Optimization of 3G Mobile Network Design Using a Hybrid Search Strategy

Yufei Wu and Samuel Pierre

Abstract: **This paper proposes an efficient constraint-based optimization model for the design of 3G mobile networks, such as universal mobile telecommunications system (UMTS). The model concerns about finding a set of sites for locating radio network controllers (RNCs) from a set of pre-defined candidate sites, and at the same time optimally assigning node Bs to the selected RNCs. All these choices must satisfy a set of constraints and optimize an objective function. This problem is NP-hard and consequently cannot be practically solved by exact methods for real size networks. Thus, this paper proposes a hybrid search strategy for tackling this complex and combinatorial optimization problem. The proposed hybrid search strategy is composed of three phases: A constraint satisfaction method with an embedded problem-specific goal which guides the search for a good initial solution, an optimization phase using local search algorithms, such as tabu algorithm, and a post-optimization phase to improve solutions from the second phase by using a constraint optimization procedure. Computational results show that the proposed search strategy and the model are highly efficient. Optimal solutions are always obtained for small or medium sized problems. For large sized problems, the final results are on average within 5.77% to 7.48% of the lower bounds.**

Index Terms: **3G universal mobile telecommunications system (UMTS), constraint programming, local search, mobile network design.**

I. INTRODUCTION

The third generation mobile system, e.g., universal mobile telecommunications system (UMTS), will provide a wide variety of sophisticated services, over the wide service area, and will play an important role in future telecommunications. From the viewpoint of the mobile users, the success of UMTS systems [1] will rely on the quality of service provided. From the service providers' perspective, the aim will be to provide the good services in the most economical manner. Effective design of UMTS networks has a significant impact on the investment costs and the quality of service. The UMTS radio access network (UTRAN), as illustrated in Fig. 1, consists of one or more radio network sub-system (RNS). An RNS is a subnetwork within UTRAN and consists of one radio network controller (RNC) and one or more node Bs. The node B is effectively a UMTS base station, while an RNC is comparable with a GSM base station controller (BSC). Each RNC is connected to the core network by the lu interface; RNCs are connected together with the lur interface. RNCs have the task of managing the radio channels of the connected node Bs, and concentrating or relaying their traffic to the core network. Each node B is connected to an RNC by the lub interface. There are some fundamental limits on the number of node Bs that can be supported by each RNC [2]. In a UMTS network, the area of coverage is geographically divided into cells. Each cell uses a node B for communications among mobile users. A handover occurs when a mobile user is in the overlapping cell coverage area and carries out the required updates. When a handover occurs between two cells linked to the same RNC, it is called a simple handover, because there are few necessary updates. On the other hand, a complex handover describes a handover between two cells related to different RNCs because, in this case, the update procedures consume more resources than in the case of a simple handover. Note that the concepts of node Bs and cells are used interchangeably in this paper.

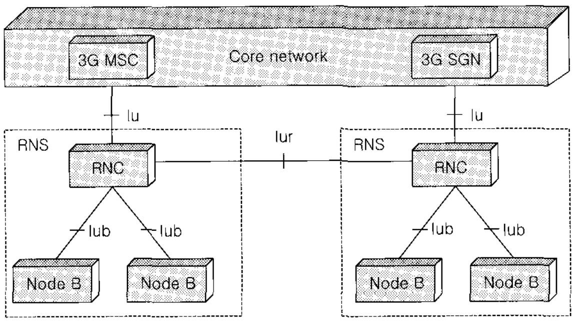


Fig. 1. UTRAN architecture.

If the handover frequency between adjacent cells is very high, it is then reasonable to keep these cells in the same RNS, and vice versa. At the same time, a further cost efficiency can be achieved by using minimum number of RNCs. Reducing the number of RNCs can not only reduce the investment on RNCs, but also the cost on implementation of lur interfaces. Both problems can be integrated in a single model. Thus the model could be summarized as follows. Given a set of node Bs and the potential locations of RNCs, the objective is to find a set of RNCs and their locations, as well as assigning node Bs to RNCs in such a way that minimizes an objective function and satisfies a set of constraints. The objective function is composed of a physical link related component and a handover related cost component. The assigning node Bs must take into account RNC's capacity constraints, which limit the number of node Bs and total traffic an RNC can handle.

We will build a constraint-based model for this problem. One of the key ideas behind the constraint-based model is the use of constrained variables with a larger domain, while the corresponding integer programming model for this problem is the use of a 0–1 binary variable [3]. Although the two models serve

Manuscript received February 13, 2003; approved for publication by Masakazu Sengoku, Division II Editor, April 24, 2005.

The authors are with the Mobile Computing and Networking Research Laboratory (LARIM), the Department of Computer Engineering, Ecole Polytechnique of Montreal, Canada, email: {yufei.wu, samuel.pierre}@polymtl.ca.

the same purpose, based on the complexity analysis for the two models, the constraint-based model is found to be more efficient for our problem. To solve the constraint-based model efficiently, we propose a hybrid search strategy which consists of three phases. The first phase is to find a good initial solution by using the constraint satisfaction method [4]. The second phase uses local search algorithms [5], to improve the solutions from the first phase. The third is called post optimization phase. The best objective function value obtained in local search is used as the upper bound. Thus, local search provides a method for pruning the search space. We use a standard constraint programming search procedure to post optimize the solutions. It is believed that the hybrid search method can be applied to a wide range of NP-hard problems [6] which are subject to intrinsic combinatorial growth.

Similar models have been proposed in the literature. Merchant and Sengupta [7] propose an integer programming model for cell assignment problem in PCS network, which is solved by a proposed heuristic algorithm. Pierre and Houeto [8] approach the same cell assignment problem by using tabu search. Amaldi *et al.* [9], [10] propose discrete optimization models and algorithms aimed at supporting the decisions in the process of planning where to locate new node Bs. These models consider the signal-to-interference ratio as quality measure and capture at different levels of detail the signal quality requirements and the specific power control mechanism of the WCDMA air interface. Demirkol *et al.* [11] propose a mobile location area planning problem as a discrete optimization problem, which is efficiently solved by a heuristic based on simulated annealing technique. The constraint-based model proposed in this paper differs from previous research in that it is not a 3G radio part problem as in [9] and [10] and a location area planning problem as in [11], although there are some similarities in problem formulations and solution processes. The constraint-based model also differs from the models proposed in [7] and [8], because it sites RNCs and assigns node Bs simultaneously while the models in [7] and [8] do not.

This paper is organized as follows. In Section II, we present a constraint-based design model. Based on the complexity analysis, the constraint-based design model stands out as the better choice. In Section III, we propose a hybrid search strategy. The first subsection of Section III illustrates the constraint satisfaction method to generate good initial feasible solutions for our problem; the second subsection is dedicated to the description of local search algorithms; the third subsection lists the post optimization procedure. In Section IV, we present the detailed results of an extensive experimental evaluation.

II. PROBLEM FORMULATION

In this section, we present a constraint-based formulation, which define the constraint-based decision variables, design constraints, and the objective function for mobile network design, and formulate the task as a discrete optimization problem.

A. Decision Variables

In this constraint-based model, we create several arrays of integer variables. These arrays of integer variables are extensible. They can also be organized into matrices. The constrained decision variable array assign is used to store the number of the RNC to which each cell is assigned. The array assign contains one element for each cell. The value of a variable from the array assign indicates the RNC that serves the cell corresponding to that element. The 0–1 binary variable array open is used to store the values that indicate whether an RNC is installed or not. The numerical array *linkcosts* is used to store the monthly leasing cost of establishing a physical link from the chosen RNC. The numerical array *handcosts* is used to store the handover cost between cells, while the numerical array *installcosts* represents the monthly amortization cost for each open RNC.

B. Design Constraints

Let n be the number of node Bs to be assigned to m potential locations of RNCs. The locations of node Bs and potential locations for installing RNCs are fixed and known. Let t_{ij} take the value 1 if node Bs i and j are both connected to the same RNC and the value 0 if node Bs i and j are connected to different RNCs.

$$t_{ij} = (assign[i] == assign[j]) \text{ for } i,\ j = 1, \cdots, n. \quad (1)$$

The variable array t_{ij} is introduced as intermediate variable array in the constraint-based formulation for the purpose of convenience. The logical operator $==$ is used to compare the values of $assign[i]$ and $assign[j]$. Let t_{ij} take 1 if both values are equal, and 0 otherwise. The constraint-based model uses variables of the same form $open[j]$ to specify whether an RNC is installed at site j. However, it is not possible to express the problem constraints as linear constraints with this choice of variables, and features from constraint programming in obtaining concise and clear models are used instead. An important feature is to use variables or expressions involving variables to index arrays [12]. Thus, constraints describing that RNC k must be installed if it serves the cell i can be expressed as

$$open[assign[i]] = 1 \text{ for } i = 1, \cdots, n. \quad (2)$$

If λ_i denotes the number of calls per time unit destined to cell i, the limited traffic-handling capacity M_k of RNC k imposes the following constraints.

$$\sum_{i=1}^{n} \lambda_i (assign[i] == k) \leq M_k open[k] \text{ for } k = 1, \cdots, m. \quad (3)$$

According to constraints (3), the total load of all cells, which are assigned to RNC k is less than the traffic handling capacity M_k of the RNC. The constraints (4) say that the total number of connections to RNC k can not exceed its total number of ports N_k.

$$\sum_{i=1}^{n} (assign[i] == k) \leq N_k open[k] \text{ for } k = 1, \cdots, m. \quad (4)$$

C. Objective Function

The objective function establishes a relationship between the objective function f itself and other constrained variables and is represented in (5). The first term of the objective function uses the variable $assign[i]$ to index the array *linkcosts*. Because our purpose is to minimize the objective function, each modification of this objective function propagates its effects to the other variables, and any modification of a constrained variable propagates to the objective function as well. These two essential processes eliminate parts of the search space that won't produce better solutions [4]. Our 3G mobile network design problem can be formulated as a discrete optimization problem.

$$\begin{aligned}\text{Minimize } f \; = \; & \sum_{i=1}^{n} \mathit{linkcost}[i, assign[i]] \\ & + \sum_{k=1}^{m} \mathit{open}[k]\mathit{installcost}[k] \\ & + \sum_{i=1}^{n}\sum_{j=1}^{n}(1 - t_{ij})\mathit{handcost}[i,j]. \qquad (5)\end{aligned}$$

Subject to the constraints (1)–(4).

D. Comparison with Mathematical Programming Model

We formulate a constraint-based model for this problem. For the comparative purpose, the corresponding mathematical integer programming decision variables are also presented as follows.

$$x_{ik} = \begin{cases} 1, & \text{if cell } i \text{ assigned to RNC } k \\ 0, & \text{otherwise.} \end{cases} \qquad (6)$$

$$y_k = \begin{cases} 1, & \text{if an RNC installed at } k \\ 0, & \text{otherwise.} \end{cases} \qquad (7)$$

The data representations for two models are the same. The basic principle of the integer programming model for this problem is the use of 0–1 binary variables. In contrast, one of the key ideas behind the constraint programming model is the use of constrained variables with a larger domain. To evaluate the combinatorial complexity of the mathematical integer model, we consider the maximum number of configurations of the model. If the search space were represented as a binary tree, that is, the worst case complexity of a generate and test algorithm, assume nm binary variables x_{ik} with domain $[0,1]$ such that $x_{ik} = 1$ if cell i connects to RNC k, where n and m represent the number of node Bs and potential locations for installing RNCs, respectively. The maximum number of configurations is 2^{nm}.

The proposed constraint-based model reduces the solution search space significantly, because it uses fewer decision variables by using constraint programming formulation. In constraint programming, the decision variables are typically represented by constrained variables, which are practical ways of cutting down the number of decision variables and thus reducing the size of the model and its search space [13]. Because the size of search space is polynomial to the domain size of constrained variables, but exponential to the number of constrained variables, we can reduce the complexity of our problem as follows. Create n constrained integer variables $assign[i]$ with domain $[1, m]$, so that $assign[i] = j$ if cell i connects RNC j. In this constraint-based model, the maximum number of solutions is m^n. For example, if $n = m = 10$, the mathematical integer programming model has a maximum solution space of 2^{100}, which is around 10^{30}, while the constraint-based model has a maximum solution space of 10^{10}. This value is much smaller than 10^{30} of the mathematical integer model.

III. HYBRID SEARCH STRATEGY

The proposed hybrid search strategy consists of three stages. In the first stage, a good feasible solution to the problem is found by using constraint satisfaction method [12] embedded with a problem-specific search guidance. After the solution with a specific objective function value can be found, the second stage is to apply a good local search procedure, such as tabu search, simulated annealing, or greedy algorithm, to improve this solution. The last stage is to make an improvement to the solution derived from the second stage. The best objective function value obtained from the second stage is used as the upper bound, then a constraint optimization procedure is applied to improve the solution.

A. Constraint Satisfaction

With regards to optimization, constraint satisfaction methods are often applied in situations where a quick feasible solution to a problem is required, as opposed to finding a provably optimal solution, or a feasible solution is hard to find, such as a nonlinear combinational integer problem. If our problem is regarded as a constraint satisfaction problem, the problem can be defined formally by using some of the terminology of mathematical programming. Given a set of n decision variables $x_1, x_2, \cdots, x_n$, the set D_j of allowable values for each decision variable $x_j, j = 1, \cdots, n$, is called the domain of the variable x_j. The domain of a decision variable can be any possible set, operating over any possible set of symbols. We can define a constraint as a mathematical function $f : D_1 \times D_2 \times \cdots \times D_n \rightarrow \{0,1\}$ such that $f(x_1, x_2, \cdots, x_n) = 1$ if and only if $f(x_1, x_2, \cdots, x_n) = 1$ is satisfied. Using this functional notation, we can then define a constraint satisfaction problem as follows. Given n domains $D_1, D_2, \cdots, D_n$ and $f_1, f_2, \cdots, f_n$, find $x_1, x_2, \cdots, x_n$ such that $f(x_1, x_2, \cdots, x_n) = 1$ and $x_j \in D_j$ for all $1 \leq k \leq m$, $1 \leq j \leq n$. Note that a constraint satisfaction problem is only a feasibility problem, and that no objective function is defined. A solution to a constraint satisfaction problem is simply a set of values of the variables such that the values are in the domains of the variables, and all of the constraints are satisfied.

Consider our problem of assigning node Bs to RNCs, while simultaneously deciding which RNCs to open, subject to a number of constraints. This is an NP-hard combinatorial optimization problem. In order to have an initial feasible solution quickly, it is desirable to formulate the problem as a constraint satisfaction problem. Thus, an initial solution may be found by using the constraint propagation and domain reduction algorithm [13]. However, in many cases, simple propagation of constraints to reduce the domains of variables is not sufficient to get a unique value for each of the variables [12]. In other words, propagation and reduction may not be enough to find a unique solution to the problem.

There are some methods that are applicable in such a situation to search for a good feasible solution to a problem. A common search method in constraint programming performs as follows [14]. First, takes arrays of constrained variables; then, as long as there exists any unknown variables, choose one of those variables; choose a value to assign to this variable; propagate the effects of that binding. However, there is a hidden problem. In general, we don't know which value in the domain is consistent with the constraints. If that binding leads to inconsistencies, it must be undone and another value must be tried. This activity of undoing a trial and returning to a previous state to make another effort is called as backtracking. It is obvious that the binding of a variable blindly is not efficient and flexible. In order to efficiently search for a solution, it is necessary to have a good search strategy which normally can eliminate a great many possibilities sooner. In this paper, a good search strategy is to choose first the variable with the smallest domain. Therefore, the hybrid search strategy for a solution to our problem favors node Bs located near an RNC since the cost of serving them would be minimal. The capacity of an RNC is deduced automatically from the number of node Bs associated with it. Each node B has a set of RNCs still available for it, with associated serving costs. The proposed strategy chooses a node B with the smallest number of available open RNCs (an open RNC is available if it has sufficient remaining capacity to satisfy the node B's demand). Once a node B has been selected, the problem is to assign to it the RNC corresponding to the least possible cost. It is implemented as follows. First, assign the least possible cost to the constrained variable linkcosts. The constraints of the problem restrict the set of RNCs to the one with the least associated costs. If this choice fails, the chosen RNC is removed from the domain, and the second least cost possible RNC is chosen, and execution proceeds. It is clear that the solution found by this search strategy often has the property of local minimum.

B. Local Search Optimization

We now present the second stage of the hybrid search strategy. This stage is based on local search algorithms, such as greedy algorithm, tabu search or simulated annealing. Local search is a mechanism for searching for improvements to the solution of a problem by making small changes to the current solution. This new solution is tested against the problem constraints for feasibility, and its objective function cost is computed. If the new solution is feasible and has reduced cost, it is accepted as the current one, otherwise the current solution remains unchanged.

An important concept in local search algorithms is the neighborhood definition. A subset of the solution space S of a problem, designated as $N(x)$, may be associated with each point $x \in S$. $N(x)$ is referred as the neighborhood of x. At each step, most search algorithms for optimization proceed from the current point x by considering a point or a set of points in $N(x)$. Local search algorithms are then differentiated from each other by the way in which the considered points are accepted. In this paper, each point in the solution space consists of a set of RNCs and an assignment pattern of node Bs to RNCs. Simple local search algorithms such as greedy algorithms move from point to neighboring point, accepting each successive point as a solution only if it has a lower cost than the current solution. This causes entrapment in the point with the lowest objective function value in the neighborhood—a local optimum. In order to escape from a local optimum, local search algorithms based on meta-heuristics, such as tabu search or simulated annealing, are commonly used.

The idea of tabu search [5] is to explore the search space through a sequence of moves. At each iteration of the algorithm, a solution is selected as the best one, even if the selected solution has the degrading cost. This means that, in a local minimum, tabu search will move out, since it can accept cost degrading moves. A set of moves is applied on this solution, and hence, a number of neighbouring solutions are obtained. A subset of these solutions (moves) is classified as forbidden, which is called the tabu tenure. The tabu tenure holds information on the solutions recently visited and forbids the search from moving back to a solution which is the same as (or shares some features with) one it has recently visited. The length of the tabu tenure is usually important to the performance of tabu search. Moreover, a subset of the tabu moves may be overridden, according to the aspiration criteria, which can maintain some promising solutions.

C. Post Optimization and Lower Bounds

The solutions from local search algorithms are not guaranteed to be optimal. This means that the solutions can still be improved somehow. Generally speaking, there are two ways for the post-optimization to make improvements. The first is to optimize any objective or to enhance constraint satisfaction in case of constraint violation by fine tuning some parameters of the problem. The second is to use the best objective function value obtained in a local search algorithm as the upper bound. In this case, the local search algorithm provides a method for pruning the search space used by a follow-up search method which is used to improve the solution. By running this search method, such as branch-and-bound, for certain amount of time to improve the solution obtained from the local search algorithm, the intermediate results can be seen as improvements.

In this paper, when the solutions from a local search algorithm are obtained, the best solution is used as the upper bound. A standard constraint programming search procedure [12] is used to search for an improvement. This standard search procedure is described as follows. Let x represent a feasible point in the search space, and $f(x)$ its corresponding objective function. The search space can then be pruned by adding the constraint $f(y) > f(x)$ to the problem, and continuing the search. The constraint that is added specifies that any new feasible point must have a better objective value than the current point. Propagation of this constraint may cause the domains of the decision variables to be reduced, thus reducing the size of the search space. As the search progresses, new points will have progressively better objective values. The procedure concludes until no feasible point is found. When this happens, the last feasible point can be taken as the optimal solution. If this does not happen for a long period of time, an intermediate solution can be obtained as an improvement by interrupting the search process. If the hybrid search strategy does not find a global optimum, it is necessary to find a lower bound to evaluate the quality of the solution found. In our case, the lower bound is obtained by using the CPLEX linear programming optimizer [15] to solve our

Table 1. Test cases used to execute the plan of experiments.

Test cases	No. of potential locations for RNCs	No. of node Bs	No. of sample tests
1	20	50	30
2	30	100	30
3	40	150	30
4	50	200	30
5	80	300	30

problem without considering the integrality constraints.

IV. EXPERIMENTAL RESULTS

A. Test Cases

Computational experiments are carried out on 5 sets of test cases, with the number of cells varying between 50 and 300 and the number of RNCs between 20 and 80. Each test case contains 30 test samples as shown in Table 1. These test case sizes are larger than the ones in [7] and [8], and difficult enough to evaluate the complexity of the problem. These test cases are generated by a MATLAB program by supposing that the cells are arranged on a hexagonal grid of almost equal length and width. The node B 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 locations where RNCs could be potentially located are generated randomly with a uniform distribution. It is assumed that a RNC, with a single cabinet supporting up to 180 node Bs, 310 Mbps of raw data throughput, is used in our simulation. It is also assumed that all potential locations have the same monthly amortization building cost, that is, *installcosts*$[k] = \$1,000$. The cost of connecting node B to RNC, denoted by the matrix *linkcosts*, is assumed to be proportional to the distance between them. We take a proportionality coefficient equal to the unit. The *handcosts* can be calculated by using the following method. The call rate γ_i of cell i follows a gamma law of average and variance equal to the unit. The call durations inside cells are distributed according to an exponential law of parameter equal to 1. If cell j has w neighbors, the $(0,1)$ interval is divided into $w+1$ sub-intervals by choosing w random numbers distributed evenly between 0 and 1. At the end of the service period in cell j, the call could be either transferred to the i-th neighbour $(i = 1, \cdots, w)$ with a handover probability γ_{ij} equal to the length of the $(w+1)$-th interval. To find the call volumes and the rates of coherent handovers, the cells are considered as $M/M/1$ queues forming a Jackson's network [16]. The incoming rates α_i in cells are obtained by solving the following system.

$$\alpha_i - \sum_{j=1}^{n} \alpha_j \gamma_{ji} = \gamma_i \quad \text{for } i = 1, \cdots, n. \tag{8}$$

If the incoming rate α_i is greater than the service rate, the distribution is rejected and chosen again. The handover rate is defined as

$$h_{ij} = \lambda_j \gamma_{ij} \quad \text{for } i = 1, \cdots, n. \tag{9}$$

Table 2. Average performance of post-optimization and relative distance between final solutions and optima or lower bounds.

Test cases	Improvement from post-optimization process (%)	Distance between final solutions and optima* or lower bounds (%)
1	0.68	0.00*
2	0.81	0.00*
3	5.40	0.00*
4	5.91	5.77
5	6.23	7.48

B. Performance of the Hybrid Search Strategy

From the previous experiments [18], we conclude that tabu search performs best and is more stable than simulated annealing and greedy algorithms. Thus, the best objective function value from tabu search is set as the upper bound for the constraint programming optimization procedure. The constraint programming optimization procedure provided by ILOG Solver [17] is run over the solutions immediately from tabu search. During the process of searching for improvements, we set limits of 30 minutes on the amount of time spending on each sample test run. For each of five test cases, 30 sample tests are run. When it reaches that time limit, the solution returned from the search is regarded as the best solution. In order to evaluate how good the best solution is, it is necessary to compare it to the optimal solution or a lower bound. Through the post optimization process, optimal solutions are obtained for test cases 1, 2, and 3 for every single sample test, while no optimal solution could be obtained for every sample test of test case 4 and 5 within 12 hours. It is difficult to evaluate good lower bounds for larger sized instances. Such an evaluation becomes time-consuming as the problem instances increase in size. In this paper, lower bounds are obtained by using the CPLEX linear programming optimizers [15] by relaxing the integrality constraints on the decision variables x_{ik} and y_k from the corresponding mathematical programming formulation. Table 2 shows the average results from this post-optimization process.

Because this problem is NP-hard, this implies that it is not guaranteed to have a feasible solution in a reasonable time. Mathematical integer programming method has been applied to the simplified version of our problem [7]. Numerical results show that the mathematical integer programming method fails to find any solution after the problem sizes goes beyond 35 cells, which is much smaller than our test cases in Table 1. The results also show that when the mathematical integer programming method works, the CPU time requirements is high, while the heuristic method in [7] requires very little time. The simulation results from a similar problem [19] also illustrate that for relatively larger instances, the mathematical integer programming method could not find a feasible solution within days, while a simulated annealing algorithm is able to quickly find near-optimal solutions for all large instances. Thus, our focus is to compare the performance of the hybrid search strategy with other existing heuristics in literature.

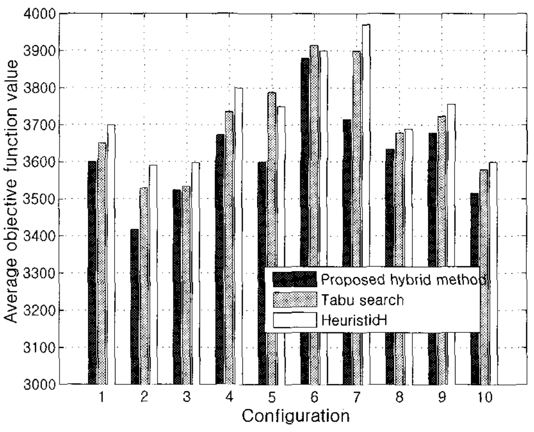


Fig. 2. 100 cells, 3 RNCs.

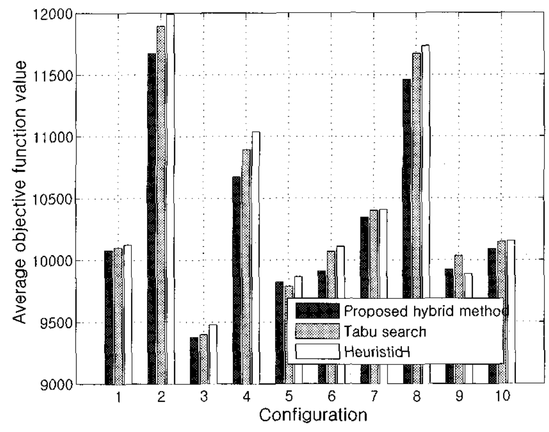


Fig. 4. 300 cells, 5 RNCs.

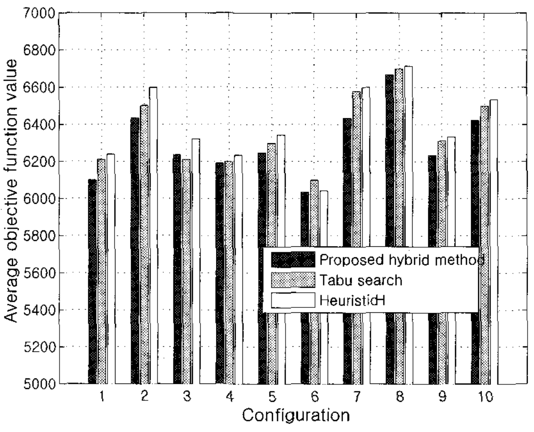


Fig. 3. 200 cells, 4 RNCs.

Table 3. Average improvement of the proposed hybrid search strategy over tabu search and heuristic H.

	100 cells, 3 RNCs (%)	200 cells, 4 RNCs (%)	300 cells, 5 RNCs (%)
Tabu search	1.61	1.73	2.16
Heuristic H	2.22	2.74	3.95

C. Comparison with Other Heuristics

In order to evaluate the performance of the hybrid search strategy, two types of experiments are performed. One set of experiments is to compare it with the heuristics used in the special case of our problem, which is the cell assignment problem [7], [8], while the other set is to compare it with local search methods with randomly generated initial input solutions of our problem. The most important measurement criterion is the objective function value.

In [7], the authors design a heuristic, which we call H, for solving the cell assignment problem. The cell assignment problem is a simplified version of our problem, because it assumes that the number and locations of RNCs are predefined. In [8], authors use tabu search for solving the same cell assignment problem. We compare tabu search and heuristic H with our proposed hybrid search strategy. These three methods are executed over a set of test cases with three different numbers of cells (100, 200, and 300) and RNCs (3, 4, and 5) that means the search space size is between 3100 and 5300. Varying the RNCs' geographical location by maintaining the number of cells and the RNCs fixed, we obtain a different configuration. For each configuration, 30 sample test runs are experimented. In order to make comparison fairly, for each sample test run, 10,000 function evaluations are performed for each heuristic. For each of the three test cases, we analyze 10 different configurations. Figs. 2–4 show that in most of the test configurations, the hybrid constraint-based search strategy yields better than tabu search and H, sometimes with big differences. Table 3 summarizes the average improvement results. Nevertheless, given the initial large investment, the handoff and the annual maintenance costs for large-sized mobile networks (in the order of hundreds of millions of dollars), these small improvements translate into a large cost saving over a long-period of time. In terms of CPU time, for a large number of cells, tabu search is a bit faster than heuristic H. Conversely, for smaller size problems, tabu search is a bit slower. The hybrid search strategy is slower than both heuristics H and tabu search. However, this is not an important factor because designing a mobile network is not a time-critical mission.

In order to further examine the effectiveness of the proposed hybrid search strategy, we compare the solutions from it with those obtained by some local search algorithms. In these experiments, the initial solutions for these local search algorithms are generated randomly with a uniform distribution. However, the proposed hybrid search strategy uses the constraint satisfaction method to obtain a good initial feasible solution as the input for the immediate local search algorithm. Fig. 5 illustrates the average results over the sample test runs on the five test cases in Table 1. In order to make fair comparisons, for each algorithm, each test case is run 30 sample tests for 2,000 solution evaluations for each sample test run. It is clear that the hybrid search strategy is best-suited for this problem.

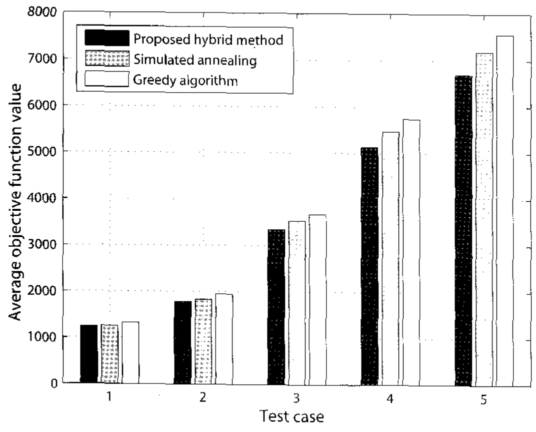


Fig. 5. Comparison between the hybrid search strategy, simulated annealing and greedy algorithm.

V. CONCLUSION

In this paper, we have proposed a new constraint-based 3G UMTS radio network design model. This constraint-based model is more efficient in terms of the combinatorial complexity when compared to that of the traditional mathematical model. We have also proposed a hybrid search strategy. Computational results show that the hybrid search strategy is more efficient than the existing heuristics. Optimal solutions are always obtained for small or medium sized problems. For large sized problems, the final results are on average within 5.77% to 7.48% of the lower bounds. It is clear that the hybrid search strategy is better-suited for this problem.

REFERENCES

[1] K. W. Richardson, "UMTS overview," *IEEE Electron. Commun. Eng. J.*, vol. 12, no. 3, pp. 93–100, 2000.

[2] H. Holma and A. Toskala, *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*, 2nd ed. John Wiley & Sons, Ltd., 2002.

[3] G. L. Nemhauser and L. A. Worlsy, *Integer and Combinatorial Optimization*, Wiley, 1988.

[4] M. Wallace, *Survey: Practical Applications of Constraint Programming*, William Penney Laboratory, Imperial College, London, Sept. 1995.

[5] E. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*, Wiley, 1997.

[6] M. R. Garey and D. S. Johson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, NY, 1979.

[7] A. Merchant and B. Sengupta, "Assignment of cells to switches in PCS networks," *IEEE/ACM Trans. Networking*, vol. 3, no. 5, pp. 521–526, 1995.

[8] S. Pierre and F. Houeto, "A tabu search approach for assigning cells to switches in cellular mobile networks," *Computer Commun.*, 25, pp. 464–477, 2002.

[9] E. Amaldi, A. Capone, and F. Malucelli, "Discrete models and algorithms for the capacitated location problem arising in UMTS network planning," in *Proc. ACM Dial-M*, 2001, pp. 1–8.

[10] E. Amaldi, A. Capone, and F. Malucelli, "Planning UMTS base station location: Optimization models with power control and algorithms," *IEEE Trans. Wireless Commun.*, vol. 2, no. 5, pp. 939–952, Sept. 2003.

[11] I. Demirkol, C. Ersoy, M. Caglayan, and H. Delic, "Location area planning in cellular networks using simulated annealing," in *Proc. INFOCOM 2001*, 2001.

[12] P. Van Hentenryck, "Search and strategies in OPL," *ACM Trans. Computational Logic*, vol. 1, no. 2, pp. 282–315, Oct. 2000.

[13] K. Marriott, *Programming with Constraints: An Introduction*, MIT Press, Cambridge, MA, 1998.

[14] I. J. Lustig and J. F. Puget, "Program does not equal program: Constraint programming and its relationship to mathematical programming," *Interfaces*, vol. 31, no. 6, pp. 29–53, 2001.

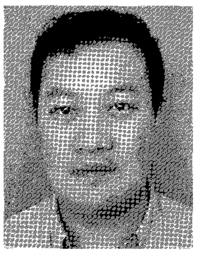
[15] ILOG CPLEX 7.1 User's Manual.

[16] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*, Wiley, New York, 1985.

[17] ILOG Solver 5.1 User's Manual.

[18] Y. Wu and S. Pierre, "Optimization of access network design in 3G networks," in *Proc. IEEE CCECE 2003*, 2003, pp. 781–784.

[19] R. Mathar and T. Niessen, "Optimum positioning of base stations for cellular radio networks," *Wireless Networks*, pp. 421–428, 2000.



Yufei Wu received the B.S. and M.Sc. degrees from Nanjing University, Nanjing, China, in 1988 and 1990, respectively, and the MBA and Ph.D. in computer engineering from McGill University and Ecole Polytechinique de Montreal, Montreal, Canada, in 1995 and 2004, respectively. He is a postdoctoral research fellow with the department of electrical engineering , Ecole Polytechinique de Montreal, Canada. His interests include mobile computing and networking.

Samuel Pierre received the B.Eng. degree in civil engineering in 1981 from École Polytechnique de Montréal, Québec, the B.Sc. and M.Sc. degrees 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 École Polytechnique de Montréal. Dr. Pierre is currently a Professor of Computer Engineering at École Polytechnique de Montréal where he is Director of the Mobile Computing and Networking Research Laboratory (LARIM) and NSERC/Ericsson Industrial Research Chair in Next-generation Mobile Networking Systems. He is the author of four books, co-author of two books and seven book chapters, as well as over 250 other technical publications including journal and proceedings papers. His research interests include wireline and wireless networks, mobile computing, performance evaluation, artificial intelligence, and electronic learning. He is a Fellow of Engineering Institute of Canada, senior member of IEEE, a member of ACM and IEEE Communications Society. He is a Fellow of Engineering Institute of Canada, senior member of IEEE, a member of ACM and IEEE Communications Society. He is an Associate Editor of IEEE Communications Letters and IEEE Canadian Review, a Regional Editor of Journal of Computer Science, and he serves on the editorial board of Telematics and Informatics published by Elsevier Science.