively to switches 1 and 2, even if the overall move improves the solution. The problem is solved when we exchange the assignment of both cells. On the right, we can also imagine that cell A cannot be assigned to switch 2, even if it would improve the current solution and that cell B can be assigned to switch 3 with a little deterioration of the solution. The resulting shift solves the problem and globally improves the current solution.

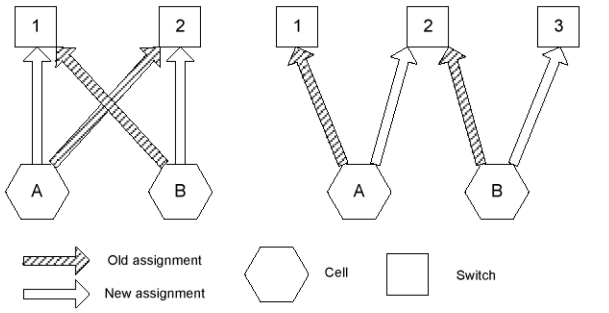

Fig. 4: Example of Double Moves

This double move may also combine two feasible simple moves and so speed up the simple TS algorithm. So during the choice of the next move, the algorithm will take the best move in cost. A double move will be considered as tabu if it deteriorates the current solution and if one of the two simple moves used is tabu. These two simple moves are added to the same tabu list if the double move deteriorates the solution.

Beyond Double Move: The addition of the double move improves the algorithm, because it extends the neighborhood and it allows bypassing the capacity constraint. This constraint bypass may be an important key to finding better solutions to the problem. One can also notice that this double move is a simple particular case of the well-known ejection chains technique [10], where there is one ejection. The idea is then to apply the principle of ejection chains to our problem. Doing so, the chain would begin with a move that assigns a cell i_1 to switch k_1. Such a move would violate the capacity constraint of k_1, so another cell i_2 would have to be ejected from k_1 and reassigned to another switch k_2. Eventually, this ejection move could violate the capacity constraint of k_2 and so on. The ejection process would then continue until all constraints are satisfied. One way to implement such a process consists of allowing a violation of the capacity constraint and then applying the TS algorithm with an objective function composed of the residual capacity of switches. Doing so, it is true that we violate our first rule, that was to ensure the feasibility of each intermediate solution, but actually we ensure that we will obtain a feasible final solution, so globally we do not violate the idea of the first rule.

With this technique, we can replace the use of double moves. Indeed it extends the idea of double moves allowing an arbitrary number of simple moves. But the choice of move is linked to a bit of chance, whereas the double move technique takes the best move in cost. It will be interesting to compare the two techniques later.

RESULTS

Our experiments were performed on Sun Microsystems Ultra 10 Model 440, 64 bit stations, equipped with a 440 MHz UltraSPARC-III processor and 1024 MB of memory.

We used 5 sets of 15 instances of data corresponding to problems with respectively 30 cells and 3 switches, 50 cells and 4 switches, 100 cells and 5 switches, 150 cells and 6 switches and 200 cells and 7 switches. These instances are the same as [6]. We thus cover a large set of configurations.

A Constraint Programming (CP) algorithm [7] was first used to construct a feasible initial solution to each instance. This CP algorithm provides a good initial solution in a very short time (less than half a second for the biggest problems). The average deviation from the best known solution is less than 2 %.

The good quality of this initial solution may clearly be a factor that helps our method to find good solutions so we added another possibility of initial solution: it is obtained with a "maximal gap" algorithm that sequentially affects cells to switches maximizing at each step the greatest remaining capacity. This other initial solution is also feasible but its cost is far from the optimum. It is obtained in approximately the same time as the CP one.

On this set of tests, we found that the best tabu parameters with the simple reassignment neighborhood are a size of 13 for the tabu list and a stopping criterion of 100 fails. Experiments have shown that better results can be obtained with this choice of values. We decided to apply the same parameters to all the neighborhoods. Thus, we can note than a further calibration of tabu parameters for other neighborhoods could improve the results.

Figure 5 shows the impact of redistribution, double move and ejection chain neighborhoods on the quality of results. These experimentations were performed with the CP initial solution. We consider here the average percent deviation from the best known solution of all our experimentations as the criteria for the quality of solutions. Note that, for most instances under 150 cells, the CP algorithm [7] gives us the optimal solution. So we have gap from optimal in that case. We can observe in Fig. 5 that both redistribution and double move neighborhoods increase the quality of our results. However, in general ejection chains appear less efficient than double moves. Actually ejection chains proceed to find a combination of moves that may improve the current solution. So with this technique not every chain will improve the solution, whereas each double move improves it. Thus it is not surprising that double moves are in general more efficient than ejection chains. The only real advantage with ejection chains over double moves is that they are quicker. But the difference between the two methods is only of one or two seconds.

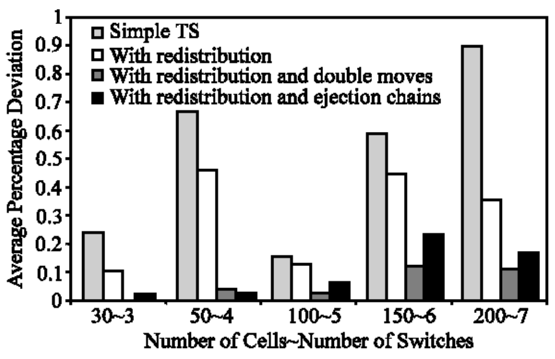

Fig. 5: Average Percent Deviation from Best Known Solution

We can also compare our heuristic with others from the literature: a TS algorithm [6] equipped with intensification and diversification techniques, a simulated annealing (SA) algorithm [14-15] and a limited time version of a CP algorithm [7].

Tests have been performed on the same equipment, with the same instances and with parameters recommended by the authors of these other algorithms. We first have to note that initial solutions of other local search algorithms are different from ours. We chose to keep these initial solutions to respect the authors' choices. A comparison between all these methods requires a special attention on both computing time and solution cost. We so chose to observe the evolution of the average percent deviation of the best feasible solution found by each algorithm over time. The first point (with the lower time) in the observation represents the first feasible solution found by the algorithm. To do so, each algorithm was executed for a duration in seconds equal to the number of cells multiplied by the number of switches. We thus set the stopping criterion to the time limit with our TS/VNS algorithm. Figures 6 and 7 with logarithmic scales present the results obtained by these methods for two series of tests.

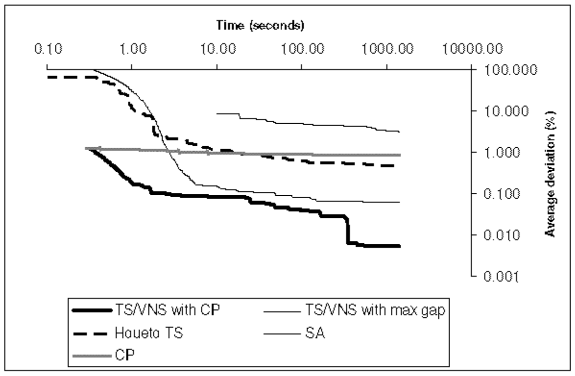

Fig. 6: Comparison with other Methods with 200 Cells and 7 Switches

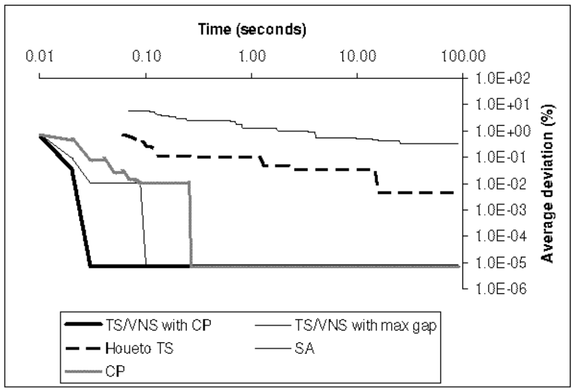

Fig. 7: Comparison with other Methods with 30 Cells and 3 Switches

We can observe on both figures the good performance of our method. At Fig. 5, instances have a complexity of 7^{200}. With a CP initial solution, after 0.5 second, our algorithm is the quickest and gives the best results. With the other solution, it needs much more time to give good results but after 3 seconds it is better than other methods except the one with the CP initial solution. With different initial solutions, our results are therefore very near. Figure 6 seems to confirm our observations, even with a lower complexity of 3^{30}. It is important to note that for all these instances of Fig. 6, the optimum found is proved by CP. We must note that because of the logarithmic scale, the null deviation of the optimum cannot appear and is thus set to nearly 0.00001 %. Thus we can also observe that our method is the only heuristic that always finds the optimum within the time limit.

CONCLUSION

We presented another heuristic algorithm based on TS and VNS to solve the assignment of cells to switches in a wireless network. A constraint programming (CP) algorithm was first used to construct a feasible initial solution to each instance. This CP algorithm provides a good initial solution in a very short time (less than half a second for the biggest problems).

We first evaluated the impact of redistribution, double move and ejection chain neighborhoods on the quality of results. Both redistribution and double move neighborhoods increase the quality of our results. However, in general ejection chains appear less efficient than double moves. We also compared our heuristic with others from the literature: a TS algorithm equipped with intensification and diversification techniques, a simulated annealing (SA) algorithm and a limited time version of a CP algorithm.

Combining ideas from variable neighborhood search and tabu search paradigms, our approach outperforms the other algorithms on a standard set of benchmarks. A key element in the success of this approach is the use of several neighborhood structures that complement each other well and that remain within the feasible region of the search space. Though the benchmark problems used were carefully generated with realistic parameters, additional experimentation based on actual data from the telecommunications industry would be a logical next step in order to assess more clearly the performance of the algorithm proposed. We only tested one combination of TS and VNS. It could therefore be interesting to experiment an algorithm using TS within VNS on this problem in a further work.

REFERENCES

1. Bhattacharjee, P.S., D. Saha and A. Mukherjee, 1999. Heuristics for assignment of cells to switches in a PCSN: a comparative study. IEEE Int. Conf. on Personal Wireless Comm., pp: 331-334.
2. Cox, L.A., Jr. and J.R. Sanchez, 2000. Designing least-cost survivable wireless backhaul networks. J. Heuristics, 6: 525-540.
3. Glover, F., E. Taillard and D. de Werra, 1993. A user's guide to tabu search. Annals of Operations Research, 41: 3-28.
4. Merchant. A. and B. Sengupta, 1994. Multiway graph partitioning with applications to PCS networks. IEEE Infocom'94, 2: 593-600.
5. Merchant, A. and B. Sengupta, 1995. Assignment of cells to switches in PCS networks. IEEE/ACM Trans. on Networking, 3: 521-526.
6. Pierre, S. and F. Houeto, 2002. A tabu-search approach for assigning cells to switches in cellular mobile networks. Computer Communications, 25: 465-478.
7. Amoussou, G., M. André, G. Pesant and S. Pierre, 2003. Une approche basée sur la programmation par contraintes pour affecter des cellules à des commutateurs dans les réseaux cellulaires pour mobiles. Annales des Télécommunications, 58 : 584-604.
8. Rardin, R.L. and R. Uzsoy, 2001. Experimental evaluation of heuristic optimization algorithms: A tutorial. J. Heuristics, 7: 261-304.
9. Barr, R.S., B.L. Golden, J. Kelly, W.R. Stewart and M.G.C. Resende, 1995. Designing and reporting on computational experiments with heuristic methods. J. Heuristics, 1: 9-32.
10. Glover, F., 1992. New ejection chain and alternating path methods for traveling salesman problems. Computer Sci. Operations Res., pp: 449-509.
11. Kirkpatrick, S. and Gelatt Jr. C. D. et Vecchi M.P., 1983. Optimization by simulated annealing. Science, 220: 671-680.
12. Hansen, P. and N. Mladenovic, 2002. Variable neighbourhood search. State-of-the-art Handbook of Metaheuristics, Glover and Kochenagen (Eds.), Kluwer Academic Publishers, Dordrecht 2002 (In Press).
13. Rodriguez, I., M. Moreno-Vega and J. Moreno-Perez, 1999. Heuristics for routing-median problems. SMG Report, Université Libre de Bruxelles, Belgium.
14. Pierre, S. and F. Houeto, 2002. Assigning cells to switches in cellular mobile networks using tabu search. IEEE Transactions on Systems, Man and Cybernetics: Part B (Cybernetics), 32: Part B,: 351-356.
15. Beaubrun, R., S. Pierre and J. Conan, 1999. An efficient method for optimizing the assignment of cells to mscs in pcs networks. Proceedings of 11th International Conference on Wireless Communication, Wireless 99, 1: 259-265. Calgary, Canada.