# The Application of a Genetic Algorithm to Trunk Network Routing Table Optimisation

Mark C. Sinclair[1]

## 1. Introduction

Traditionally, research into circuit-switched (C-S) networks has been divided into two areas, reliability and performance, each making little reference to the other. More recently, however, research has focused on unification of the two. One possible choice for a combined performance and reliability measure is the overall network grade-of-service (NGOS), i.e. the overall probability of blocking, averaged over all node pairs, taking into account both the offered traffic between node pairs and the link (trunk group) availabilities.

In a 1991 paper [1], this author gave a method for the overall NGOS, by single-moment analysis, of fixed-alternative-routing (FAR) circuit-switched communications networks with reliable nodes but unreliable links. The method employs a Markov model for an unreliable link, in combination with earlier work [2-4], so as to greatly reduce execution time compared to prior algorithms.

In two subsequent papers [5,6], this author used the analysis method as part of three heuristic routing table optimisation algorithms, which obtain routing tables such that the overall NGOS is minimised under different call control rules. These algorithms were then used to demonstrate the improvements in trunk network fault-tolerance (assessed in terms of the overall NGOS) that can be achieved both through routing table updating in the event of link failure [5], and through network augmentation i.e. the addition of extra links to a network, either in parallel with existing links, or between nodes that previously had no direct connection [6].

In the current paper, an alternative approach to routing table optimisation is taken, based on combining a genetic algorithm with Sinclair's analysis method. "Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest [amongst a population of] string structures with a structured yet randomised information exchange [between population members] to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. [7]"

The paper begins by describing the network model used and, briefly, both the underlying analysis method and the earlier heuristic optimisation methods. The basic principles of genetic algorithms are presented, and the details of the particular genetic algorithm used here are described. The paper concludes with a comparison of the execution times and results obtained for an example network with those from Sinclair's heuristic method [5,6], and some suggestions for possible future work.

## 2. Network Description

The unreliable trunk network model, used as a basis for analysis and optimisation in this and earlier papers [1,5,6], consists of the following six elements:

1. Routing Table
2. Call Control Rule
3. Network Configuration Table
4. Link Capacity & Availability Table
5. Busy-Hour Traffic Matrix
6. Loading

The Routing Table (e.g. Table 2 in Section 9 below) consists of an n x n matrix of entries, where n is the number of nodes in the network, except that the forward diagonal is empty. Entries are ordered sets of node numbers, $(N_1,N_2,\ldots,N_m)$, such that the entry at position (S,T) represents links $(S\text{-}N_1,S\text{-}N_2,\ldots,S\text{-}N_m)$. These are searched in the given order to find a free circuit for a call arriving at S destined for T.

[1] British Telecom Lecturer in Communications Network Modelling , Department of Electronic Systems Engineering, University of Essex

The Call Control Rule describes how the routing table will be used to find paths through the network. Four such routing disciplines can be analysed using the methods of [1]: Successive Office Control (SOC), Originating Office Control (OOC), Originating Office Control with Spill-Forward (OOC-SF) [2], and Tandem Node Matrix routing (TNM) [3]. In this paper, however, we will only be considering OOC. Using this call control rule, if a call is at its originating node, a free circuit will be searched for, trying the first-choice, and then the other links in order. If a free circuit is found, then the call is passed to the next node over that link, otherwise the call is lost. At a tandem (intermediate) node, however, only the first-choice link will be tried. If this link is unavailable, the call will not be lost, but will drop back to the originating node, where the next choice link will be tried. The call is only lost if the last choice link (and all previous choices of course!) from the originating node is unavailable. For a more detailed discussion, and examples, see Lin *et al.* [2].

The Network Configuration Table (e.g. Table 3) is an n x n array of link numbers, with an empty forward diagonal, the (S,T) entry being either the link number of the link connecting these two nodes, or -1 if there is no direct link.

The Link Capacity & Availability Table (e.g. Table 1) contains the capacities (i.e. number of circuits) and availabilities of the b links connecting the network.

The Busy-Hour Traffic Matrix (e.g. Table 4) is an n x n array, with an empty forward diagonal, of the busy-hour traffic in Erlangs between each node pair (S,T).

The Loading is used to obtain different traffic levels. Each entry in the Busy-Hour Traffic Matrix is multiplied by the loading, and thus, for example, with a loading of 1.1, the traffic is increased by 10% over the nominal level. Unless otherwise stated, the loading is always set to 1.

## 3. Analysis of a Network Using a Markov Model for Unreliable Links

A method for the single-moment overall NGOS analysis of FAR C-S communications networks with reliable nodes but unreliable links was described in [1]. It was developed using techniques given by Lin *et al.* [2] and Chan [3], in combination with a recursive algorithm for path-loss sequences and a Markov model for unreliable links both developed by this author. The method uses the iterative approach to network analysis first developed by Katz & Cooper [8], and it relies on the standard eight assumptions discussed by Lin *et al.*, as well as the assumption that all the nodes in the network are perfectly reliable. For details of the method, and a comparison with a method based on earlier work, see [1].

## 4. Heuristic Routing Table Optimisation

Using Sinclair's method for the necessary network analysis, three heuristic routing table optimisation algorithms were described in [5] for C-S networks with unreliable links employing OOC, SOC and OOC-SF routing. The three methods do not carry out an exhaustive search of all possible loop-free routing tables, and therefore cannot provide the optimum routing table for the chosen call control rule. They were devised, however, to provide the best possible routing table within the assumptions made. The optimisation criterion used by all three methods is the minimisation of overall NGOS. For details of the three routing table optimisation algorithms, and their application to a larger example network, see [5].

## 5. Genetic Algorithm Based Routing Table Optimisation

One possible alternative to the heuristic approach to routing table optimisation, is to use genetic algorithms. These are general search methods, based on a simplification of the mechanisms of natural selection and genetics [7].

### 5.1. Basics of Genetic Algorithms

Instead of searching directly through the desired problem space for an optimum (in this case, a routing table of minimum overall NGOS), genetic algorithms decouple the problem into genotypes, analogous to the chromosomes inherited by an individual, and phenotypes, analogous to the organism that develops from those chromosomes. The only link between them in the algorithm is that any given genotype can be decoded into a phenotype and its raw fitness assessed; this is analogous to the organism's ability to survive and reproduce in the environment. In this case, the genotypes are strings of integers, and the phenotypes are the routing tables decoded from them, with their raw fitness assessed in terms of their overall NGOS.

The genetic algorithm searches by starting with a randomly generated population of genotypes, each of which has its raw fitness assessed, and then randomly selecting pairs of genotypes, with their probabilities of selection determined by their fitness. Each such pair is mated to produce two child genotypes, which are modified from their parents using two mechanisms: crossover and mutation. Rather than being direct copies of their parents, in any given mating, with a given probability, there will be an exchange of genetic material between the genotypes passed down by the parents to the two children. In this case, one of the children will consist of a copy of the start of one parent genotype (with the break at a random point), and the end of the other, and vice versa for the other child: crossover has occurred. The other mechanism is mutation: with a given probability any of the integers in the child genotypes may be altered to another valid value. The selection and mating of pairs of genotypes from the original population continues until a new population of identical size has been generated; the old population is then discarded, and replaced by the new one. The process continues until a given stopping condition, such as a maximum number of such generations, is reached. In addition, the genotype with the highest fitness found during the run will be recorded.

The mechanisms of selection and crossover ensure that short highly-fit substrings are searched out much faster than might be supposed. This is as a result of implicit parallelism: all the substrings at different positions within the genotypes are competing simultaneously. Mutation ensures that new substrings, not present in the original population, are introduced into the competition as the algorithm proceeds.

One modification to the basic algorithm described is fitness scaling. Rather than using the raw fitness directly, which can lead to rapid domination of the entire population by one 'super-fit' genotype, the raw fitness is scaled appropriately. The simplest approach to fitness scaling is linear scaling, where the average population raw fitness remains constant, but the fitnesses themselves are scaled linearly such that the genotype with the largest raw fitness has a scaled fitness equal to a given factor (say 2.0) times the average fitness. Some care must be taken, however, to avoid introducing negative fitness values [7].

For more details of both basic and more advanced genetic algorithms, see [7].

### 5.2. Routing Table Encoding

For the example network of Section 9, genotypes of total length 64 were used. For each of the valid node pairs, i.e. (0,1), (0,2), (0,3), (0,4), (1,0), (1,2), ..., an entry of d integers, equal to the degree of the originating node, was used. The range of integers permitted in each position, within an entry, was from 0 up to a maximum of d-1 for the first and second, between 0 and d-2 for the third, down to between 0 and 1 for the last. For example, node 0, of degree 4, would have a four integer entry to encode the routing for node pair (0,1), with the first and second positions in the range 0 to 3, the third in the range 0 to 2 and the fourth in the range 0 to 1.

When decoding a genotype, for any given entry, the numbers of the d nodes directly connected to the originating node would be placed in numeric order. An integer x in the first position in the genotype entry being decoded would determine the first-choice route as being via the $(x+1)^{th}$ node in the list. For example, for node pair (1,3), the directly connected node numbers are: 0,2,3,4. An entry of 2121 would mean that the first choice route was the 3rd node i.e. 3. For the second and subsequent positions, a 0 is taken to indicate no further choices (routes), and an integer x (greater than 0) to indicate the $x^{th}$ remaining node. The full routing for (1,3) would thus be decoded as (3,0,4,2). If the entry had instead been 2101, then this would have been decoded as (3,0); any integers beyond a zero in the second or subsequent positions are ignored.

Having decoded the genotype into a routing table, Sinclair's analysis method [1] was used to determine the overall NGOS. Raw fitness values are maximised by the genetic algorithm, and must always be positive. Overall NGOS is (being a probability) always in the range 0 to 1, and hence positive. However, minimisation of overall NGOS is the objective of the optimisation, so the raw fitness of the phenotype was taken to be 1.0 - overall NGOS.

### 5.3. Penalty Function for Looped Routing Tables

The example network, for OOC routing, has some $1.85\times10^{26}$ valid routing tables. However, the vast majority of these are not loop free: when routes are generated from the table, several of these would, if they were not eliminated, traverse the same node more than once! Clearly, the random nature of the initial genotype population will result in many, if not all, the initial routing tables containing a large number of such loops. As a result of the automatic removal of these routes in the course of analysis [1], the fitness of looped routing tables is comparatively low, but the differential is insufficient to ensure the rapid elimination of looped tables from the population. Consequently, a penalty function [7] must be introduced to reduce the raw fitness returned by the phenotype in the event that the routing table is looped. Several penalty functions were examined, and the best and worst are illustrated in Figures 1 and 2 below. These show the maximum, in a given generation, and mean raw fitness (plain and dashed lines, respectively) of the population over 50 generations.

The penalty function used in Figure 1 was a step function: the raw fitness was divided by 2 if the routing table was looped. In Figure 2, the raw fitness was divided by the number of loops detected (plus 1); this was the penalty function adopted for the remainder of the study.

**Figure 1** Raw Fitness vs. Generation with a Step Penalty Function

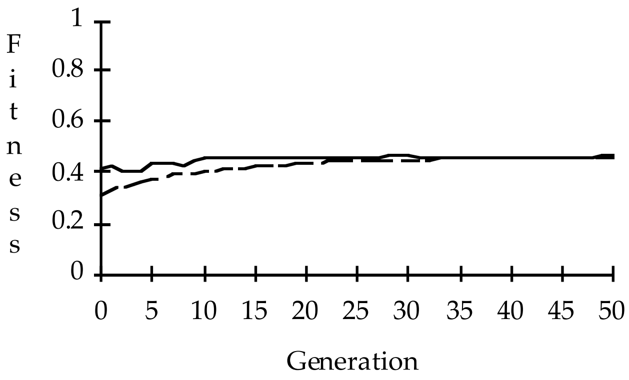
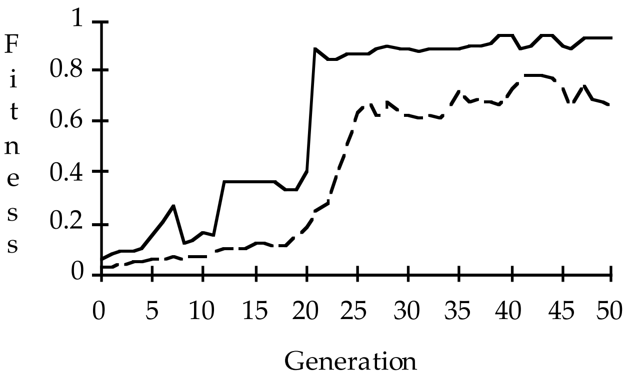



**Figure 2** Raw Fitness vs. Generation with an Offset Inverse Penalty Function




In Figure 1, both the mean population fitness and the maximum fitness in each generation improved only slightly, and a loop-free routing table was not found, over the 50 generations. In Figure 2, however, despite the much lower initial fitness values (as a result of the higher penalties, given the highly-looped nature of the initial population), both the maximum and mean fitness improved rapidly. By generation 21, the population contained at least one loop-free table, and thereafter further improvements were made in both maximum and mean fitness, although this process was limited by the continuing re-emergence of looped routing tables, as a result of both crossover and mutation.

5.4. Parameter Choices

Based on earlier work by De Jong, Goldberg [7] recommends a high choice of crossover probability (0.6), a low mutation probability (inversely proportional to the population size, 0.0333), a moderate population size (30), and a fitness scaling factor between 1.2 and 2.0. Initial trials were carried out with a population of 50 (given the large size of the problem space), a crossover probability of 0.6, a mutation probability of 0.02, a scaling factor of 2.0 and run lengths of 50 generations. However, after several trial runs with variations on these initial parameter choices, and graphical examination of the results, the mutation probability was reduced to 0.015.

More extensive trials were then carried out, and the results are given in Figures 3, 4 and 5 below. These show the best (i.e. minimum) NGOS obtained for various values of crossover probability (Fig. 3), mutation probability (Fig. 4) and scaling factor (Fig. 5). Each of the points on the graph represents the mean of values obtained over ten to twenty different initial seed values for the random number generator, with the 90% confidence intervals shown.

**Figure 3** Best Overall NGOS vs. Crossover Probability

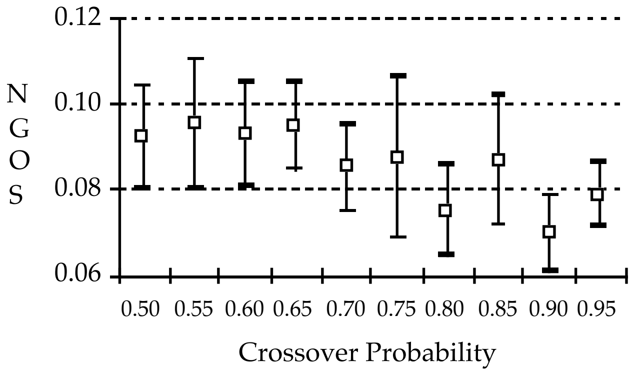



**Figure 4** Best Overall NGOS vs. Mutation Probability

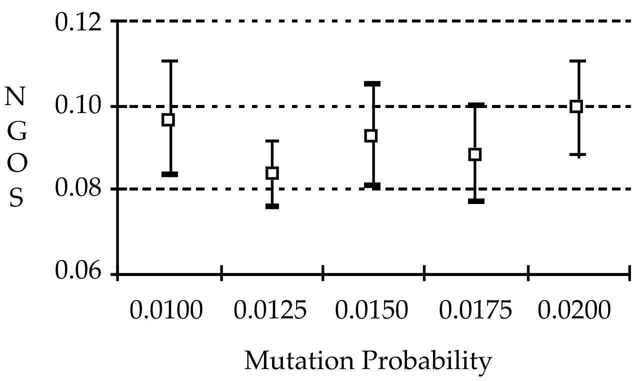
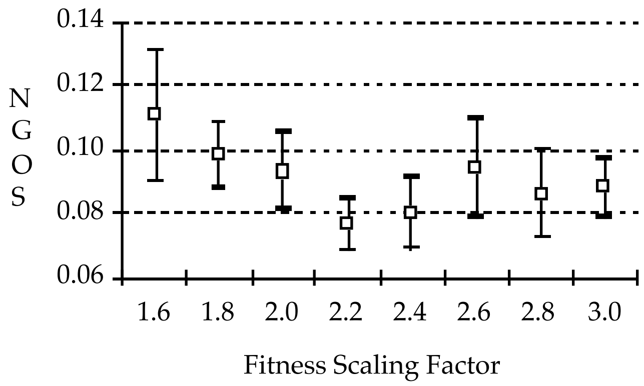


**Figure 5** Best Overall NGOS vs. Fitness Scaling Factor




The optimum combination of parameters for the example network would appear to be a crossover probability of 0.9, a mutation probability of 0.0125, and a scaling factor of 2.2.

6. Results and Execution Times

The best result obtained from twenty different initial seed values, over runs of 50 generations, with the parameters chosen in Section 5.4, was on overall NGOS of 0.0532. However, each such run required some 36 minutes to execute on a Sun SPARCstation-ELC. A single run of 1000 generations was also carried out with the same parameters and, after some 11.6 hours, the best overall NGOS obtained was 0.0489.

In contrast, Sinclair's heuristic OOC routing table optimisation algorithm [5] obtained an overall NGOS of 0.0243 with an execution time of just 23 seconds.

7. Conclusions and Further Work

Despite the promise of genetic algorithms, before the commencement of the study, to be highly suitable for this type of optimisation problem, they have yet to fulfil that promise. Sinclair's heuristic optimisation algorithm, at least for the example network examined, would appear to be far superior. Nevertheless, there are still some possibilities unexplored.

With regard to the genetic algorithm itself, three areas of further work occur to this author. Firstly, the penalty function could be modified to include a penalty for routing tables that result in one or more high individual node-to-node grades-of-service; such a penalty might serve to enhance the overall convergence. Secondly, other routing table encodings could be explored, possibly leading to improved algorithm performance. Finally, some of the more advanced genetic algorithms reviewed by Goldberg [7] could be tried.

An additional possibility, departing from the 'pure' use of genetic algorithms, would be to use some hybrid of heuristics and a genetic algorithm: the initial population could be generated by using heuristics combined with random selection, and/or the final population, after a given number of generations, could be further optimised by a heuristic approach.

One final thought is that, even if one, or a combination, of these approaches can provide reasonably rapid optimisation of the example network considered here, the scaling of algorithm performance with network size must also be examined.

8. References

1 Sinclair, M.C. **"Single-Moment Analysis of Unreliable Trunk Networks: Two Methods for the Network Grade-of-Service"** Proc 8th UKTS, Beeston, Apr 91, p8/1-8/11

2 Lin, P.M., Leon, B.J. & Stewart, C.R. **"Analysis of Circuit-Switched Networks Employing Originating-Office Control with Spill-Forward"** IEEE Trans Commun vCOM-26 n6 Jun 78 p754-765

3 Chan, W.S. **"Recursive Algorithms for Computing End-to-End Blocking in a Network with Arbitrary Routing Plan"** IEEE Trans Commun vCOM-28 n2 Feb 80 p153-164

4 Bonaventura, V., Cacopardi, S., Decina, M. & Roveri, A. **"Service Availability of Communication Networks"** Proc 1980 Nat Telecommun Conf, Houston, Dec 80, p15.2.1-15.2.6

5 Sinclair, M.C. **"Trunk Network Fault-Tolerance through Routing Table Updating in the Event of Trunk Failure"** Proc 9th UKTS, University of Surrey, Guildford, Apr 92, p17/1-17/9

6 Sinclair, M.C. **"Trunk Network Fault-Tolerance through Network Augmentation"** Proc 4th Bangor Communications Symposium, University of Wales, Bangor, May 92, p252-256

7 Goldberg, D.E. **"Genetic Algorithms in Search, Optimization, and Machine Learning"** Addison-Wesley, 1989

8 Katz, S. **"Statistical Performance Analysis of a Switched Communications Network"** Proc 5th Int Teletraffic Cong, New York, Jun 67, p566-575

## 9. Appendix: The Example Network

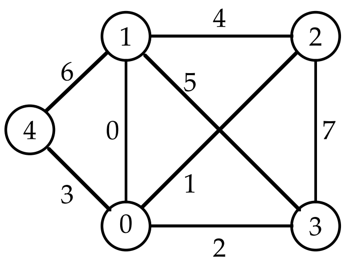
**Table 1** Link Capacity & Availability Table

| Link Number | Capacity | Avail. |
|---|---|---|
| 0 | 23 | 0.98 |
| 1 | 10 | 0.98 |
| 2 | 12 | 0.98 |
| 3 | 6 | 0.98 |
| 4 | 31 | 0.98 |
| 5 | 9 | 0.98 |
| 6 | 5 | 0.98 |
| 7 | 22 | 0.98 |

**Table 2** Routing Table

| From \ To | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | | (1,4,2,3) | (2,1,3,4) | (3,2,1,4) | (4,1,2) |
| 1 | (0,2,4,3) | | (2,3,0) | (3,2,0) | (4,0) |
| 2 | (0,1,3) | (1,3,0) | | (3,1,0) | (1,0) |
| 3 | (0,2,1) | (1,2,0) | (2,1,0) | | (2,0,1) |
| 4 | (0,1) | (1,0) | (1,0) | (1,0) | |

**Figure 6**




A Small Circuit-Switched Communications Network

Links marked with link number

**Table 3** Network Configuration Table

| From \ To | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | | 0 | 1 | 2 | 3 |
| 1 | 0 | | 4 | 5 | 6 |
| 2 | 1 | 4 | | 7 | -1 |
| 3 | 2 | 5 | 7 | | -1 |
| 4 | 3 | 6 | -1 | -1 | |

**Table 4** Busy-Hour Traffic Matrix

| From \ To | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | | 5.56 | 4.17 | 6.94 | 1.39 |
| 1 | 10.42 | | 9.72 | 5.56 | 1.11 |
| 2 | 3.47 | 8.33 | | 9.03 | 0.56 |
| 3 | 5.56 | 1.67 | 1.67 | | 0.56 |
| 4 | 1.11 | 0.83 | 0.28 | 0.56 | |