Optimization of Fixed Network Design in Cellular Systems using Local Search Algorithms

Bhaskar Krishnamachari and Stephen B. Wicker
School of Electrical and Computer Engineering
Wireless Multimedia Laboratory
Cornell University, Ithaca, NY 14853, USA
{bhaskar, wicker}@ee.cornell.edu

Abstract

Search techniques such as Genetic Algorithms, Simulated Annealing, Tabu Search and Random Walk Algorithms have been used extensively for global optimization. This paper presents an experimental analysis of the performance of these algorithms for the problem of designing the fixed portion of a cellular network. We first investigate the effect of various algorithm specific parameters on their performance. The algorithms are then compared to each other using criteria that ensure fairness. Under the given problem formulation and assumptions regarding the location of nodes in the network, we find that Tabu Search and Genetic Algorithms provide good, robust solutions.

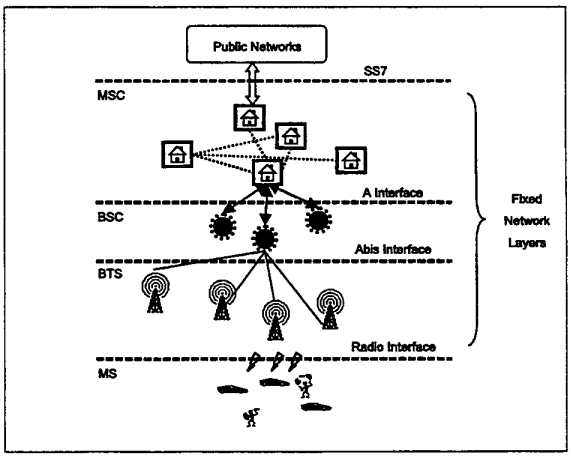

Figure 1. GSM Network Architecture

1. Introduction

In cellular networks, the mobile units are connected to the public communication networks via a series of interconnections determined by the system architecture. Figure 1 shows the architecture for the European GSM standard. The mobile stations (MS) in each cell communicate over the radio interface with the Base Transceiver Stations (BTS). Some low-level functions such as mobile handoffs are made by the Base Station Controllers (BSC), each of which may control several hundred BTS. The BSC in turn are connected to and controlled by the Mobile Switching Centers (MSC). Thus there is a large part of the communication hierarchy in a mobile cellular system which takes the form of a fixed wired network. The cost of the topology of this fixed network depends upon several factors such as the cost of the nodes and links, and the maximum number of links allowed per node. It is desirable to design the network topology in such a way as to minimize this cost.

This problem is closely related to the NP-hard *facility location problem* in Operations Research where the goal is to place facilities on a 2-D plane in such a way as to minimize the average distance between them and existing client locations [1]. Local Search Algorithms such as Genetic Algorithms, Simulated Annealing, Tabu Search and Random Walk have been applied extensively for such difficult optimization problems [2], [3], [6]. In fact, both Genetic Algorithms and Simulated Annealing have been utilized previously to find low-cost topologies [4], [5] for GSM and DCS1800 systems. However, an experimental comparison of these algorithms and the other search techniques has not yet been performed for this problem.

2. Problem Description and Parameters

The network to be minimized consists of N_{BTS} base transmitting stations which are connected to N_{BSC} base station controllers in such a way that no BTS is connected to more than one BSC, and no more than MAX_BTS of

0-7803-6507-0/00/$10.00 ©2000 IEEE

them may be connected to each BSC. It is further assumed that all BSC are connected to exactly 1 MSC. The locations of all BTS and the MSC is pre-specified. The object of the optimization process is to find the best location for each BSC and determine which BTS should be connected to it. It is assumed that α, the ratio of the cost per unit length of the higher bandwidth links between each BSC and MSC to the cost per unit length of the links between the BSC and BTS is some constant ≥ 1.

The cost function to be minimized is:

$$f = \sum_{i=1}^{N_{BTS}} l_{BTS_i \rightarrow BSC(i)} + \alpha \sum_{j=1}^{N_{BSC}} l_{BSC_j \rightarrow MSC} \quad (1)$$

where $l_{BTS_i \rightarrow BSC(i)}$ is the length of the direct link between the i^{th} BTS and the BSC it is assigned to, and $l_{BSC_j \rightarrow MSC}$ is the length of the direct link between the j^{th} BSC and the MSC. For the purpose of investigating the performance of the various search algorithms on this optimization problem, we choose an instance with $N_{BSC} = 3, N_{BTS} = 50, MAX\_BTS = 20, \alpha = 5$. All nodes are assumed to be located in an area that is of size 10x10 square units. The locations of the BTS are generated randomly using a uniform distribution over the service area. A typical starting point for the search algorithms is shown in figure 2. The rectangle represents the MSC, the BSCs are represented by filled circles, and the BTS are represented by the small squares.

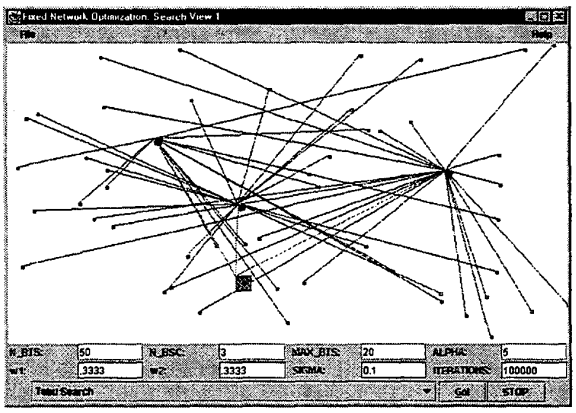

Figure 2. A Typical Initial Point

Figure 3 shows the histogram of costs obtained by taking 10^6 samples from a search space with $N_{BSC} = 3, N_{BTS} = 50, MAX\_BTS = 20$, and $\alpha = 5$. It is interesting to note that for this problem the distribution of costs is quite smooth and appears bell-shaped. The probability of locating low cost solutions falls off quite sharply.

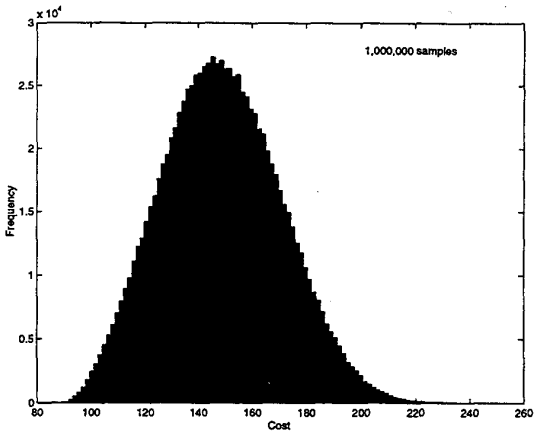

Figure 3. Histogram of Costs

3. Neighborhood Definition

Each point in the search space consists of i) a particular set of locations for each BSC, and ii) a particular assignment of BTSs to each BSC satisfying the MAX_BTS constraint. To generate a new neighbor from this point two neighborhood generating operators are required - one that moves the locations of the BSCs and another that changes the BTS assignments for each BSC.

To move the BSCs, two independent Gaussian random variables, N_x and N_y, are generated with zero mean and standard deviation σ. At each step, one BSC is randomly picked and if it is located at the co-ordinates (X_j, Y_j), in the generated neighbor this BSC is displaced by (N_x, N_y) and has co-ordinates

$$(X_j', Y_j') = (X_j + N_x, Y_j + N_y) \quad (2)$$

To ensure that the generated BSC location is within the boundaries of the service area, the following hold:

$$\begin{aligned} X_j + N_x > 10 &\Rightarrow X_j' = 10 \\ X_j + N_x < 0 &\Rightarrow X_j' = 0 \\ Y_j + N_y > 10 &\Rightarrow Y_j' = 10 \\ Y_j + N_y < 0 &\Rightarrow Y_j' = 0 \end{aligned}$$

To modify the assignments of the various BTS to each BSC, three kinds of moves are possible:

- BTS move A: Do nothing. This "null" move is important to have because it allows the search to optimize the location of the BSCs without changing BTS assignments.
- BTS move B: Pick two BSCs, BSC_1 and BSC_2, and if the number of BTSs allocated to BSC_2 is less than

MAX_BTS, pick one BTS assigned to BSC_1 at random and assign it to BSC_2 instead.

- BTS move C: Pick two BSCs, exchange one BTS assigned to each (picked randomly with uniform probability) with the other.

Weights may be assigned to each of these kinds of moves. This is accomplished by using two probabilistic parameters ω_1, ω_2 which are chosen such that

$$0 \leq \omega_1 + \omega_2 \leq 1 \tag{3}$$

At each step, move A is carried out with probability ω_1, move B with probability ω_2 and move C with probability $1 - (\omega_1 + \omega_2)$.

Thus the parameters that affect the definition of the neighborhood for the various search algorithms are σ - the standard deviation of the BSC displacements, and ω_1, ω_2, which control the changes in BTS assignments to each BSC. For the experiments described in this paper, these parameters have the following values: $\sigma = 0.1, \omega_1 = \omega_2 = \frac{1}{3}$.

4. Effect of Algorithm Parameters on Performance

As mentioned in the introduction, local search algorithms have been used extensively for difficult optimization problems such as this mixed continuous-discrete problem. We investigate four local search algorithms in this paper: Random Walk, Simulated Annealing, Tabu Search and Genetic Algorithms. Random Walk (RW) is the simplest of these: make local moves in the search space, such that uphill moves are accepted with a fixed probability p throughout the search, while downhill moves are accepted unconditionally. Simulated Annealing (SA) is an extension of this idea - a temperature T is decreased as the search proceeds, in such a way as to decrease the probability of accepting uphill moves. In Tabu Search (TS), a number of neighboring points in the search space are considered as possible moves. There is also a list of tabu moves that is maintained dynamically; these moves may not be considered unless some *aspiration criterion* is satisfied. In genetic algorithms (GA), we consider a population of individuals at each generation, each corresponding to a single point in the search space. A fitness selection process determines a subset of these individuals, which then mutate give rise to the next generation. The mutation simply consists of generating a neighboring point. We do not utilize the crossover operator in this paper. For a more detailed description of these algorithms and their implementation, see [7].

Before the search algorithms can be compared to each other, it is necessary to determine the best parameter values for each algorithm independently via initial experimentation. These experiments are described below. In all algorithms, we use the neighborhood scheme described in section 3 to generate local moves. This ensures fairness of comparison.

4.1. Performance of Random Walk

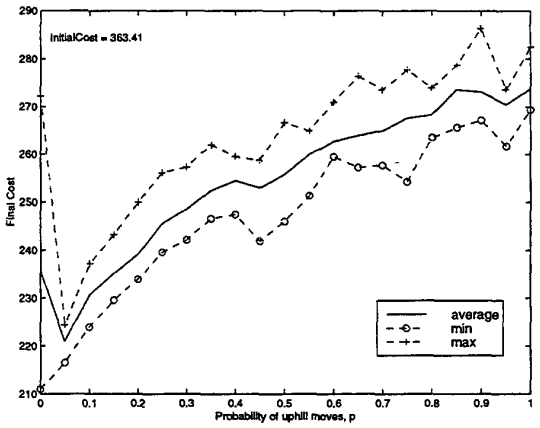

Figure 4. Performance of Random Walk

We first investigate the performance of the RW algorithms for varying values of p, the probability of accepting uphill moves. The RW algorithms thus range from being a purely greedy search ($p = 0$) to a purely random one ($p = 1$). The algorithms are run 10 times for 100,000 function evaluations for each value of p ranging from 0 to 1 in increments of 0.05. The results are shown in figure 4. It is observed that the variance of final solutions obtained is greatest with greedy search; however, the average final cost is worse than that observed with a low uphill move acceptance probability of 0.05. A small non-zero value of p dislodges the search from some local minima. As p is increased further, the search performance deteriorates in a monotonic fashion, with the worst performance observed when $p = 1$ (purely random movement). This is fairly typical of search spaces with multiple local optima, since in the absence of such local optima, a greedy search would provide the best performance.

4.2. Performance of Simulated Annealing

Some of the parameters to be determined for SA are the initial temperature, the temperature cooling schedule used, and the number of moves at each temperature. If I is total number of iterations to run the algorithm and n the number of discrete temperature levels in each run, the number of

moves at each temperature level L is set as

$$L = \frac{I}{n} \tag{4}$$

The number of temperature values in the experiments is chosen to be $n = 20$; hence if $I = 100,000$ then $L = 5000$). The initial temperature T_o needs to be high enough. It is determined by first obtaining L cost values via a purely random search (equivalent to RW with $p = 1$) and finding the standard deviation σ_∞ of costs at this temperature and setting $T_o = 10\sigma_\infty$. The geometric cooling schedule is employed:

$$T_{new} = \alpha_c T_{old} \tag{5}$$

where α_c is the cooling rate parameter.

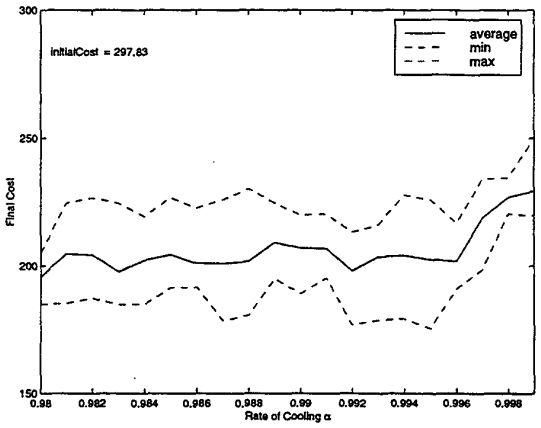

Figure 5. Performance of Simulated Annealing with Geometric Cooling Schedule

To the investigate the performance of the geometric cooling schedule, SA with different values of α_c (which must be between 0 and 1) are each tested for 10 runs and 100,000 iterations. Figure 5 shows the final costs obtained for α_c in the range [0.980, 1.000]. It was observed that SA performance i the same for all values of α_c from 0 to about 0.996, after which it is seen to deteriorate a little as it is increased to 1.0, when the cooling is too slow. Further, a visual examination of the lowest cost solutions obtained by SA (not shown here) indicated that they were far from optimal. This behavior of SA suggests that the search space may be quite rough and full of local minima. This means that during the initial stages of the annealing process, when the temperature is high, the algorithm is unable to locate good regions of the space to settle down to. If the cooling rate is fast enough to allow the SA to settle down to a greedy search, the net performance of the algorithms depends highly on the greedy portion of its search, which, because of the existence of multiple local minima fails to find good solutions. Further experiments using exponential and logarithmic schedules confirmed that the performance of SA is not significantly affected by the choice of cooling schedule [7].

4.3. Performance of Tabu Search

As there are essentially two kinds of moves that generate neighboring points (moving the locations of BSCs and changing the BTSs allocated to BSCs), two different tabu lists were implemented for the Tabu Search algorithm, both with the same tabu tenure value K. These tabu lists are defined as follows:

- BSC Location Tabu: When the j^{th} BSC is moved to a location (X_j, Y_j) in iteration n, movement of the j^{th} BSC to any point within a radius of 0.1σ of (X_j, Y_j) before iteration $(n + K)$ is tabu.
- BTS Allocation Tabu: When the i^{th} BTS is taken from the j^{th} BSC and allocated to some other BSC in iteration n, the re-assignment of the i^{th} BTS to the j^{th} BSC is tabu until iteration $(n + K)$.

The simple aspiration criteria , whereby a tabu move is accepted if it leads to the lowest cost function evaluation to date, is utilized in conjunction with these tabu definitions.

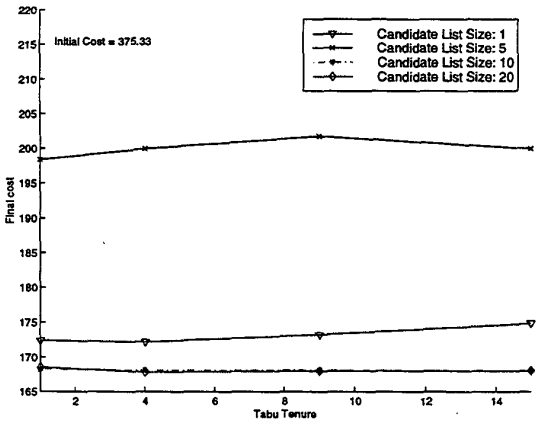

Figure 6. Performance of Tabu Search

The effect of the value of the tabu tenure K and the size of the candidate list v on the performance of Tabu Search for this problem is investigated. The first experiment analyzes the average final cost over 10 runs of 100,000 iterations each obtained by running Tabu Search algorithms with candidate list sizes v = [1, 5, 10, 20] for values of the tabu tenure K = [1, 4, 9, 15]. The results are shown in figure 6 and are somewhat intriguing. First of all, it appears that the

performance for the various values of K is nearly identical suggesting that in this case the tenure value of 1 may be sufficient to prevent cycling. Further, while the performance for candidate list size $v = 1$ is fairly good, the performance for $v = 5$ is worse, and the final costs obtained with $v = 10$ and $v = 20$ are nearly identical and better than that obtained with $v = 1$, suggesting that a candidate list size of $v = 10$ is good. Since the algorithm appears fairly independent of the value of K, the variation of the final costs with v was further explored by an experiment where K is fixed at a value of 4. Figure 7 shows the average, minimum and maximum cost values obtained over 10 runs of 100,000 iterations of Tabu Search for v values from 1 to 10. The average results obtained with the greedy search algorithm (RW with $p = 0$) are also plotted for comparison.

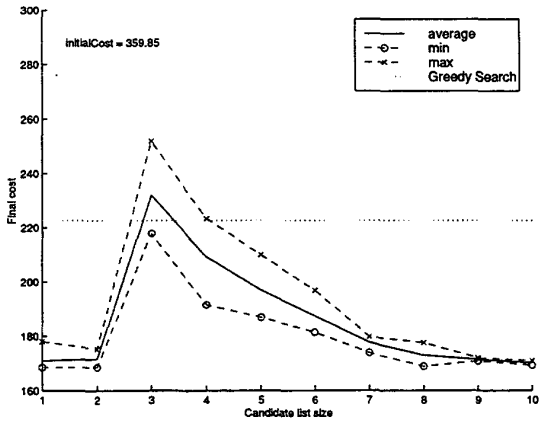

Figure 7. Performance of Tabu Search (2)

It appears from figure 7 that the performance of Tabu Search is fairly good for candidate list sizes of 1 and 2. Tabu Search performs quite poorly for $v = 3$ (even worse than greedy search on average), and thereafter improves in a monotonic fashion as v is increased to 10. It also appears that the variance in the final solution obtained is very small for $v = 10$. This is certainly odd behaviour. How can it be explained? v is effectively the branching factor at each step, indicating how the neighborhood of each point is sampled before making a move. One possible explanation for the observation is that on average 1 in 3 neighbors of each point in the space are misleading in that they that lead the search into local minima. When only one point is sampled, the search will accept that point unconditionally if it is not in the tabu list. However, when at each step about 3 neighbors are sampled, chances are that picking the lowest of those points leads to a trap (local minimum). As the candidate list size increases beyond 3, more of the neighborhood is being sampled and the better exploration of the neighborhood allows the search to escape such traps by finding relatively

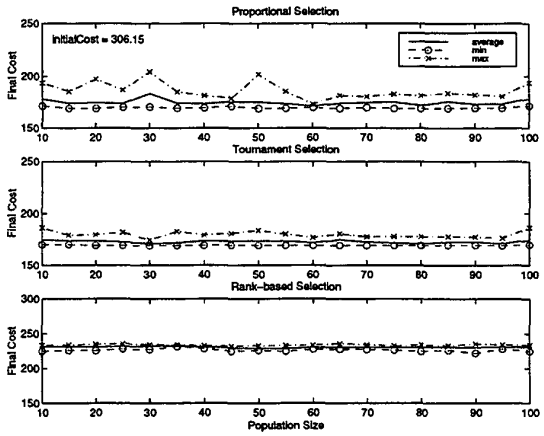

Figure 8. Performance of Genetic Algorithms

rarer lowest points in the neighborhood (perhaps about 1 in 10) that do not lead to local minima.

4.4. Performance of Genetic Algorithms

The influence of population size and selection technique on the performance of Genetic Algorithms is investigated next. Three selection techniques - proportional selection, tournament selection, and rank-based selection - are compared for this problem with the same parameters. The three sub-plots Figure 8 show the performance of each technique for population sizes ranging from 10 to 100 in increments of 5. For each selection technique and population size parameter value 10 independent runs consisting of 100,000 function evaluations were carried out. It is clear that the performance of the Genetic Algorithms is quite robust and independent of the size of the population for this problem. The performance of the proportional and binary tournament selection schemes is about the same with the latter performing a little better. The rank-based selection technique fares rather poorly for this problem.

From an inspection of the final results produced by Genetic Algorithms with tournament selection, it appears that they are very likely the globally optimal solutions or very close to it. One support for this speculation is the fact that the minimum cost function solution returned by the GA with tournament selection for each population value is exactly the same.

5. Comparison of Algorithms

Once the best parameters for each algorithm have been determined, we can compare them with each other. Greedy Search (RW with $p = 0$), Random Walk ($p = 0.05$), Simulated Annealing (with geometric cooling schedule, $\alpha_c =$

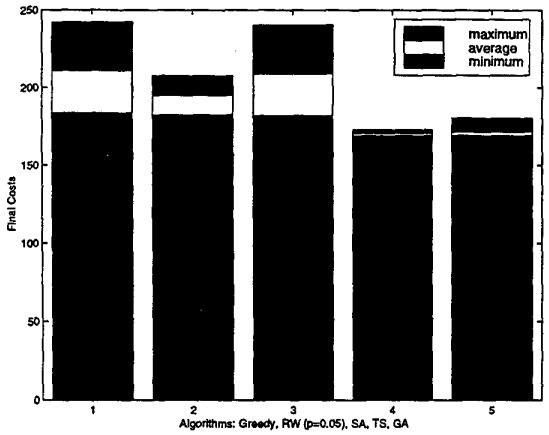

Figure 9. Comparison of Search Algorithms ($N_{BSC} = 3, N_{BTS} = 50, MAX\_BTS = 20$)

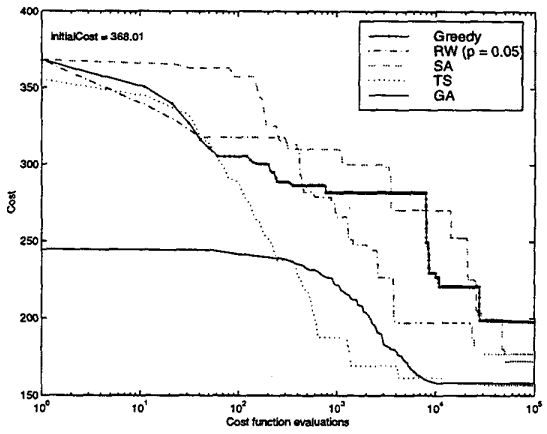

Figure 10. Comparison of Search Algorithms – a sample run ($N_{BSC} = 3, N_{BTS} = 50, MAX\_BTS = 20$)

0.99), Tabu Search (with $K = 4$, $v = 10$), and Genetic Algorithm (with a population size of 50, and elitist binary Tournament Selection) are compared with each other for their performance on the problem. We use the same number of cost function evaluations for each algorithm. Figure 9 shows the average, minimum and maximum final cost obtained with these algorithms in 100 runs, each consisting of 100,000 cost function evaluations.

It is observed that Tabu Search and Genetic Algorithms provide good final solutions for this problem. Random Walk with a small probability of uphill moves performs better than Greedy Search, while the performance of Simulated Annealing lies somewhere between the two. None of these last three algorithms, however, appear to be able to provide good quality solutions. This is probably due to the search space being really rough and full of local minima. Tabu Search with its aggressive, non-cycling, exploration of the space and Genetic Algorithms with their inherent parallelism both perform better for such a problem. Figure 10 shows the best cost seen to date versus number of cost function evaluations for a sample run of each algorithm for this problem. The Genetic Algorithms start off with a low initial cost because one of the 50 points chosen initially at random happened to have a low cost.

Figure 11 shows a sample final solution obtained in one of the runs by Genetic Algorithms after 100,000 cost function evaluations. It appears that GA has found a good, possibly near-optimal, solution which consists of BSCs that are i) spaced apart from each other, and ii) in the midst of clusters of base stations that are connected to them.

6. Conclusions

We have presented an experimental comparison of various local search algorithms to the problem of minimizing the cost of the fixed network in a cellular network. The results suggest that Genetic algorithms and Tabu Search are both well-suited for this problem which appears to contain multiple local minima. This may be attributed to the fact that they both investigate multiple points in the search space at each step. Other experiments conducted by us, that were not presented in this paper, suggest that genetic algorithms are particularly scalable to problems involving larger networks.

One observation we must make is that the problem formulation is a little simplified here. We have assumed, for example, that costs of the links are linear functions of their lengths. In real situations, there may be a piecewise-linear relationship between the cost of wiring and the length of the links. Another assumption is that the links between point to point are straight lines – this is rather unlikely in any realistic setting since the geography of the service region, the layout of the roads etc. plays an important role in determining the exact wiring of the links. Despite these shortcomings, we believe that the general results presented here regarding the performance of the algorithms will hold under a more realistic scenario as well. This is a topic for future research.

References

[1] P.B. Mirchandani and R.L. Francis, eds. *Discrete Location Theory*, John Wiley and Sons, 1988.

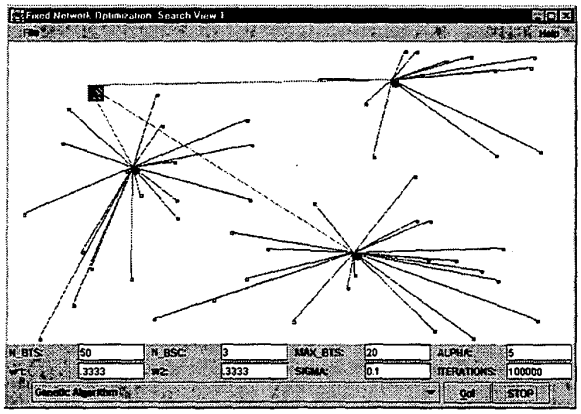

Figure 11. Sample Final Solution obtained using GA

[2] E. Aarts and J. K. Lenstra, *Local search in Combinatorial Optimization*, John Wiley and Sons, 1997.

[3] B. Selman, H.A. Kautz, and B. Cohen, "Noise Strategies for Improving Local Search," *Proceedings of Twelfth National Conference on Artificial Intelligence, AAAI-94*, vol.1, pp. 337-43, July 1994.

[4] M. Shahbaz, "Fixed network design of cellular mobile communication networks using Genetic Algorithms," *4th IEEE International Conference on Universal Personal Communications*, pp. 163-7, 1995.

[5] M. Shahbaz, "Random Guided Algorithms for Fixed Network Design of Cellular Mobile Communication Networks," *Teletraffic Contributions for the Information Age. 15th International Teletraffic Congress*, vol. 1, pp. 255-64, June 1997.

[6] B. Krishnamachari and S.B. Wicker, "Global search techniques for problems in mobile communications," *Telecommunications Optimization: Adaptive and Heuristic Methods*, ed. D. Corne et al., John Wiley and Sons, (expected) 2000.

[7] B. Krishnamachari, *Global Optimization in the Design of Mobile Communication Systems*, Masters Thesis, Electrical Engineering, Cornell University, 1999.