Wireless Personal Communications **3:** 273–286, 1996.
© 1996 *Kluwer Academic Publishers. Printed in the Netherlands.*

Channel Assignment in Cellular Radio Using Genetic Algorithms

J.-S. KIM, S. PARK, P. DOWD and N. NASRABADI
Department of Electrical and Computer Engineering, State University of New York at Buffalo, Buffalo, NY 14260, U.S.A.

Abstract. The channel assignment problem has become increasingly important in mobile telephone communication. Since the usable range of the frequency spectrum is limited, the optimal assignment problem of channels has become increasingly important. Recently Genetic Algorithms (GAs) have been proposed as new computational tools for solving optimization problems. GAs are more attractive than other optimization techniques, such as *neural networks* or *simulated annealing*, since GAs are generally good at finding an *acceptably good* global optimal solution to a problem *very quickly*. In this paper, a new channel assignment algorithm using GAs is proposed. The channel assignment problem is formulated as an energy minimization problem that is implemented by GAs. Appropriate GAs operators such as reproduction, crossover and mutation are developed and tested. In this algorithm, the cell frequency is not fixed before the assignment procedures as in the previously reported channel assignment algorithm using neural networks. The average generation numbers and the convergence rates of GAs are shown as a simulation result. When the number of cells in one cluster are increased, the generation numbers are increased and the convergence rates are decreased. On the other hand, with the increased minimal frequency interval, the generation numbers are decreased and the convergence rates are increased. The comparison of the various crossover and mutation techniques in a simulation shows that the combination of two points crossover and selective mutation technique provides better results. All three constraints are also considered for the channel assignments: the co-channel constraint, the adjacent channel constraint and the co-site channel constraint. The goal of this paper is the assignment of the channel frequencies which satisfied these constraints with the lower bound number of channels.

Key words: Cellular mobile network, channel assignment, Genetic Algorithms, wireless communication.

1. Introduction

There is an increasing demand for mobile telephone communication. At the same time, the usable range of the frequency spectrum is limited. Optimal frequency channel assignment is an increasingly important problem in order to use the available frequency spectrum most efficiently. In this paper, we describe the channel assignment algorithm which uses the Genetic Algorithms (GAs) [1].

GAs are adaptive search techniques that can find the global optimal solution by manipulating and generating recursively a new population of solutions from an initial population of sample solutions. The appropriate GAs operators such as reproduction, crossover and mutation, are developed by using a natural selection.

The following three conditions are considered as the electromagnetic compatibility constraints which were adapted in [2–5]: 1) co-channel constraint (CCC): for a certain pair of radio cells, the same frequency can not be used simultaneously; 2) adjacent channel constraint (ACC): the adjacent frequencies in the frequency domain cannot be assigned to adjacent radio cells simultaneously; 3) co-site constraint (CSC): any pair of frequencies assigned to a cell should have a minimal distance between frequencies.

The channel assignment problem is the assignment of the required number of frequency channels to each radio cell such that the above constraints are satisfied. When the co-channel constraint is considered only, this channel assignment problem is known to be equivalent to a graph coloring problem [6]. The graph coloring problem is a NP-complete problem [7], so an exact search for the best solution is impossible and the complexity of searching computation for the optimum solution grows exponentially as the problem size increases. Recently, *neural networks* [2, 5, 8, 9] and *simulated annealing* [10] have been considered for the channel assignment problems. Neural network algorithms [11, 12] are based on the behavior of the neurons in the brain. Simulated annealing [13] is a searching optimization technique which is based on a physical, rather than a biological process. One major disadvantage of a neural network is that it gives the local optimal value rather than the global optimal value, and the solution varies depending on the initial values. Simulated annealing is a stochastic algorithm which gives an optimal solution but may take a long time to find an optimal solution. On the other hand, GAs are generally good at finding an *acceptably good* solution to a problem *very quickly*, although GAs are not guaranteed to find the global optimum solution to a problem [14].

Gamst [4] defined the compatibility matrix $C = (c_{ij})$, which is an $N \times N$ symmetric matrix where N is the number of cells in the mobile network, and c_{ij} is the minimum frequency separation between a frequency in cell i and cell j. For example, $c_{ij} = 0$ indicates that the same channel can be used in cell i and cell j. Hence, CCC and ACC can be represented in matrix C by $c_{ij} = 1$ and $c_{ij} = 2$, respectively. CSC also can be represented by the certain value of c_{ii}. The number of required channels for each cell i is presented by the demand vector $M = (m_i)$ where $1 \leq i \leq N$. Let f_{ik} indicate the assigned frequency for the kth call in cell i where $1 \leq i \leq N$ and $1 \leq k \leq m_i$. The condition for the compatibility constraints is the following:

$$|f_{ik} - f_{jl}| \geq c_{ij} \tag{1}$$

where $1 \leq i, j \leq N$ and $1 \leq k, l \leq m_i$. The channel assignment problem is to find the value of f_{ik} which satisfies the constraint conditions when the number of cells in the mobile network N, the demand vector M and the compatibility matrix C are given.

The contribution of this paper is in the application of GAs to assign frequency channels with appropriate genetic operators and an energy function. The proposed assignment technique uses the implicit parallelism of GAs to find the frequencies which satisfy all constraints. An energy function is derived which represents the constraints that should be satisfied in order to find the best assignment. The various genetic operators and representation structure for a population are developed and compared.

Also in this paper, the cell frequencies are not fixed before the assignment procedure as in the previously reported channel assignment algorithm [5]. In [5], wide variations of performance could occur with its algorithm depending on which cells are fixed and how the frequencies of the fixed cells are assigned.

Section 2 presents a short introduction to GAs. In section 3, GAs are applied to channel assignment problems and the various GAs operators are explained. Section 4 presents the simulation results.

2. Genetic Algorithm

Genetic Algorithms (GAs) [14–16] are adaptive methods which may be used to solve search and optimization problems. GAs are algorithms based on an analogy with nature as are neural networks. GA techniques are robust and they can deal successfully with a wide range of problem areas such as image processing [17], routing problems [18, 19], and load balancing for distributed systems [20].

The idea of a GA is that the combination of good characteristics from different ancestors can produce *superfit* offspring whose fitness to the new environment is greater than that of either parent. In this way, the species can evolve to become more and more suited to their environments. The highly suitable individuals are given opportunities to *reproduce*, by *cross breeding* with other individuals in the population. This makes new individuals as *offspring*, and this offspring shares some features taken from each *parent*. By favoring the mating of the more suitable individuals, the most promising areas of the search space are explored. If a GA has been designed well, the population will *converge* to an optimal solution to the problem [14]. *Crossover* is the randomly chosen point which cuts two individual chromosome strings in two pieces, *head* segments and *tail* segments. Two tail segments are swapped over to produce two new full length chromosomes. The *mutation* is applied to each child individually with very small probability, after the *crossover*. The mutation provides a small amount of random search and helps ensure that no point in the search space has a zero probability of being examined. However, the crossover is more important for rapidly exploring a search space [14]. When a GA has been correctly implemented, the fitness of the best and the average individual in each generation increases towards the global optimum. *Convergence* is the progression towards increasing uniformity.

3. Genetic Algorithm Approach

Generally, GAs have two steps in the algorithm. First, the initial population is needed. Second, for each generation, GAs are operated for the solution by evaluating the *energy function*. If one has a smaller value of this energy function, it is more desirable for the optimization of the problem. It will be described later in greater detail.

GAs are randomized parallel search strategies which can find the optimal solution for a particular problem by seeking the maximum/minimum of an appropriate energy function. The strength of GAs lies in finding very quickly good optimal solutions in a complex search space. This is the reason for using GAs over other optimization techniques, such as neural networks [11, 12] and the simulated annealing [13].

3.1. Population and Strings

A GA is an iterative procedure that maintains a set of candidate solutions called the *population* $P(t)$ for each iteration t. At each iteration a new population $P(t+1)$ is created from the previous population $P(t)$ using a set of genetic operators.

A population consists of a number of possible candidate solutions called *strings* S_r where $1 \leq r \leq P$ and P is the population size. Each string consists of an array of genes which may take some values called *alleles*. A GA starts with $P(0)$, a randomly generated initial population of possible solutions. During each iteration t, called a generation, strings in the current population $P(t)$ are evaluated on the basis of a fitness function. After evaluation, strings that have high performance (goodness factor) relative to the performance of the other strings in the population are selected to be used in the next generation $(t+1)$ to create a new population, $P(t+1)$. This iterative process of creating new strings is called *reproduction*. To generate a new population for next generation, usually two selected strings are recombined by specific genetic

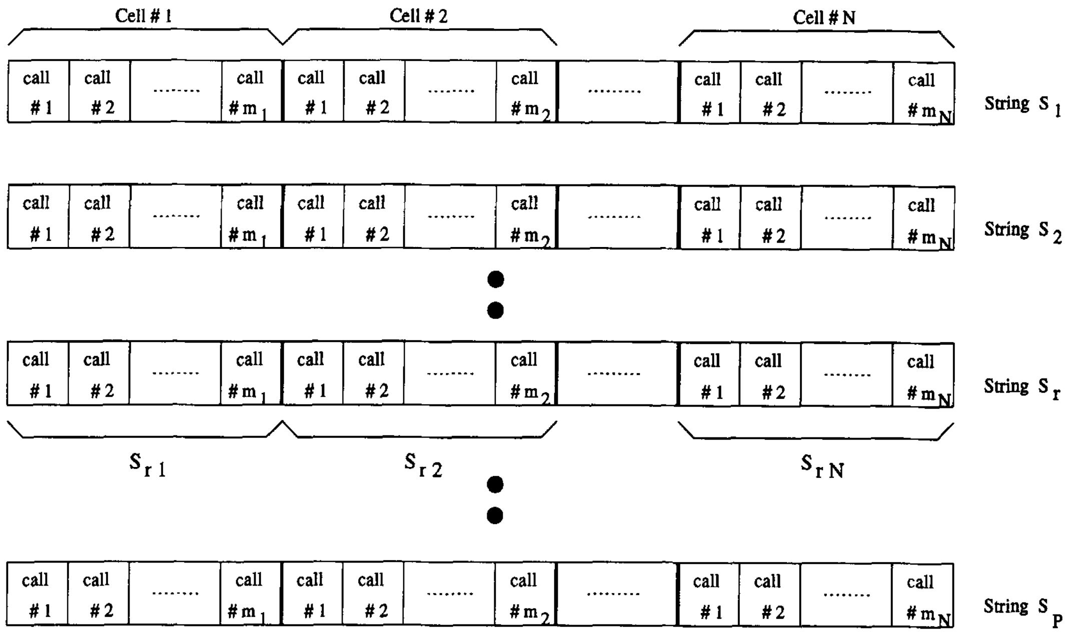


Figure 1. Structure of the population strings.

operators such as *crossover* and *mutation.* This process is called recombination. In GAs, the choice of the fitness function and the genetic operators will dominate the performance.

A population can be represented by an array of strings (individuals) as depicted in Figure 1. The rows of the array represent strings in a population, and the columns represent the channel numbers which will be assigned. There are P strings for a population and each string has Q calls which is the total number of calls in the system. The total number of calls in the system, Q in the model-database, is a sum of the number of calls in all cells which is given by

$$Q = \sum_{i=1}^{N} m_i \tag{2}$$

where each cell i has m_i calls.

A string S_r is composed of N substrings which is the number of cells in the network. Each substring S_{ri} (for cell i) is composed of m_i calls.

For example, the rth string S_r in a population $P(t)$ may be $S_r = (1, 2, 3, \ldots, \mathrm{j}, \ldots, Q)$ and a substring S_{ri} for the cell i in string S_r may be $S_{ri} = (1, 2, \ldots, m_i)$, therefore, $S_r = S_{r1}$ $S_{r2} \ldots S_{rN}$. A $P{\times}$Q two-dimensional array is constructed to implement a number of strings (a population) as shown in Figure 1. In classical GAs, each gene is represented as a binary bit, however, in our algorithm we assign an integer value to each gene which represents the frequency numbers.

3.2. Energy Function

We define the energy function for each constraint. CCC and ACC are considered together since both constraints can be represented by the value of c_{ij} when $i \neq j$.

1) Energy function for CSC:

For CSC, the energy function E_{CSC_i} for each cell i is given by

$$E_{CSC_i} = \sum_{k=1}^{m_i} V_{ik} \tag{3}$$

where

$$V_{ik} = \begin{cases} 1 & \text{if } |f_{ik} - f_{i(k+1)}| < c_{ii} \text{ or } |f_{ik} - f_{i(k-1)}| < c_{ii} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

V_{ik} represents the satisfaction state for CSC. If CSC is not satisfied, then $V_{ik} = 1$. On the other hand, if CSC is satisfied, then $V_{ik} = 0$.

The energy function for each string S_r which includes all cells is given by

$$E_{CSC} = \sum_{i=1}^{N} \sum_{k=1}^{m_i} V_{ik} \tag{5}$$

2) Energy function for ACC and CCC:

Energy function for ACC and CCC for each string, E_{AC} is given by

$$E_{AC} = \sum_{i=1}^{N} \sum_{k=1}^{m_i} \sum_{j=1}^{N} \sum_{l=1}^{m_j} V_{ikjl} \tag{6}$$

where

$$V_{ikjl} = \begin{cases} 1 & \text{if } |f_{ik} - f_{jl}| < c_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

i and j indicate the cell numbers and k and l are call numbers in each cell.

Finally, we consider the total energy function for each string S_r

$$\begin{aligned} E_{S_r} &= E_{CSC_{S_r}} + E_{AC_{S_r}} \\ &= \sum_{i=1}^{N} \sum_{k=1}^{m_i} V_{ik} + \sum_{i=1}^{N} \sum_{k=1}^{m_i} \sum_{j=1}^{N} \sum_{l=1}^{m_j} V_{ikjl} \end{aligned} \tag{8}$$

To reproduce offspring, each string of parents should have a probability to be selected. Our fitness for reproduction consists of probabilities of selection in order to choose more strings which have good candidate solutions. The fitness function for this selection is given by

$$F_{S_r} = \frac{\frac{1}{E_{S_r}}}{\sum_{n=1}^{P} \frac{1}{E_{S_n}}} \tag{9}$$

where S_r is the string number and P is the population size.

Initial population: The procedure of the initial population for each string is as follows:

1. For the cell i^* with the largest number of calls, the channel frequency for the kth call is given by $f_{i^*k} = (k-1) \times \gamma + 1$. γ is the minimal frequency interval in the maximum demand cell i^*, and is given by $\gamma = \left\lfloor \frac{LB}{m_{i^*}} \right\rfloor$ where LB is the lower bound of the total number of required frequencies in the system and $\gamma > c_{i^*i^*}$.

2. For the cell i with the next largest number of calls,
   a) compute the number of available frequencies in the subgroup whose size is γ.
   b) In the available frequency block from step a), randomly choose the frequency.
   c) Randomly choose the frequency subgroup to assign the frequency.
   Assign the frequency in the chosen subgroup and assign the next frequency for the next call by the interval of γ with the previously assigned frequency. The assignment also should satisfy the condition of c_{ii^*} according to CCC or ACC.
   d) Repeat step c) until all calls have the assigned frequencies.
3. Repeat step 2 until all cells have the assigned frequencies.

In the simulation of this paper, the population size is set to be 200. Hence, the initial population procedures are repeated 200 times.

Reproduction: After calculation of the fitness function, a certain pair of strings should be selected for the parents. A simple biased roulette wheel is used to select strings for our experiments. Each string in the population has a roulette wheel slot size in proportion to the ratio of its fitness over the total sum of fitness in the population. A random number between 0 and 1 is generated for each selection. A string is selected for reproduction if the random number is within the range of its roulette wheel slot. The copy of the selected string is gathered into a mating pool, in which they are mated for further genetic operation. Strings which have higher fitness have higher probabilities of selection so that those with higher fitness produce more offsprings than those with lower fitness in the next generation.

Crossover: The reproduced strings in the mating pool are mated under crossover operation at random. Crossover operation is performed with a pair of substrings in the mated strings for each model. Three crossover techniques are considered: uniform crossover [21], one point crossover and two point crossover.

- Uniform crossover (X0): If the random generated probability P_{rf} for each assigned frequency in a substring S_{ri}, is greater than the crossover probability P_X ($P_{rf} > P_X$), the assigned frequencies are swapped in the mated strings. It is repeated for all frequencies in all substring S_{ri} where $1 \leq i \leq N$.
- One point crossover (X1): If the random generated probability P_r is greater than the crossover probability P_X ($P_r > P_X$), the crossover point is generated randomly and the frequency strings after the crossover point are swapped in the mated strings.
- Two point crossover (X2): If $P_r > P_X$, two crossover points are generated randomly and the assigned frequencies between two crossover points are swapped in the mated strings.

Mutation: Mutation is a process to find a new search space by changing the value of a randomly chosen position in a substring chosen at random. Although reproduction and crossover operators will search the solution space effectively, occasionally they may lose some useful solution patterns. The mutation operator will prevent such an irrecoverable loss and will protect the algorithm from becoming trapped in a local minimum. This will enable it to jump to the global minimum. In this paper, we consider the five mutation techniques:

- Mutation 1 (M1): If the random generated probability P_r is greater than the mutation probability P_M ($P_r > P_M$), assign the randomly selected frequency f_{i1} to the 1st call of the cell i and assign $(f_{i(j-1)} + \gamma)$ to the next call j where $2 \leq j \leq m_i$, according to CSC.
- Mutation 2 (M2): If $P_r > P_M$, move the every assigned frequency f_{ij} for the call j in the cell i to the call $((j + l) \bmod m_i)$ where l is the randomly chosen integer and $l < \gamma$.
- Mutation 3 (M3): If $P_r > P_M$, change the every assigned frequency f_{ij} of the call j in the cell i to $(f_{ij} \pm \Delta)$ where Δ is the randomly chosen small integer.
- Mutation 4 (M4) and Mutation 5 (M5):

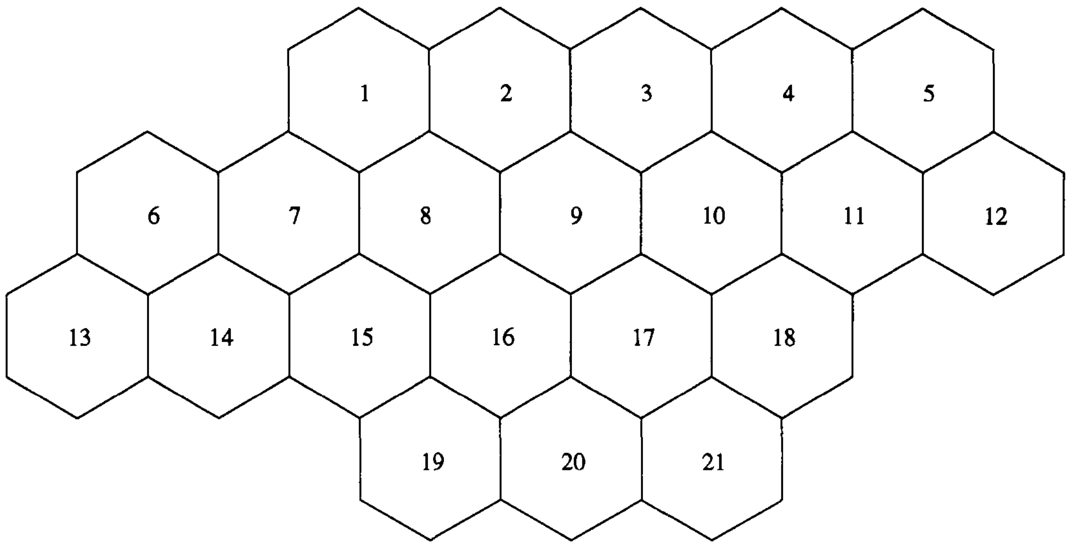


Figure 2. The 21 cell system used in this paper.

Table 1. Specification of the problems and simulation results with two point crossover (X2) and mutation technique M5.

Problem No.	Cell #	N_c	ACC	c_{ii}	LB	Ave. Gen. No.	Conv. Rate
1	21	12	1	5	381	26.95	49/50
2	21	7	1	5	381	0.28	50/50
3	21	12	1	7	533	5.07	50/50
4	21	7	1	7	533	0.0	50/50
5	21	7	2	7	533	7.43	50/50

1. For the already assigned frequency of each cell, compare its frequency with the frequency of other cells.
2. Calculate the frequency difference from step 1 and compare it with the compatibility matrix.
3. If the frequency difference from step 2 is the less than the value of the compatibility matrix and $P_r > P_M$,
   a) M_4: it is mutated by M_1.
   b) M_5: it is mutated by M_1 and M_3.

In mutation techniques M4 and M5, only when the already assigned frequencies do not fit the compatibility matrix, it executes the mutation. Therefore, if the already assigned frequencies prior to the mutation procedure satisfy the compatibility matrix, the mutation process does not occur since they are already *fitted* offsprings for the environment.

The energy function, the fitness function and the creating procedures of new strings generate new populations until the termination condition is reached. Once termination condition is reached, the best string in the final population will be chosen as the solution. Iteration of GAs may terminate by determining the maximum number of iterations or after finding the string S_r which has $E_{S_r} = 0$.

4. Simulation Results

The cellular network system for this paper is the 21 cell system [22] which is shown in Figure 2. The total number of calls is 481, which is the sum of the calls in the total number of cells. The population size is 200 and the maximum number of generations is fixed at 100. The algorithm can be terminated prior to the $100th$ generation if and only if the algorithm could find the string S_r which has $E_{S_r} = 0$. The crossover and mutation probabilities are set to be $P_X = 0.9$ and $P_M = 0.03$, respectively. When multiple mutation techniques are used, the sum of all mutation probabilities is set to 0.03. Table 1 shows the specification of the problems. The 5 cases, Problems #1-#5 in Table 1 which were used in [5, 23], are experimented as a benchmark. N_c is the number of cells in one cell cluster and ACC implies the presence of adjacent channel constraint (ACC) on adjacent cells. A "2" or "1" in ACC column implies the presence and absence of ACC respectively. The co-site constraint (CSC) is indicated by the value in column c_{ii}. LB is the theoretical lower bound of the number of required channels which is obtained using the bounds in [22]. In [23], 8 problems are considered since the constants N_c, ACC and c_{ii} have two values each. In [23], various algorithms are proposed and their performance compared in terms of total number of required channels which is sometimes greater than theoretical LB. However, our paper has a different objective. Our paper shows the feasibility of GAs in channel assignment problems and the various crossover and mutation techniques in GAs are proposed and investigated with the *given* LB as an input. If the number of available channels are increased as in [23], which is larger than theoretical LB, the channels for all calls in each cell for remaining three cases, can be assigned without violation of constraints. The performance of these techniques in GAs are compared in terms of convergence rate and average generation number to the solution. GA is a heuristic optimization technique which is not always guaranteed to find the global minimum (*e.g.*, successful channel assignment without violation of constraints). Therefore, the convergence rate is an important parameter to compare the performance. A similar objective was pursued in [5, 2, 24] using neural networks. In [5], the results are also compared in terms of convergence rate and average iteration numbers, for the cases whose channel assignments are possible with the theoretical LB. Hence, the remaining three cases are not considered in [5].

Figure 3 (a) and (b) show the compatibility matrices for Problems #1 and #5 as examples, which have $ACC = 1$ and $ACC = 2$ respectively. Figure 3 (c) shows the demand vector which is used for the simulations. Table 1 also shows the results of simulations with two point crossover (X2) and mutation technique M5. The average generation number is the average number of generations until $E_{S_r} = 0$. Convergence rate is the probability that the experiment has $E_{S_r} = 0$ before the maximum generation number. To investigate the number of generations and the convergence rate, 50 simulation runs were performed from the different initial seed values for random generators for each of the five problems.

In our GAs, the frequencies for the cells are assigned in order of the number of calls in the cell. In Table 1, Problem #1 with the large N_c has a larger average generation number and a smaller convergence rate than Problem #2 with the small N_c, when they have the same value of c_{ii} and same application of ACC. When the value of N_c is large, there is more cells in one cluster and the reusable distance of the same frequency is increased. With the increased reusable frequency distance, the number of assignable frequencies for other cells within the minimal frequency interval whose size is γ, is decreased. It causes more generation numbers and less convergence rates. On the other hand, in the case of Problems #1 and #3, Problem # 1 with the small c_{ii} has a larger average generation number and a smaller convergence rate than

```
5 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0     7 2 0 0 0 0 2 2 0 0 0 0 0 0 0 0 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     2 7 2 0 0 0 0 2 2 0 0 0 0 0 0 0 0 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     0 2 7 2 0 0 0 0 2 2 0 0 0 0 0 0 0 0 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     0 0 2 7 2 0 0 0 0 2 2 0 0 0 0 0 0 0 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     0 0 0 2 7 0 0 0 0 0 2 2 0 0 0 0 0 0 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     0 0 0 0 0 7 2 0 0 0 0 0 2 2 0 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     2 0 0 0 0 2 7 2 0 0 0 0 0 2 2 0 0 0 0 0 0     18
1 1 1 0 0 1 1 5 1 1 0 0 0 1 1 1 1 0 1 1 0     2 2 0 0 0 0 2 7 2 0 0 0 0 0 2 2 0 0 0 0 0     52
1 1 1 1 0 0 1 1 5 1 1 0 0 0 1 1 1 1 1 1 1     0 2 2 0 0 0 0 2 7 2 0 0 0 0 0 2 2 0 0 0 0     77
0 1 1 1 1 0 0 1 1 5 1 1 0 0 0 1 1 1 0 1 1     0 0 2 2 0 0 0 0 2 7 2 0 0 0 0 0 2 2 0 0 0     28
0 0 1 1 1 0 0 0 1 1 5 1 0 0 0 0 1 1 0 0 1     0 0 0 2 2 0 0 0 0 2 7 2 0 0 0 0 0 2 0 0 0     13
0 0 0 1 1 0 0 0 0 1 1 5 0 0 0 0 0 1 0 0 0     0 0 0 0 2 0 0 0 0 0 2 7 0 0 0 0 0 0 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     0 0 0 0 0 2 0 0 0 0 0 0 7 2 0 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     0 0 0 0 0 2 2 0 0 0 0 0 2 7 2 0 0 0 0 0 0     15
1 1 0 0 0 1 1 1 1 0 0 0 1 1 5 1 1 0 1 1 0     0 0 0 0 0 0 2 2 0 0 0 0 0 2 7 2 0 0 0 0 0     36
1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 5 1 1 1 1 1     0 0 0 0 0 0 0 2 2 0 0 0 0 0 2 7 2 0 2 2 0     57
0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 5 1 1 1 1     0 0 0 0 0 0 0 0 2 2 0 0 0 0 0 2 7 2 0 2 2     28
0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 5 0 1 1     0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 0 2 7 0 0 2      8
0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 5 1 1     0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 7 2 0     10
0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 5 1     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 2 7 2     13
0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 5     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 2 7      8

                  (a)                                            (b)                           (c)
```

Figure 3. Compatibility matrix and Demand vector (a) Compatibility matrix for Problem #1. (b) Compatibility matrix for Problem #5. (c) Demand vector for all problems.

Problem #3 with the large c_{ii}, when they have the same value of N_c and same application of ACC. When the value of c_{ii} is larger, LB is increased and the minimal frequency interval γ is increased. Within the increased interval γ, there will be more assignable frequencies and it is easier to assign the frequencies without the violation of the constraints for the calls in other cells

In Problem #4, the generation number is 0. This means that during the initial population procedure, the frequencies of all calls in every cell, which satisfy the constraints condition, are found. Problem #5 is the same as Problem #4 except the application of ACC. It shows that the ACC application causes more generation numbers for the solution with no decrease in the convergence rate. Table 1 also shows that all problem cases have 100 % convergence rate except Problem #1 which has 98 %. The convergence rates of our results are equal to or higher than the ones given in [5]. For example, the convergence rates of Problems #1 and #3 in [5] are 93 % and 100 %, respectively. Also in [5], they fixed a couple of frequencies in certain cells for the acceleration of the convergence time. The convergence time and rates will be totally different depending on which cells are fixed and how the frequencies of the fixed cells are assigned. In our GAs, the frequencies are not fixed. The average iteration number to the solution for Problems #1 and #3 in [5] is 147.8 and 117.5, respectively. Although the average generation numbers in GAs and the average iteration numbers in neural networks cannot be compared directly since they are different in computational complexity, it shows that the average generation numbers of GAs are relatively smaller.

Another advantage of using GAs in this problem, is the simple hardware implementation since GAs needs the function of only swapping and shifting, rather than the adder, the comparator and the inverter as in the neural networks. The second advantageous characteristic of GAs is its implicit parallelism. In GAs, the all crossover (or mutation) operators of the strings can be processed at the same time since the outputs of these operators are not used as the inputs of other operators like in the neural networks. Refer to [3] for the detailed comparison of genetic algorithms and neural networks.

In [23], the algorithm results are shown from 8 different cases. The 8 difference cases are from the combination of 3 different techniques in which each of them has two kinds of methods

Table 2. Comparison of the various crossover and mutation techniques for Problems #1 and #3.

Crossover	Mutation	Problem #1		Problem #3	
		Ave.Gen.No.	Conv.Rate	Ave.Gen.No.	Conv.Rate
X0	M1	23.81	33/50	4.89	39/50
X0	M2	20.91	38/50	5.28	49/50
X0	M3	27.60	28/50	5.54	50/50
X0	M4	17.84	44/50	4.44	50/50
X0	M5	23.54	42/50	4.88	50/50
X1	M1	27.31	35/50	4.98	50/50
X1	M2	22.99	33/50	4.89	48/50
X1	M3	25.86	23/50	5.51	50/50
X1	M4	20.17	45/50	4.58	50/50
X1	M5	25.37	45/50	4.74	50/50
X2	M1	28.78	33/50	7.08	48/50
X2	M2	21.61	29/50	8.80	47/50
X2	M3	28.64	28/50	6.95	49/50
X2	M4	22.26	46/50	5.15	50/50
X2	M5	26.95	49/50	5.07	50/50

($2 \times 2 \times 2 = 8$). The used constraints in [23] are 3 constraints: the co-channel constraint, the adjacent channel constraint and the co-site channel constraint. The same constraints are applied in our paper. The basic idea of their algorithm is to list the calls in the cells in some order. The 3 different techniques used in [23] are follows:

1. According to the frequency assignment methods [25, 4].

   In *frequency exhaustive strategy*, assign the *least* possible frequency to each call starting at the top of the list. In *requirement exhaustive strategy*, assign the frequency #1 to the first call in the list and reassign the same frequency #1 to all calls in the list if the assignment does not violates the constraints. Assign the frequency #2 to the next call and other possible calls, and so on.
2. According to ordering method of the cells.

   In *node-degree ordering*, the cells are arranged in the decreasing order of the degrees where *degree* is the heuristic measure of the difficulty of a frequency to a call in that cell. In *node-color ordering*, the cells are arranged by the 'highest degree first' and 'least degree last' heuristics in graph coloring.
3. According to ordering methods of all calls after ordering the cells by using one of two cell ordering methods.

   The matrix of the calls can be listed by *row-wise ordering* or *column-wise ordering*.

In [23] the required number of frequencies is larger than the lower bound in many cases. In our algorithm, the total number of frequencies required was equal to the lower bound number (LB) since the available frequency spectrum was fixed as a maximum usable frequency number.

Table 2 shows the average generation number and convergence rate of Problems #1 and #3 for each crossover and mutation techniques. It is shown that the performance of the algorithm is more greatly affected by the mutation technique rather than the crossover technique for the given examples. When the selective mutation techniques (M4 or M5) are used, the convergence rates are higher since the mutation only occurs when the already assigned frequencies do not fit the compatibility matrix. With non-selective mutation techniques (M1, M2 and M3), the assigned frequencies can be changed by the mutation even when the frequencies already fit the compatibility matrix, and it causes a lower convergence rate.

5. Conclusion

The results observed in this paper show that Genetic Algorithms (GAs) can be applied to obtain the optimal solution for the channel assignment in mobile cellular environment. GAs have an advantage over neural networks or simulated annealings in that GAs are generally good at finding an *acceptably good* global optimal solution to a problem *very quickly*. The average generation numbers and the convergence rates of GAs are shown as a simulation result. The optimal solution can be found with the small generation numbers and high convergence rates. An expected outcome of the simulation was that the performance of the algorithm varied depending on the GAs operators. The comparison of the various crossover and mutation techniques illustrates that the combination of two point crossover and selective mutation technique provides the relatively better results. When the number of cells in one cluster are increased, the generation numbers are increased and the convergence rates are decreased due to the increased reusable frequency distances. On the other hand, when the minimal frequency interval is increased with the increased LB, the generation numbers are decreased and the convergence rates are increased since there will be more assignable frequencies within the minimal frequency interval for the cells.

References

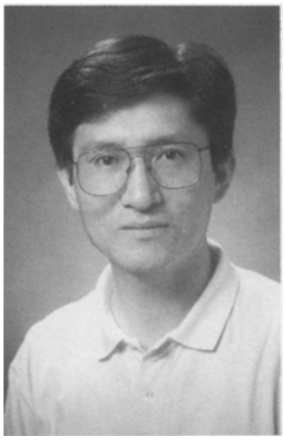
1. J.-S. Kim, S. H. Park, P. W. Dowd, and N. M. Nasrabadi, "Genetic algorithm approach to the channel assignment problem," in *Proc. 1995 Asia-Pacific Conference on Communications*, pp. 564–567, Jun. 14-16 1995.
2. J.-S. Kim, S. H. Park, P. W. Dowd, and N. M. Nasrabadi, "A modified hopfield network approach for cellular radio channel assignment," in *Proc. 45th IEEE Vehicular Technology Conference*, pp. 589–593, Jul. 26-28 1995.
3. J.-S. Kim, S. H. Park, P. W. Dowd, and N. M. Nasrabadi, "Comparison of two optimization techniques for channel assignment in cellular radio network," in *Proc. IEEE International Conference on Communications*, pp. 1850–1854, Jun. 18-22 1995.
4. A. Gamst and W. Rave, "On frequency assignment in mobile automatic telephone systems," in *Proc. IEEE GLOBECOM'82*, pp. 309–315, 1982.
5. N. Funabiki and Y. Takefuji, "A neural network parallel algorithm for channel assignment problems in cellular radio networks," *IEEE Transactions on Vehicular Technology*, vol. VT-41, pp. 430–436, Nov. 1992.
6. W. K. Hale, "Frequency assignment: Theory and applications," *Proceedings of IEEE*, vol. 68, pp. 1497–1514, Dec. 1980.
7. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guid to the Theory of NP-Completeness.* New York: W.H.Freeman and Co., 1979.
8. D. Kunz, "Channel assignment for cellular radio using neural networks," *IEEE Transactions on Vehicular Technology*, vol. VT-40, pp. 188–193, Feb. 1991.
9. 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. VT-43, pp. 279–288, May 1994.
10. M. Duque-Anton, D. Kunz, and B. Ruber, "Channel assignment for cellular radio using simulated annealing," *IEEE Transactions on Vehicular Technology*, vol. VT-42, pp. 14–21, Feb. 1993.
11. J. E. Dayhoff, *Neural Network Architectures: an Introduction.* New York, NY: Van Nostrand Reinhold, 1990.
12. J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

13. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, May 1983.
14. D. Beasley, D. R. Bull, and R. R. Martin, "An Overview of Genetic Algorithms:Part I, Fundamentals," *University Computing*, vol. 15, no. 2, pp. 58–69, 1993.
15. J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, 1975.
16. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison Wesley, 1989.
17. A. K. Bhattacharjya, D. E. Becker, and B. Roysam, "A genetic algorithm for intelligent imaging from quantum-limited data," *Signal Processing*, vol. 28, pp. 335–348, 1992.
18. Z. Michalewicz, "A step towards optimal topology of communication networks," *Proc. of SPIE*, vol. 1470, pp. 112–122, Apr. 1991.
19. R. Chandrsekharam, S. Subhramanian, and S. Chaudhury, "Genetic algorithm for node partitioning problem and applications in VLSI design," *IEE Proceedings E*, vol. 140, pp. 255–260, Sept. 1993.
20. A. V. Sannier and E. D. Goodman, "Midgard:a genetic approach to adaptive load balancing for distributed sytems," in *Proc. of* 5^{th} *International Conference on Machine Learning*, pp. 174–180, June 1988.
21. G. Syswerda, "Uniform crossover in genetic algorithm," in *Proc. of International Conference on Genetic Algorithms*, pp. 2–9, 1989.
22. A. Gamst, "Some lower bounds for a class of frequency assignment problems," *IEEE Transactions on Vehicular Technology*, vol. VT-35, pp. 8–14, Feb. 1986.
23. K. N. Sivarajan, R. J. McEliece, and J. W. Ketchum, "Channel assignment in cellular radio," in *Proc. 39th IEEE Vehicular Technology Conference*, pp. 846–850, May 1989.
24. 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*, (Submitted and Revised), 1994.
25. J. Zoellner and C. Beall, "A breakthrough in spectrum conserving frequency assignment technology," *IEEE Transactions on Electromagnetic Compatibility*, vol. EMC-19, pp. 313–319, Aug. 1977.

Jae-Soo Kim received the B.S. degree in electronics engineering from Hanyang University, Seoul, Korea, the M.S. degree from Iowa State University, Ames, Iowa, and the Ph.D. degree from State University of New York at Buffalo, all in electrical engineering. Now he is working as a Member of Technical Staff at AT&T Bell Laboratories.

His current research interests are mobile and personal communication networks, wireless ATM and high speed optical communications.

He is a member of IEEE and ACM.



Sahng H. Park received B.S. degree in electronics from Kyungpook National University, Taegu, Korea, M.S. degree in electronics from Yeung-nam University, Taegu, Korea, M.S. degree in computer engineering from Syracuse University, and Ph.D. degree in electrical engineering at State University of New York at Buffalo.

His research interests are in image and video compression, and wireless communications.

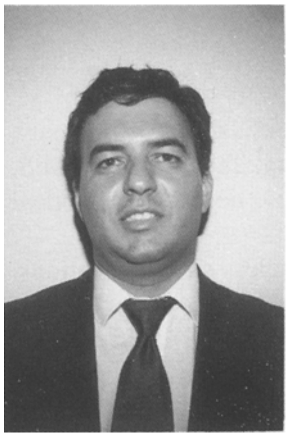
Patrick W. Dowd received the Bachelor of Science in Electrical Engineering and Computer Science (Summa Cum Laude) from the State University of New York at Buffalo, and the M.S. and Ph.D. degrees in Electrical Engineering from Syracuse University.

Dowd was with the IBM Corporation between 1983–1989 as a Staff Engineer with System Design in Processor Development at the IBM Endicott Laboratory. His principal design effort was in the area of communication subsystems for future computer systems, involving Token Ring, FDDI, Fiber Channel, and ISDN. Early assignments were in the areas of processor design: microcode, fault detection, isolation and diagnosability. Patrick joined the Department of Electrical and Computer Engineering at the State University of New York at Buffalo as an Assistant Professor in 1989.

Dowd has served as Conference Co-Chair of the SPIE OE/Fiber'92 Conference on High Speed Fiber Networks and Channels, Program Chair of the 27th Annual IEEE Simulation

Symposium, Program Chair of IEEE International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'95), the Technical Committee on Computer Communications Representative to GLOBECOM'95, and currently serves on the Program Committee of IEEE INFOCOM, IEEE GLOBECOM, ACM SIGCOMM, IEEE HPDC, IEEE MPP–OI, and IEEE MASCOTS.

He is a Member of the IEEE, ACM and SPIE, with research interests in reconfigurble networks, high-speed switching, optical communication, computer communication, multimedia based networks, and distributed databases.



Nasser M. Nasrabadi received the B.Sc. (Eng.) and Ph.D. degrees in Electrical Engineering from Imperial College of Science and Technology (University of London), London, England, in 1980 and 1984, respectively.

From 1986 to 1991 he was assistant professor in the Department of Electrical Engineering at Worcester Polytechnic Institute, Worcester, MA. Since 1991, he has been an associate professor in the Electrical and Computer Engineering Department at State University of New York at Buffalo, Buffalo, NY. He is an associate editor for the IEEE Transactions on Image Processing. His primary research interests are in image processing/video, and neural networks applications to image processing.