Frequency Assignment for Cellular Mobile Systems Using Constraint Satisfaction Techniques

Makoto Yokoo† and Katsutoshi Hirayama‡

† NTT Communication Science Laboratories
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237 Japan
e-mail: yokoo@cslab.kecl.ntt.co.jp, http://www.kecl.ntt.co.jp/csl/ccrg/members/yokoo/

‡ Kobe University of Mercantile Marine
5-1-1 Fukae-minami-machi, Higashinada-ku, Kobe 658-0022, Japan
e-mail: hirayama@ti.kshosen.ac.jp, http://jos2.ti.kshosen.ac.jp/~hirayama/

Abstract - This paper presents a new algorithm for solving frequency assignment problems in cellular mobile systems using constraint satisfaction techniques. The characteristics of this algorithm are as follows: 1) instead of representing each call in a cell (a unit area in providing communication services) as a variable, we represent a cell (which has multiple calls) as a variable that has a very large domain, and determine a variable value step by step, 2) a powerful cell-ordering heuristic is introduced, 3) a branch-and-bound search that incorporates forward-checking is performed, and 4) the limited discrepancy search is introduced to improve the chance of finding a solution in a limited amount of search. Experimental evaluations using standard benchmark problems show that this algorithm can find optimal or semi-optimal solutions for these problems, and most of the obtained solutions are better than or equivalent to those of existing methods using simulated annealing, tabu search, or neural networks. These results show that state-of-the-art constraint satisfaction/optimization techniques are capable of solving realistic application problems when equipped with an appropriate problem representation and heuristics.

I. Introduction

With growth in the demand of mobile telephone services, the efficient use of available spectrums is becoming increasingly important. The studies of a frequency assignment problem (also called a channel assignment problem) in cellular mobile systems have a long history [1], [6], [7], [14], [15]. Various AI techniques, including constraint satisfaction, simulated annealing, neural networks, tabu search, and GA, have been applied to this problem [2], [4], [5], [8], [11]–[13], [16].

An overview of a frequency assignment problem is as follows. There exists a set of geographically divided, typically hexagonal regions called cells. Frequencies (channels) must be assigned to each cell according to the number of call requests. There exist following three types of electro-magnetic separation constraints.

- co-channel constraint: the same frequency cannot be assigned to pairs of the cells that are geographically close to each other.
- adjacent channel constraint: similar frequencies cannot be simultaneously assigned to adjacent cells.
- co-site constraint: any pair of frequencies assigned to the same cell must have a certain separation.

The goal is to find a frequency assignment that satisfies the above constraints using a minimum number of frequencies (more precisely, using the minimum span of the frequencies).

It must be noted that there exist several variations of frequency assignment problems. The benchmark problems provided by the EUCLID-project Combinatorial ALgorithms for Military Applications (CALMA) project are well-known in the constraint satisfaction/optimization research community[1]. This type of problem arises from a military application, and geographical information including cells is not described in the problem specification. Constraint satisfaction/optimization techniques can solve this type of problem quite efficiently.

On the other hand, according to [16], constraint satisfaction techniques are not very effective for solving frequency assignment problems discussed in this paper. The most straightforward way for solving such problems using constraint satisfaction techniques would be to represent each call as a variable (whose domain is available frequencies), then to solve the problem as a generalized graph-coloring problem. However, solving real-life, large-scale problems using this simple formulation seems rather difficult without avoiding the symmetries between calls within one cell. In our new algorithm, instead of representing each call as a variable, we represent a cell as a variable that has a very large domain. Furthermore, we determine the variable value step by step instead of determining a variable value at one time.

A standard method for solving constraint optimization problems, such as partial constraint satisfaction problems [3], is a depth-first branch-and-bound

[1]The benchmark problems and technical reports are available from ftp://ftp.win.tue.nl/pub/techreports/CALMA/

search algorithm. In a branch-and-bound search, we usually assume the existence of a heuristic function that evaluates a node in a search tree. This function estimates the quality of the solutions that exist under the node in the search tree. However, creating a good heuristic function (i.e., estimating the number of required frequencies) is rather difficult in frequency assignment problems. To solve problems efficiently without a good heuristic function, we use limited discrepancy search techniques [10], [17] so that the algorithm can limit the search efforts to only the part of the search tree where a solution is likely to exist.

In this paper, we show the formal definition of the problem in Section II, then describe the algorithm that utilizes constraint satisfaction techniques in Section III. Furthermore, we show experimental evaluations using standard benchmark problems in Section IV, and discuss the relation to other research in Section V.

II. Frequency Assignment Problem

A frequency assignment problem is formalized as follows. We follow the formalization used in [5], [6], [11], [13], [15], [16]. Frequencies are represented by positive integers $1, 2, 3, \ldots$.

Given: N: the number of cells

d_i, $1 \leq i \leq N$: the number of requested calls (demands) in cell i

c_{ij}, $1 \leq i, j \leq N$: the frequency separation required between a call in cell i and a call in cell j

Find: f_{ik}, $1 \leq i \leq N$, $1 \leq k \leq d_i$: the frequency assigned to the kth call in cell i.

such that,
subject to the separation constraints,
$| f_{ik} - f_{jl} | \geq c_{ij}$, for all i, j, k, l except for $i = j$ and $k = l$,

minimize
$max f_{ik}$ for all i, k.

These constraints can be represented as an $N \times N$ symmetric compatibility matrix C. In addition, a set of requested calls can be represented by an N-element demand vector D.

Example 1. The number of cells is $N = 4$,

$$C = \begin{pmatrix} 5 & 3 & 0 & 0 \\ 3 & 5 & 0 & 2 \\ 0 & 0 & 5 & 1 \\ 0 & 2 & 1 & 5 \end{pmatrix}, \quad D = \begin{pmatrix} 2 \\ 1 \\ 2 \\ 3 \end{pmatrix}.$$

Positive integers (frequencies) must be assigned to f_{ik}, such that their maximum is a minimum, subject to the separation constraints C.

In this example, there exist constraints between cell 1 and cell 2, cell 2 and cell 4, cell 3 and cell 4, namely, the required minimum separations are 3, 2, and 1, respectively.

In addition, a frequency assignment problem can be represented using a graph. Figure 1 shows a graph representing a problem in Example 1 (co-site constraints are not described in the graph). In this graph, a vertex represents a cell, and an edge represents a constraint between cells. The number in the vertex represents the number of call requests of the cell (d_i), and the weight of an edge represents a required separation (c_{ij}). This graph representation is called a *macro-graph* [14].

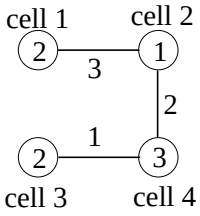

Fig. 1. Example of Macro-graph

III. Algorithm

A. Basic Algorithm

We are going to describe the basic algorithm developed in this paper. In this algorithm, we first fix the number of available frequencies M to a certain upper-bound[2], then find an assignment that satisfies given separation constraints and a set of constraints $f_{ik} \leq M$ for all i, k using a backtracking algorithm. When a solution is found, we set M to $max f_{ik} - 1$, where f_{ik} are frequencies used in the obtained solution, and continue the search process. This algorithm can be considered as a depth-first branch-and-bound algorithm.

In this algorithm, there exists an M-element vector for each cell. An element of this vector represents one frequency, where a value can be "assigned" (the frequency is used at the cell) or "not-assigned" (the frequency is not used). Figure 2 shows an example of these vectors for the problem described in Example 1. A blank element is "not-assigned", and "a" means "assigned". This is an optimal assignment for this problem.

By using this representation, the domain size of a variable becomes 2^M. Since this size will be very large, determining the vector value of a cell at one time is not practical. Therefore, we determine the vector value step by step. We introduce tentative values used during the search process for each vector

[2]Such an upper-bound can be obtained using the heuristic sequential methods described in [15].

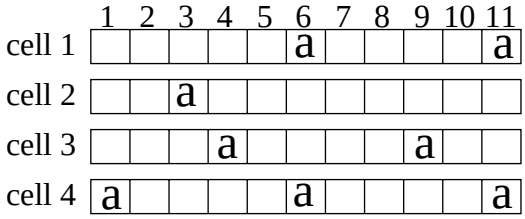

Fig. 2. Example of Cell Representation

element. A vector element can be one of the following three values, "assigned", "free", or "forbidden". The value "assigned" means the frequency is used at the cell, "forbidden" means the frequency cannot be used at the cell due to separation constraints, and "free" means the frequency is available at the cell.

We show the outline of the basic algorithm in Figure 3. In the initial state, all vector elements of a cell are free, and *stack* is empty. This algorithm is basically a depth-first branch-and-bound algorithm that incorporates *forward-checking* [9]. Therefore, as long as the cell-ordering heuristic and the frequency-ordering heuristic are exhaustive, this algorithm can eventually find an optimal solution. The worst-case time complexity is $O(M_{init}^S)$, where M_{init} is the initial upper-bound of frequencies, and $S = \sum_{i=1}^{N} d_i$.

B. Cell-ordering Heuristic

Now, we are going to describe the cell-ordering heuristic used in Step 3 of the main procedure. A commonly used rule of thumb for selecting a variable in constraint satisfaction problems is to select the variable that is most strongly constrained. For example, we can select the variable with the smallest domain after constraint propagation.

A simple extension of this heuristic calculates the average number of free frequencies per one call for each cell. More specifically, let us represent the number of vector elements that are "assigned" for cell i as $assign_i$, and the number of vector elements that are "free" as $free_i$. We calculate the value $free_i/(d_i - assign_i)$ for each cell i (we call this value *average available frequencies*, AAF), and select the cell with the smallest AAF. For example, when selecting a cell in Example 1, AAF is 11/2 for cell 1 and cell 3, 11 for cell 2, 11/3 for cell 4. As a result, cell 4 is selected first.

However, we found that using only the AAF heuristic is not very effective. This is because it tends to select a cell that simply has many demands in the shallow search nodes in the search tree, and does not appropriately consider the strength of the constraints among cells. For example, let us consider the situation after assigning frequencies of three calls of cell 4 (Figure 4). In the figure, "x" means a vector element that becomes "forbidden". In this case, the AAF for

main procedure

1. If the demands are fully satisfied for all cells, then a solution is found, call **procedure reduce-frequency**. Go to Step 1 of this procedure.
2. If a demand cannot be satisfied for a cell, call **procedure backtrack**, go to Step 1 of this procedure.
3. Select *cell* whose demands are not fully satisfied using a cell-ordering heuristic.
4. Select *frequency* for assigning one call of *cell* using a frequency-ordering heuristic, and set the vector element of the cell corresponding to *frequency* to "assigned". Push ("choice", *cell*, *frequency*) to *stack*.
   Propagate constraints, i.e., set the vector elements that interfere with *frequency* of *cell* to "forbidden". Go to Step 1 of this procedure.

procedure backtrack

1. If *stack* is empty, then there is no solution. Finish the algorithm and return *best-solution*.
2. Pop an element (*flag*, *cell*, *frequency*) from *stack*.
3. If *flag*="choice", then set the vector element of the cell corresponding to *frequency* to "forbidden", unpropagate constraints.
   Push ("forbidden", *cell*, *frequency*) to *stack*, and finish this procedure.
4. If *flag*="forbidden", set the vector element of the cell corresponding to *frequency* to "free". Go to Step 1 of this procedure.

procedure reduce-frequency

1. Record the current assignment as *best-solution*.
2. Set *max-frequency* to the maximal frequency used in the current assignment.
3. If no cell uses *max-frequency*, for each cell, set each vector element larger than or equal to *max-frequency* to "forbidden", set M to *max-frequency*-1, finish this procedure.
4. Pop an element (*flag*, *cell*, *frequency*) from *stack*.
5. If *flag*="choice", then set the vector element of the cell corresponding to *frequency* to "free", unpropagate constraints.
   Go to Step 3 of this procedure.
6. If *flag*="forbidden", set the vector element of the cell corresponding to *frequency* to "free". Go to Step 3 of this procedure.

Fig. 3. Basic Algorithm

both cell 2 and cell 3 is 4. However, selecting cell 3 does not help to reduce the search space, since it does not have any constraints between other cells except cell 4, which is already fully assigned.

To take into account the strength of constraints among cells, we invented an evaluation value for cell i called generalized weighted degree (GWD) defined as follows.

$$\sum_j c_{ij} \times (assign_j + 1)$$

In this formula, $assign_j$ is the number of assigned frequencies of another cell j, and c_{ij} is the weight of the edge[3]. We select the cell that has the maximal GWD value. For example, in Figure 4, the GWD for cell 2 is 11, while the GWD for cell 3 is 4. Therefore, we prefer cell 2 to cell 3.

However, we found that using only the GWD heuristic is also not very effective since it does not take into account the demands, and it tends to make poor decisions in the deep search nodes in the search tree. Thus, we decided to combine these two heuristics. Since we prefer a cell with a smaller AAF and a larger GWD, we use the evaluation value obtained by AAF/GWD, and select the cell with a minimal AAF/GWD.

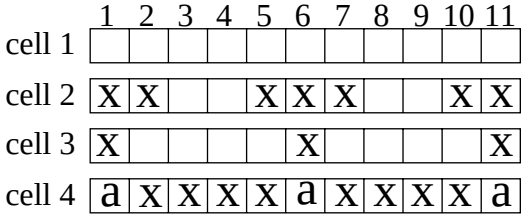

Fig. 4. Cell Status during Search Process

C. Frequency-ordering Heuristic

We are going to describe the frequency-ordering heuristic used in Step 4 in the main procedure. The simplest way would be to select the first (smallest) free frequency (*first-free*). A more sophisticated method would be to consider the impact of selecting a frequency for other cells. Namely, for each frequency, we calculate the number of vector elements of other cells that turn from "free" to "forbidden" by selecting the frequency, and select the one that minimizes this value (*least-impact*).

If the number of frequencies is large, this calculation can be very costly. However, we can reduce the number of possibilities using co-site constraints. For example, let us assume we are selecting the first frequency for cell 4 in Example 1, and the number of possible frequencies is 11. Since there are three demands for cell 4, we cannot assign frequencies larger than 6 for the second call, otherwise there is no free frequency for the third call. Then, if we assign 6 to the second call, we have only one possibility, i.e., 1 for the first call.

We found that the least-impact heuristic is more powerful than the first-free heuristic in finding a solution. On the other hand, the first-free heuristic can find a better solution (a solution with a smaller maximal frequency) if it can find a solution. Therefore, in our algorithm, we first use the first-free heuristic in the *0-discrepancy* search described in the next subsection. When the 0-discrepancy search fails to find a solution, we switch to the least-impact heuristic.

D. Limiting Search Efforts

Although this method uses constraint propagation (forward-checking) to reduce the search space, the search tree of a real-life, large-scale problem is still too large to perform an exhaustive search. We need to limit the search efforts to only the part of the search tree where a solution is likely to exist.

One method for limiting the search efforts is the *limited discrepancy search* [10]. In the limited discrepancy search, the search process initially chooses only the best nodes according to a given heuristic at each decision point in a search tree (this is called 0-discrepancy search). If a solution cannot be obtained by the 0-discrepancy search, the search process is allowed to select a sub-optimal node only once at a decision point (1-discrepancy search). If a solution cannot be obtained, then the search process is allowed to select sub-optimal nodes twice (2-discrepancy search). The number of allowed discrepancies is increased one by one.

In our algorithm, we limit the number of discrepancies for the frequency selection (in Step 4 of the main procedure). More specifically, if a solution cannot be found by selecting the best frequencies according to the frequency-ordering heuristic, the search process is allowed to choose the second-best or the third-best frequencies. The original limited discrepancy search algorithm is developed for searching a binary tree, i.e., the branching factor is two. As shown in [10], there are several alternatives for modifying the limited discrepancy search algorithm to non-binary search tree. In our algorithm, we weight discrepancies depending on the order in the heuristic, i.e., the second-best value is counted as 1-discrepancy, and the third-best value is counted as 2-discrepancy. If the current limit of discrepancies is 2, the algorithm is allowed to choose a third-best value once, or choose the second-best value twice[4].

Furthermore, since the least-impact heuristic is less informative in shallow nodes in the search tree, we use

[3]We add one to $assign_j$, otherwise this value becomes 0 for all cells in the initial state.

[4]Another way is to count all values except the best as 1-discrepancy. This is not practical in our frequency assignment problems since the branching factor of a tree node can be very large.

a modified version of the depth-bounded limited discrepancy search described in [17]. In our algorithm, a discrepancy is allowed only in the nodes whose depth is shallower than or equal to a given depth-limit. In addition, we introduce a modified version of the bounded backtracking described in [17] to allow a quick recovery from mistakes deep in the tree. In the original version of the bounded backtracking, the algorithm is allowed to perform backtracking up to a fixed level. However, in the frequency assignment problems, the branching factor of a node may vary significantly. Therefore, allowing a fixed level of backtracking is inappropriate, since the searched subtree can be too large or too small. Therefore, we set the limit to the total number of backtracking in the subtree. The algorithm is allowed to perform a certain number of backtracking in a subtree. If the number of backtracking exceeds the limit, the subtree is discarded.

In Figure 5, we show an example of the nodes visited in a search tree. We assume that the heuristic prefers left branches, and there exists no solution in this search tree. Furthermore, we assume the depth-limit is two and the limit of the total number of backtracking is one. In the 0-discrepancy search, the algorithm follows the left branches and reaches node 1, which is a dead-end. Since the algorithm is allowed to perform backtracking once, it goes to node 2, which is another dead-end. Since the algorithm reaches the limit of backtracking, it discards the subtree and increases the number of allowed discrepancies. In the 1-discrepancy search, the algorithm first follows the left, then the right branch, and visits node 3 and node 4. Note that discrepancies are allowed only at the nodes within the depth-limit. The algorithm discards the subtree, and visits node 5 and node 6. In the 2-discrepancy search, the algorithm visits node 7 and node 8.

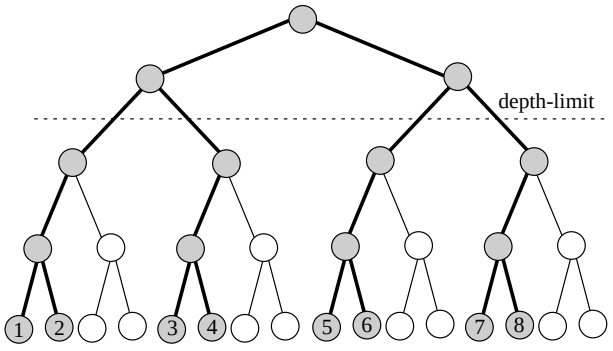

Fig. 5. Limited Discrepancy Search with Bounded Backtracking

IV. Evaluations

We use benchmark problems called Philadelphia problems, which have been used widely in previous researches including [4]–[6], [11], [15], [16]. These problems are formulated based on an area in Philadelphia, Pennsylvania. The network consists of 21 cells as shown in Figure 6.

There are many variations for setting constraints and demands. The parameter settings used in our evaluations are described in Table I. In the table, "Nc" means the square of required distance for co-channel constraints, assuming that the distance between adjacent cells is 1. For example, if Nc=12, while cell 1 and cell 5 can use the same frequency (the distance is 4), cell 1 and cell 4 cannot (the distance is 3). "acc" represents the separation required for adjacent channel constraints, and "c_{ii}" represents co-site constraints. The demand vectors used in the table are as follows (case 3 and case 4 are obtained by multiplying 2 and 4 to case 1, respectively):

case 1: (8 25 8 8 8 15 18 52 77 28 13 15 31 15 36 57 28 8 10 13 8)

case 2: (5 5 5 8 12 25 30 25 30 40 40 45 20 30 25 15 15 30 20 20 25)

case 3: (16 50 16 16 16 30 36 104 154 56 26 30 62 30 72 114 56 16 20 26 16)

case 4: (32 100 32 32 32 60 72 208 308 112 52 60 124 60 144 228 112 32 40 52 32).

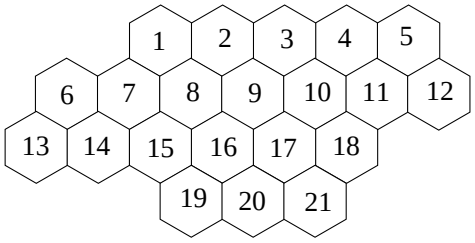

Fig. 6. Cellular Geometry of Philadelphia Problems

TABLE I
Specifications for Philadelphia Problems

Instance	Nc	acc	c_{ii}	Demand Vector
P1	12	2	5	case 1
P2	7	2	5	case 1
P3	12	2	7	case 1
P4	7	2	7	case 1
P5	12	2	5	case 2
P6	7	2	5	case 2
P7	12	2	7	case 2
P8	7	2	7	case 2
P9	12	2	5	case 3
P10	12	2	5	case 4

Table II shows the theoretical lower-bounds reported in [11], [15] and the results obtained with our constraint satisfaction method (CS). In this method,

to finish the algorithm execution within a reasonable amount of time, we set the limit of the visited nodes to 10,000. We terminate the execution when the algorithm exceeds this limit, and use the best solution obtained so far. Also, we set the depth-limit where the discrepancy is allowed to 10, and the number of allowed backtracking to 100.

To the extent of the authors' knowledge, the best published results for these problems have been obtained by FASoft [11], [16] and [4]. FASoft is an integrated package of various methods for solving frequency assignment problems, such as heuristic sequential methods, methods using constraint satisfaction techniques, Simulated Annealing, GA, tabu search, etc. We show the results obtained with Simulated Annealing (SA) and tabu search (TS) reported in [11]. These two methods are the most efficient among the various components of FASoft. Furthermore, we show the best results obtained with a set of heuristic sequential methods (SE) reported in [15], and the results obtained with neural networks (NN) reported in [4] ("..." in the table means that the result is not reported).

As shown in the table, our algorithm obtains optimal solutions for the instances of P1, P2, P3, P4, P7, P8, P10, and obtains semi-optimal solutions that are very close to the optimal for other problem instances. Moreover, this method can obtain better or equivalent solutions compared with existing methods for all problem instances except P5 (where NN is better),

TABLE II
Comparison of Solution Quality (Philadelphia Problems)

Instance	Lower Bound	CS	TS	SA	SE	NN
P1	427	427	429	429	460	427
P2	427	427	430	439	447	427
P3	533	533	...	...	536	533
P4	533	533	...	...	533	533
P5	258	261	270	261	283	258
P6	253	258	258	260	270	258
P7	309	309	...	...	310	309
P8	309	309	...	...	310	309
P9	856	857	859	859	...	...
P10	1714	1714	1725	1725	...	...

Furthermore, to examine the efficiency of the proposed algorithm in larger-scale problems, we show the evaluation results for the benchmark problems presented in [4], [12]. There are 7×7 symmetrically placed cells (49 cells in all) in these problems. Problem parameters are described in Table III, where "c_{ij}" is the minimal frequency separation between any pair of cells whose distance is less than $\sqrt{N_c}$, except for adjacent cells. The demand vector is: (19 14 11 13 15 23 21 25 19 20 21 17 10 18 27 23 29 10 17 16 22 14 19 14 22 27 28 25 30 14 18 28 26 12 10 27 29 11 18 24 24 20 25 12 22 25 29 19 14). This vector is randomly generated from a uniform distribution between 10 and 30. There are 976 calls in total.

TABLE III
Specifications for Kim's Benchmark Problems

Instance	Nc	c_{ij}	acc	c_{ii}
K1	7	1	1	3
K2	7	2	3	5
K3	7	3	4	7

TABLE IV
Comparison of Solution Quality (Kim's Benchmark Problems)

Instance	CS	NN	SE
K1	168	168	178
K2	422	435	473
K3	619	630	673

Table IV shows the results obtained with our new method (CS). The parameter settings of the algorithm are identical to those of the Philadelphia Problems. For comparison, we show the results described in [4], i.e., the results obtained using neural networks (NN), and the best results obtained with a set of heuristic sequential methods (SE). Our method obtains much better solutions than those of NN for K2 and K3.

Table V shows the total execution time of this algorithm. Although the execution time is obtained by a naive LISP implementation on a Sun Ultra 30 Model 300 (Ultra SPARC-II 296MHz), we can see that very high-quality solutions are obtained within a reasonably short running time. Since the optimality of the obtained solution is guaranteed before reaching the limit of the visited nodes in P3 and P4, the execution time for these instances is very short. It must be noted the primary evaluation criterion in these benchmark problems is the solution quality. The execution time of the algorithms seems less important and not often reported in the literature. One reason for this is that most algorithms (including our algorithm) show rapid improvements in the early stage of the search process, then the improvements saturate very quickly, and the solution quality cannot be significantly improved even after a very long execution time (e.g., a day).

V. Discussion

In [11], [16], it is reported that optimal solutions for the Philadelphia problems can be obtained using *cliques*. In this method, a maximal or some large clique in a *micro-graph* (where each call is represented as a vertex) is identified first, then frequencies are assigned to the calls in the clique. Subsequently, this partial solution is iteratively extended by adding calls to the partial solution until it becomes a complete solution. One drawback to this method is that finding the maximal clique is another NP-complete problem and time-consuming (note that a micro-graph is much larger than a macro-graph). Furthermore, this clique

TABLE V
Algorithm Execution Time

Instance	Execution Time (sec)
P1	49.3
P2	59.4
P3	0.1
P4	0.1
P5	35.3
P6	37.0
P7	85.4
P8	90.9
P9	112.9
P10	188.3
K1	79.4
K2	133.0
K3	198.9

method is not fully automated, i.e., it requires human trial-and-error selections of cliques and extension methods. Our method and this clique method are not mutually exclusive, namely, our method can be used for finding a partial solution for the clique and extending the partial solution.

A macro-graph is introduced in [14] to solve frequency assignment problems. However, this method uses a macro-graph not to directly solve a problem, but to obtain upper-bounds of the problem. A similar problem representation used in this paper is introduced in [4], [5] for solving the problem using neural networks.

VI. Conclusion and Future Issues

We have developed a new algorithm for solving frequency assignment problems in cellular mobile systems. This algorithm is basically a depth-first branch-and-bound procedure that incorporates forward-checking. In this algorithm, we represent a cell as a variable with a very large domain, and determine the variable value step by step. Furthermore, we have developed a powerful cell-ordering heuristic and introduced the limited discrepancy search to cope with large-scale problems. Experimental evaluations using standard benchmark problems showed that for most of the problem instances, this algorithm can find better or equivalent solutions compared with existing optimization methods. These results imply that state-of-the-art constraint satisfaction/optimization techniques are capable of solving realistic application problems, if we choose the appropriate problem representation and heuristics.

There is plenty of room for improvement in this algorithm. Currently, various algorithm parameters (e.g., the depth-limit, the limit of the number of backtracking) are adjusted by hand. It would be desirable if these parameters can be dynamically tuned according to the characteristics of the solved problem instances. Our future works also include introducing local-consistency algorithms that are stronger than forward-checking, introducing the limited discrepancy search to the cell-ordering heuristic, and using hybrid type algorithms of backtracking and iterative improvement (e.g., [18]).

References

[1] Box, F.: A Heuristic Technique for Assignment Frequencies to Mobile Radio Nets, *IEEE Transactions on Vehicular Technology*, Vol. 27, No. 2, (1978) 57–64

[2] Carlsson, M. and Grindal, M.: Automatic Frequency Assignment for Cellular Telephones Using Constraint Satisfaction Techniques, *Proceedings of the Tenth International Conference on Logic Programming* (1993) 647–663

[3] Freuder, E. C. and Wallace, R. J.: Partial Constraint Satisfaction, *Artificial Intelligence*, Vol. 58, No. 1–3, (1992) 21–70

[4] Funabiki, N., Okutani, N., and Nishikawa, S.: A Three-stage Heuristic Combined Neural Network Algorithm for Channel Assignment in Cellular Mobile Systems, *IEEE Transactions on Vehicular Technology* (2000): (to appear)

[5] Funabiki, N. and Takefuji, Y.: A Neural Network Parallel Algorithm for Channel Assignment Problems in Cellular Radio Networks, *IEEE Transactions on Vehicular Technology*, Vol. 41, No. 4, (1992) 430–437

[6] Gamst, A.: Some Lower Bounds for a Class of Frequency Assignment Problems, *IEEE Transactions on Vehicular Technology*, Vol. 35, No. 1, (1986) 8–14

[7] Hale, W. K.: Frequency Assignment: Theory and Application, *Proceedings of the IEEE*, Vol. 68, No. 12, (1980) 1497–1513

[8] Hao, J. K., Dorne, R., and Galinier, P.: Tabu Search for Frequency Assignment in Mobile Radio Networks, *Journal of Heuristics*, Vol. 4, (1998) 47–62

[9] Haralick, R. and Elliot, G. L.: Increasing Tree Search Efficiency for Constraint Satisfaction Problems, *Artificial Intelligence*, Vol. 14, (1980) 263–313

[10] Harvey, W. D. and Ginsberg, M. L.: Limited discrepancy search, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (1995) 607–613

[11] Hurley, S., Smith, D. H., and Thiel, S. U.: FASoft: A system for discrete channel frequency assignment, *Radio Science*, Vol. 32, No. 5, (1997) 1921–1939

[12] Kim, S. and Kim, S. L.: A Two-Phase Algorithm for Frequency Assignment in Cellular Mobile Systems, *IEEE Transactions on Vehicular Technology*, Vol. 43, No. 3, (1994) 542–548

[13] Kunz, D.: Channel Assignment for Cellular Radio Using Neural Networks, *IEEE Transactions on Vehicular Technology*, Vol. 40, No. 1, (1991) 188–193

[14] Pennotti, R. J. and Boorstyn, R. R.: Channel Assignments for Cellular Mobile Telecommunications Systems, *Proceedings of National Telecommunications Conference* (1976) 16:5–1–16:5–5

[15] Sivarajan, K. N., McEliece, R. J., and Ketchum, J. W.: Channel Assignment in Cellular Radio, *Proceedings of 39th IEEE Vehicular Technology Society Conference* (1989) 846–850

[16] Smith, D. H., Hurley, S., and Thiel, S. U.: Improving Heuristics for the Frequency Assignment Problem, *European Journal of Operational Research*, Vol. 107, (1998) 76–86

[17] Walsh, T.: Depth-bounded Discrepancy Search, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (1997) 1388–1393

[18] Yokoo, M.: Weak-commitment Search for Solving Constraint Satisfaction Problems, *Proceedings of the Twelfth National Conference on Artificial Intelligence* (1994) 313–318