Fixed Channel Assignment in Cellular Radio Networks Using a Modified Genetic Algorithm

Chiu Y. Ngo, *Member, IEEE,* and Victor O. K. Li, *Fellow, IEEE*

***Abstract*—With the limited frequency spectrum and an increasing demand for cellular communication services, the problem of channel assignment becomes increasingly important. However, finding a conflict-free channel assignment with the minimum channel span is NP hard. Therefore, we formulate the problem by assuming a given channel span. Our objective is to obtain a conflict-free channel assignment among the cells, which satisfies both the electromagnetic compatibility (EMC) constraints and traffic demand requirements. We propose an approach based on a modified genetic algorithm (GA). The approach consists of a genetic-fix algorithm that generates and manipulates individuals with fixed size (i.e., in binary representation, the number of ones is fixed) and a minimum-separation encoding scheme that eliminates redundant zeros in the solution representation. Using these two strategies, the search space can be reduced substantially. Simulations on the first four benchmark problems showed that this algorithm could achieve at least 80%, if not 100%, convergence to solutions within reasonable time. In the fifth benchmark problem, our algorithm found better solutions with shorter channel span than any existing algorithms. Such significant results indicate that our approach is indeed a good method for solving the channel-assignment problem.**

***Index Terms*— Cellular network, channel assignment, generic algorithm, wireless communication.**

I. Introduction

AN IMPORTANT issue in the design of a cellular radio network is to determine a spectrum-efficient and conflict-free allocation of channels among the cells while satisfying both the traffic demand and the electromagnetic compatibility (EMC) constraints. This issue is commonly referred to as channel assignment or frequency assignment. Generally, there are three types of EMC constraints [7], namely: 1) the cochannel constraint (CCC), where the same channel cannot be assigned to certain pairs of radio cells simultaneously; 2) the adjacent channel constraint (ACC), where channels adjacent in the frequency spectrum cannot be assigned to adjacent radio cells simultaneously; and 3) the cosite constraint (CSC), where channels assigned in the same radio cell must have a minimal separation in frequency between each other. These EMC constraints are normally determined by the characteristics of the radio frequency propagation and spatial density of the expected traffic requirements.

Manuscript received December 19, 1994; revised October 29, 1996.

C. Y. Ngo is with Philips Research Laboratories, Briarcliff Manor, NY 10510 USA.

V. O. K. Li is with the Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong, China.

Publisher Item Identifier S 0018-9545(98)00711-7.

The problem of channel assignment has had a brief history in the literature. Basically, it can be classified into two categories: 1) fixed channel assignment (FCA), where channels are permanently allocated to each cell and 2) dynamic channel assignment (DCA), where all channels, which are available for every cell, are allocated dynamically upon request. Normally, DCA gives better performance than FCA except under heavy traffic load condition, where FCA outperforms DCA [2]. Since heavy traffic load is expected in the future generation of cellular radio networks, an efficient FCA scheme that can provide high spectrum usage efficiency is desired. The FCA problem has been studied extensively for the past three decades. A comprehensive summary of the work done before 1980 can be found in [3]. It has been shown that this problem is equivalent to a generalized graph-coloring problem, which is NP hard (e.g., [1]). As a result, various approximate algorithms have been proposed. These include some graph-theoretic approaches [3]–[8]. Most of these methods are based on a heuristic ranking of cells according to the difficulty of meeting the EMC constraints. Recently, some approaches based on the Hopfield neural network [9] and simulated annealing [10] have been proposed. The first neural network for solving the channel-assignment problem was proposed probably by Kunz [11]. Following that, several other neural networks were studied. These include the work by Sengoku *et al*. [12], Funabiki and Takefuji [13], and Lochite [14]. An inherent disadvantage of this approach is that it easily converges to local optima and, hence, suboptimal solutions. To overcome this difficulty, a simulated annealing approach was suggested by Duque-Anton *et al*. [15] and Mathar *et al*. [16]. Although this approach is guaranteed to achieve the global optimum asymptotically, its rate of convergence is rather slow, and a carefully designed cooling schedule is required.

In this paper, we propose yet another approach—the genetic algorithms (GA's) [17], [18]—for solving the channel-assignment problem. We consider a general cellular radio network subjected to all three kinds of EMC constraints described earlier. In addition, the traffic is assumed to be inhomogeneous, i.e., each cell has different traffic requirements. Our problem formulation follows the stem of the aforementioned neural-network and simulated annealing approaches. The objective is to obtain a conflict-free channel assignment among the cells such that the total number of channels used is close to the minimum channel span required for the whole network. The approach is based on a modified GA called the genetic-fix algorithm. Unlike the conventional GA's [17],

[18] that generate subsets of all possible sizes, the genetic-fix algorithm can generate fixed-size subsets (i.e., in binary representation, the number of ones is fixed). Furthermore, we propose a minimum-separation encoding scheme that can eliminate redundant zeros in the solution representation. These two strategies allow us to reduce the search space substantially and, hence, greatly speed up the computation.

Recently, it has come to our attention that two GA approaches have been independently considered in [19] and [20]. In [19], the author first defined an asexual crossover and a special mutation and then represented the solution space in a way similar to our genetic-fix algorithm such that the traffic requirement was fulfilled inherently. A disadvantage of such asexual crossover is that it can easily destroy the structure and, thus, make the problem harder to converge. In [20], the authors represented the channel-assignment solution as a string of channel numbers (instead of binary string). These numbers were grouped in such a way that each cell had a specified number of channels and, hence, satisfied the traffic requirement. The evolution was then proceeded via the partially matched (PMX) crossover operator and basic mutation. Our work differs from [19] and [20] in that our crossover operator is based on a concept similar to that in conventional binary crossover. Furthermore, our solution representation allows us to further reduce the search space by eliminating the CSC using the minimum-separation encoding scheme, which will not be possible with the representation in the above two approaches. In addition, instead of performing simulation on small- or medium-size problems for demonstration purposes, we consider several "hard" realistic benchmark problems.

The rest of the paper is organized as follows. In Section II, we formulate the channel-assignment problem with several assumptions. We first present a graph-theoretic formulation of the problem and then formulate the problem as an unconstrained optimization problem by assuming that the total number of available channels is given. In Section III, we discuss the genetic-fix algorithm with a brief review of the conventional simple GA (SGA). In Section IV, we present the minimum-separation encoding scheme that can eliminate the CSC. In Section V, we discuss some implementation issues. These include the determination of a near-optimal minimal channel span and a local-search routine that can improve the rate of convergence of the genetic-fix algorithm. In Section VI, we apply the algorithm to solve the channel-assignment problem and perform some simulations. Finally, we conclude our work in Section VII.

II. Problem Formulation

The cellular radio network to be considered consists of n arbitrary cells. Without loss of generality, it is assumed that channels are evenly spaced in the radio frequency spectrum. Using an appropriate mapping, channels can be represented by consecutive positive integers. Therefore, the EMC constraints can be described by an $n \times n$ symmetric matrix called the compatibility matrix C, where 1) each diagonal element c_{ii} represents the CSC, i.e., the minimum separation distance between any two channels at cell i and 2) each nondiagonal element c_{ij} represents the minimum separation distance in frequency between any two frequencies assigned to cells i and j, respectively. In this matrix, CCC is represented by $c_{ij} = 1$, ACC is represented by $c_{ij} = 2$, and cells that are free to use the same channels are represented by $c_{ij} = 0$. In all cases, $c_{ii} \geq 1$.

By analyzing the traffic at each cell, the traffic demand requirement can be obtained. This can be represented by an n-element demand vector denoted as $\boldsymbol{d}$. In this vector, each element d_i represents the number of channels to be assigned to cell i.

In short, given the compatibility matrix C and the traffic demand vector $\boldsymbol{d}$, the optimal channel-assignment scheme involves the determination of the minimum channel span required and the way to distribute these channels among the cells while satisfying the EMC constraints and the traffic demand requirements.

A. Graph-Theoretic Formulation

Conventional graph-theoretic approaches formulate the problem as a "minimum span" problem, i.e., given the compatibility matrix C and the traffic demand vector $\boldsymbol{d}$ of an arbitrary n-cell radio network, find a conflict-free channel assignment with the minimum channel span. Mathematically, it means that if s_{ik} is the channel number of the kth call assigned to cell i, then the problem can be expressed as

$$\begin{aligned} \text{minimize} \quad & m = \max_{i,k} s_{ik} \\ \text{subject to} \quad & |s_{ik} - s_{jl}| \geq c_{ij} \\ & \text{for } i, j = 1, 2, \cdots, n \\ & \text{and } k = 1, 2, \cdots, d_i, l = 1, 2, \cdots, d_j \\ & \text{and } (i,k) \neq (j,l). \end{aligned} \tag{1}$$

It has been shown that this problem is equivalent to a generalized graph-coloring problem [1], [3]. When only CCC is considered, it is reduced to a graph-coloring problem, which is known to be NP complete, i.e., the computation time grows exponentially with the number of the nodes in the graph. Consequently, an exact search of the optimal solution is impractical for problems of a reasonable size.

B. Our Formulation

The graph-theoretic approach aims only at minimizing the used spectrum. However, in practice, the determination of a conflict-free assignment pattern is more important. Consequently, some methods that assume a given channel span have been proposed. These include several "natural" approaches such as neural network [11], [13], [14] and simulated annealing

[15], [16]. Our formulation follows the stem of these approaches. We assume that the cellular network is composed of n cells and each cell is capable of carrying any of the available channels. We formulate the channel-assignment problem as an unconstrained optimization problem.

In particular, we represent the solution space F as an $n \times m$ binary matrix, where n is the number of radio cells and m is the total number of available channels. (Here, we assume m is given. This formulation is different from the "minimum span" problem.) Each element f_{jk} in the matrix is either one or zero such that

$$f_{jk} = \begin{cases} 1 \\ 0 \end{cases} \text{ if channel } k \text{ is } \begin{Bmatrix} \text{assigned} \\ \text{not assigned} \end{Bmatrix} \text{ to cell } j.$$

Diagrammatically, the admissible channel assignment F can be described, as given at the bottom of the page.

Basic requirements for the cellular network are the ability to serve the expected traffic and the avoidance of interference. The first requirement imposes a demand constraint on F. A total of d_i channels are required for cell i. This implies that the total number of ones in row i of F must be d_i. Mathematically, it means that if the assignment to cell i violates the demand constraint, then

$$\left(\sum_{q=1}^{m} f_{iq} - d_i \right) \neq 0.$$

The second requirement is modeled by the compatibility matrix C. It is composed of CSC, CCC, and ACC. For CSC, if channel p is within distance c_{ii} from an already assigned channel q in cell i (i.e., $|p-q| < c_{ii}$), then channel p must not be assigned to cell i. Mathematically, it means that if the assignment of channel p to cell i violates CSC, then

$$\sum_{\substack{q=p-(c_{ii}-1) \\ q \neq p \\ 1 \leq q \leq m}}^{p+(c_{ii}-1)} f_{iq} > 0.$$

For CCC and ACC, if channel p in cell i is within distance c_{ij} from an already assigned channel q in cell j, where $c_{ij} > 0$ and $i \neq j$, (i.e., $|p-q| < c_{ij}$), then channel p must not be assigned to cell i. Mathematically, it means that if the assignment of channel p to cell i violates CCC and/or ACC, then

$$\sum_{\substack{j=1 \\ j \neq i \\ c_{ij} > 0}}^{n} \sum_{\substack{q=p-(c_{ij}-1) \\ 1 \leq q \leq m}}^{p+(c_{ij}-1)} f_{jq} > 0.$$

Therefore, a generic choice of cost function can be expressed as

$$\begin{aligned} C(F) = & \sum_{i=1}^{n} \sum_{p=1}^{m} \left(\sum_{\substack{j=1 \\ j \neq i \\ c_{ij} > 0}}^{n} \sum_{\substack{q=p-(c_{ij}-1) \\ 1 \leq q \leq m}}^{p+(c_{ij}-1)} f_{jq} \right) f_{ip} \\ & + \alpha \sum_{i=1}^{n} \sum_{p=1}^{m} \left(\sum_{\substack{q=p-(c_{ii}-1) \\ q \neq p \\ 1 \leq q \leq m}}^{p+(c_{ii}-1)} f_{iq} \right) f_{ip} \\ & + \beta \sum_{i=1}^{n} \left(\sum_{q=1}^{m} f_{iq} - d_i \right) \end{aligned} \tag{2}$$

where α and β are weighting factors.

It is noted that $C(F)$ achieves its minimum of zero when all constraints are satisfied. Therefore, our problem is to find an F such that $C(F)$ is zero. To speed up the search process, the structure of the solution space is examined. A special form of solution representation is used such that the traffic demand and the CSC requirements are fulfilled inherently. In the next two sections, we will discuss how a modified GA, called the genetic-fix algorithm, and a special encoding technique, called the minimum-separation encoding scheme, can accomplish this task.

f_{jk}

Cell Number	Channel Number 1	2	3	⋯			m
1	0	1	0	⋯	0	1	0
2	0	0	1	⋯	0	0	1
3	0	0	1	⋯	1	0	0
⋯				⋮			
n	1	0	0	⋯	1	0	1

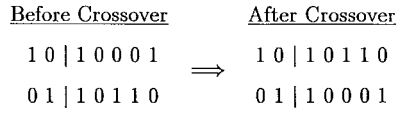


Fig. 1. Crossover in SGA.

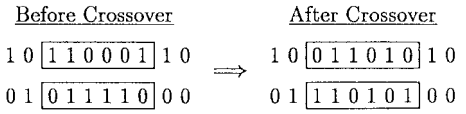


Fig. 2. Crossover in genetic fix.

III. Principle of Generic Genetic Algorithms

Although there are many variants of GA's, their mechanisms are similar as illustrated in the following pseudocodes:

```
GA( )
{
    generate randomly an initial population;
    evaluate fitness of each individual;
    while convergence not achieved and
      max. number of generations not exceeded
    {
      select individuals probabilistically according to
        their fitness;
      perform genetic operations on the selected
        individuals;
      evaluate fitness of the newly obtained
        individuals;
    }
}
```

In its most fundamental form, called the SGA [17], each individual is represented by a binary vector. Proportional selection is adopted and two genetic operators, called mutation and crossover, are used. These operators are applied to the selected individuals with fixed probabilities p_m and p_c corresponding to mutation and crossover, respectively. Unlike crossover that does not create new genetic material, mutation can introduce new information in the population. This peculiar characteristic of mutation allows GA's to overcome local optima. Basically, mutation involves the "flipping" of each bit of an individual with probability p_m. Its role is to restore lost or unexplored genetic information into the population to prevent premature convergence. Crossover, on the other hand, involves the exchange of portions of two selected individuals. The idea is to allow the offspring to preserve part of the first parent while incorporating information from the second parent. This implementation in SGA is accomplished by choosing a crossover point at random and exchanging the segments on the right of this point as illustrated in Fig. 1.

Another common way to implement crossover is to choose two crossover points and exchange the segments between these points. It has been shown in [21] that such crossover has some theoretical advantages over the one-point case. Hence, in our subsequent discussion, we will focus on crossover operators with two crossover points.

D. Principle of Genetic-Fix Algorithm

It is noted that generic GA's generate subsets of all possible sizes. However, there are some combinatorial optimization problems whose feasible solutions are fixed-size subsets. By taking this information into account, we can greatly reduce the search space of the problem and, hence, shorten the computation time of the algorithm. We introduce the genetic-fix algorithm that can generate a fixed number of ones for each individual and preserve this property during the genetic operations. Of course, this requires special crossover and mutation operators that can maintain the property of a fixed number of ones. One way to implement these operators is proposed as follows.

A. Crossover in Genetic Fix

Given two parents A and B, we create a first-in last-out (FILO) stack to store the bit position k corresponding to the opposite bit pair (A_k, B_k). A_k and B_k are said to be opposite if $A_k \oplus B_k = 1$, where $\oplus$ denotes the exclusive or operator. The crossover is performed by first generating two crossover points c_1 and c_2 at random along the string length such that $c_1 < c_2$ and then moving right from c_1 until an i is found such that $A_i \oplus B_i = 1$. We push i into the FILO stack and continue the process until we find a j such that $A_j \oplus B_j = 1$. Then, we compare A_j with A_{s1}, where $s1$ is the top element in the stack. If they are the same, we push j into the stack, otherwise, we swap the pair indexed by j with the pair indexed by $s1$ and pop $s1$ from the stack. The process continues until c_2 is reached. A pseudocode of this operator is illustrated as follows:

```
Crossover_Genetic_Fix (parents A and B)
{
    generate randomly two crossover points c1 and c2
     along the string length such that c1 < c2;
    for (i = c1; i ≤ c2; i++)
    {
      if Ai and Bi are opposite
      {
        if stack is empty, push i into stack;
        else
        {
          if Ai and As1 are different
                (s1 = top element in stack)
          {
                swap (Ai, Bi) with (As1, Bs1);
                pop s1 from stack;
          }
          else push i into stack;
        }
      }
    }
}.
```

Let us consider an example using binary vector representation. Suppose that we have two 10-b strings. Each of them has five ones with the crossover sites $c_1 = 3$ and $c_2 = 8$. The genetic-fix crossover operation can be illustrated by Fig. 2.

The same rule applies to the binary array representation if the two crossover points are in the same row. However, if they belong to different rows (say, row x and row y, where $x<y$), then genetic-fix crossover will apply from c_1 to the end of row x and from the beginning of row y to c_2 and will swap those in-between rows, if any.

B. Mutation in Genetic Fix

In order to balance the number of ones in an individual, the mutation operation must always be done in pairs of opposite bits. This can be implemented as follows. Let b_i be the ith bit position of an individual. To mutate b_i, we need to find a random b_j such that $b_i \oplus b_j = 1$. Then, we swap b_i with b_j. In the case of binary array representation, both b_i with b_j must be in the same row.

IV. Minimum-Separation Encoding Scheme

There exists some combinatorial optimization problems, which may require a minimum separation between consecutive elements in the solution. By encoding each element properly, the solution space can be greatly reduced. In the sequel, we will present an encoding technique, called the minimum-separation encoding scheme, which can accomplish this objective.

Let an individual be represented by a p-bit binary string with q fixed elements and let $d_{\min}$ be the minimum separation between consecutive elements. The idea of the encoding scheme is to represent the solution in such a way that a one is followed by $(d_{\min}-1)$ zeros encoded as a new "one," denoted as $\tilde{1}$. For example, if $d_{\min}=3$, then $\tilde{1}$ is equivalent to "100" in the old representation. Consequently, for an individual with $p=10$ and $q=3$, it can be encoded as follows:

$$\underline{\text{Original Rep.}}\quad 1\,0\,0\,0\,1\,0\,0\,1\,0\,0 \Longrightarrow \underline{\text{Encoded Rep.}}\quad \tilde{1}\,0\,\tilde{1}\,\tilde{1}.$$

Hence, the length of representation can be substantially reduced. However, a problem still remains if a "one" is at a position within $(d_{\min}-1)$ from the end of the string. To cope with this shortcoming, we need to first augment the original individual with $(d_{\min}-1)$ "zero" before performing the encoding scheme. In this way, the total bit length is increased to $(p+d_{\min}-1)$. An example is illustrated as follows:

$$\begin{array}{c}\underline{\text{Original Rep.}}\\ 1\,0\,0\,0\,1\,0\,0\,0\,1\,0\\ \Downarrow\\ \underline{\text{Augmentation}}\\ 1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\\ \Downarrow\\ \underline{\text{Encoded Rep.}}\\ \tilde{1}\,0\,\tilde{1}\,0\,\tilde{1}\,0\end{array}$$

Using the minimum-separation encoding scheme, a p-bit binary string with q elements separated by a distance of $d_{\min}$ can be encoded by a binary string of $[p-(q-1)(d_{\min}-1)]$ bits only.

V. Impelmentation Issues

By representing each individual as a binary array F and restricting the number of ones in each row to its corresponding traffic demand, i.e., given at the bottom of the page, the traffic demand requirement can be fulfilled automatically. This property can be maintained throughout the iterative process using the genetic-fix operators.

Further improvement can be achieved by noting that the CSC requirement is nothing more than a minimum separation between consecutive elements in each row of an individual. Therefore, using the minimum-separation encoding scheme, we can eliminate the CSC requirement from the cost function and further reduce the search space. As a result, row i of the solution matrix F will consist of d_i $\tilde{1}$'s with a total length of $[m-(d_i-1)(c_{ii}-1)]$ bits.

Hence, using the genetic-fix algorithm and the minimum-separation encoding scheme, the cost function of the channel-assignment problem can be simplified to

$$C(F)=\sum_{i=1}^{n}\sum_{p=1}^{m}\left(\sum_{\substack{j=1\\ j\neq i\\ c_{ij}>0}}^{n}\sum_{\substack{q=p-(c_{ij}-1)\\ 1\le q\le m}}^{p+(c_{ij}-1)} f_{jq}\right) f_{ip}. \tag{3}$$

f_{jk}

Cell Number	Channel Number 1	2	3	$\cdots$			m	Row Sum
1	0	1	0	$\cdots$	0	1	0	d_1
2	0	0	1	$\cdots$	0	0	1	d_2
3	0	0	1	$\cdots$	1	0	0	d_3
$\cdots$				$\vdots$				$\vdots$
n	1	0	0	$\cdots$	1	0	1	d_n.

Compared with the problem formulation in the neural network and simulated annealing approaches (e.g., [13]), our formulation is much simpler. However, there are still some implementation issues to be considered. These include the determination of the total number of available channels m and the ways of improving the performance of the genetic-fix algorithm.

A. Determination of m

Our problem formulation assumes that the total number of available channels m is given. This number can be determined by either the available radio spectrum or the lower bound estimated by a graph-theoretic method [22]. Alternatively, a rough estimate of the least upper bound of m can be obtained by multiplying c_{ii} in C with the maximum d_j in ***d***. Starting from this estimate, we decrease it gradually until the genetic-fix algorithm cannot find a "feasible" solution for the channel-assignment problem within a "reasonable" time. The smallest feasible number of available channels thus far obtained will be used as m in our problem.

B. Simplification of the Cost Function

By exploiting the symmetry of the compatibility matrix C, the cost function (3) can be further simplified to

$$C(F) = \sum_{i=1}^{n-1} \sum_{\substack{j=i+1 \\ c_{ij}>0}}^{n} \left(\sum_{p=1}^{c_{ij}-1} \sum_{q=1}^{p-1} f_{jq} f_{ip} + \sum_{p=c_{ij}}^{m} \sum_{q=p-c_{ij}+1}^{p-1} f_{jq} f_{ip} + \frac{1}{2} \sum_{p=1}^{m} f_{jp} f_{ip} \right). \quad (4)$$

Cf. Local-Search Routine

Generally, GA's do not perform a finely tuned local search. In order to improve the performance and increase the rate of convergence, we introduce the following local-search routine. The basic idea is rather simple. It involves nothing more than finding a zero along a row of the binary array solution and swapping it with the most "violated" one (i.e., the "one," which gives the highest increase in the cost function) on that row. Of course, this procedure should result in better performance with a little more expense in computation. The routine starts with a given binary array solution F and an empty penalty vector with size n_p and proceeds as follows.

1) Save n_p most "violated" ones in the penalty vector.
2) Select randomly a nontagged element, say indexed by i, from that vector and tag it.
3) Find a nontagged zero at random along the ith row of F, tag it, and swap it with the most corresponding "violated" one pointed to by element i.
4) Evaluate the new structure.
5) If no improvement is revealed, restore the old structure and repeat steps 3)–4) until all zeros in row i are tagged.
6) Repeat steps 2)–5) until all elements in the penalty vector are tagged.

To make the computation manageable, n_p should not be large, and the local-search routine should only be done when necessary. In our case, we perform a local search only when the best solution thus far achieved has not been changed for K generations. Once a better solution is found, this counter will be reset or the local search will continue.

VI. Simulations

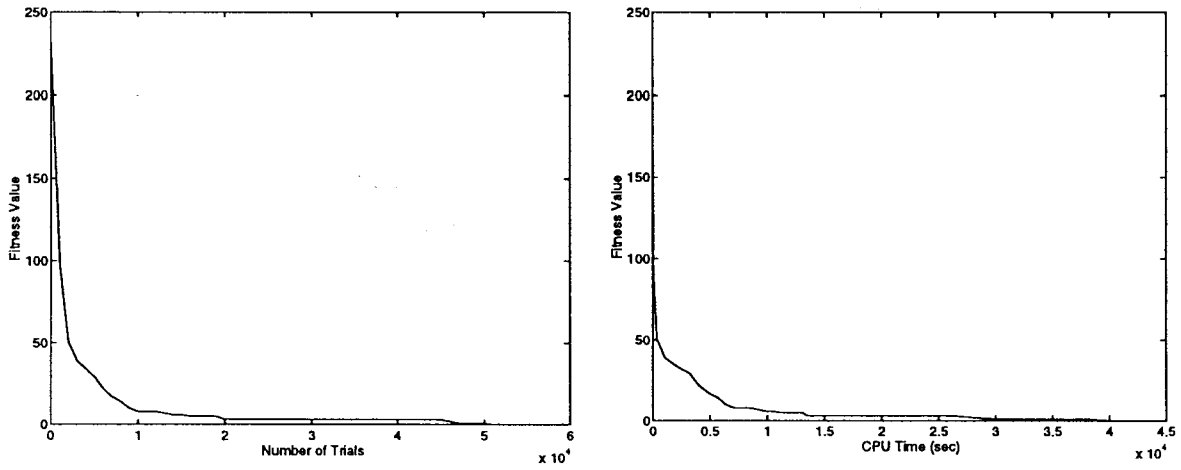
Our simulator, called GENESIS_F and written in C, is based on the GENESIS system [23] developed by Grefenstette for general-purpose unconstrained optimization using genetic-search techniques. Similar to GENESIS, the user needs only to provide an evaluation function that gives the "fitness" of a given individual. There are several enhancements over GENESIS. First, it allows users to think about the genetic structures as binary arrays in addition to vectors of real numbers and bit strings. This representation enables an easier application of GENESIS to some problems, like the scheduling problem. Second, variable bit length is allowed in each row of a binary array. Such an arrangement may help reduce the search space by removing some redundant zeros (see [24]). Third, an option that allows users to choose fixed-element manipulation, based on the concept of the genetic-fix algorithm, is added. This includes the generation of fixed-size individuals and preserves the fixed-size property throughout various genetic operations such as selection, crossover, and mutation. Fourth, it allows users to solve constrained optimization problems via a penalty method, which uses the "implied objective function" constraint [24].

Five benchmark problems were examined. Problems 1, 2, 3, and 5 were taken from [7], and problem 4 was from [11]. Table I [11] summarizes the characteristics of these five problems, where all the demand vectors and the compatibility matrices except C_6 are given in [13]. The compatibility matrix C_6 is the same as C_5 in [13] with all the diagonal elements being replaced by problem 5, while problems 2 and 4 consider only CSC and CCC, and problems 1, 3, and 5 also consider ACC. Furthermore, for each problem, the required channel span m was determined via pilot trials as described in the previous section. While problems 1–4 give the same m as in [13], problem 5 has an even shorter channel span than the existing algorithms in [7]. (The best existing channel span is 269.) In order to justify the effectiveness of the minimum-separation encoding scheme, the bit reduction in representing the solution space is also included in the table.

One hundred Monte Carlo runs were performed on problems 1–4, and ten runs were performed on problem 5. In order to avoid the disappearance of the best individual after some generations, the "elitist" selection strategy was adopted so that the best individual survives with probability one, i.e., it always survives intact from one generation to the next.

TABLE I
PROBLEM SPECIFICATIONS AND SOLUTION REPRESENTATION

Problem	No. of Cells	No. of Channels	Compatibility Matrix	Demand Vector	Required Number of Bits		
					Previous Approach	Our Approach	Difference
	n	m	C	$\boldsymbol{d}$	A	B	$A-B$
1	4	11	C_1	D_1	44	36	8
2	21	221	C_3	D_4	4641	2845	1796
3	21	309	C_5	D_4	6489	3795	2694
4	25	73	C_2	D_2	1825	1683	142
5	21	268	C_6	D_4	5628	3832	1796



(a) (b)

Fig. 3. A typical rate of convergence trajectory for problem 3 based on (a) number of trials and (b) CPU time used.

TABLE II
SIMULATION PARAMETERS

Problem	T	p_c	p_m	N_s	n_p	K
1	100	0.95	0.0005	10	5	10
2	200000	0.95	0.0005	10	25	10
3	200000	0.95	0.0005	10	30	10
4	200000	0.95	0.0005	10	25	10
4	400000	0.95	0.0005	10	30	10

Several parameters need to be set, including the maximum number of trials per run (T trials), the crossover probability (p_c), the mutation probability (p_m), the population size (N_s), the size of the penalty vector (n_p), and the counter for "igniting" the local-search routine (K generations). As for any GA's, the settings of these parameters are generally quite *ad hoc*. Nevertheless, a general rule was suggested in [17] for using a relatively small population size, high crossover probability, and low mutation probability. In our simulation, this rule was first applied, and then the parameters were fine tuned through pilot experiments, ensuring that the computation was manageable. Table II summarizes the simulation parameters of all five problems.

TABLE III
SUMMARY OF SIMULATION RESULTS

Problem	Neural Network	Genetic-Fix		
	Frequency of Convergence (%)	Frequency of Convergence (%)	No. of Trials	CPU Time (sec)
1	100	100	1 (± 0.0)	0.0 (± 0.0)
2	79	92	63152 (± 40521)	19365 (± 16782)
3	24	80	79502 (± 40778)	89196 (± 64846)
4	9	100	26382 (±19639)	2181 (±1722)
5	–	20	382770 (±159014)	596790 (±76716)

Cell Number

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
2	19	16	18	8	5	8	6	9	2	4	1	7	1	4	15	5	19	7	3	1
45	64	59	66	34	11	14	12	24	7	10	6	13	16	10	27	11	28	13	20	25
162	133	77	83	61	26	20	22	32	13	15	12	30	23	18	50	17	33	42	39	31
204	188	112	90	79	33	36	29	37	21	23	20	37	28	25	61	47	38	59	45	36
223	235	227	96	101	47	41	34	52	26	30	25	62	44	31	93	58	44	69	54	41
0	0	0	197	110	60	54	43	68	35	40	32	87	51	38	101	65	53	77	72	56
0	0	0	251	138	69	65	48	73	42	46	37	97	58	46	128	75	60	107	82	63
0	0	0	262	178	75	81	56	85	49	51	43	110	67	63	185	87	67	115	90	70
0	0	0	0	200	101	92	76	91	55	57	48	115	73	71	190	97	78	124	113	80
0	0	0	0	210	113	98	83	99	62	64	54	121	78	86	197	125	84	134	121	95
0	0	0	0	233	132	103	88	104	71	69	59	127	84	96	212	135	89	139	143	103
0	0	0	0	239	143	109	106	110	81	74	72	137	89	112	218	182	100	158	148	123
0	0	0	0	0	148	116	114	119	94	86	77	146	94	120	242	187	109	166	169	133
0	0	0	0	0	155	122	126	140	102	92	82	153	105	131	250	203	120	181	175	145
0	0	0	0	0	161	129	138	146	107	98	88	180	118	145	262	236	127	188	194	150
0	0	0	0	0	166	135	149	154	115	105	95	207	125	151	0	0	139	204	207	156
0	0	0	0	0	173	159	156	161	122	113	103	212	140	168	0	0	147	235	223	165
0	0	0	0	0	196	170	165	167	132	118	108	219	157	174	0	0	153	244	228	171
0	0	0	0	0	201	177	202	173	137	124	116	255	182	179	0	0	162	252	258	200
0	0	0	0	0	209	186	214	180	142	129	121	260	193	195	0	0	168	260	266	209
0	0	0	0	0	226	191	225	192	151	134	126	0	198	200	0	0	174	0	0	220
0	0	0	0	0	232	206	231	199	158	144	131	0	203	208	0	0	179	0	0	234
0	0	0	0	0	238	211	237	210	164	149	136	0	215	227	0	0	193	0	0	248
0	0	0	0	0	253	217	254	216	170	155	141	0	224	247	0	0	198	0	0	261
0	0	0	0	0	266	229	259	221	176	160	146	0	236	265	0	0	206	0	0	268
0	0	0	0	0	0	234	0	233	184	166	152	0	241	0	0	0	211	0	0	0
0	0	0	0	0	0	245	0	240	189	172	157	0	249	0	0	0	217	0	0	0
0	0	0	0	0	0	251	0	246	195	181	163	0	258	0	0	0	226	0	0	0
0	0	0	0	0	0	256	0	257	201	186	169	0	263	0	0	0	232	0	0	0
0	0	0	0	0	0	261	0	264	208	191	175	0	268	0	0	0	245	0	0	0
0	0	0	0	0	0	0	0	0	213	204	183	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	219	215	188	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	224	222	196	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	230	228	202	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	238	235	207	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	243	241	212	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	249	247	218	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	255	253	225	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	260	258	231	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	267	265	237	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	244	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	250	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	256	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	263	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	268	0	0	0	0	0	0	0	0	0

Simulations were performed on an HP Apollo 9000/700 workstation using our proposed genetic-fix algorithm and the minimum-separation encoding scheme in GENESIS_F. Performances were measured based on: 1) average central processing unit (CPU) time used per successful convergence in which a solution is found before exceeding T trials; 2) average number of fitness evaluations, called trials, per successful convergence; and 3) average frequency of convergence to solutions defined as the ratio of the total number of successful convergence to the total number of runs. Because not every offspring needs to be evaluated in each generation, performance measure based on trials is more meaningful than generations. Table III summarizes the results. For ease of comparison, the corresponding neural-network performance in [13] is also included. In addition, a typical trajectory of problem 3 that demonstrates the rate of convergence is shown in Fig. 3, and an example of channel assignment for problem 5 is given in the Appendix.

The results show that the genetic-fix algorithm is, indeed, a good method for solving the FCA problem. Compared with the small number of convergences in the neural-network approach, our algorithm gives 100% convergence in problems 1 and 4, 92% convergence in problem 2, and 80% convergence in problem 3. With a shorter channel span than any existing algorithms [7], our algorithm gives 20% convergence in problem 5. Such a result already demonstrated its effectiveness in finding good solutions. With an unlimited number of trials, one can be almost sure that the algorithm gives 100% convergence to the solutions as proved in [24].

VII. Conclusions

We have studied the problem of conflict-free FCA in cellular radio networks. We proposed an approach based on a modified GA. This algorithm, called the genetic-fix algorithm, generates and manipulates individuals with fixed size and, hence, greatly reduces the search space. Furthermore, using the minimum-separation encoding scheme, the required number of bits for representing the solutions decreases substantially. These two strategies enable us to eliminate the traffic demand requirement and the CSC from the cost function and, hence, greatly improve the computation speed. Simulations on the first four benchmark problems showed that this algorithm could achieve, respectively, 100%, 92%, 80%, and 100%. In the fifth benchmark problem, our algorithm found better solutions with a shorter channel span than any existing algorithms. Such significant results indicate that our algorithm is indeed a good one for solving the channel-assignment problem.

One should note that our simulation is based on a sequential implementation of the genetic-fix algorithm and is, by no means, optimized in terms of computation time. By coding the algorithm more efficiently and implementing it in a parallel fashion, it is expected that this costly process can be greatly reduced and, hence, substantially speed up the search.

Appendix

For an example of channel assignment for problem 5, see the previous page.

Acknowledgment

The authors would like to thank the reviewers for their constructive comments and suggestions.

References

[1] A. Gamst and W. Rave, "On frequency assignment in mobile automatic telephone systems," in *Proc. IEEE GLOBECOM '82*, pp. 309–315.
[2] P. Raymond, "Performance analysis of cellular networks," *IEEE Trans. Commun.*, vol. 39, no. 12, pp. 1787–1793, 1991.
[3] W. K. Hale, "Frequency assignment: theory and applications," *Proc. IEEE*, vol. 68, no. 12, pp. 1497–1514, 1980.
[4] F. Box, "A heuristic technique for assignment frequencies to mobile radio nets," *IEEE Trans. Veh. Technol.*, vol. 27, pp. 57–64, May 1977.
[5] S. Kim and S. L. Kim, "A two-phase algorithm for frequency assignment in cellular mobile systems," *IEEE Trans. Veh. Technol.*, vol. 43, no. 3, pp. 542–548, 1994.
[6] M. Sengoku, H. Tamura, S. Shinoda, and T. Abe, "Development in graph- and/or network-theoretic research of cellular mobile communication channel assignment problems," *IEICE Trans. Fundamentals*, vol. E77-A, no. 7, pp. 1117–1123, July 1994.
[7] K. N. Sivarajan, R. J. McEliece, and J. W. Ketchun, "Channel assignment in cellular radio," in *Proc. 39th IEEE Vehicular Technology Conf.*, May 1989, pp. 846–850.
[8] J. A. Zoellner and C. L. Beall, "A break-through in spectrum conserving frequency assignment technology," *IEEE Trans. Electromagn. Compat.*, vol. 19, pp. 313–319, Aug. 1977.
[9] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Bio. Cybern.*, vol. 52, pp. 141–152, 1985.
[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, May 1983.
[11] D. Kunz, "Channel assignment for cellular radio using neural networks," *IEEE Trans. Veh. Technol.*, vol. 40, no. 1, pp. 188–193, 1991.
[12] M. Sengoku, K. Nakano, K. Shinoda, Y. Yamaguchi, and T. Abe, "Cellular mobile communication systems and channel assignment using neural networks," in *Proc. IEEE 33rd Midwest Symp. Circuits and Syst.*, Aug. 1990, pp. 411–414.
[13] N. Funabiki and Y. Takefuji, "A neural network parallel algorithm for channel assignment problems in cellular radio networks," *IEEE Trans. Veh. Technol.*, vol. 41, no. 4, pp. 430–437, 1992.
[14] G. D. Lochite, "Frequency channel assignment using artificial neural networks," in *Proc. 8th IEE Int. Conf. Antennas and Propagation*, 1993, vol. 2, pp. 948–951.
[15] M. Duque-Anton, D. Kunz, and B. Ruber, "Channel assignment for cellular radio using simulated annealing," *IEEE Trans. Veh. Technol.*, vol. 42, no. 1, pp. 14–21, 1993.
[16] R. Mathar and J. Mattfeldt, "Channel assignment in cellular radio networks," *IEEE Trans. Veh. Technol.*, vol. 42, no. 4, pp. 647–656, 1993.
[17] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[18] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.
[19] M. Cuppini, "A genetic algorithm for channel assignment problems," *Eur. Trans. Telecommun. Related Technol.*, vol. 5, no. 2, pp. 285–294, Mar.–Apr. 1994.
[20] W. K. Lai and G. G. Coghill, "Channel assignment through evolutionary optimization," *IEEE Trans. Veh. Technol.*, vol. 45, no. 1, pp. 91–96, 1996.
[21] K. A. DeJong, "Analysis of the behavior of a class of genetic adaptive systems," Ph.D. thesis, Dept. Comp. Commun. Sci., Univ. Michigan, Ann Arbor, MI, 1975.
[22] A. Gamst, "Some lower bounds for a class of frequency assignment problems," *IEEE Trans. Veh. Technol.*, vol. 35, pp. 9–14, Feb. 1986.
[23] J. J. Grefenstette, "A user's guide to genesis version 5.0," Oct. 1990.
[24] C. Y. Ngo, "Genetic algorithms for discrete optimization and their applications to radio network design," Ph.D. thesis, Dept. Electr. Eng. Syst., Univ. Southern California, Los Angeles, CA, Aug. 1995.

Chiu Y. Ngo (S'82–M'86) was born in Hong Kong. He received the B.Sc. (Hons.) degree in electrical engineering from the University of Hong Kong, Hong Kong, in 1984 and the M.E. degree in electronic engineering from the NUFFIC/Philips International Institute, Eindhoven, The Netherlands, in 1986. He received the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the University of Southern California (USC), Los Angeles, in 1993 and 1995, respectively.

From 1986 to 1989, he was a Project Engineer in the Transmission Department of the Hong Kong Telephone Company. Since 1995, he has been a Senior Member of Research Staff, Philips Research Laboratories, Briarcliff Manor, NY, engaging in the research of wireless ATM. His current research interests include wireless communication networks, personal communication systems, multimedia communications, signal processing, and applications of GA's and fuzzy logic.

Victor O. K. Li (S'80–M'81–SM'86–F'92) was born in Hong Kong in 1954. He received the S.B., S.M., and Sc.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1977, 1979, and 1981, respectively.

In February 1981, he joined the University of Southern California (USC), Los Angeles, where he became Professor of Electrical Engineering and Director of the USC Communication Sciences Institute. Since September 1997, he has been with the University of Hong Kong, Hong Kong, China, where he is Chair Professor of Information Engineering, Department of Electrical and Electronic Engineering, and also the Managing Director of Versitech, Ltd., the University contract research and consulting company. His research interests include high-speed communication networks, personal communication networks, distributed multimedia systems, distributed databases, queueing theory, graph theory, and applied probability. He has lectured and consulted extensively around the world.

Dr. Li chaired the Computer Communications Technical Committee of the IEEE Communications Society from 1987 to 1989 and the Los Angeles Chapter of the IEEE Information Theory Group from 1983 to 1985. He was the Steering Committee Chair of the International Conference on Computer Communications and Networks (IC^3N), General Chair of the 1st Annual IC^3N, June 1992, Technical Program Chair of the Institution of Electrical Engineers (IEE) Personal Communication Services Symposium, June 1995, and Chair of the 4th IEEE Workshop on Computer Communications, October 1989. As a Member of ACM, he has served as an Editor of *IEEE Network* and *Telecommunication Systems* and Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNCATIONS and *Computer Networks and ISDN Systems*. He is now serving as an Editor of ACM Wireless Networks. He serves on the International Advisory Board of IEEE TENCON '90, IEEE TENCON '94, IEEE SICON '91, IEEE SICON '93, IEEE SICON/ICIE '95, the International Conference on Microwaves and Communications '92, and the International Symposium on Communications '91.