JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 21, 39-58 (2005)

A Novel Approach to the Fixed Channel Assignment Problem*

REI-HENG CHENG, CHANG WU YU+ AND TUNG-KUANG WU++
Department of Information Management
Hsuan Chuang University
Hsinchu, 300 Taiwan
E-mail: rhc@hcu.edu.tw
+*Department of Computer Science and Information Engineering*
Chung Hua University
Hsinchu, 300 Taiwan
E-mail: cwyu@chu.edu.tw
++*Department of Information Management*
Ming Hsin University of Science and Technology
Hsinchu, 304 Taiwan
E-mail: tkwu@mis.must.edu.tw

A critical task in the design of a cellular radio network is to determine a spectrum-efficient and conflict-free allocation of channels among the cells. In this paper, we propose a new approach to the Fixed Channel Assignment (FCA) problem. By preserving the co-site channel constraint throughout our algorithm and adopting a fine-tuning procedure to escape from a local minimum, we reduce the overall execution time and improve the convergence rate. Simulation results show that our algorithm achieves a very high rate of convergence to solutions for eight benchmark problems. Furthermore, the number of iterations our algorithm requires is fewer than previous results.

Keywords: channel assignment, FCAP, cellular network, algorithm, mobile communication

1. INTRODUCTION

The growing popularity for mobile communication services increases the need for efficient use of the limited frequency spectrum. A common way to increase frequency spectrum utilization is to adopt the cellular structure approach, which spatially divides the geographical region into a number of cells. A base station (BS) is established in each cell, and each mobile station (MS) in the cell communicates through the BS via a channel. In such a frequency reuse system, MSs in different non-interfering cells may simultaneously use the same frequency channel. A critical task in the design of such a cellular radio network is to determine a spectrum-efficient and conflict-free allocation of channels among the cells.

Generally, there are three types of constraints that need to be considered, which include (1) the *co-channel constraint* (CCC), where the same channels cannot be assigned

---

Received June 4, 2002; revised February 17 & November 17, 2003; accepted April 15, 2004.
Communicated by Nen-Fu Huang.
* This research was supported by the National Science Council, Taiwan, R.O.C., under grant NSC 89-2213-E-364-002.

to certain pair of cells simultaneously; (2) the *adjacent channel constraint* (ACC), where channels adjacent in the frequency spectrum cannot be assigned to adjacent cells simultaneously; and (3) the *co-site constraint* (CSC), where channels assigned in the same cell must have a minimal separation in frequency from every other channel. We call these three constraints the *electromagnetic compatibility constraints* (EMC) [4, 9-11, 13, 17]. Allocating the channels among the cells while satisfying both the traffic demand and EMC is the main objective of the *channel assignment problem*.

The channel assignment problem takes two forms: (1) the *fixed channel assignment problem* (FCAP), where channels are permanently allocated to each cell [4, 10, 13]; and (2) the *dynamic channel assignment problem* (DCAP) [16, 18], where all channels that are available for every cell are allocated dynamically upon request [1].

The FCA problem has been studied extensively [4, 9-13, 17]. It has been shown that this problem is equivalent to a generalized graph-coloring problem, which is NP-hard [11, 12]. Such problems require the use of extremely time-consuming algorithms to obtain exact solutions. It is, therefore, necessary to use more time-efficient algorithms that, however, cannot guarantee optimal solutions. Most of the previous efforts have focused on developing approximation algorithms [17]. These include graph-theoretic [7], neural network [4, 11], simulated annealing [3], and genetic algorithms [13]. However, neural-network-based algorithms typically yield only sub-optimal solutions [17]. The simulated annealing approach, although it may be more flexible, is easily trapped in a local minimum, which it cannot escape without expending a lot of computation time [17]. In [6], Gamst derived some lower bounds for the minimal number of channels required. Other authors [8, 15, 17] provided other lower bounds, which are tighter in some cases.

In [4], the authors proposed a neural network-based parallel algorithm. It first fixes all the output values and updates the input values sequentially. Then it fixes all the input values and updates all the output values. Four heuristics are used to improve the convergence rate. Channel assignments in one or more cells are also fixed in order to reduce the convergence time. In [10], the authors proposed an algorithm that is based on a modified Hopfield neural network. Two different initialization methods and three update procedures with various updating orders were developed in the hope of improving the convergence rate and reducing the number of iterations. In [13], a genetic algorithm was proposed to solve the FCA problem. In this approach, the minimum separation encoding scheme is used to ensure that CSC constraints are not violated and to speed up the genetic computing process.

In this paper, we propose a new approach to solving the fixed channel assignment problem. By preserving the CSC throughout our algorithm, we can reduce the time required to reach convergence. A number of new techniques are also employed to increase the rate of convergence to a feasible solution. Simulation results show that our algorithm achieves a rate of convergence to solutions of 100% for eight benchmark problems. To the best of our knowledge, no previous algorithms could achieve such a high convergence rate. Furthermore, the number of iterations our algorithm requires is fewer than all previous results. Such significant results indicate that our new approach is indeed an effective and efficient method for solving the fixed channel assignment problem.

The remainder of this paper is organized as follows. In section 2, some necessary definitions are given. Our new approach is described in section 3, while simulation results and analysis are provided in section 4. Finally, section 5 concludes the paper.

2. DEFINITIONS

A system of n cells is represented by n vectors $X = \{x_1, x_2, \ldots, x_n\}$. We assume that the channels are equally spaced in the frequency domain and are ordered from the low-frequency band to the high-frequency band with numbers 1, 2, …, m. We use an $n \times n$ nonnegative symmetric matrix C, called a *compatibility matrix* [4, 7, 9, 10, 13, 17], to represent EMC. Each diagonal element c_{ii} (where $i = j$) in C represents the CSC, and the rest of the elements, c_{ij} (where $i \neq j$), represent the ACC or CCC. A demand vector $R = \{r_1, r_2, \ldots, r_n\}$ describes the channel requirements for each cell. Each element r_i in R represents the minimal number of channels to be assigned to cell x_i.

The FCAP is specified by the triple (X, R, C), where X is a cell system, R is a requirement vector, and C is a compatibility matrix. Let $N = \{1, 2, \ldots, m\}$ be a set of available channels, and let H_i be the subset of N assigned to x_i. The objective of FCAP is to find an assignment $H = \{H_1, H_2, \ldots, H_n\}$, which satisfies the following conditions:

$$|H_i| = r_i, \text{ for } 1 \leq i \leq n, \tag{1}$$

$$|h - h'| \geq c_{ij}, \text{ for all } h \in H_i, h' \in H_j, \text{ where } 1 \leq i \leq n \text{ and } 1 \leq j \leq n, i \neq j, \tag{2}$$

and

$$|h - h'| \geq c_{ii}, \text{ for all } h, h' \in H_i, \text{ where } h \neq h', \tag{3}$$

where $|H_i|$ denotes the number of channels in the set of H_i. We call such an assignment an *admissible assignment*. Notations used in this paper are summarized in Table 1.

Table 1. Notations used in this paper.

n	Number of cells in the mobile network
m	Number of available channels in the mobile network
R	Required channel number matrix, $R = (r_i)$
r_i	Required number of channels for cell i, $1 \leq i \leq n$
C	Compatibility matrix, $C = (c_{ij})$
c_{ij}	Minimal channel separation between channels in cells i and j, $1 \leq i, j \leq n$
d_{csc}	Minimal distance between channels of the same cell for CSC. The d_{csc} for cell i is c_{ii}.
ACC	Adjacent channel constraint
CCC	Co-channel constraint
CSC	Co-site constraint

In our channel assignment algorithm, we define an energy function E to keep track of the status of the simulation system. When E falls to zero, we find an admissible assignment. The energy function E is defined as follows:

$$E = \sum_{i=1}^{n}\sum_{a=1}^{m}\sum_{j=1, j\neq i}^{n}\sum_{b=1}^{m} p(i,a) * Obj(i,a,j,b) * p(j,b) + \sum_{i=1}^{n}\sum_{a=1}^{m}\sum_{b=1, b\neq a}^{m} p(i,a) * Obj(i,a,i,b) * p(i,b), \tag{4}$$

where

$$p(i,a) = \begin{cases} 1, & \text{if the channel } a \text{ is assigned to cell } x_i, \\ 0, & \text{otherwise.} \end{cases}$$

$$Obj(i,a,j,b) = \begin{cases} 0, & \text{if the distance between channel } a \text{ and } b \text{ is greater or equal to } c_{ij}, \\ & \text{that is, } |a-b| \geq c_{ij}, \\ 1, & \text{otherwise.} \end{cases}$$

$Obj(i, a, j, b)$ is set to 1 if the assignment of channel a to cell x_i and channel b to cell x_j violates the EMC. As a result, E represents the total number of such violations in an assignment. We can then use E to evaluate the extent of violations in current assignments.

3. THE CSC-PRESERVING FCA ALGORITHM

Our new FCA algorithm includes three main procedures. First, the *initializing procedure* assigns a set of channels that satisfy the CSC to each cell without considering ACC or CCC. Then the *updating procedure* tries to reduce the energy function E by changing the current assignment without violating the CSC constraints. If the energy function E is reduced to zero, then we can successfully obtain an admissible assignment. On the other hand, if E stays unchanged, the system is in a *locked* state. An *unlocking procedure* is then activated. The unlocking procedure first tries to find free channels from neighboring cells to resolve these conflicts. If it is not successful, the simulation system randomly resets the channel assignment of each cell, again without violating the CSC constraints. Then it goes back to the updating procedure until the simulation process stops. If the energy function of the system is greater than zero after the maximum number of iterations is reached, the simulation system terminates and fails to converge (the maximum number of iterations was set to 500 in our simulations).

The essential ideas behind our algorithm are described as follows. First of all, we preserve the CSC constraints for each cell throughout the channel assignment procedure. Assigning channels to cells in the FCA problem involves the consideration of many inter-dependent factors. If we can limit one or more of these factors, we might have more control over the FCA process and improve the efficiency of our algorithm. We choose to preserve CSC because it is the only EMC constraint that can be locally determined within each cell. The other essential feature of our algorithm is the adoption of a fine-tuning procedure to escape from local minima if certain conditions are met (e.g., if the simulation system is very close to obtaining a successful assignment). As will be shown later, by preserving the CSC and adopting a fine-tuning procedure, we can ensure that the energy function E will converge to zero much more quickly than without doing so.

Without considering the CCC or ACC constraints, we notice that cells with a high channel requirement or with large d_{csc} have more difficulty obtaining an assignment that satisfies the CSC constraints. According to our simulation results, these difficult cells are bottlenecks in the channel assignment procedure. If we can tackle these difficult cells first, it is more likely that we will be able to obtain an admissible assignment quickly. Based upon this intuition, we define the *feasibility function* of a cell as follows:

$$feasibility(x_i) = m - r_i * c_{ii}. \tag{5}$$

A cell with more feasible combinations of channel assignments that satisfy the CSC constraints has a higher feasibility value. Intuitively, the value of the feasibility function indicates the flexibility of a cell in terms of channel assignment. The channel assignment procedure will then proceed according to the ascending order of the calculated feasibility values of the cells.

Our new approach is summarized in the CSC-preserving algorithm described as follows. (The flow chart of the algorithm is shown in Fig. 1.) Note that, throughout the entire process, none of these temporary assignments violates the CSC constraints. As a result, the second part of the energy function E will always be zero. Finally, the energy function can be further simplified as shown below:

$$E = \sum_{i=1}^{n}\left(\sum_{a=1}^{m}\left(\sum_{j=1,\, j\neq i}^{n}\left(\sum_{\substack{b=a-(c_{ij}-1)\\ 1\le b\le m}}^{a+(c_{ij}-1)} p(i,a)*Obj(i,a,j,b)*p(j,b)\right)\right)\right). \tag{6}$$

The CSC-Preserving Algorithm

Step 1: Determine an assignment to each cell using the initializing procedure.

Step 2: Process all the cells sequentially in ascending order according to the value of the feasibility function of each cell. For each cell, we determine a new assignment by applying the updating procedure.

Step 3: Compute the energy function E. If $E = 0$, the algorithm ends. An admissible assignment is then successfully obtained.

Step 4: If E causes no changes when compared with the assignment prior to Step 2, launch the unlocking procedure and go to Step 3. Otherwise, go to Step 2 for another iteration until the maximum number of iterations is reached.

In the following, we provide more detailed descriptions of the three major procedures, including the initializing, updating, and unlocking procedures.

3.1 The Initializing Procedure

Initially, we assign channel set $H_i = \{s, s + c_{ii}, s + 2 \times c_{ii}, \ldots, s + (r_i - 1) \times c_{ii}\}$ to the cell x_i where s is a random number between 1 and $m - (r_i - 1) \times c_{ii}$. This is a straightforward way to assign each cell so that the assignment satisfies the CSC and the channel requirement. The procedure is shown below.

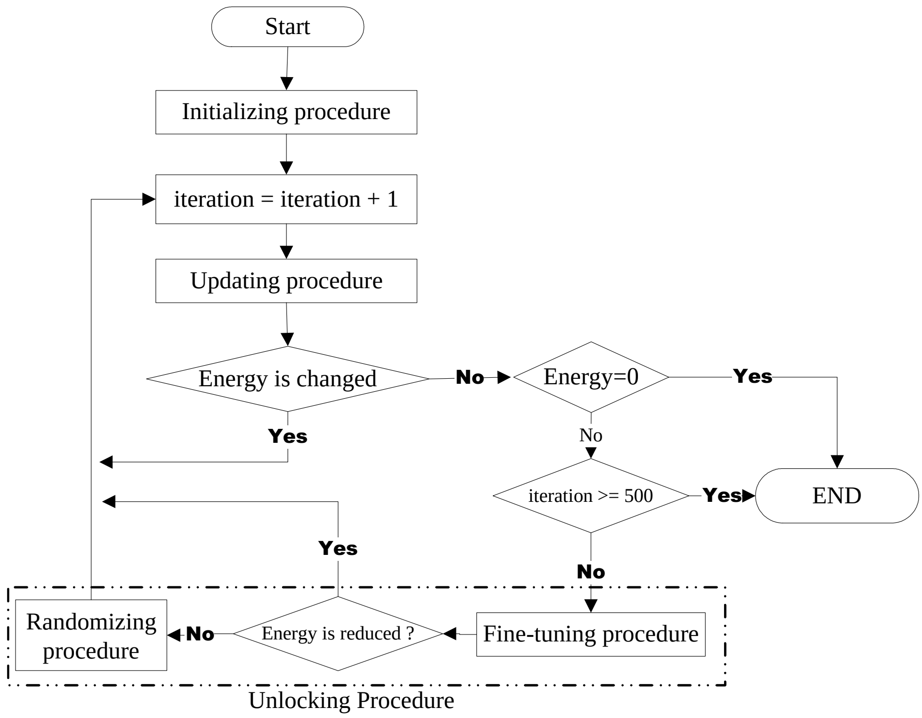

Fig. 1. The flow chart of the CSC-preserving FCA algorithm.

The Initializing Procedure

Perform the following steps for each cell x_i.

Step 1: Randomly set s to be an integer between 1 and $m - (r_i - 1) \times c_{ii}$.

Step 2: Assign channel set $H_i = \{s, s + c_{ii}, s + 2 \times c_{ii}, \ldots, s + (r_i - 1) \times c_{ii}\}$ to cell x_i.

3.2 The Updating Procedure

The updating procedure proceeds by processing each cell sequentially to reduce the energy E and to preserve the CSC constraints at the same time.

The support function defined below is used to select the most promising channel set for each cell. The degree of support for assigning a channel to a cell is computed by counting the votes from the neighbors of that cell. A channel a that is assigned to a cell x_i without violating the EMC will be given a positive vote by its neighboring cells. In the opposite case, a negative vote will be given. Accordingly, the *support* for assigning channel a to cell x_i can be defined as follows:

$$support(a, x_i) = \sum_{j=1,\, j \neq i}^{n} \left(r_j - \sum_{\substack{b=a-(c_{ij}-1) \\ 1 \le b \le m}}^{a+(c_{ij}-1)} S(i, a, j, b) * p(j, b)\right), \tag{7}$$

where

$$p(j,b)=\begin{cases}1, & \text{if the channel } b \text{ is assigned to cell } x_j,\\ 0, & \text{otherwise.}\end{cases}$$

$$S(i,a,j,b)=\begin{cases}1, & \text{if } |a-b|\geq c_{ij},\\ -1, & \text{otherwise.}\end{cases}$$

Intuitively, the value of the support function of a channel assigned to a particular cell represents the degree of support from its neighboring cells for the particular channel assignment.

The updating procedure is described as follows.

The Updating Procedure

Perform the following steps for each cell x_i in ascending order of the value of the feasibility function of each cell.

Step 1: Calculate the support function of the cell for each channel.

Step 2: Find a set of channels that have high support values and preserve CSC at the same time.

- **2.1** Create a list *ChList* of channel set {1, 2, …, m} sorted in descending order of the support function calculated in Step 1.
- **2.2** Create an empty list *SeList*, where *SeList* represents one of potential sets of channels that can be assigned to cell x_i.
- **2.3** Pick the first channel a and delete it from *ChList*.
- **2.4** If channel a conflicts with the channel(s) in *SeList* (due to violation of the CSC constraints), go to Step 2.3.
- **2.5** Insert channel a into *SeList* so that *SeList* is sorted in ascending order.
- **2.6** Determine whether channel a can be preserved in *SeList* or not. By means of function f (defined below), calculate that the maximal possible number of channels which preserves the CSC constraints and contains all the channels in *SeList*. If the value of f is less than r_i, remove channel a from *SeList*.

$$f=\lfloor Sel_1/d_{\mathrm{csc}}\rfloor+\sum_{n=2}^{top}\lfloor (Sel_n-Sel_{(n-1)})/d_{\mathrm{csc}}\rfloor+\lfloor (m-Sel_{top})/d_{\mathrm{csc}}\rfloor. \tag{8}$$

In practice, Eq. (8) can be further modified to save computation time, as illustrated in Eq. (9):

$$f = oldf + \lfloor(Sel_i - Sel_{(i\text{-}1)})/d_{\mathrm{csc}}) + \lfloor(a - Sel_{(i\text{-}1)})/d_{\mathrm{csc}}\rfloor + \lfloor(Sel_i - a)/d_{\mathrm{csc}}\rfloor. \tag{9}$$

Here, *oldf* is the value of f computed in the pervious channel selection, Sel_i is the ith channel in *SeList*, and *top* is the number of channels in *SeList*.

- **2.7** If the number of channels in *SeList* is equal to r_i, then go to Step 2.8. Otherwise, go to Step 2.3
- **2.8** The channels in *SeList* comprise a set of channels that have a high support value and preserve the CSC.

Step 3: Compared with the previous channel assignment, the new assignment to cell x_i receives more support from the other cell x_j; thus the new assignment is adopted. Otherwise, the channel assignments of cell x_i remain unchanged.

3.3 The Unlocking Procedure

When the value of the energy function E is not equal to zero and can not be reduced any further, the system converges to a local minimum. Usually, other algorithms will change some of channel assignments randomly in order to make system unstable (i.e., to escape from the local minimum) [10], and re-apply the updating procedure. As a result, all the efforts made in the previous steps are wasted.

Since the updating procedure makes decisions based only on a cell's neighboring state in order to reduce conflicts in the current assignment, it seems reasonable to start from a local minimum state and initiate a more global check to resolve the remaining conflicts. In this way, we expect that the simulation system may get around the local minimum and save all the efforts that were made in the previous steps. In fact, according to our analysis of the simulation process, a few channel adjustments to cells with inter-conflicting channels usually cause a chain of actions that will resolve the current conflict and lead to a state with fewer (or even no) conflicts. The fine-tuning procedure is based on the above idea.

However, there may exist complicated conflicts that the fine-tuning procedure can not resolve. In this case, we apply a CSC preserving randomizing procedure to randomly generate CSC preserving channel assignments as the initial state for the next run of the updating procedure.

In summary, the unlocking procedure that we use to escape from a local minimum consists of the fine-tuning and randomizing procedures described below and depicted in Fig. 1.

3.3.1 The fine-tuning procedure

The fine-tuning procedure is designed to find out channel assignments (if any exist) that do not satisfy the ACC or the CCC constraints and to try to resolve these conflicts.

We say that a channel a (of cell x_i) *conflicts with* channel b (of cell x_j, where $i \neq j$) if the pair of channels a and b does not satisfy the ACC or CCC constraints.

The *important set* I_i of cell x_i is defined as follows:

$$I_i = \{x_k \mid \text{where } k \neq i \text{ and } c_{ik} \neq 0\}.$$

By the above definition, the important set I_i contains cells (except for cell x_i) whose channels might conflict with cell x_i.

A channel f of cell x_i is *free* if $f \notin H_i$, and if f does not conflict with any selected channel for each cell $x_k \in I_i$. We define $\Gamma(f) = \{g \mid g \text{ conflicts with } f, \text{ where } f \notin H_i, g \in H_j, x_j \in I_i \text{ and } i \neq j\}$.

Suppose $a \in H_i$. We say that channel a is *replaceable* if there exists $f \notin H_i$ such that $H_i - \{a\} \cup \{f\}$ satisfies CSC, and

(1) f is free in cell x_i or
(2) each channel in the set $\Gamma(f)$ is replaceable.

When f makes a replaceable, we call f the *successor* of a.

An example is shown in Fig. 2. Here, nodes denote cells, and a circle on each node represents the current channel assignment. An arc with an arrow between two channels indicates that these two channels conflict with each other. In addition, all free channels are enclosed in rectangles. In Fig. 2, channel b conflicts with channel a_0, while f_3 and f_4 are both free. Suppose that $H_i - \{a_i\} \cup \{f_i\}$ satisfies CSC for all i. In the following, we will show that a_0 is replaceable. First of all, since f_3 and f_4 are free and $\Gamma(f_2) = \{a_3, a_4\}$, a_2 is replaceable by definition. Secondly, the fact that $\Gamma(f_0) = \{a_1, a_2\}$, a_2 is replaceable, and that f_1 is free also means that a_0 is replaceable. In addition, f_0, f_1, f_2, f_3, and f_4, are successors of a_0, a_1, a_2, a_3, and a_4 respectively. We call the tree-like graph in Fig. 2 a *replacing tree* of a_0, since it represents a possible sequence of channel re-assignments (replacements) for resolving a conflict due to the assignment of a_0 to some cell. Note that if a_0 is replaceable, the leaves of its replacing tree are free.

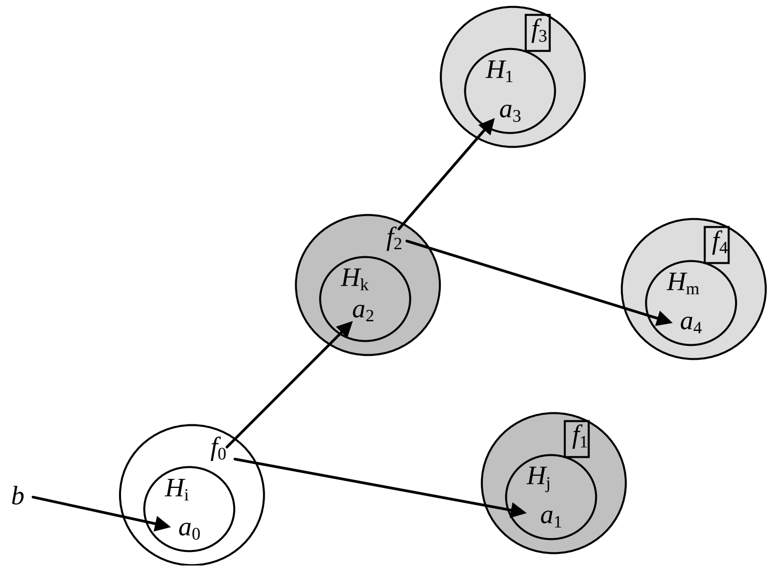

Fig. 2. An example showing that channel a_0 is replaceable when $\Gamma(f_0)$ is replaceable.

It is easy to see that if channel a conflicts with channel b, and if a is replaceable, then the conflict resulting from a and b can be removed when all conflicting channels on the replacing tree of a are replaced with their corresponding successors. Further conflicts might be introduced but will be limited to the set $\{b\} \cup \{f_i \mid f_i$ is a successor in the replacing tree of $a\}$.

The fine-tuning procedure is described as follows.

Supposed that a channel $a \in H_i$ conflicts with b

Step 1: Try to select an unmarked channel f from $N - H_i$ such that $H_i - \{a\} \cup \{f\}$ satisfy the CSC of cell x_i. Mark f. If no such f exists, go to Step 4.

Step 2: If f is free, replace channel a with channel f in cell x_i; that is, let $H_i = H_i - \{a\} \cup \{f\}$ and go to Step 4.
Otherwise, if each channel in $\Gamma(f)$ is replaceable and no conflict occurs in $\{b\} \cup \{f_i \mid f_i$ is a successor in this replacing tree$\}$, then let $H_j = H_j - \{a_j\} \cup \{f_j\}$ for all f_j, where f_j is the successor of a_j in the replacing tree of a. Go to Step 4.

Step 3: Go to Step 1.

Step 4: END

Our fine-tuning procedure extends the replacing tree dynamically. Based on our simulation results (shown in section 4), this procedure resolves conflicts in most cases, which in turn leads the simulating system away from the local minimum quickly and raises the rate of convergence dramatically.

3.3.2 The CSC preserving randomizing procedure

The randomizing procedure processes cells one by one. First, we separate all available channels into r_i channel intervals for each cell x_i. The kth channel interval of x_i can be represented by the pair ($minpos_k$, $maxpos_k$), $k = 0, \ldots, r_i - 1$. Next, we randomly select an arbitrary channel interval of the processed cell. Within the selected channel interval, we select an arbitrary channel. The neighboring channel interval will be adjusted so that the next selection of a channel from this channel interval does not violate CSC. The remaining channels can be decided similarly. Note that the procedure randomly changes the channel assignment of each cell in order to make the system unstable but still preserves CSC for each cell. The procedural steps for changing the channel assignments of cell x_i are listed as follows.

The Randomizing Procedure

Step 1: Set every channel interval ($minpos_k$, $maxpos_k$) for the processed cell, where $0 \le k \le r_i - 1$, $minpos_k = k \times c_{ii}$, and $maxpos_k = m - (r_i - k - 1) \times c_{ii}$.

Step 2: Set $currentpos_x$ as the xth channel according to the increasing order of the values of channel set currently assigned to the processed cell.

Step 3: Randomly select a channel interval ($minpos_x$, $maxpos_x$), where $0 \le x \le r_i - 1$.

Step 4: Randomly select a channel y from ($minpos_x$, $maxpos_x$). That is, $minpos_x \le y \le maxpos_x$.

Step 5: If $currentpos_x \le y$, set $dir = 1$ and $limit = r_i$. Otherwise, set $dir = -1$ and $limit = -1$.

Step 6: Set $currentpos_x = y$.

Step 7: If $dir = 1$, let $minpos_x + 1 = max\{minpos_x + 1, y + c_{ii}\}$. Otherwise, let $maxpos_x - 1 = min\{maxpos_x - 1, y - c_{ii}\}$.

Step 8: Determine the next channel interval by setting $x = x + dir$. If $x = limit$, then stop this procedure.

Step 9: Randomly select a channel y from ($minpos_x$, $maxpos_x$). Go to Step 6.

4. SIMULATION RESULTS AND COMPLEXITY ANALYSIS

4.1 Simulation Results

Our simulator was developed on a personal computer with an Intel Pentium IV 2G CPU and 256 MB of main memory (RAM). The program was coded in Delphi 5. Eight benchmark problems taken from [4], which were also used as test samples in [10, 13], were examined. Table 2 summarizes the characteristics of these eight problems, together with all the demand vectors, D_i, and the compatibility matrices, C_i. A compatibility matrix specifies the CSC, CCC and ACC constraints among the base stations. On the other hand, a demand vector specifies the number of channels that each site demands. Take problem 1 in Table 2 as an example; it specifies that there are 4 sites, with a total of 11 available channels. In addition, each site's channel demand is listed in D_1, with EMC constraints specified in C_1. The iteration number is the number of updating procedures executed for all cells before E reaches zero. To acquire the average number of iterations and the convergence rates, one hundred simulation runs were performed.

The simulation results show that our algorithm achieved 100% convergence to solutions for all eight benchmarking problems. Furthermore, the number of iterations our algorithm required was fewer than all previous results. (Please refer to Table 3; the outcomes reported in the three related papers [4, 10, 13] are also listed for comparison.) It is worth noting that our algorithm has an updating procedure that is similar to those used in [4] and [10]. This means that fewer iterations also translates into less computation time. On the other hand, the genetic algorithm proposed in [13] tends to converge slower since it uses a more random method to generate new combinations for potential solutions. Furthermore, our proposed fine-tuning procedure would try carefully to replace some of the assignments to escape from a local minimum. Other algorithms, in contrast, randomly break their current combination and start all over again from scratch. This is another reason why our algorithm performs better than the others overall. The results and analysis presented above indicate that our approach is indeed an effective and efficient method for dealing with the fixed channel assignment problem.

The effect of preserving CSC constraints throughout the channel assignment process can be clearly seen in Fig. 3 (the result of problem 1 is omitted since it takes only one iteration even though the CSC constraints are not preserved), which shows the energy variation in the course of simulation for seven benchmarking problems. Take benchmark problem 8 as an example; we can see that if the CSC constraints are preserved, the values of the energy function stay at a much lower level. If the fine-tuning procedure is further applied, the simulation process quickly converges. Similar phenomena are also noted for the other benchmark problems.

To further investigate the effect of the fine-tuning procedure, a few terms are defined as follows. The depth of a replacing tree that corresponds to a channel assigned to a cell is defined as the *T-depth*. The number of conflict pairs t is used (in our simulation) as a threshold for determining whether to launch the fine-tuning or the randomizing procedure. If the number of conflicting pairs contained in one stable state exceeds a pre-defined value t, we skip the fine-tuning procedure and start the randomizing procedure. Otherwise, the fine-tuning procedure is initiated. First of all, as one can see in Table 4, five (out of the eight) benchmark problems achieve a convergence rate of 100%

Table 2. Specifications of the eight benchmark problems used in our simulation.

Problem #	number of radio cells (n)	number of channels (m)	Compatibility matrix (C)	Demand vector (D)
1	4	11	C_1	D_1
2	25	73	C_2	D_2
3	21	381	C_3	D_3
4	21	533	C_4	D_3
5	21	533	C_5	D_3
6	21	221	C_3	D_4
7	21	309	C_4	D_4
8	21	309	C_5	D_4

C_1	D_1
5 4 0 0	1
4 5 0 1	1
0 0 5 2	1
0 1 2 5	3

C_2	D_2
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	10
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	11
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	9
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	5
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	9
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	4
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	5
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	7
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	4
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	8
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	8
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	9
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	10
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	7
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	7
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	6
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	4
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	5
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	5
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	7
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	6
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	4
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	5
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	7
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	5

C_3	D_3
5 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0	8
1 5 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0	25
1 1 5 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0	8
0 1 1 5 1 0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 0	8
0 0 1 1 5 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0	8
1 0 0 0 0 5 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0	15
1 1 0 0 0 1 5 1 1 0 0 0 1 1 1 1 0 0 1 0 0	18
1 1 1 0 0 1 1 5 1 1 0 0 0 1 1 1 1 0 1 1 0	52
1 1 1 1 0 0 1 1 5 1 1 0 0 0 1 1 1 1 1 1 1	77
0 1 1 1 1 0 0 1 1 5 1 1 0 0 0 1 1 1 0 1 1	28
0 0 1 1 1 0 0 0 1 1 5 1 0 0 0 0 1 1 0 0 1	13
0 0 0 1 1 0 0 0 0 1 1 5 0 0 0 0 0 1 0 0 0	15
0 0 0 0 0 1 1 0 0 0 0 0 5 1 1 0 0 0 0 0 0	31
1 0 0 0 0 1 1 1 0 0 0 0 1 5 1 1 0 0 1 0 0	15
1 1 0 0 0 1 1 1 1 0 0 0 1 1 5 1 1 0 1 1 0	36
1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 5 1 1 1 1 1	57
0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 5 1 1 1 1	28
0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 5 0 1 1	8
0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 5 1 1	10
0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 5 1	13
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 5	8

C_4	D_4
7 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0	5
1 7 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0	5
1 1 7 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0	5
0 1 1 7 1 0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 0	8
0 0 1 1 7 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0	12
1 0 0 0 0 7 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0	25
1 1 0 0 0 1 7 1 1 0 0 0 1 1 1 1 0 0 1 0 0	30
1 1 1 0 0 1 1 7 1 1 0 0 0 1 1 1 1 0 1 1 0	25
1 1 1 1 0 0 1 1 7 1 1 0 0 0 1 1 1 1 1 1 1	30
0 1 1 1 1 0 0 1 1 7 1 1 0 0 0 1 1 1 0 1 1	40
0 0 1 1 1 0 0 0 1 1 7 1 0 0 0 0 1 1 0 0 1	40
0 0 0 1 1 0 0 0 0 1 1 7 0 0 0 0 0 1 0 0 0	45
0 0 0 0 0 1 1 0 0 0 0 0 7 1 1 0 0 0 0 0 0	20
1 0 0 0 0 1 1 1 0 0 0 0 1 7 1 1 0 0 1 0 0	30
1 1 0 0 0 1 1 1 1 0 0 0 1 1 7 1 1 0 1 1 0	25
1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 7 1 1 1 1 1	15
0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 7 1 1 1 1	15
0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 7 0 1 1	30
0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 7 1 1	20
0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 7 1	20
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 7	25

C_5
7 2 1 0 0 1 2 2 1 0 0 0 0 1 1 1 0 0 0 0 0
2 7 2 1 0 0 1 2 2 1 0 0 0 0 1 1 1 0 0 0 0
1 2 7 2 1 0 0 1 2 2 1 0 0 0 0 1 1 1 0 0 0
0 1 2 7 2 0 0 0 1 2 2 1 0 0 0 0 1 1 0 0 0
0 0 1 2 7 0 0 0 0 1 2 2 0 0 0 0 0 1 0 0 0
1 0 0 0 0 7 2 1 0 0 0 0 2 2 1 0 0 0 0 0 0
2 1 0 0 0 2 7 2 1 0 0 0 1 2 2 1 0 0 1 0 0
2 2 1 0 0 1 2 7 2 1 0 0 0 1 2 2 1 0 1 1 0
1 2 2 1 0 0 1 2 7 2 1 0 0 0 1 2 2 1 1 1 1
0 1 2 2 1 0 0 1 2 7 2 1 0 0 0 1 2 2 0 1 1
0 0 1 2 2 0 0 0 1 2 7 2 0 0 0 0 1 2 0 0 1
0 0 0 1 2 0 0 0 0 1 2 7 0 0 0 0 0 1 0 0 0
0 0 0 0 0 2 1 0 0 0 0 0 7 2 1 0 0 0 0 0 0
1 0 0 0 0 2 2 1 0 0 0 0 2 7 2 1 0 0 1 0 0
1 1 0 0 0 1 2 2 1 0 0 0 1 2 7 2 1 0 2 1 0
1 1 1 0 0 0 1 2 2 1 0 0 0 1 2 7 2 1 2 2 1
0 1 1 1 0 0 0 1 2 2 1 0 0 0 1 2 7 2 1 2 2
0 0 1 1 1 0 0 0 1 2 2 1 0 0 0 1 2 7 0 1 2
0 0 0 0 0 0 1 1 1 0 0 0 0 1 2 2 1 0 7 2 1
0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 2 2 1 2 7 2
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 2 2 1 2 7

Table 3. Comparison of our simulation results with previous solutions.

Problem	Our method (t = 24, *T-Depth* = 2)		Result from [4]		Result from [10]		Result from [13]	
	# of iteration	convergence rate(%)	# of teration	convergence rate(%)	# of iteration	convergence rate(%)	# of iteration	convergence rate(%)
1	1	100	21.2	100	NA	NA	1	100
2	5	100	294.0	9	279.9	62	26382	100
3	3	100	147.8	93	67.4	99	NA	NA
4	1	100	117.5	100	64.2	100	NA	NA
5	5	100	100.3	100	126.8	98	NA	NA
6	8.6	100	234.8	79	62.4	97	63152	92
7	4	100	85.6	100	127.7	99	NA	NA
8	5	100	305.6	24	151.9	52	79502	80

Table 4. Effect of fine-tuning (t = 24 & *T-depth* = 2) on the performance of the FCA simulation for the eight benchmark problems.

Fine-tuning? \ Problem #	1	2	3	4	5	6	7	8
Without fine-tuning								
Convergence rate	100%	20%	100%	100%	100%	93%	100%	97%
# of iterations	1	201.2	22.1	1	53.5	168.3	29.7	101
Average time spent in a simulation (*msec*)	7.3	14322.2	18520.6	1327.8	70455.9	69025	17924.5	61890
With fine-tuning								
Convergence rate	100%	100%	100%	100%	100%	100%	100%	100%
# of iterations	1	5	3	1	5	8.6	4	5
Average time spent in a simulation (*msec*)	7.3	355.5	2751.7	1327.8	6988.1	3466.6	2575.1	3233.7
Time spent in fine-tuning of a simulation (*msec*)	0	12	3	0	20	1.3	0.3	99.4

without application of the fine-tuning procedure. This means that for most of the benchmarking problems, preservation of CSC alone results in pretty good outcomes for our simulation. Secondly, based on the numbers associated with the eight benchmark problems, a little more time spent on the fine-tuning procedure turns out to improve the performance of the simulation very much. Not only does the simulation time drop significantly, but the number of iterations required to converge is also reduced. In other words, the fine-tuning procedure does effectively help lead the simulation system away from a local minimum and help the system converge quickly.

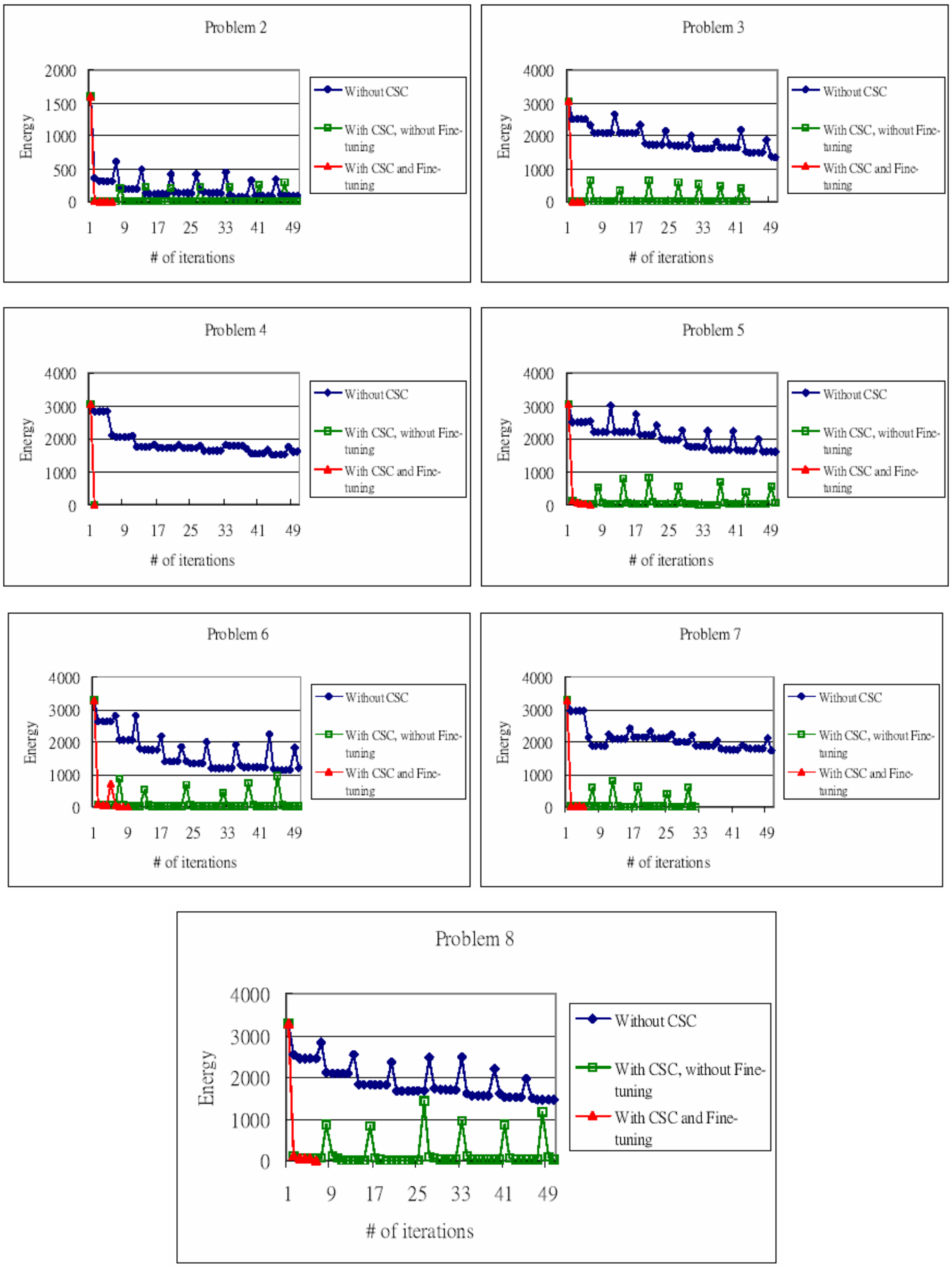

Fig. 3. Effects of preserving CSC constraints and using the fine-tuning procedure in our simulation.

An intuitive example is given in the following to illustrate how the fine-tuning procedure helps the simulation system escape from a local minimum. Table 5 depicts a snapshot of one of the stable states in our simulation for the solution of problem 6. According to the specifications of problem 6 (shown in Table 2), cells 11, 7 and 17 are neighbors of cell 9. This means that channels cannot be assigned to cell 9 and to cell 11, 7 or 17 simultaneously. As a result, assigning channel 0 to cell 9 and 11 introduces an ACC conflict as shown in Table 4. Unfortunately, during the updating procedure, this situation would not be improved any further since the arrangement induces only one conflict (with cell 11), and since any other combinations of assignments are no better than the current one. However, examining the channel assignment map (Table 5) further, we find out that even though assigning channel 21 to cell 11 introduces two ACC conflicts (with cells 7 and 17), these conflicts are resolvable since both cells 7 and 17 have unused free channels. By replacing channel 21 with its successor in cells 7 and 17, we can free up channel 21 for use in cell 9. As we can see in this example, the fine-tuning procedure indeed successfully leads the simulation system away from a local minimum state when the updating procedure can not make any further improvement.

Table 5. One of the stable states of problem 6.

# cell	Channels assigned (**A**) and Free channels (**F**)
7	**A:** 3 ,101,106,111,116,121,126,131,136,141,146,151,156,161,166,171,176,181,186,191,196,201,206,211,216 **F:**
9	**A:** 0,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,110,115,120,125,130,135,140,145,150, 179,184,189,194,199,204,209,214,219 **F:**
11	**A:** 0,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,110,115,120,125,130,135,140,145,150, 155,160,165,170,175,180,185,190,195,200,205,210,215,220 **F:**
17	**A:** 43,48,53,58,63,68,73,78,83,88,93,98,103,108,113,118,123,128,133,138,143,148,153,158,163,168,173,178, 183,188 **F:** 193,198,203,208,213,218

We also performed a number of simulations to study the effects of adjusting the values of *t* and *T-depth*. The results are shown in Figs. 4 and 5, respectively. (The results for both problems 1 and 4 are omitted from Figs. 4 and 5 since they take only one iteration and no fine tuning procedure is required.) Fig. 4 shows that the number of iterations and the simulation time decrease when we set the threshold for triggering the fine-tuning procedure lower (i.e., set the value of *t* higher). However, at some point, the increase of the *t* value does not affect the number of iterations or the duration of the simulation any further. Fig. 5 shows that variation of *T-depth* also has an effect on the overall performance though it does not seem to be very significant. This might be due to the choices of benchmark problems, in which cases the use of a low *T-depth* value can resolve most conflicts. A much more complex channel assignment might be needed to explore the effectiveness of adjusting the *T-depth* value.

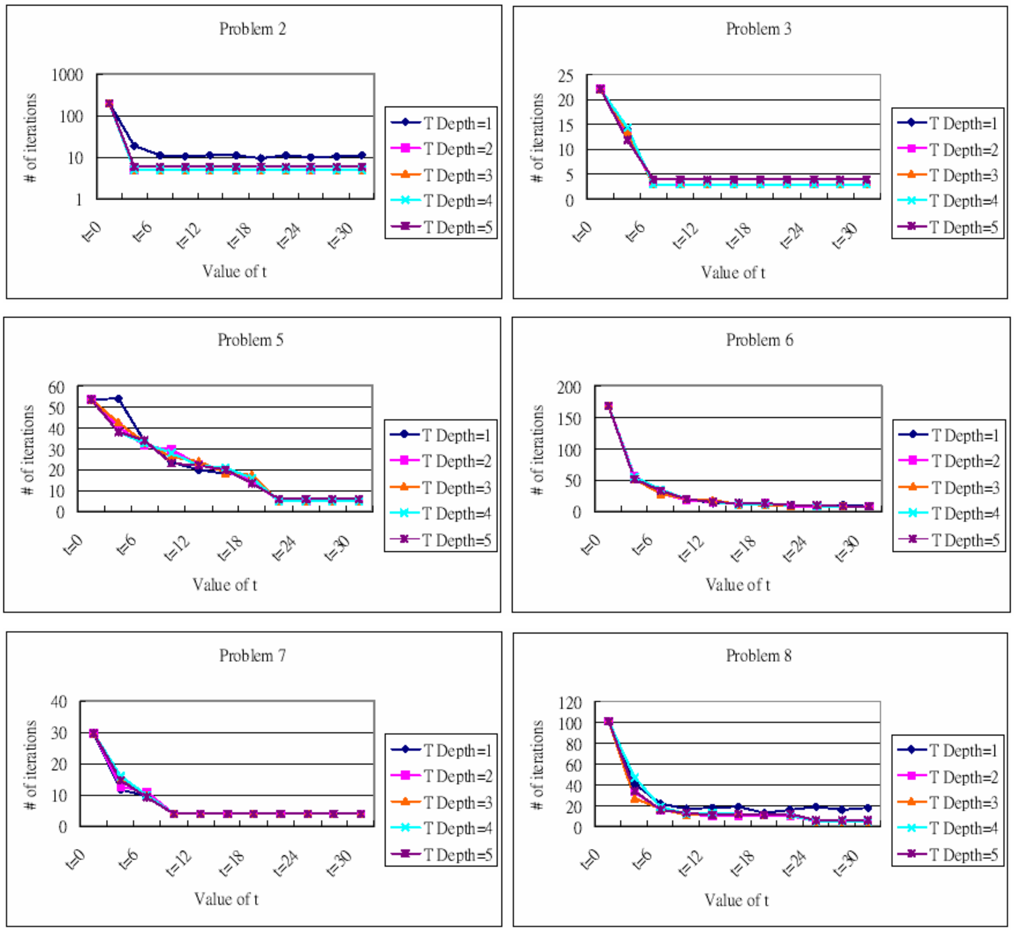

Fig. 4. Effect of the *t* value (used in the fine-tuning procedure) on the performance of our FCA simulation.

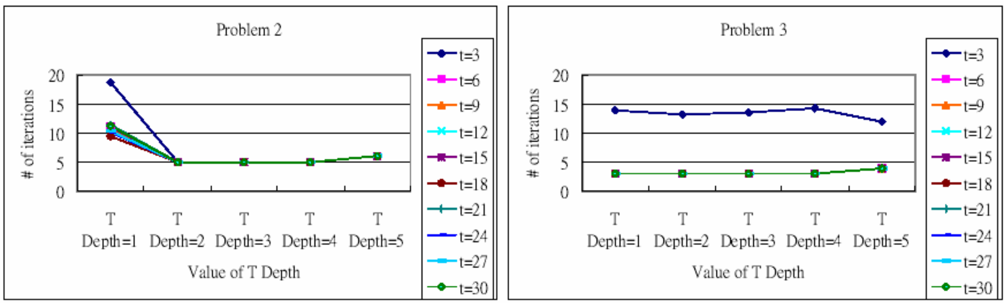

Fig. 5. Effect of *T-depth* (used in the fine-tuning procedure) on the performance of our FCA simulation.

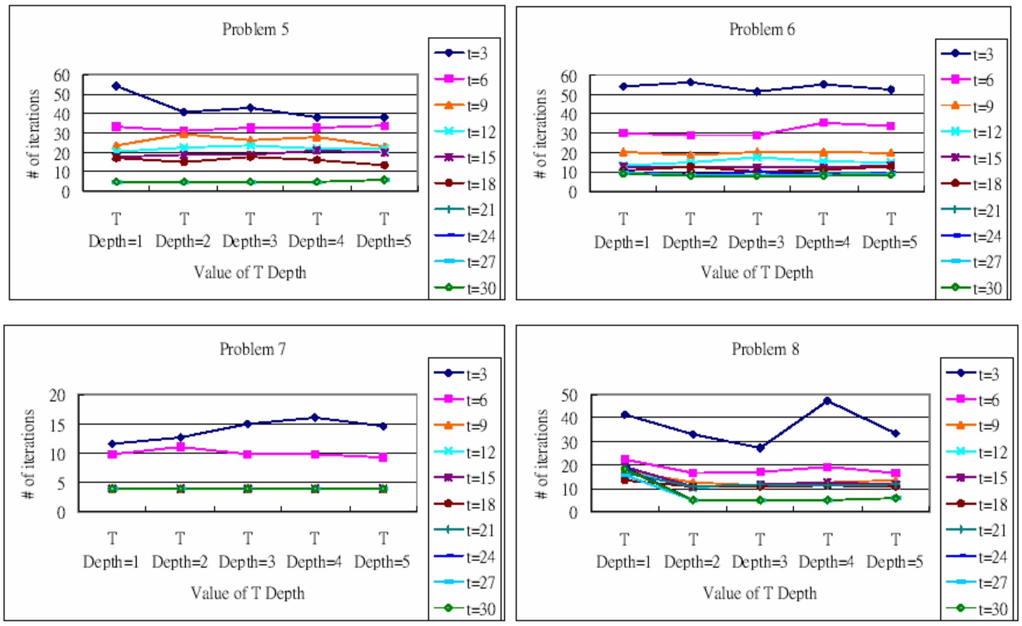

Fig. 5. (Cont'd) Effect of *T-depth* (used in the fine-tuning procedure) on the performance of our FCA simulation.

4.2 Complexity Analysis

The simulation results presented in the previous section show that our proposed algorithm can achieve a better (or the same) convergence rate with many fewer iterations. However, fewer iterations do not necessarily correspond to better performance. It remains to be proven that each iteration of our algorithm requires a similar amount (or even less) computation than the other ones require.

As shown in section 3, our CSC-preserving channel assignment algorithm consists of four steps. The computation order of each step is analyzed as follows.

Step 1: The initializing procedure takes $O(n * \max(r_i))$.

Step 2: The updating procedure can be further divided into three sub-steps as follows.

- **2.1** Compute the degree of support for m channels using Eq. (7). The computational complexity is $nm * (\max(c_{ij}))$.
- **2.2** Sort the m channels according to the degree of support. The computational complexity is $O(m\log m)$.
- **2.3** For each m channel, determine if the new channel assignment of cell i receives more support from the rest of the cells. Again, use Eq. (7) to do this. As a result, the computation complexity is the same as in Substep 2.1.

The combined complexity of Step 2 (Substeps 2.1, 2.2 and 2.3) should be multiplied by n since the updating procedure is processed once for every cell in each iteration.

Step 3: Compute the energy function E using Eq. (6). This results in $O(n^2m * \max(c_{ij}))$ complexity.

Step 4: For each conflict pair, the fine-tuning procedure is as follows.

4.1 Find a replaceable free channel f among the m channels. Since CSC has to be checked, this results in $O(m * d_{csc})$ complexity.

4.2 For each found replaceable free channel f, check if it conflicts with the assigned channels of the rest of the cells. This takes $O(n * \max(c_{ij}))$ time.

The combined computational complexity of the fine-tuning procedure is, thus, $O((m * (d_{csc} + n * \max(c_{ij})))^{T\text{-}depth})$. However, since this step is not usually required in every iteration, it does not contribute to the overall complexity order.

Similar analyses were also applied to the other three algorithms listed in [4, 10, 13]. In [4], the researchers computed $U_{ij} = U_{ij} + \Delta U_{ij}$ (where $1 \le i \le n$, $1 \le j \le m$) in each iteration, so the computational complexity of ΔU_{ij} being $O(2 * m + n * \max(c_{ij}))$. That made the overall complexity $O(2nm^2 + n^2m * \max(c_{ij}))$. In [10], it took $O(n^2m^2)$ time to compute the total energy (Eqs. (11) and (21) listed in [10]). The resulting order of computation complexity was, thus, not less than $O(n^2m^2)$. Finally, in [13], the overall complexity was $O(C_2^P n^2m * \max(c_{ij}))$, where C_2^P denotes the number of ways of selecting 2 objects from P distinct objects. When P was set as 10 in their experiment, the computational complexity was $O(45n^2m * \max(c_{ij}))$.

As one can see, for each iteration, our proposed algorithm's overall required computational complexity is $O(n(nm(\max(c_{ij})) + m\log m + nm(\max(c_{ij}))) + n^2m(\max(c_{ij}))) = O(3n^2m(\max(c_{ij})) + nm\log m)$. With $n \le 25$, $m \le 533$, and $\max(c_{ij}) \le 2$ for the eight benchmark problems, it is clear that each iteration of our algorithm is less complex than that of the other three algorithms.

5. CONCLUSIONS

In this paper, we have proposed an algorithm that preserves CSC throughout the FCA process. We use many techniques to improve the rate of convergence and to speed up the simulation process. According to the simulation results obtained for the eight benchmark problems, our algorithm outperforms the other existing algorithms. In addition, the freedom of adjusting the depth of the replacing tree in the fine-tuning procedure allows us to tackle problems of varying difficulty. Though the simulations are run in the sequential mode, all the operations can be arranged for parallel execution through careful design. Improving the initializing and fine-tuning stages of the new approach will be a subject of our future work. It would also be quite interesting to study whether the techniques developed in this paper can be applied to solve other optimization problems.

REFERENCES

1. P. T. H. Chan, M. Palaniswami, and D. Everitt, "Neural network-based dynamic channel assignment for cellular mobile communication systems," *IEEE Transactions on Vehicular Technology*, Vol. 43, 1994, pp. 279-287.
2. K. Chang, J. Kim, C. Yim, and S. Kim, "An efficient borrowing channel assignment

scheme for cellular mobile system," *IEEE Transactions on Vehicular Technology*, Vol. 47, 1998, pp. 602-608.
3. M. Duque-Anton, D. Kunz, and B. Ruber, "Channel assignment for cellular radio using simulated annealing," *IEEE Transactions on Vehicular Technology*, Vol. 42, 1993, pp. 14-21.
4. N. Funabiki and Y. Takefuji, "A neural network parallel algorithm for channel assignment problems in cellular radio networks," *IEEE Transactions on Vehicular Technology*, Vol. 41, 1992, pp. 430-436.
5. H. Furukawa and Y. Akaiwa, "Design of underlaid microcells in umbrella cell system," *IEICE Transactions on Communication*, Vol. E81-B, 1998, pp. 762-769.
6. A. Gamst, "Some lower bounds for a class of frequency assignment problem," *IEEE Transactions on Vehicular Technology*, Vol. VT-35, 1986, pp. 8-14.
7. A. Gamst and W. Rave, "On frequency assignment in mobile automatic telephone system," in *Proceedings of IEEE GLOBECOM*, 1982, pp. 309-315.
8. S. C. Ghosh, B. P. Sinha, and N. Das, "Channel assignment using genetic algorithm based on geometric symmetry," *IEEE Transactions on Vehicular Technology*, Vol. 52, 2003, pp. 860-875.
9. S. Kim and S. L. Kim, "A two-phase algorithm for frequency assignment in cellular mobile systems," *IEEE Transactions on Vehicular Technology*, Vol. 43, 1994, pp. 542-548.
10. J. S. Kim, S. H. Park, P. W. Dowd, and N. M. Nasrabadi, "Cellular radio channel assignment using a modified Hopfield network," *IEEE Transactions on Vehicular Technology*, Vol. 46, 1997, pp. 957-967.
11. D. Kunz, "Channel assignment for cellular radio using neural network," *IEEE Transactions on Vehicular Technology*, Vol. 40, 1991, pp. 188-193.
12. R. Mathar and J. Mattfeldt, "Channel assignment in cellular radio networks," *IEEE Transactions on Vehicular Technology*, Vol. 42, 1993, 647-656.
13. Y. Ngo and V. O. K. Li, "Fixed channel assignment in cellular radio networks using a modified genetic algorithm," *IEEE Transactions on Vehicular Technology*, Vol. 47, 1998, pp. 163-171.
14. K. Sallberg, B. Stavenow, and B. Eklundh, "Hybrid channel assignment and reuse partitioning in a cellular mobile telephone system," in *Proceedings of 37th IEEE Vehicular Technology Society Conference*, 1987, pp. 405-411.
15. S. Sarkar and K. N. Sivarajan, "Channel assignment algorithms satisfying cochannel and adjacent channel reuse constraints in cellular mobile networks," *IEEE Transactions on Vehicular Technology*, Vol. 51, 2002, pp. 954-967.
16. K. Shoarinejad, J. L. Speyer, and G. J. Pottie, "Integrated predictive power control and Dynamic channel assignment in mobile radio systems," *IEEE Transactions on Wireless Communications*, Vol. 2, 2003, pp. 976-988.
17. W. Sung and W. S. Wong, "Sequential packing algorithm for channel assignment under cochannel and adjacent channel interference constraint," *IEEE Transactions on Vehicular Technology*, Vol. 46, 1997, pp. 676-685.
18. K. L. Yeung and T. S. Peter Yum, "Compact pattern based dynamic channel assignment for cellular mobile systems," *IEEE Transactions on Vehicular Technology*, Vol. 43, 1994, pp. 892-891.

Rei-Heng Cheng (鄭瑞恆) received the M.S. and Ph.D. degrees in Computer Science and Information Engineering from National Chiao Tung University, Taiwan, R.O.C. in 1989 and 1995 respectively. Since 1999, he has been on the faculty of Hsuan Chuang University, Taiwan, R.O.C. He served as the director of Department of Information Management from 2002 to 2004. His research interests include wireless network, image processing, and character recognition.

Chang Wu Yu (俞征武) was born in Taoyuan, Taiwan, R.O.C. in 1964. He received the B.S. degree from Soochow University in 1985, M.S. degree from National Tsing Hua University in 1989, and Ph.D. degree from National Taiwan University in 1993, all in Computer Sciences. From 1995 to 1998, he was an Associate Professor at the Department of Information Management, Ming Hsin University of Science and Technology. In 1999, he joined the Department of Computer Science and Information Engineering, Chung Hua University. His current research interests include graph algorithms, wireless networks, and distributed computing.

Tung-Kuang Wu (吳東光) was born in Taichung, Taiwan, R.O.C. in 1962. He received his B.S. degree in Electrical Engineering from National Taiwan University in 1984, M.S. and Ph.D. degrees in Computer Engineering from the Pennsylvania State University in 1991 and 1995 respectively. He joined the Computer and Communications Lab. in ITRI since 1994. He is currently an Associate Professor at the Department of Information Management, Ming Hsin University of Science and Technology. His current research interests include special education technology, eLearning, and wireless networks.