A Hybrid Genetic Algorithm for the Fixed Channel Assignment Problem

Mark Ryan, Justin Debuse, George Smith and Ian Whittley
School of Information Systems, University of East Anglia
Norwich, NR4 7TJ. U.K.
Email: {mdr,jcwd,gds,imw}@sys.uea.ac.uk
Tel: +44 1603 592308

Abstract

This paper describes a hybrid genetic algorithm for solving instances of the Fixed Channel Assignment Problem (FCAP), a problem that is frequently encountered by designers of mobile telecommunication networks. The hybrid GA manipulates solutions which model networks directly, allowing it to provide realistic assignments for highly constrained problems. Unfortunately, such solutions can be very expensive to evaluate. Algorithms such as simulated annealing often speed up the evaluation process by using delta evaluation. Whilst such an approach is not normally adopted by genetic algorithms, this paper demonstrates that delta evaluation can be incorporated into a GA, to deliver dramatic speed increases. We have found that delta evaluation can improve the speed of our GA by a factor of 90. This improved performance allows the GA to produce good results for large and complicated networks in a reasonable amount of time. The results obtained by the GA are compared to previous GA algorithms proposed for the FCAP and to a highly tuned simulated annealing algorithm.

1 THE FIXED CHANNEL ASSIGNMENT PROBLEM

Solving the Fixed Channel Assignment Problem (FCAP) is one of the major obstacles which must be overcome by designers of mobile telecommunication networks. Mobile telecommunication networks are composed of a number of cells. For the problems described in this paper, each cell contains a single base station. Clients of the network rely on the base station of the cell in which they are situated, to provide them with a channel through which they can make a call. Each cell must be allocated a sufficiently large number of channels, known as the cell's demand, to cope with the level of network traffic it typically experiences. These channels are then delegated by the cell's base station to its clients' calls. In solving the FCAP, network designers must allocate channels to all the cells in the network as efficiently as possible so that the demand of each cell is satisfied and the number of violations of electro-magnetic compatibility constraints (EMCs) in the network is minimised. This paper considers three types of electro-magnetic constraints:

Co-Channel Constraints (CCCs) Certain pairs of cells may not be assigned the same channel simultaneously.

Adjacent Channel Constraints (ACCs) Adjacent cells cannot be simultaneously assigned channels which are adjacent in the electro-magnetic spectrum.

Co-Site Constraints (CSCs) Channels assigned to the same cell must be separated by a minimum frequency distance.

Solving instances of the FCAP is not a trivial exercise. If we consider a simple network that exhibits just one of the constraints described above, the CCCs, then the problem is identical to Graph Colouring. The Graph Colouring problem is known to be NP-Complete [1] and consequently it is very unlikely that a polynomial time algorithm exists that can solve all instances of the FCAP. Previously, researchers have applied a variety of heuristic techniques to the FCAP, including Neural Networks, [2, 3], Genetic Algorithms, [4, 5, 6, 7, 8], Simulated Annealing [9, 10], Local Search [11] and various greedy and iterative algorithms [12, 13, 14]. Typically, modern heuristic search techniques, such as simulated annealing and genetic algorithms, tend to adopt one of two strategies.

- A direct approach which uses solutions that model the network directly, i.e. they contain information about which channels are assigned to which cells.
- An indirect approach whose solutions do not model the network directly. Typically the solutions represents a list of all the calls required to satisfy the demand of the network. Algorithms such as those described by Sivarajan [12] are used to transform the indirect solutions into real network models which can be used to evaluate the quality of the proposed solutions.

This paper describes a GA which adopts a direct approach to solving the FCAP. Previous experiments with both approaches have demonstrated that the direct approach generally yields better results. See [15] for details.

2 A HYBRID GENETIC ALGORITHM FOR THE FCAP

Standard GAs, using a direct representation, have been found to perform quite poorly on the FCAP. A few genetic algorithms employing a direct representation appear in the FCAP literature. Papers by Cuppini [8] and Lai and Coghill [7] attempt to solve only reasonably trivial FCAP problems. Ngo & Li [4] successfully apply their GA to more difficult problems but they report run times of over 24 hours for a single run of some of the simple problem instances. In addition they also employ a local search algorithm which fires when the GA gets stuck in local optima. In short, the literature does not provide much evidence that an efficient and scalable channel assignment system could be based on a standard GA.

2.1 THE HYBRID GA

Designing a genetic algorithm for the FCAP using a direct approach that will execute in a reasonable amount of time is very difficult. The main obstacle to efficient optimisation of assignments using a traditional genetic algorithm is the expense of evaluating a solution. Complete evaluation of a solution to the FCAP can be extremely time consuming. See Section 2.1.5 for details. Neighbourhood search algorithms such as simulated annealing can typically bypass this obstacle using delta evaluations. Each new solution created by the neighbourhood search algorithm differs only slightly from its predecessor. Typically the contents of only a single cell are altered. By examining the effects these changes have on the assignment, the fitness of the new solution can be computed by modifying the fitness of its predecessor to reflect these changes. A complete evaluation of the assignment is avoided and huge gains in execution times are possible. Unfortunately such delta evaluations are difficult to incorporate in the GA paradigm. At each generation a certain proportion of the solutions in a population are subject to crossover. Crossover is a binary operator which combines the genes of two parents in some manner to produce one or more children. The products of a crossover operator can often be quite different from their parents. For example consider the genetic fix crossover operator employed by Ngo & Li [4]. So long as the two parents are quite different from each other, their children are also likely to be quite dissimilar from both parents. Consequently it is generally impractical to use delta evaluations to compute the fitnesses of offspring from their parents. After a child has been produced by crossover it must be completely re-evaluated to determine its fitness.

In the light of this shortcoming, a GA using a simple crossover operator, such as the one employed by Ngo & Li [4], which requires a large amount of time to evaluate a single solution and which does not guide the population towards a speedy convergence, will be comprehensively outperformed by a local search algorithm, such as simulated annealing, which uses delta evaluations. In order to be competitive with local search techniques a GA must utilise operators that allow it either to converge very quickly so few evaluations are required or to explore the search space efficiently using delta evaluations. Section 2.1.2 describes a greedy crossover operator which uses delta evaluations to explore a large number of solutions, cheaply, in an attempt to find the best way to combine two given parents.

2.1.1 Representation

The solution representation employed by the Hybrid GA is based on the basic representation used by Ngo and Li [4]. Each solution is represented as a bit-string. The bit-string is composed of a number of equal sized segments. Each segment represents the channels which are assigned to a particular cell. The size of each segment is equal to the total number of channels available. If a bit is switched on in a cell's segment, then the channel represented by the offset of the bit from the start of the segment is allocated to the cell. Each segment is required to have a specific number of bits set at any one time which is equal to the cell's demand. Genetic operators must not violate this constraint. The length of a solution is equal to the product of the number of cells in the network and the number of channels available.

2.1.2 Crossover

A good crossover operator for the FCAP must create good offspring from its parents quickly. Producing good quality solutions will drive the GA towards convergence in a reasonable number of generations, thus minimising the amount of time the GA will spend evaluating and duplicating solutions. Mutation can be relied on to maintain diversity in the population and prevent the GA from converging too quickly. A research group at the University of Limburg [15] devised such a crossover operator for the Radio Link Frequency Assignment Problem (RFLAP), which proved to be very successful. In essence, they used a local search algorithm to search for the best uniform crossover that could be performed on two parents to produce one good quality child. Once found, the best crossover was performed and the resulting child took its place in the next generation. Whilst the FCAP and the RFLAP problems are significantly different to prevent this crossover operator being employed in the FCAP, this research does illustrate how a similar sort of operator may be applied to achieve good results for the FCAP.

The crossover operator employed here uses a greedy algorithm to attempt to find the best combination of genes from two parents to produce one good quality child. The greedy algorithm is seeded with an initial solution consisting of two individual solutions to the FCAP.

The greedy algorithm works by maintaining two solutions. It attempts to optimize only the solution with the best fitness. It achieves this by swapping genetic information between the two solutions. Information can only be swapped between corresponding cells in each of the solutions. When a swap is performed two channels, one from each solution, are selected. The channels are then removed from the solution from which they were originally selected and replaced by the channel chosen from the other solution.

The manner in which these swaps are performed is defined by a neighbourhood. The greedy algorithm explores a neighbourhood until it finds an improving swap. When such a swap is found both solutions are modified and the neighbourhood is updated. The greedy algorithm continues to explore the remainder of the neighbourhood searching for more improving moves. The process continues until the entire neighbourhood is explored. At this juncture the solution being optimized is returned as an only child.

The neighbourhoods are constructed in the following fashion. Each solution is essentially a sequence of sets, one for each cell in the network. Each set contains a certain number of channels which are assigned to the cell corresponding to this set. Two new sequences of sets are created by performing set subtractions on each of the sets in both parents. These new sequences of sets, referred to as the channel lists, again contain a set for each cell. Each set in channel list 1 contains channels which have been assigned to the cell, represented by this set, in the first parent but not to the corresponding cell in the second parent and vice versa for channel list 2. The neighbourhood is then constructed from these lists in the following manner: (See Figure 1)

for each cell c
    for each channel i in cell c in channel list 1
        for each channel j in cell c in channel list 2
            Generate move which swaps channels i
                and j in cell c

Since the parents and the channel lists are represented as bit-strings the set subtractions can be efficiently performed as a sequence of ANDs and XORs.

The order in which the greedy algorithm explores the moves in the neighbourhood is important. Experimentation has shown that the moves are best explored in a random fashion. Consequently if the crossover operator is applied to the same parents more than once there is no guarantee that the resulting children will be identical.

The process of neighbourhood construction is depicted in Figure 1. Figure 1(a) illustrates the two parents. These are real solutions to problem 1 as described in Section 3. This toy problem has only four cells which have demands of 1, 1, 1 and 3 respectively. The cell segments are denoted by the numbers appearing above the solutions. The solutions are not depicted in bit-string form for reasons of clarity. Performing the set subtraction operations described above yields two channel lists which are displayed in Figure 1(b). Finally Figure 1(c) depicts all the moves which are generated from the channel lists. These moves define the neighbourhood.

Computing the channel lists is a very important part of the crossover operation. It guarantees that each move in the neighbourhood will alter the two solutions, maintained by the crossover operator, in some

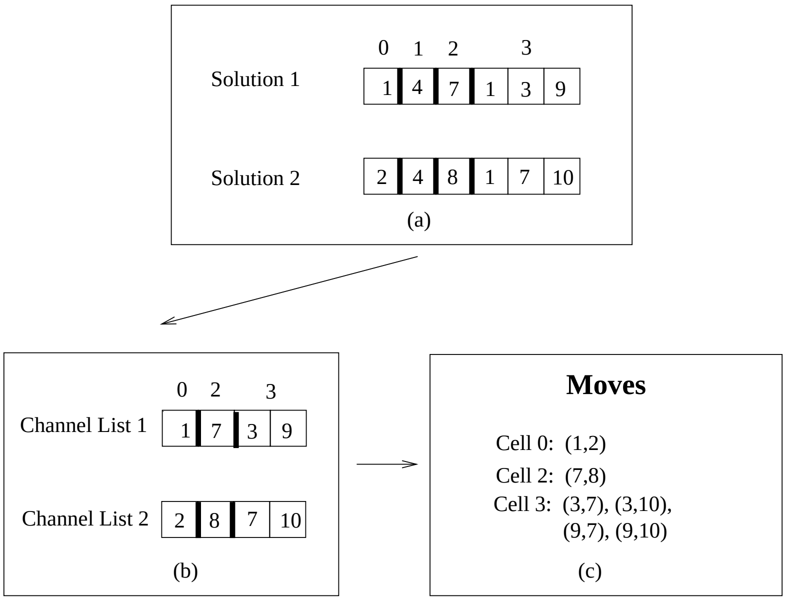

Figure 1: Neighbourhood Construction

way. Hence moves that will not effect the solution we are trying to optimise will not be generated and consequently we will waste no time evaluating the solutions they produce. Interestingly, this aspect of the crossover operator does have an advantageous side effect. As the size of the neighbourhood depends on the size of the channel lists, the number of solutions evaluated by a crossover operator depends on the similarity between the parents upon which it was invoked. As the population of the GA begins to converge the crossover operators performs less work and the GA speeds up.

There is one huge advantage of using the neighbourhood described above. Each new solution explored differs only slightly from its predecessor. Consequently it is entirely practical for the greedy algorithm to employ delta evaluations allowing it to search its neighbourhoods incredibly quickly. So rather than performing one slow evaluation on two solutions as a normal crossover operator would do, it performs quick evaluations on many solutions. The GA can now search the solution space cheaply in the fashion of a local search algorithm.

The effect that delta evaluation has on our hybrid GA is dramatic. Some experiments were performed on the first problem set, described in Section 3, to assess the impact of delta evaluation on the genetic search. The results of these experiments demonstrated that the GA runs about 90 times faster when using delta evaluations. This result illustrates the most important feature of the hybrid GA presented in this paper. Its ability to explore the search space very efficiently allows the GA to produce effective assignments for large and complicated networks in a reasonable amount of time.

Finally, relatively low crossover rates of 0.2 and 0.3 have been found to work well with this crossover operator. Due to the greedy nature of the operator, higher crossover rates cause the GA to converge prematurely.

2.1.3 Mutation

The nature of the crossover operator described above tends to cause the GA's population to converge very quickly. Mutation plays an essential role in the hybrid GA by maintaining sufficient diversity in the population, allowing the GA to escape from local optima. Mutation iterates through every bit in a solution and modifies it with a certain probability. If a bit is to be modified, the associated cell is determined. A random bit is then chosen in the same cell which has an opposite value to the original bit. These bits are then swapped and the process continues. The mutation operator cannot simply flip bits because this would violate the demand constraints of the cell. A maximimum of 100 mutations is permitted per bit-string. Without this limit, mutation would cause the GA to execute very slowly on some of the larger problems.

2.1.4 The GA

The hybrid GA is loosely based on Goldberg's simple GA [16]. Each generation the individuals in the population are ranked by their fitnesses. Solutions are selected for further processing using a roulette wheel. A certain proportion of solutions for the next generation will be created by the crossover operator described above. The remaining slots in the next generation are filled by reproduction. Mutation is only performed on solutions produced by reproduction. Allowing mutation an opportunity to mutate the products of crossover was found to have a negative impact on the genetic search. Due to the highly epistatic nature of our representation even a single mutation can have a very detrimental impact on the fitnesses of the solutions produced, after considerable effort, by the crossover operator. If mutation were applied to all solutions, an excellent assignment produced by crossover could be corrupted completely before it found its way into the next generation.

2.1.5 Fitness

The fitness of a solution to the FCAP is determined by the *number* of EMCs that it violates. It does not include any information as to whether the network demand is satisfied because this constraint is enforced by the representation. More precisely the fitness may be defined as follows:

$$F(S) = \sum_{i=0}^{n-1} \sum_{p=0}^{m-1} \left(\sum_{\substack{j=0 \\ j \neq i \\ C_{ij} > 0}}^{n-1} \sum_{\substack{q = p-(C_{ij}-1) \\ 0 \leq q < m}}^{p+(C_{ij}-1)} S_{jq} \right) S_{ip} \quad (1)$$

$$+ \sum_{i=0}^{n-1} \sum_{p=0}^{m-1} \left(\sum_{\substack{q = p+1 \\ 0 \leq q < m}}^{p+(C_{ii}-1)} S_{iq} \right) S_{ip} \quad (2)$$

where S is a solution, S_{ij} is 1 if channel j has been assigned to cell i, otherwise it is 0. C_{ij} is the minimum separation between a channel assigned to cell i and a channel assigned to cell j. The letters n and m represent the number of cells and the number of available channels in the network, respectively.

The first part of the fitness equation is responsible for computing ACC and CCC violations. The second part calculates CSC interference.

2.2 HEURISTIC ENHANCEMENTS

A number of problem specific enhancements can be made to the basic genetic operators, described in the previous section, to improve the performance of the GA on the FCAP problem instances. These enhancements are described below.

2.2.1 Ignore Good Channels

An important enchancement can be made to the crossover operator in an attempt to improve its efficiency. During its execution, the crossover operator constructs a neighbourhood which defines the work that it is to perform. However, this neighbourhood is going to be used by a greedy algorithm which will only perform an improving move. Since we are just optimizing the first solution, we do not need to bother considering channels which are assigned to it without violation. Replacing these channels cannot possibly lead to an improvement in the solution as they were not responsible for any interference in the first place. Thus we can omit these channels from the list of channels that can be swapped out of the first solution. Determining which channels in the first channel list are involved in violations is actually quite expensive and involves a partial evaluation of the first solution. However if it can prevent the crossover operator performing more than a few swaps, some performance gains might be made. It should be noted that, even though delta evaluation is used to recompute the value of solutions after a swap has occurred, the process is still quite slow.

2.2.2 Eliminating CSCs

Ngo & Li [4] demonstrate that it is possible to eliminate CSCs completely from the search process for some problems. They achieved this by modifying their representation so that CSCs could not be violated. The hybrid GA attempts to employ a similar heuristic without altering the representation. It constructs an initial population which does not contain any CSC violations. This can be achieved by ensuring that for each solution all the channels assigned to a single cell are separated from each other by the CSC frequency separation for that cell. The GA then ensures that neither the crossover nor the mutation operator can perform a swap that can violate a CSC. This heuristic allows us to effectively reduce the size of the search space for certain problems.

3 PROBLEM INSTANCES

The performance of the hybrid GA has been evaluated on two problem sets. The first set has been compiled from the FCAP literature and contains 11 problems of varying difficulty. The second problem set consists of four benchmark problems presented at the NCM_2 Workshop on Optimization methods for Wireless Networks [17]. These problems represent some very large real world networks. More information about the problem instances is presented in Table 1. Problems 1 to 11 constitute problem set 1, problems 12 to 15 form the second problem set. In Table 1, *Cells* indicates the number of cells in a network, *Ch* represents the total number of channels available and *D* is the total demand of a network. The column entitled, *Source*, provides references to various papers in the FCAP literature where these problems have previously appeared.

Table 1: Problem Instances

Problem	*Source*	*Cells*	*Ch*	*D*
1	[2, 4, 12]	4	11	6
2	[2, 3, 7, 4, 11]	25	73	167
3	[2, 4, 12, 11]	21	221	470
4	[2, 4, 12, 11]	21	309	470
5	[2, 12, 11]	21	533	481
6	[2, 12, 11]	21	533	481
7	[2, 12, 11]	21	381	481
8	[2, 12, 11]	21	309	470
9	[12, 11]	21	414	481
10	[12, 11]	21	258	470
11	[12, 11]	21	529	470
12	[17]	100	373	677
13	[17]	170	373	1231
14	[17]	214	373	1221
15	[17]	718	373	5101

4 EXPERIMENTATION

The results of a number of experiments conducted to assess the performance of the hybrid GA are presented in Table 3. Table 3 also places the GA's performance in perspective by presenting the results obtained by a highly tuned simulated annealing algorithm, which is discussed briefly below. The outcome of these experiments are analysed in Sections 4.2 and 4.3. Ten separate runs of both the GA and the SA were performed on each problem[1]. Table 3 presents the best and the average fitnesses obtained by each algorithm. The average time taken to converge, in CPU seconds, is also reported. All experiments were performed on a 433Mhz DEC Alpha. Unless otherwise stated in the sub sections below, the parameters detailed in Table 2 were used to guide the genetic search.

[1]Due to time constraints, only one run was performed for the GA on problem 15

Table 2: GA Parameter Settings

Parameter	*Value*
Population Size	50
Crossover Rate	0.2
Mutation Rate	0.01
Selection Method	Roulette Wheel

4.1 SIMULATED ANNEALING

Simulated annealing (SA) is a modern heuristic search method which is often applied to combinatorial optimisation problems, such as the FCAP. The reader is referred to [18] for a description of the SA algorithm. The SA uses the same representation employed by the GA. It uses a neighbourhood operator which randomly selects a channel that has been allocated to a cell and attempts to replace it with the best unused channel that is not currently assigned to the cell in question. Delta evaluation is used in the SA.

4.2 PROBLEM SET 1

The results obtained by the hybrid GA for the first problem set are very encouraging. It can produce interference free solutions to nine out of the eleven problems after only a couple of minutes. These run times are admirable, especially when compared to previous GAs that adopted a direct method for solving the FCAP. For instance, Ngo & Li [4] report average run times of $20,000$ and $90,000$ seconds for problems 3 and 4 respectively. Problems 3 to 8 and problem 11 are described as being CSC limited, i.e. it is the presence of CSCs constraints in the network that make these problems difficult. The GA finds them easy because it does not permit CSC violations in its solutions. The SA does not employ such an approach and consequently it has difficulties with these problems. On the other hand problems 9 and 10 are made difficult by the presence of ACC and CCC constraints. The SA outperforms the GA on these problems.

4.3 PROBLEM SET 2

Problem set 2 contains some very large problems. An interesting feature of these problems is that the average demand per cell is much lower than it is for the first problem set. Since it is the demand of the network that determines the number of bits set in a solution, these problems require a larger population size to ensure that the GA has enough information available to perform its search effectively. Interestingly, whilst you might suspect that the low demand of these networks

Table 3: Results

	GA			*SA*		
Problem	*Best Fit*	*Avg Fit*	*Time*	*Best Fit*	*Avg Fit*	*Time*
1	0	0	0	0	0	0
2	0	0	13	0	0	1
3	0	0	22	1	1.4	176
4	0	0	84	1	1.2	415
5	0	0	1	1	1.5	202
6	0	0	25	4	4.7	324
7	0	0	4	3	3.6	134
8	0	0	6	1	1	71
9	16	17.6	401	9	10	553
10	3	5.4	631	0	0.3	442
11	0	0	19	1	1	77
12	0	0	474	0	0	14
13	0	0	2155	0	0	216
14	57	59.6	10564	25	26.9	14610
15	12	n/a	36011	0	0	1439

would make these problems easy to solve, it actually makes them very difficult for the GA. We are forced to increase our population size to provide sufficient diversity. Unfortunately doing so has a rather negative effect on the speed of the GA. We are forced to maintain relatively large populations which contain very large solutions. Population sizes of 150 solutions are required to obtain reasonable results for this problem set. With these population sizes the GA is capable of solving the first and second problems. Although the GA is outperformed by the SA on the remaining two problems, insufficient time was available to tune the GA to work well with this problem set. We suspect that a more suitable parameter configuration will yield better results for these problems. Problem set two has been included in this paper to demonstrate that the GA can tackle large networks.

5 CONCLUSION

This paper describes a hybrid GA which we believe has made significant advances in the field of channel assignment through genetic search. The most distinguished quality of the hybrid GA is its ability to explore the search space of FCAP problems very efficiently. Consequently, the GA can be applied to large and complicated networks, producing good results in a reasonable amount of time. However, whilst the GA does compare very favourably with previous GA solutions to the FCAP problem, it is not as efficient as a well tuned SA algorithm on really large problems. Further research is required if the GA is to compete on a par with the SA. Possible enhancements to the GA might include a distributed version or the introduction of a pyramid architecture as described in [15].

Acknowledgements

This research was sponsored by Nortel, Harlow, UK.

References

[1] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* Freeman, New York, 1979.

[2] N. Funabiki and Y. Takefuji. A neural network parallel algorithm for channel assignment in cellular radio networks. *IEEE Transactions on Vehicular Technology*, 41(4), 1992.

[3] D. Kunz. Channel assignment for cellular radio using neural networks. *IEEE Transactions on Vehicular Technology*, Vol. 40, No. 1:188–193, 1991.

[4] C. Y. Ngo and V. O. K. Li. Fixed channel assignment in cellular radio networks using a modified genetic algorithm. *IEEE Transactions on Vehicular Technology*, 47(1):163–172, 1998.

[5] C. Crisan and H. Mühlenbein. The breeder genetic algorithm for frequency assignment. In Eiben et al. [19], pages 897–906.

[6] C. Valenzuela, S. Hurley, and D. Smith. A permutation based genetic algorithm for minimum span

frequency assignment. In Eiben et al. [19], pages 907–916.

[7] W. K. Lai and G. Coghill. Channel assignment through evolutionary optimization. *IEEE Transactions on Vehicular Technology*, 45(1):91–95, 1996.

[8] M. Cuppini. A genetic algorithm for channel assignment problems. *Communication Network*, 5(2):157–166, 1994.

[9] D. Kunz M. Duque-Antón and B. Rüber. Channel assignment for cellular radio using simulated annealing. *IEEE Transactions on Vehicular Technology*, 42(1):14–21, 1993.

[10] T. Clark and G. D. Smith. A practical frequency planning technqiue for cellular radio. In G. D. Smith, N. C. Steele, and R. F. Albrecht, editors, *Aritifical Neural Networks and Genetic Algorithms*. Springer, 1997.

[11] W. Wang and C. K. Rushforth. An adaptive local-search algorithm for the channel assignment problem. *IEEE Transactions on Vehicular Technology*, 45(3):459–466, 1996.

[12] K. N. Sivarajan, R. J. McEliece, and J. W. Ketchum. Channel assignment in cellular radio. *Proc. of the 39th IEEE Vehicular Technology Conf.*, pages 846–850, 1989.

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

[14] A. Gamst and W. Rave. On frequency assignment in mobile automatic telephone systems. *Globecom '82, IEEE Global Telecommunications Conference. Miami. Paper B.3.1.1-7*, pages 309–315, 1982.

[15] G. D. Smith, A. Kapsalis, V. J Rayward-Smith, and A. Kolen. Report 2.1: Implementation and testing of genetic algorithm approaches. Technical report, Euclid CALMA radio link frequency assignment project, 1995.

[16] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Compnay, Inc., 1989.

[17] Pascal Labit. Frequency assignment test problems. `http://www.crt.umontreal.ca/~brigitt/telecom/test_probs/prob1.html`, 1998.

[18] K.A. Dowsland. Simulated annealing. In *Modern Heuristic Techniques*, chapter 2, pages 20–69. Blackwell Scientific, 1993.

[19] A. E. Eiben, T. Bäck, M. Schoenauer, and H.P. Schwefel, editors. *Parallel Problem Solving from Nature - PPSN V*. Springer, 1998.