OPTIMAL CHANNEL ASSIGNMENT IN CELLULAR SYSTEMS USING TABU SEARCH

Yanjie Peng, Lipo Wang, and Boon Hee Soong

Abstract - The channel assignment problem (CAP) in cellular systems has the task of planning the reuse of the limited available frequencies in a spectrum-efficient and interference-minimal way. We adopt an integer programming representation of the static channel assignment (SCA) formulated by Smith and Palaniswami. A tabu search (TS) algorithm is implemented to solve this problem. Two forms of tabu list are used, one of which is the short-term memory; the other contains both the short- and long- term memory. Ten benchmark problems are tested and the results are compared with those obtained by using some popular optimization methods. We show that the TS approach significantly outperforms other optimization techniques.

Index Terms - Channel assignment problem, tabu search.

I. INTRODUCTION

With the ever-increasing popularity of mobile cellular communications, the limited radio spectrum places a restriction on this promising scenario. When two users are close geographically, interferences occur if they transmit on frequencies that are close in the radio spectrum. In general, interferences can be classified into three categories: a *co-channel* interference - when the same channel is assigned in several cells from different clusters simultaneously, a *adjacent-channel* interference - when channels adjacent in the radio spectrum are used in cells adjacent geographically, and a *co-site* interference - when channels assigned to the same cell are not separated from each other by a minimal separation in frequency. There are at least two approaches coping with the lack of the radio spectrum. One is to improve the design and specifications of the hardware used to transmit and receive the RF signals. If a very tight filter is available to remove the undesired frequencies effectively, it is possible to increase the traffic capacity. This approach is important but not the only way. Optimal assignment of frequency channels is the other approach to solving the scarcity of the radio spectrum, which provides motivation for the channel assignment problem (CAP). The CAP is to assign the required number of channels to each base station in a way so that the radio spectrum is used most efficiently and the interferences among users are minimized.

The channel assignment problem first appeared in the 1960s [1]. The development of new wireless services such as the first cellular networks led to scarcity of usable frequencies in the radio spectrum. In the late 1980s and early 1990s, the development of the GSM systems led to a rapidly increasing interest for channel assignment [2]. But projects on other applications such as military wireless communication and radio/TV broadcasting contributed to the literature of channel assignment. So far, many methods have been proposed to tackle the channel assignment problem. Most early approaches used so-called greedy algorithms [3], [4]. Some heuristic methods include neural networks (NNs) [5], [6], [7], [8], genetic algorithms (GAs) [9], [10], simulated annealing (SA) [11], [12], [13], and tabu search (TS) [13], [14].

In this paper, we implement a tabu search algorithm. The paper is organized as follows. Section II describes the mathematical formulation of the channel assignment problem. Section III presents the principles of tabu search. Section IV presents an implementation of the tabu search algorithm for solving the channel assignment problem. Section V discusses the simulation results. Section VI concludes the paper.

II. THE CHANNEL ASSIGNMENT PROBLEM

The channel assignment problem can be classified into two categories: the static channel assignment (SCA) and the dynamic channel assignment (DCA). A static channel assignment is made to satisfy the requirement for the immediate future based on the traffic forecast. The SCA is made only once and remains unchanged. As the traffic demand increases, it will become more and more difficult to maintain a satisfactory performance. In our future work, a dynamic assignment strategy, which will allow adaptive re-assignment of channels, will be investigated. In this paper, we only deal with the SCA.

A. Problem Description

Suppose the whole network is divided into N cells and total number of channels available is M. The channel demand is given in a one-dimensional matrix D. The non-interference constraints are stored in a symmetrical $N \times N$ compatibility matrix *C*. C_{ij} gives the minimal frequency separation between the channels assigned to cells *i* and *j*, for interference-free assignments.

As an example, suppose the number of cells in the network is $N = 4$, and the demand for channels in each of these cells is given by $D = [1, 1, 1, 3]$. Consider the compatibility matrix suggested by Sivarajan *et al* [4]

$$C = \begin{pmatrix} 5 & 4 & 0 & 0 \\ 4 & 5 & 0 & 1 \\ 0 & 0 & 5 & 2 \\ 0 & 1 & 2 & 5 \end{pmatrix}$$

The diagonal terms $C_{ii} = 5$ indicate that any two channels assigned to the same cell must be at least five frequencies apart in order that no co-site interference exists. Channels assigned to cells 1 and 2 must be at least $C_{12} = 4$ frequencies apart. Off-diagonal terms of $C_{ij} = 1$ and $C_{ij} = 2$ correspond to co-channel and adjacent-channel constraints, respectively. Thus, the solution represented in Fig. 1 is one of the interference-free assignments that can be generated for $M = 11$, and the solution in Fig. 2 incurs a weak interference.

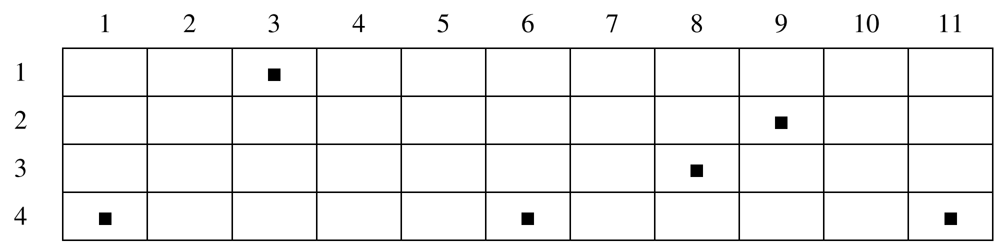

Fig. 1: An interference-free assignment for a four-cell and 11-channel network

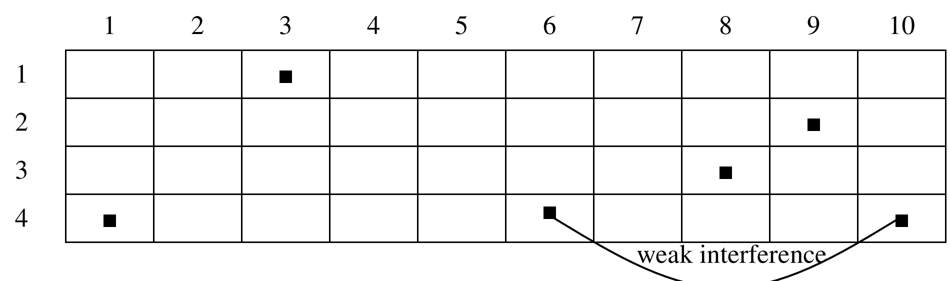

Fig. 2: A near interference-free assignment for a four-cell and 10-channel network

Given the above parameters, the SCA can be further divided into two types: the minimum span problem and the fixed spectrum problem. The minimum span problem is to minimize the span of channels (*M*) subject to the demand constraints (*D*) and the non-interference constraints (*C*), i.e., to determine the minimal number of channels (*M*) required to achieve an interference-free assignment, given *N*, *C* and *D*. The fixed spectrum problem is to minimize the severity of interferences subject to the demand constraints (*D*), i.e., to minimize the interference level given all the four parameters, *M*, *N*, *C* and *D*. In practice, the total number of channels available is normally fixed, so the algorithm developed in this paper is only applied to the fixed spectrum problem. However, with some small modifications, the algorithm can also be used to solve the minimum span problem.

B. Mathematical Formulation

The SCA problem can be formulated as various models. In this paper, the formulation by Smith and Palaniswami [5] is used.

First, a set of binary variables are defined:

$$X_{j,k} = \begin{cases} 1, \text{ if channel } k \text{ is assigned to cell } j \\ 0, \text{ otherwise} \end{cases} \quad (1)$$

$$\text{for } j = 1, \ldots, N \text{ and } k = 1, \ldots, M$$

Suppose that $X_{j,k} = X_{i,l} = 1$, i.e., calls in cells j and i have been assigned channels k and l, respectively. One way to measure the degree of interference caused by such assignments is to weight each pair of assignments by a cost tensor $P_{j,i,m+1}$, where $m = |k - l|$ is the distance between channels k and l. If $k = l$, the interference cost should be at its maximum, with the cost decreasing until the two channels are far enough apart so that no interference exists.

The SCA then can be formulated to minimize the total cost $F(X)$ of all assignments in the network, where $F(X)$ is given below.

$$F(X) = \sum_{j=1}^{N}\sum_{k=1}^{M} X_{j,k} \sum_{i=1}^{N}\sum_{l=1}^{M} P_{j,i,(|k-l|+1)} X_{i,l} \tag{2}$$

$$\text{subject to } \sum_{k=1}^{M} X_{j,k} = D_j \qquad \forall\, j = 1, \ldots, N; \tag{3}$$

$$X_{j,k} \in \{0,1\} \qquad \forall\, j = 1, \ldots, N, \text{ and } k = 1, \ldots, M. \tag{4}$$

The cost tensor P can be generated according to the recursive relation

$$P_{j,i,m+1} = \max(0, P_{j,i,m} - 1), \quad \text{for } m = 1, \ldots, M-1; \tag{5}$$

$$P_{j,i,1} = C_{ji} \qquad \text{for all } j,\ i \neq j; \tag{6}$$

$$P_{j,i,1} = 0 \qquad \text{for all } j = i. \tag{7}$$

The cost tensor is a cuboid, the front square of which is simply the compatibility matrix C with the diagonal terms overwritten to zero. This is because if $j = i$ and $k = l$, there should be no penalty incurred (there is effectively only a single call and no interference is possible). The third dimension of the tensor decreases the penalty linearly until the penalty becomes zero. Thus, the effective depth of the tensor is equivalent to the value of the maximum diagonal of C. This linear decrease in cost is a means of encouraging less severe violation of the non-interference constraints. If the network demands are such that violation is inevitable, the strength of the resulting signal is preferred to still be as strong as possible.

III. THE TABU SEARCH ALGORITHM

Glover [15] and Hansen [16] developed tabu search (TS) independently in 1986, to solve the combinatorial problem. It is a meta-heuristic method that guides a local heuristic search procedure to explore the solution space beyond a local optimality. The basic idea is to forbid certain moves that would return to recently visited solutions, by rendering them tabu. To go beyond a local optimum, TS allows a worsening solution. Given a starting feasible solution, TS computes the neighborhood, but selects the solution with the best objective function value within the neighborhood, even it's worse than the best solution obtained so far. The key components of the TS algorithm are explained below.

Initial Solution: Every tabu search algorithm starts with an initial solution. This is in the same way as an ordinary local or neighborhood search.

Move: A move is a simple modification to the current solution. A move varies from problem to problem, and even for the same problem instance, there is more than one possible case of moves.

Neighborhood: A move from the current solution produces a new solution. A number of such new solutions compose a neighborhood.

Tabu Tenure: The tabu tenure is the number of iterations for which the tabu-active status of attributes will last. It can vary for different types or combinations of attributes, and can vary over different intervals of time or stages of search.

Tabu List: The tabu list is a list containing the last several moves carried out (short-term memory) and moves that have occurred too frequently (long-term memory).

Short-Term Memory: The most commonly used short-term memory keeps track of the solution attributes that have changed during the recent past, and is also called *recency-based* memory. The short-term memory is used to prevent the search from being trapped in a local optimality.

Long-Term Memory: In some applications, the short-term memory components are sufficient to produce high quality solutions. However, in general, TS becomes significantly stronger by including the long-term memory. Normally the long-term memory is employed to encourage the search to explore regions less frequently visited. The *frequency-based* approach is most commonly implemented.

Aspiration Criterion: Aspiration criteria are introduced in tabu search to determine when tabu activation rules can be overridden, thus removing a tabu classification otherwise applied to a move. The appropriate use of such criteria can be very important for enabling a TS method to avoid missing good solutions and achieve its best performance levels.

Termination Criterion: The search procedure terminates when the termination criterion is satisfied. There are several possible stopping criteria: (a) an optimal solution is found; (b) the time interval is expired; (c) the neighborhood set is null; (d) the number of iterations is greater than the maximum number of iterations allowed; and (e) the number of iterations performed since the current solution last changed is greater than a specified maximum number of iterations.

IV. AN IMPLEMENTATION OF THE TS ALGORITHM

In this section, we present a TS implementation for the fixed spectrum problem. And the algorithm can easily be modified to solve the minimum span problem. Suitable values or forms are selected for all the parameters introduced in section III. The four-cell and 11-channel example in section II is used here for demonstration. A flowchart of the implementation is shown in Fig. 11.

Initial Solution: Assigning arbitrarily the required number of channels to the corresponding cells produces the initial solution. For the example problem, the initial solution can be generated at random by assigning one channel to cells 1, 2, and 3, and three channels to cell 4, which is displayed in Fig. 3.

	1	2	3	4	5	6	7	8	9	10	11
1									■		
2										■	
3			■								
4			■						■		■

Fig. 3: An example initial solution

Move: A move is a basic flip-flop, which is generated by replacing one used channel with one unused channel in a cell. The unused channel is generated randomly as long as it does not coincide with those used channels in the current cell.

Neighborhood: The neighborhood contains solutions that are one move from the current solution. The size of the neighborhood is equal to the total demand of channels. The size is simply 6, which is simply the sum of 1, 1, 1 and 3, for the example problem. The neighborhood is shown in Fig. 4 - 9, supposing a sequence of integers 2, 1, 8, 7, 2, 1 has been generated at random.

	1	2	3	4	5	6	7	8	9	10	11
1		■									
2										■	
3			■								
4			■						■		■

Fig. 4: The first solution of the current neighborhood

	1	2	3	4	5	6	7	8	9	10	11
1									■		
2	■										
3			■								
4			■						■		■

Fig. 5: The second solution of the neighborhood

	1	2	3	4	5	6	7	8	9	10	11
1									■		
2										■	
3								■			
4			■						■		■

Fig. 6: The third solution of the neighborhood

	1	2	3	4	5	6	7	8	9	10	11
1									■		
2										■	
3			■								
4							■		■		■

Fig. 7: The fourth solution of the neighborhood

	1	2	3	4	5	6	7	8	9	10	11
1									■		
2										■	
3			■								
4		■	■								■

Fig. 8: The fifth solution of the neighborhood

	1	2	3	4	5	6	7	8	9	10	11
1									■		
2										■	
3			■								
4	■		■						■		

Fig. 9: The sixth solution of the neighborhood

The cost values for the six solutions are calculated. They are 5, 5, 7, 10, 10, and 8 respectively. The smallest cost value is 5 in this neighborhood. In case of a tie, the solution occurred earliest will be chosen. So the first solution is selected to be the starting solution of the next neighborhood, which is reproduced in Fig. 10.

	1	2	3	4	5	6	7	8	9	10	11
1		■									
2										■	
3			■								
4			■						■		■

Fig. 10: The starting solution of the next neighborhood

Tabu Tenure: The tabu tenure varies with cell. If it is too small, the algorithm will not be so effective. This is because the small tabu tenure cannot guide the search to explore the unvisited regions. If the tenure is too large, on the other hand, the number of free channels will be small and the flip-flops will be restricted to a small range. This may unnecessarily take long time to select a move. Even worse, if the cells except the active cell (in which the flip-flop is being done) have a bad combination, then the solution will not pass the aspiration check, although the active cell has been optimized. Hence, we choose the tabu tenure to be

$$TN_j = \frac{M - D_j}{D_j \times 2} \times S \tag{8}$$

where M is the total number of channels available;

D_j is the number of channels required in cell j;

S is the size of the neighborhood.

This ensures that there is always sufficient number of channels to select. With only the short-term memory, more than half of the $M - D_j$ unused channels are not in the tabu list. If both the short- and long-term memory are implemented, the number of free channels decreases slightly.

Tabu List: Every attribute is reserved a space in the tabu list, which is a two-dimensional array, *TL[][]*. The two indices represent the cell and channel numbers, respectively. Every element in the tabu list decreases by one after each iteration until it becomes zero. Two forms of tabu list are implemented in this paper. One is the short-term memory; the other contains both the short- and long-term memory. Results of both implementations are obtained and compared in the next section.

Short-Term Memory: In this paper, an unused channel that replaced a used one in a flip-flop is placed in the tabu list after the current iteration. For example, in the neighborhood of the initial solution, channel 2 (cell 1), channel 1 (cell 2), channel 8 (cell 3), channels 7, 2, 1 (cell 4) are the unused channels in the six flip-flops. They are put in the tabu list after the corresponding iteration.

However, for the local optimum, there are some small alterations to the tabu list. The unused channel in that flip-flop, which has been labeled tabu after that iteration, should be "dis-tabu". This is because, as an attribute of the local optimum, the channels occur in the following iterations must be different from it. It is unnecessary to tabu it. And the used channel should be put in the tabu list simply because it occurred recently. In the example, channel 2 of cell 1 is "dis-tabu" and channel 9 of cell 1 is put in the tabu list at the end of the current neighborhood.

Long-Term Memory: The frequency-based approach is employed in this paper. Frequency is conceived to consist of ratios, whose numerators represent counts expressed in two different measures: a *transition* measure - the number of iterations where an attribute changes (enters or leaves) the solutions visited on a particular trajectory, and a *residence* measure - the number of iterations where an attribute belongs to solutions from a particular subset. The denominators generally represent the total number of occurrences of all events represented by the numerators (such as the total number of associated iterations). The residence frequency is used here. The residence measure for each attribute is

$$FM_{jk} = \frac{A_{jk}}{T} \tag{9}$$

where A_{jk} is the number of iterations where an attribute belongs to solutions;

T is the total number of iterations so far.

Frequency threshold: Attributes with frequency measures exceeding the frequency thresholds will be labeled tabu. We find that approximately half of the unused channels in a cell are in the tabu list, and the other half is free for flip-flops. On average, every free channel is selected once every $\frac{M - D_j}{2 \times D_j} \times S$ iterations. Therefore, we select the threshold value to be

$$TR_j = \frac{D_j \times 2}{(M - D_j) \times S} \tag{10}$$

where M is the total number of channels available;

D_j is the number of channels required in cell j;

S is the size of the neighborhood.

The frequency measure is checked against the corresponding threshold at the end of a neighborhood. If it is greater than the threshold, this associated attribute is placed in the tabu list.

Aspiration Criterion: Aspiration by objective is used in this paper. If a move is in the tabu list, it must be checked against the aspiration criterion to determine whether it is used or discarded. If it passes the criterion, i.e., it leads to a solution with a smaller cost value than the minimum cost obtained so far, the tabu status of the move is revoked for the current iteration. Otherwise, the move is discarded and another random channel is generated.

Termination Criterion: The search procedure terminates when an interference-free solution is found or the preset six-hour time interval expires, whichever comes earlier. For the example problem, an interference-free assignment can be achieved within a few seconds, so the search progress stops at the moment the solution with zero cost is found.

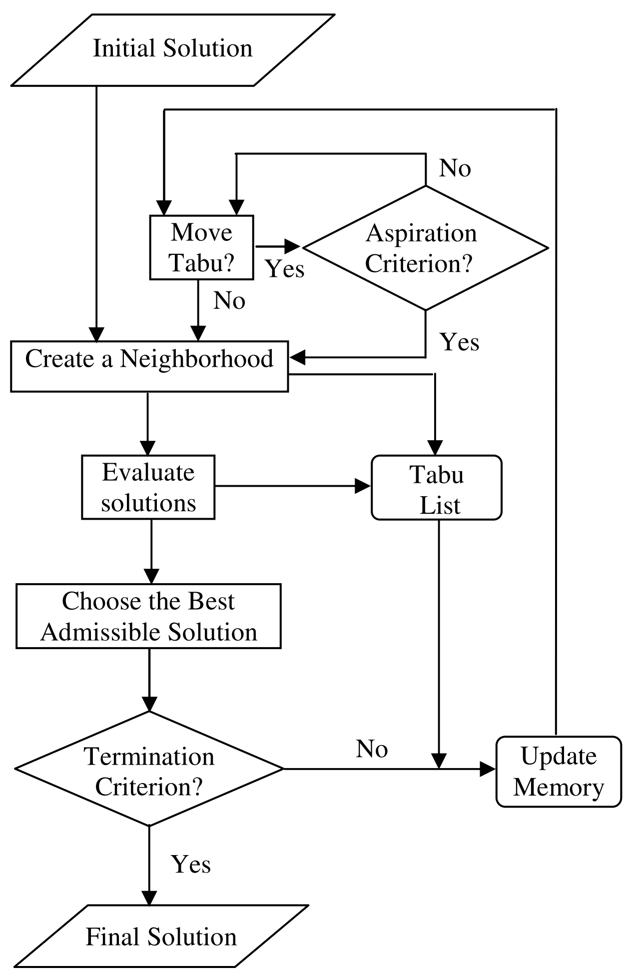

Fig 11: Implementation flowchart of the TS algorithm

5. RESULTS AND DISCUSSION

The implemented TS algorithm was tested on ten benchmark problems, which are divided into three groups. The first group contains two standard examples [4], the first of which is the small example used for demonstration, denoted by EX1. The second (denoted by EX2) is a slightly larger extension of EX1 with N = 5, M = 17, D = [2, 2, 2, 4, 3]. And the compatibility matrix is given by

$$C = \begin{pmatrix} 5 & 4 & 0 & 0 & 1 \\ 4 & 5 & 0 & 1 & 0 \\ 0 & 0 & 5 & 2 & 1 \\ 0 & 1 & 2 & 5 & 2 \\ 1 & 0 & 1 & 2 & 5 \end{pmatrix}$$

The second group (denoted by HEX 1 – 4) is based on the 21-cell regular hexagonal network used by Sivarajan *et al* [4]. The details are shown in Table I. The two sets of demands are shown in D_1 and D_2, and the compatibility matrices for the second group are stored in C_1 and C_2.

The final set is derived from the topo-graphical data of an actual 24 × 21 km area around Helsinki, Finland, as used by Kunz [6]. This set is described in Table II. The demand vector is D_3 and the compatibility matrix is C_3.

D_1 = [2, 6, 2, 2, 2, 4, 4, 13, 19, 7, 4, 4, 7, 4, 9, 14, 7, 2, 2, 4, 2]
D_2 = [1, 1, 1, 2, 3, 6, 7, 6, 10, 10, 11, 5, 7, 6, 4, 4, 7, 5, 5, 5, 6]
D_3 = [10, 11, 9, 5, 9, 4, 5, 7, 4, 8, 8, 9, 10, 7, 7, 6, 4, 5, 5, 7, 6, 4, 5, 7, 5]

TABLE I
PROBLEM DESCRIPTIONS FOR HEX DATA SETS

Problem	N	M	D	co-channel	adjacent	C_{ii}
HEX1	21	37	D_1	yes	no	2
HEX2	21	91	D_1	yes	yes	3
HEX3	21	21	D_2	yes	no	2
HEX4	21	56	D_2	yes	yes	3

TABLE II
PROBLEM DESCRIPTIONS FOR KUNZ DATA SETS

Problem	N	M	C	D
KUNZ1	10	30	$[C_3]_{10}$	$[D_3]_{10}$
KUNZ2	15	44	$[C_3]_{15}$	$[D_3]_{15}$
KUNZ3	20	60	$[C_3]_{20}$	$[D_3]_{20}$
KUNZ4	25	73	$[C_3]$	$[D_3]$

Ten runs were performed for each of the ten testing problems. Table III compares the results obtained in this paper by using TS with the results given in [5] by using GAMS/MINOS-5 (labeled GAMS), steepest descent (SD), simulated annealing (SA), self-organizing neural network (SONN), Hopfield network without hill-climbing (HN), and hill-climbing Hopfield network (HCHN). “Min” represents the minimum interference found during the ten runs, while “Ave” is the average interference value. TS obtains better average results for all the data sets except EX1 and KUNZ3. In term of minimum results, TS improves much for the HEX data set.

Table IV compares the results obtained by implementing only the short-term memory (STM) and both the short- and long-term memory (SLM). The results for the first and third groups are exactly the same, so they are not shown here. For the second group, SLM performs slightly better than STM, as expected. The actual channel assignments for selected problems for which improvement were obtained are shown in Table V – IX.

6. CONCLUSIONS

In this paper, a Tabu Search algorithm has been successfully implemented to solve the channel assignment problem. Simulation results on ten benchmark CAPs were presented and compared with some popular optimization techniques. Most of the ten problems tested had better average results and substantial improvements have been achieved for the HEX data sets. TS became slightly stronger when the long-term memory was included. From the comparison, TS outperforms the existing methods and proves to be the most effective algorithm for the channel assignment problem.

$$C_1 = \begin{pmatrix}
2&1&1&0&0&1&1&1&1&0&0&0&0&1&1&1&0&0&0&0&0\\
1&2&1&1&0&0&1&1&1&1&0&0&0&0&1&1&1&0&0&0&0\\
1&1&2&1&1&0&0&1&1&1&1&0&0&0&0&1&1&1&0&0&0\\
0&1&1&2&1&0&0&0&1&1&1&1&0&0&0&0&1&1&0&0&0\\
0&0&1&1&2&0&0&0&0&1&1&1&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&2&1&1&0&0&0&0&1&1&1&0&0&0&0&0&0\\
1&1&0&0&0&1&2&1&1&0&0&0&1&1&1&1&0&0&1&0&0\\
1&1&1&0&0&1&1&2&1&1&0&0&0&1&1&1&1&0&1&1&0\\
1&1&1&1&0&0&1&1&2&1&1&0&0&0&1&1&1&1&1&1&1\\
0&1&1&1&1&0&0&1&1&2&1&1&0&0&0&1&1&1&0&1&1\\
0&0&1&1&1&0&0&0&1&1&2&1&0&0&0&0&1&1&0&0&1\\
0&0&0&1&1&0&0&0&0&1&1&2&0&0&1&0&0&1&0&0&0\\
0&0&0&0&0&1&1&0&0&0&0&0&2&1&1&0&0&0&0&0&0\\
1&0&0&0&0&1&1&1&0&0&0&0&1&2&1&1&0&0&1&0&0\\
1&1&0&0&0&1&1&1&1&0&0&0&1&1&2&1&1&0&1&1&0\\
1&1&1&0&0&0&1&1&1&1&0&0&0&1&1&2&1&1&1&1&1\\
0&1&1&1&0&0&0&1&1&1&1&0&0&0&1&1&2&1&1&1&1\\
0&0&1&1&1&0&0&0&1&1&1&1&0&0&0&1&1&2&0&1&1\\
0&0&0&0&0&0&1&1&1&0&0&0&0&1&1&1&1&0&2&1&1\\
0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&1&1&1&1&2&1\\
0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&1&1&1&1&2
\end{pmatrix}$$

$$C_2 = \begin{pmatrix}
3&2&1&0&0&1&2&2&1&0&0&0&0&1&1&1&0&0&0&0&0\\
2&3&2&1&0&0&1&2&2&1&0&0&0&0&1&1&1&0&0&0&0\\
1&2&3&2&1&0&0&1&2&2&1&0&0&0&0&1&1&1&0&0&0\\
0&1&2&3&2&0&0&0&1&2&2&1&0&0&0&0&1&1&0&0&0\\
0&0&1&2&3&0&0&0&0&1&2&2&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&3&2&1&0&0&0&0&2&2&1&0&0&0&0&0&0\\
2&1&0&0&0&2&3&2&1&0&0&0&1&2&2&1&0&0&1&0&0\\
2&2&1&0&0&1&2&3&2&1&0&0&0&1&2&2&1&0&1&1&0\\
1&2&2&1&0&0&1&2&3&2&1&0&0&0&1&2&2&1&1&1&1\\
0&1&2&2&1&0&0&1&2&3&2&1&0&0&0&1&2&2&0&1&1\\
0&0&1&2&2&0&0&0&1&2&3&2&0&0&0&0&1&2&0&0&1\\
0&0&0&2&2&0&0&0&0&1&2&3&0&0&1&0&0&1&0&0&0\\
0&0&0&0&0&2&1&0&0&0&0&0&3&2&1&0&0&0&0&0&0\\
1&0&0&0&0&2&2&1&0&0&0&0&2&3&2&1&0&0&1&0&0\\
1&1&0&0&0&1&2&2&1&0&0&0&1&2&3&2&1&0&2&1&0\\
1&1&1&0&0&0&1&2&2&1&0&0&0&1&2&3&2&1&2&2&1\\
0&1&1&1&0&0&0&1&2&2&1&0&0&0&1&2&3&2&1&2&2\\
0&0&1&1&1&0&0&0&1&2&2&1&0&0&0&1&2&3&0&1&2\\
0&0&0&0&0&0&1&1&1&0&0&0&0&1&2&2&1&0&3&1&1\\
0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&2&2&1&2&3&2\\
0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&2&2&1&2&3
\end{pmatrix}$$

$$
C_3 = \begin{pmatrix}
2 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 2 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 2 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 2 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 2
\end{pmatrix}
$$

TABLE III
COMPARISON OF TS WITH GAMS, SD, SA, SONN, HN, AND HCHN

Problem	GAMS	SD		SA		SONN		HN		HCHN		TS	
	Min	Ave	Min	Ave	Min	Ave	Min	Ave	Min	Ave	Min	Ave	Min
EX1	2	0.6	0	0.0	0	0.4	0	0.2	0	0.0	0	0	0
EX2	3	1.1	0	0.1	0	2.4	0	1.8	0	0.8	0	0	0
HEX1	54	56.8	55	50.7	49	53.0	52	49.0	48	48.7	48	47.0	46
HEX2	27	28.9	25	20.4	19	28.5	24	21.2	19	19.8	19	18.4	15
HEX3	89	88.6	84	82.9	79	87.2	84	81.6	79	80.3	78	73.8	73
HEX4	31	28.2	26	21.0	17	29.1	22	21.6	20	18.9	17	12.8	12
KUNZ1	28	24.4	22	21.6	21	22.0	21	22.1	21	21.1	20	20	20
KUNZ2	39	38.1	36	33.2	32	33.4	33	32.8	32	31.5	30	29	29
KUNZ3	13	17.9	15	13.9	13	14.4	14	13.2	13	13.0	13	13	13
KUNZ4	7	5.5	3	1.8	1	2.2	1	0.4	0	0.1	0	0	0

TABLE IV
COMPARISON OF SLM TO STM

Problem	STM	SLM

	Ave	Min	Ave	Min
HEX1	47.4	47	47.0	46
HEX2	18.7	17	18.4	15
HEX3	73.8	73	73.8	73
HEX4	12.8	12	12.8	12

REFERENCES

[1] B. H. Metzger, “Spectrum management technique,” presented at the 38th National ORSA meeting, Detroit, Michigan, USA, Fall 1970.

[2] A. Eisenblätter, “Frequency assignment in GSM networks: models, heuristics, and lower bound,” Ph.D. thesis, Technische Universität Berlin, Berlin, Germany, 2001.

[3] J. A. Zoellner and C. L. Beall, “A breakthrough in spectrum conserving frequency assignment technology,” IEEE Trans. Electromagn. Compat., vol. 19, pp. 313 – 319, 1977.

[4] K. N. Sivarajan and R. J. McEliece, “Channel assignment in cellular radio,” in Proc. IEEE Veh. Technol. Conf., 1989, pp. 846 – 850.

[5] K. Smith and M. Palaniswami, “Static and dynamic channel assignment using neural networks,” IEEE J. Select. Areas Commun., vol. 15, no. 2, pp. 238 – 249, Feb. 1997.

[6] D. Kunz, “Channel assignment for cellular radio using neural networks,” IEEE Trans. Veh. Technol., vol. 40, no. 1, pp. 188 – 193, 1991.

[7] J. –S. Kim, S. Park, P. Dowd and N. Nasrabadi, “Cellular radio channel assignment using a modified Hopfield network,” IEEE Trans. Veh. Technol., vol. 46, no. 4, pp. 957 – 967, 1996.

[8] A. Hanamitsu and M. Ohta, “A maximum neural network with self-feedbacks for channel assignment in cellular mobile systems,” In Proc. Int. Joint Conf. Neural Networks, 2002, vol. 3, pp. 2814 – 2818.

[9] D. Beckmann and U. Killat, “A new strategy for the application of genetic algorithms to the channel-assignment problem,” IEEE Trans. Veh. Technol., vol. 48, no. 4, pp. 1261 – 1269, Jul. 1999.

[10] C. Y. Ngo and V. O. K. Li, “Fixed channel assignment in cellular radio networks using a modified genetic algorithm,” IEEE Trans. Veh. Technol., vol. 47, no. 1, pp. 163 – 171, 1998.

[11] M. Duque-Antón, D. Kunz and B. Rüber, “Channel assignment for cellular radio using simulated annealing,” IEEE Trans. Veh. Technol., vol. 42, pp. 14 – 21, 1993.

[12] A. Dawood, M. M. Oo, K. Ahmed and N. Rajatheva, “Efficient channel assignment algorithm under cochannel and adjacent channel interference constraint,” in Proc. IEEE Int. Conf. Pers. Wireless Commun., 2000, pp. 424 – 428.

[13] D. H. Smith, R. K. Taplin and S. Hurley, “Frequency assignment with complex co-site constraints,” IEEE Trans. Electromagn. Compat., vol. 43, no. 2, pp. 210 – 218, May 2001.

[14] A. Capone and M. Trubian, “Channel assignment problem in cellular systems: a new model and a tabu search algorithm,” IEEE Trans. Veh. Technol., vol. 48, no. 4, pp. 1252 – 1260, Jul. 1999.

[15] F. Glover, "Future paths for integer programming and links to artificial intelligence," Comp. and Oper. Res., vol. 13, pp. 533-549, 1986.

[16] P. Hansen, “The steepest ascent and mildest descent heuristic for combinational programming,” Congress on Numerical Methods in Combinational Optimization, Capri, Italy, 1986.

[17] F. Glover and M. Laguna, *Tabu search.* Boston: Kluwer Academic Publishers, 1997.

[18] I. Katzela and M. Naghshineh, “ Channel assignment schemes for cellular mobile telecommunication systems: a comprehensive survey,” *IEEE Pers. Commun.*, pp. 10 – 31, Jun. 1996.

[19] A. Eisenblätter, M. Grötschel and A. M. C. A. Koster, “Frequency planning and ramifications of coloring,” ZIB, Berilin, Germany, Tech. Rep. 00 – 47, Dec. 2000.

[20] K. I. Aardal, S. P. M. Van Hoesel, A. M. C. A. Koster, C. Mannino and A. Sassano, “Models and solution techniques for frequency assignment problems,” ZIB, Berilin, Germany, Tech. Rep. 01 – 40, Dec. 2001.

TABLE V
CHANNEL ASSIGNMENT FOR HEX1 WITH INTERFERENCE VALUE 46

Cell No.	Number of Channels	Assigned Channels
1	2	24,26
2	6	2,10,12,29,31,35
3	2	6,22
4	2	30,32
5	2	5,10
6	4	3,12,19,37
7	4	4,8,15,22
8	13	1,5,7,9,14,16,18,20,28,30,32,34,36
9	19	1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37
10	7	4,8,13,17,24,26,33
11	4	2,14,20,34
12	4	7,9,35,37
13	7	16,18,20,24,27,29,34
14	4	2,10,21,31
15	9	4,6,11,13,17,28,30,33,36
16	14	3,5,9,14,16,18,20,23,25,27,29,32,34,37
17	7	3,8,10,15,19,21,24
18	2	12,36
19	2	12,26
20	4	2,6,22,31
21	2	28,30

TABLE VI
CHANNEL ASSIGNMENT FOR HEX2 WITH INTERFERENCE VALUE 15

Cell No.	Number of Channels	Assigned Channels
1	2	4,53
2	6	13,22,28,44,72,80
3	2	7,62
4	2	59,64

5	2	3,84
6	4	3,33,52,88
7	4	47,64,77,90
8	13	6,11,15,19,24,32,40,50,57,61,69,83,87
9	19	3,9,16,26,30,34,38,41,46,49,54,58,65,70,75,78,84,88,91
10	7	12,20,36,51,68,82,86
11	4	15,29,80,89
12	4	22,35,66,74
13	7	25,41,45,61,73,76,91
14	4	10,39,49,68
15	9	2,8,17,27,37,55,59,71,79
16	14	5,21,29,35,43,48,52,63,67,73,76,81,85,89
17	7	1,14,18,23,39,56,60
18	2	6,31
19	2	13,45
20	4	7,10,25,33
21	2	4,42

TABLE VII
CHANNEL ASSIGNMENT FOR HEX3 WITH INTERFERENCE VALUE 73

Cell No.	Number of Channels	Assigned Channels
1	1	21
2	1	5
3	1	2
4	2	7,9
5	3	12,16,20
6	6	1,3,6,8,12,20
7	7	2,9,11,13,15,17,19
8	6	5,7,10,14,16,18
9	10	1,4,6,8,10,12,14,16,18,20
10	10	1,3,6,8,11,13,15,17,19,21
11	11	1,3,5,7,9,11,13,15,17,19,21

12	5	2,4,10,14,18
13	7	5,7,10,14,16,18,21
14	6	2,4,6,8,15,20
15	4	11,13,17,19
16	4	3,7,9,12
17	7	4,6,8,12,14,16,20
18	5	5,11,13,17,19
19	5	1,3,5,13,21
20	5	2,7,9,15,21
21	6	2,4,10,16,18,20

TABLE VIII
CHANNEL ASSIGNMENT FOR HEX4 WITH INTERFERENCE VALUE 12

Cell No.	Number of Channels	Assigned Channels
1	1	39
2	1	24
3	1	51
4	2	2,25
5	3	17,40,53
6	6	2,16,22,30,48,51
7	7	6,10,26,33,41,44,55
8	6	8,12,15,29,36,50
9	10	1,5,18,21,27,32,42,45,53,56
10	10	3,9,13,16,23,30,35,39,48,54
11	11	6,11,15,19,22,28,33,37,43,46,50
12	5	4,8,24,31,55
13	7	4,8,11,18,42,53,56
14	6	14,20,28,35,38,46
15	4	3,17,23,52
16	4	25,34,40,47
17	7	7,10,14,20,38,44,49

18	5	1,26,41,52,56
19	5	13,19,37,43,54
20	5	2,22,28,51,55
21	6	4,12,17,24,31,36

TABLE IX
CHANNEL ASSIGNMENT FOR KUNZ2 WITH INTERFERENCE VALUE 29

Cell No.	Number of Channels	Assigned Channels
1	10	1,3,16,18,26,28,33,37,39,41
2	11	1,3,12,18,20,24,28,33,35,37,41
3	9	7,10,13,22,25,27,29,32,44
4	5	1,12,26,28,37
5	9	4,6,8,15,21,26,31,34,42
6	4	3,12,18,20
7	5	4,8,15,23,42
8	7	2,6,9,17,31,34,43
9	4	11,19,21,24
10	8	5,7,14,30,32,36,38,40
11	8	3,16,18,20,33,35,39,41
12	9	10,13,16,22,27,29,35,39,44
13	10	5,11,14,19,21,24,30,36,38,40
14	7	2,4,9,17,23,34,43
15	7	10,13,22,25,27,29,44