Comparison of Two Optimization Techniques for Channel Assignment in Cellular Radio Network

JAE-SOO KIM, SAHNG H. PARK, PATRICK W. DOWD AND NASSER M. NASRABADI
Department of Electrical and Computer Engineering
State University of New York at Buffalo
Buffalo, NY 14260

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. Two optimization methods, the neural networks and the genetic algorithms are applied for the channel assignment problem. To avoid falling into the local minima in the neural networks, certain techniques such as the forced assignment and the changing cell list order, are used. In the genetic algorithms approach, the proper genetic operators are developed. All three constraints are also considered for the channel assignments: the co-channel constraint, the adjacent channel constraint and the co-site channel constraint. As simulation results, the average iteration (or generation) numbers, the convergence rates and the cpu times according to the various techniques for the two approaches are presented and compared.

Index Terms: cellular mobile network, channel assignment, neural network, genetic algorithm.

1 INTRODUCTION

Recently, *neural network* [1–4] and *genetic algorithms* [5] have been considered for the channel assignment problems. Neural networks (NNs) are based on the behavior of the neurons in the brain. The principal strength of this approach is the performing capability of the efficient local search with guaranteed convergence to feasible solutions. On the other hand, the principal weakness is the inability to perform efficient global search, or to continue the algorithm in an effort to improve upon the obtained solution.

Genetic algorithms (GAs) [6] are adaptive methods which may be used to solve search and optimization problems by manipulating and generating recursively a new population of solutions from an initial population of sample solutions. GAs are powerful precisely in the area where NNs are weak, namely, in global search. However, for constrained optimization, GAs have their own weaknesses. In certain cases, the majority of the function of the population remains far away from any feasible solution and therefore does not efficiently search the feasible solution space.

In this paper, two channel assignment algorithms are presented and compared, which use a modified Hopfield neural network [4] and GAs [5] respectively. Our modified Hopfield network algorithm uses the forced assignment and the changing cell list order technique to inhibit falling into the local minima which is the disadvantage of NNs. In GAs approach the fitness function and the genetic operators are developed according to the constrained condition in the channel assignment problem.

Three conditions are considered in this paper as in [2]: co-site constraint (CSC), co-channel constraint (CCC) and adjacent channel constraint (ACC). The goal of the channel assignment problem in a cellular radio network is the assignment of the required channel numbers (RCNs) to each radio cell such that the above constraints are satisfied.

Section 2 describes the modified discrete Hopfield network approach which is developed. The initialization and updating techniques for the network is also explored. In Section 3, GAs are applied to the channel assignment and the operators of the algorithm such as crossover and mutation is described. Section 4 discusses the simulation results and compares two approach techniques.

2 NEURAL NETWORK APPROACH

A 2-D discrete Hopfield network is implemented by $(n \times m)$ neurons to minimize the energy function where n is the number of cells and m is the lower bound number (LB) of total required frequencies. With the considerations of three constraints and the total number of assigned channel numbers (ACNs), the total energy function for the channel assignment problem is as follows:

$$\begin{aligned} E &= \sum_{i=1}^{n}((r_i - \sum_{j=1}^{m} V_{ij})^2 + \sum_{j=1}^{m}\sum_{q=1}^{m} V_{ij}V_{iq}X_{ijq} \\ &+ \sum_{j=1}^{m}\sum_{p=1}^{n}\sum_{q=1}^{m} V_{ij}V_{pq}Y_{ijpq}) \end{aligned} \quad (1)$$

where

$$X_{ijq} = \begin{cases} 1 & \text{if } q \neq j \text{ and} \\ & j-(d_{csc}-1) \leq q \leq j+(d_{csc}-1) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

and

$$Y_{ijpq} = \begin{cases} 1 & \text{if } p \neq i \text{ and } c_{ip} > 0 \text{ and} \\ & j-(c_{ip}-1) \leq q \leq j+(c_{ip}-1) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

0-7803-2486-2/95 $4.00 © 1995 IEEE

The value of the processing unit V_{ij}, $1 \leq i \leq n$ and $1 \leq j \leq m$, indicates if frequency j is assigned to cell i: $V_{ij} = 1$ meaning that the frequency j is assigned to cell i and $V_{ij} = 0$ which means that the frequency j cannot be assigned to cell i. d_{csc} is the minimal distance between frequencies in the same cell for CSC. c_{ip} is the minimum frequency separation between a frequency in cell i and one in cell p. Also with the considerations of three constraints and the total number of ACN, the total interconnection weight is

$$T_{ijpq} = -\delta_{ip} - \delta_{ip}\alpha_{jq}(d_{csc}) - |(1 - \delta_{ip})|\alpha_{jq}(c_{ij}) \quad (4)$$

where $\delta = 1$ if $i = j$; $\delta = 0$ otherwise, and $\alpha_{ij}(x) = 1$ if $|i - j| < x$; $\alpha_{ij}(x) = 0$ otherwise. Input to each neuron of the modified network is defined as

$$S_{ij} = \sum_{p=1}^{n}\sum_{q=1}^{m} T_{ijpq}V_{pq} + I_i + I_{E_{ij}} \quad (5)$$

The external input I_i is defined as $I_i = (r_i - 1)$. To protect the algorithm from becoming trapped in a local minima, we incorporate the forced assignment method [4]. The difference between the ACN and the RCN constraints is used as an additional excitatory input given by:

$$I_{E_{ij}} = (r_i - \sum_{q=1}^{m} V_{iq}) \quad (6)$$

The output function consists of a threshold operation. The final output state of the neuron is,

$$V_{ij} = f_{out}(S_{ij}) \quad (7)$$

where

$$f_{out} = \begin{cases} 1 & \text{if } S_{ij} \geq \mathsf{THD} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

and THD is the threshold value and $\mathsf{THD} = 0$ in this paper.

With applying our modified Hopfield network to the channel assignment problem, the overall algorithm is summarized in the following steps.

1. The initial state of neurons are set to 1 or 0 according to the chosen initialization method.
2. Repeat the following steps until all neurons are picked.
   (a) Pick neuron V_{ij} according to the chosen updating method.
   (b) Calculate the input to this neuron by Eqn.(5).
   (c) Decide the new state of this neuron by Eqn.(7) and Eqn.(8).
3. Compute the energy E of the current assignment. If $E = 0$, stop and go to step 4 otherwise repeat the process from step 2.
4. The output state of the neurons V_{ij} will be the final assignment based on the compatibility matrix C and the RCN matrix R.

Initialization: The total frequency spectrum consists of B blocks of block width (BW) which is the number of channels in the block. The channel number in block (CNB) is the frequency number in the block and the sequence is repeated for each block. In this paper, two methods are identified for the initialization of the neurons [4]: (1) the fixed interval initialization method (A_1), (2) the random interval initialization method (A_2).

Updating procedure: In the algorithm, both random and sequential selection techniques are used along with three updating methods, U_1, U_2 and U_3. During the iteration subroutine, the first neuron to be updated, is selected randomly and the next updating neurons are selected sequentially. The updating methods U_2 and U_3 are modified from U_1. The procedure step of updating method U_1 is as follows:

1. Make a list of cells according to the descending order of RCN for each cell.
2. Execute the iteration subroutine which use the random and sequential techniques are used until the counter a has reached to a prespecified maximum iteration number 500 or $E = 0$.
3. Repeat step 2 for the next cell in the list.

In method U_2, if the energy E is less than threshold energy E_{th} and it is continued until the threshold energy counter b_{th} during the iteration subroutine, then change the list according to ascending order of RCN of each cell. This technique is used to avoid falling in local minima area. Updating method U_3 is similar to method U_1 except that the making of cells list are different. In method U_3, the list is made with alternating the largest and smallest number of RCN of cells. This end-packing technique [7] is used for the maximum utilization of the frequency spectrum.

3 Genetic Algorithm Approach

GAs are algorithms based on an analogy with nature as are neural networks. The idea of GA is that the combination of good characteristics from different ancestors can produce *superfit* offspring and its fitness to the new environments is greater than that of either parent. In this way, the species can evolve to become more and more well suited to their environments. In this section, the population and the strings of the GAs for the channel assignment are explained. The energy function is also derived and the operators for the assignment problems are presented.

A GA is an iterative procedure that maintains a set of candidate solutions called *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_p, $1 \leq p \leq P$ where P is the population size. A population can be represented by an array of strings (individuals) as depicted in Fig. 1. The rows of the array represent strings in a

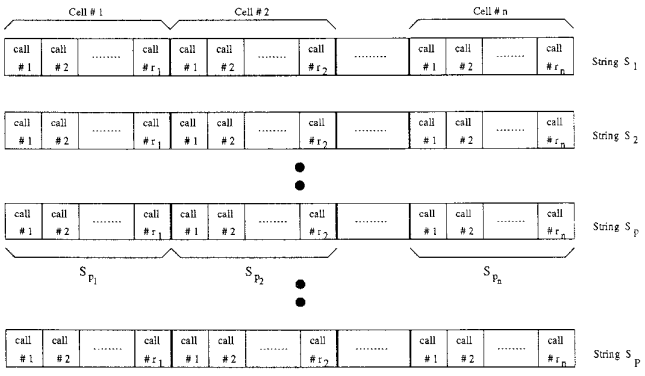

Figure 1: Structure of the population strings.

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 is a sum of the number of calls in all cells which is given by $Q = \sum_{i=1}^{n} r_i$ where each cell i has r_i calls. A string S_p is composed of n substrings which is the number of cells in the network. Each substring S_{p_i} (for cell i) is composed of r_i calls. A $P \times Q$ two-dimensional array is constructed to implement a number of strings (a population) as shown in Figure 1.

We define the energy function for each constraint. The total energy function for each string S_p

$$\begin{aligned} E_{S_p} &= E_{CSC_{S_p}} + E_{AC_{S_p}} \qquad (9) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{r_i} G_{ik} + \sum_{i=1}^{n}\sum_{k=1}^{r_i}\sum_{j=1}^{n}\sum_{l=1}^{r_j} G_{ikjl} \end{aligned}$$

where

$$G_{ik} = \begin{cases} 1 & \text{if } |f_{ik} - f_{i(k+1)}| < c_{ii} \text{ or } |f_{ik} - f_{i(k-1)}| < c_{ii} \\ 0 & \text{otherwise} \end{cases} \qquad (10)$$

and

$$G_{ikjl} = \begin{cases} 1 & \text{if } |f_{ik} - f_{jl}| < c_{ij} \\ 0 & \text{otherwise} \end{cases} \qquad (11)$$

i and j indicate the cell numbers and k and l are call numbers in each cell. $E_{CSC_{S_p}}$ is the energy function for CSC, and $E_{AC_{S_p}}$ is the energy function for CCC and ACC. CCC and ACC are considered together since both constraints can be represented by the value of c_{ij} when $i \neq j$. G_{ik} represents the satisfaction state for CSC. If the frequency distance between kth and $(k+1)th$ calls or kth and $(k-1)th$ calls in cell i is less than c_{ii}, CSC is not satisfied, then $G_{ik} = 1$. On the other hand, if the frequency distance is larger than the minimum frequency distance, CSC is satisfied, then $G_{ik} = 0$.

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.

Problem No.	Cell #	N_c	ACC	c_{ii}	LB	Comp. Matrix C
1	21	12	1	5	381	C_1
2	21	7	1	5	381	C_2
3	21	12	1	7	533	C_3
4	21	7	1	7	533	C_4
5	21	7	2	7	533	C_5

Table 1: Simulated problem specifications.

Reproduction: After evaluating the strings based on the fitness function, a certain pair of strings should be selected according to the fitness values in order to combine with other selected string for generating offspring. 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. Two crossover techniques are considered: one point crossover (X_1) and two points crossover (X_2).

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. The mutation operator will protect the algorithm from becoming trapped in a local minima. In this paper, we consider the four mutation techniques (M_1, M_2, M_3 and M_4). In M_1, assign the randomly selected frequency channels to the call of each cell according to CSC and the minimum frequency interval with the cell of maximum required channel number. In M_2, shift the assigned frequency channels by random number of frequency blocks. In M_3 and M_4 which are selective mutations, (1) for the already assigned frequency of each cell, compare its frequency with the frequency of other cells, (2) calculate the frequency differences from step (1), (3) if the frequency difference from step (2) is the less than the value of the compatibility matrix and random generated probability is the greater than mutation probability, (M_3) : it is mutated by M_1; (M_4) : it is mutated by M_1 for the first mutation and shifting the assigned frequency channels by random number of frequency blocks thereafter.

The energy function and fitness function, procedures of iteratively creating new strings would generate new populations until termination condition is reached. Once termination 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 iteration or after finding the string S_p which has $E_{S_p} = 0$.

4 Simulation Results and Discussion

In this section, the simulation results of the NNs and the GAs are presented and compared according to the different techniques. The cellular network system for this paper is the 21 cell system. Total number of calls is 481,

Initia-lization	Upda-ting	Problem #1		Problem #2		Problem #3		Problem #4		Problem #5		Ave.(#1-#5)	
		AIN	CR	AIN	CR	AIN	CR	AIN	CR	AIN	CR	AIN	CR
I_1	U_1	67.1	100%	42.3	100%	85.9	98%	33.7	100%	107.5	99%	67.3	99.4%
I_1	U_2	72.9	100%	39.3	100%	64.2	100%	31.0	100%	109.1	98%	63.3	99.6%
I_1	U_3	67.4	99%	29.8	99%	78.1	100%	39.2	100%	110.6	97%	65.0	99.0%
I_2	U_1	85.5	99%	54.4	99%	126.3	100%	71.4	98%	152.3	97%	98.0	98.6%
I_2	U_2	73.1	99%	52.8	100%	151.8	98%	77.5	99%	159.3	98%	102.9	98.8%
I_2	U_3	76.7	99%	53.6	99%	120.5	97%	62.9	97%	137.8	98%	90.3	98.0%
$Rand$	U_1	292.6	82%	423.7	27%	397.2	25%	NA	0%	379.6	36 %	NA	34.0%
Ave.(I_1,I_2)		73.78	99.3%	45.36	99.5%	104.4	98.8%	52.6	99.0%	129.4	99.8%		

Table 2: Comparison of the initialization and updating technique for Neural Networks.

Cross-over	Muta-tion	Problem #1		Problem #2		Problem #3		Problem #4		Problem #5		Ave.(#1-#5)	
		AGN	CR	AGN	CR	AGN	CR	AGN	CR	AGN	CR	AGN	CR
X_1	M_1	25.82	76%	0.25	100%	5.31	100%	0.0	100%	8.02	100%	7.88	95.2%
X_1	M_2	23.13	69%	0.24	100%	4.95	97%	0.0	100%	7.58	98%	7.18	92.8%
X_1	M_3	19.56	89%	0.24	100%	4.72	100%	0.0	100%	6.39	100%	6.18	97.8%
X_1	M_4	24.0	93%	0.25	100%	4.97	100%	0.0	100%	6.95	99%	7.23	98.4%
X_2	M_1	26.69	63%	0.24	100%	6.46	97%	0.0	100%	7.62	94%	8.20	90.8%
X_2	M_2	22.48	60%	0.24	100%	7.26	95%	0.0	100%	8.18	97%	7.63	90.4%
X_2	M_3	22.80	95%	0.24	100%	5.06	100%	0.0	100%	6.54	100%	6.92	99.0%
X_2	M_4	26.29	97%	0.24	100%	5.46	100%	0.0	100%	7.35	100%	7.86	99.4%
Ave.(X_1,X_2)		23.84	80.2%	0.24	100%	5.52	98.6%	0.0	100%	7.32	98.5%		

Table 3: Comparison of the crossover and mutation techniques for Genetic Algorithms.

which is the sum of calls in the total number of cells. Table 1 shows the specification of the problems which are taken from [2,8]. N_c is the number of cells in one cell cluster and ACC implies the presence of ACC on adjacent cells. A "2" or "1" in ACC column implies the presence and absence of ACC respectively. The CCC is indicated by the value in column c_{ii}. LB is the lower bound of required frequency numbers. In Table 2, the simulation results by the NNs approach are shown according to the various initialization and updating methods. The results are then compared with the random initialization method ($Rand$).

The average iteration number (AIN) and the convergence rate (CR) to the solution are shown in the table. The AIN is the average number of iterations which are increased until $E = 0$. The CR is the probability that the experiment has $E = 0$ before the maximum iteration number. In these simulations, the maximum iteration numbers are fixed at 500. In Table 2 the simulation results shows that the proposed initialization methods I_1 and I_2 have a smaller AIN and a higher CR than the usual random initialization method. The Ave.(#1-#5) in Table 2 is the sum total of all the AINs and CRs for the Problems #1-#5. It demonstrates that generally experiment I_1 has a better performance (*i.e.,* smaller AINs and higher CRs) than experiment I_2. When I_1 is used, U_2 has the best performance due to the changing cell list order technique. On the other hand, when I_2 is used, the case of U_2 has the larger AIN than the other two updating techniques. It shows that the performance of NNs are more greatly affected by the initialization method, rather than by the updating technique. In the case of experiment I_1 and U_2, the results have 41 % smaller AIN and 21 % higher CR than the results of the random initialization method. The Ave.(I_1,I_2) in Table 2 is the sum total of the AINs and CRs from iteration method I_1 through I_2 for each problem. It shows that the performance of Problem #2 (with compatibility matrix C_2) and Problem #4 (C_4) has a better performance than Problems #1 (C_1) and #3 (C_3) since $N_c = 7$ for C_2 and C_4, and $N_c = 12$ for C_1 and C_3. With the larger number of cells in one cluster (N_c), the size of the area where the assignment of the same frequencies are prohibited by CCC, is increased. The compatibility matrices C_4 and C_5 are the same except that ACC is applied in C_5. It shows that Problem # 5 (C_5) has the larger AIN than Problem #4 (C_4) due to the addition of ACC to the matrix.

For the GAs, the population size 200 is used and the maximum number of generation is fixed at 100. The algorithm can be terminated prior to $100th$ generation if and only if the algorithm could find the string S_p which has $E_{S_p} = 0$. Table 3 shows the simulation results according to two crossover and mutation techniques for each problem. The Average Generation Number (AGN) is the average number of generations until $E_{S_p} = 0$. The CR is the probability that the experiment has $E_{S_p} = 0$ before the maximum generation number. For the AGN and CR, 100 simulation runs were performed as in NNs. Ave.(#1-#5) in Table 3 is the sum total of all AGNs and CRs. In Problem # 4, the AGN 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. In cases where the selective mutation techniques (M_3 or M_4) have the highest CR within maximum generation numbers, mutation only occurs when the previously assigned frequencies do not fit the compatibility matrix. The Ave.(X_1, X_2) in Table 3 is the sum total of the AGNs and CRs of the various crossover and mutation techniques for each problem. It shows that the problem with the large N_c has the larger AGNs and the smaller CRs, than the problem with the small N_c, when it has the same value of c_{ii} as in the case of NNs. However, when the problems have the same value of N_c, the problem with the larger c_{ii} has a better performance

Initialization	I_1			I_2			$Rand$
Updating	U_1	U_2	U_3	U_1	U_2	U_3	U_1
Problem #1	539.064	586.895	540.328	685.418	587.262	614.877	2345.532
Problem #2	338.450	313.782	238.450	436.922	421.552	428.85	3389.650
Problem #3	1655.000	1235.836	1502.147	2431.199	2922.048	2317.627	7645.753
Problem #4	646.517	595.746	753.317	1372.072	1487.973	1208.736	NA
Problem #5	2166.068	2189.251	2221.230	3056.102	3201.821	2767.487	7629.630
Ave.(#1-#5)	1069.020	1060.294	1051.094	1596.343	1724.131	1467.515	NA

Table 4: Comparison of the cpu time for the initialization and updating techniques in Neural Networks.

Crossover	X_1				X_2			
Mutation	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
Problem #1	1145.022	1030.951	1579.272	1928.480	1182.623	1002.051	1834.184	2108.428
Problem #2	48.239	48.016	52.368	52.768	48.012	48.020	52.384	52.368
Problem #3	269.286	253.915	413.382	433.103	318.389	352.308	440.261	471.700
Problem #4	42.560	42.560	42.560	42.560	42.560	42.560	42.560	42.560
Problem #5	224.380	214.618	303.368	326.342	215.526	228.377	309.601	342.675
Ave.(#1-#5)	345.897	318.012	478.19	556.65	361.422	334.663	535.798	603.546

Table 5: Comparison of the cpu time for the crossover and mutation techniques in Genetic Algorithms.

than the one with the smaller c_{ii}. Problem #1 has the worst performance in Problems #1-#5 since it has the larger N_c and the smaller c_{ii}. These characteristics are not shown in the given problems for NNs approaches. Tables 4 and 5 shows the cpu time measured in *sec* unit for the various techniques in NNs and GAs, respectively. The cpu time for the initialization is 0.05 *sec* and 42.46 *sec* for NNs and GAs respectively. The initialization time of GAs is large since GAs have the large population size 200. It only takes an initialization time in Problem #4 of GAs since all the frequencies are assigned during the initialization period. Ave.(#1-#5) is the average cpu time of all problems with each technique used. Generally, the cpu time of GAs are shorter than the cpu time of NNs, except the case of Problem#1. In the case of Problem#1, GAs have the relatively higher CPU time measurement since its CRs are relatively low and many generations should be run until the predetermined maximum generation number. In Table 5 the cpu times of the two mutation methods M_3 and M_4 are larger than M_1 and M_2 since the difference between the already assigned frequencies and the frequencies of other cells should be computed for the selective mutation.

5 Conclusion

Two optimization methods, neural networks (NNs) and the genetic algorithms (GAs), are applied. The results observed in this paper show that NNs and GAs can be applied to obtain the optimal solution for the channel assignment in mobile cellular environment. The AIN and the CR in NNs and the AGN and the CR of the GAs are shown as simulation results. An expected outcome of the simulation was that the performance of the algorithm varied depending on the initialization and the updating techniques in the NNs and the operators in GAs. However, it is more greatly affected by the initialization method, rather than by the updating method in NNs. For both of the approaches, when the numbers of cells in one cluster are increased, the iteration numbers are increased and the convergence rates are decreased. From our simulation results with the chosen problem examples, it was shown that the convergence rates of NNs are slightly higher than GAs, although the cpu time of GAs is shorter than NNs.

References

[1] D. Kunz, "Channel assignment for cellular radio using neural networks," *IEEE Transactions on Vehicular Technology*, vol. VT-40, pp. 188–193, Feb. 1991.

[2] 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-31, pp. 430–436, Nov. 1992.

[3] 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.

[4] 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*, (Under Review), 1994.

[5] J.-S. Kim, S. H. Park, P. W. Dowd, and N. M. Nasrabadi, "Channel assignment in cellular radio using Genetic Algorithm," *Wireless Personal Communications*, (Under Review), 1995.

[6] 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.

[7] F. Box, "A heuristic technique for assigning frequencies to mobile radio nets," *IEEE Transactions on Vehicular Technology*, vol. VT-27, pp. 57–64, May 1978.

[8] K. N. Sivarajan, R. J. McEliece, and J. W. Ketchum, "Channel assignment in cellular radio," in *Proc. of* 39^{th} *Vehecular Technology Conference*, pp. 846–850, May 1989.