Design of Routing Tables for a Survivable Military Communications Network using Genetic Algorithms

Squadron Leader S. J. Moore
Defence Communications Services Agency
Basil Hill Barracks,
Park Lane, Corsham, Wilts SN13 9NR, UK

Mark C. Sinclair
Dept. of Electronic Systems Engineering,
University of Essex,
Wivenhoe Park, Colchester, Essex CO4 3SQ, UK
mcs@essex.ac.uk

Abstract- One of the vital areas in the design and operation of a survivable military telecommunications network is the selection of its routing tables. In this paper, both a bit-string genetic algorithm (GA) and an iterative stochastic hill climber (ISHC) are applied to two problem scenarios: with and without existing routing tables. Experimental results are reported for one destination node in an 18-node model network. Comparisons are made both between the GA and ISHC approaches, and with the pre-existing routing table. Overall, both the GA and the ISHC provide substantial improvements over the existing routing table, for both problem scenarios. However, the ISHC consistently obtains better results than the GA, often at reduced computational cost.

1 Introduction

This paper reports an investigation into the use of both a bit-string genetic algorithm (GA) and an iterative stochastic hill climber (ISHC) to optimise circuit-switched routing, in a military telecommunications network, for survivability. While GAs, and evolutionary algorithms in general, have been used in many areas of telecommunication routing before, the particular focus here is on providing survivability for a military network.

2 EC for telecommunications routing

Over the years, more than 30 papers have been published on the application of evolutionary computation (EC) to telecommunications routing. Four of the earliest papers are by Cox *et al.* [1, 2], Pan & Wang [3] and Sinclair [4]. In 1991, Cox *et al.* [1] used a permutation-based GA for the bandwidth-packing component of an heuristic for dynamic-anticipatory routing in circuit-switched networks. Two years later, almost the same group of authors [2] tackled both minimum-cost link capacity assignment and routing in a network subject to a reliability constraint. They employed a hybrid GA, with three-part chromosomes representing both link capacities and routing permutations, a chromosome-repair mechanism, a stochastic Lamarckian replacement technique and problem-specific operators. Pan & Wang [3], also in 1991, used a GA for minimum-delay bandwidth allocation, *i.e.* link capacity assignment and routing, in an ATM network. Their GA encodes individuals as a string of indices into a combined look-up table of both link capacities and routes. Then Sinclair [4], in 1993, applied a bit-string based encoding of static-routing tables in an unreliable circuit-switched network, employing a penalty function to eliminate infinite-routing loops, and aiming for minimum network grade-of-service.

More recent work of interest includes that of Bentall *et al.* [5], who used a GA with a two-dimensional order-based representation to obtain restoration paths for a heavily-loaded network. Their aim was not so much to apply the GA to real-time restoration, but rather for the results obtained to serve as a benchmark in assessing rapid heuristic algorithms with only local knowledge. Shami *et al.* [6], applied genetic programming to obtain both primary and restoration paths in a network subject to single-link failures. Munetomo *et al.* [7], described a real-time dynamic-routing algorithm that employs path-based genetic operators to create alternative routes in a packet-switched network, thereby balancing link loads and reducing delay. Sinclair [8] developed a hybrid GA for routing, fibre choice and wavelength allocation in multi-wavelength all-optical transport networks, using both an object-oriented representation and problem-specific operators. Finally, Zhu *et al.* [9] employed a permutation-based GA on the point-to-multipoint routing problem, *i.e.* routing a set of circuit-switched connection requests, each of which has a single source and one or more destinations.

3 Military & commercial networks

The continued operation of a communications network is an important design issue. Military networks are concerned with survivability, *i.e.* the ability of a network to perform to a specification after it is has been damaged, whereas in commercial networks the emphasis is on reliability to protect revenues. Military networks must be survivable to avoid possible loss of life, or damage to national security, as a result of lost command and control facilities. However, the solutions to the need for reliability and survivability are often similar in both types of network. The differences arise in the types of network failure, and the probabilities of their occurring.

A commercial network is more likely to suffer from link failures, as opposed to node failures, because nodes are reliable, have high levels of redundancy, and are designed to be maintained whilst on-line and carrying traffic. A military network is likely to suffer network damage, due to an attack by an aggressor. The symptoms of such an attack would be similar to those for simple network failures. The difference is in the scale. Failures can generally be assumed to be independent, and due in the main to simple component malfunction. Network damage is more likely to manifest itself as replacement of many nodes with smoking piles of debris, as an aggressor would attack in a manner planned to cause the most disruption, with the aim of completely preventing network operation. Therefore, the effects of the attack would be highly correlated and widespread.

The measures that should be taken to provide the required level of survivability for a military network are in addition to those measures taken to provide reliability in a commercial network, and may well require considerable additional cost. The level of redundancy introduced in a commercial network could be the minimum necessary to deal with the most probable traffic loads; whereas the redundancy provided in a military network should be the minimum necessary to deal with the worst-case attack, which is a highly unlikely eventuality except during conflicts.

The survivability requirement for military networks adopted in this paper was based on a statement made by the MOD [10], but further refined to: the proportion of working node pairs that can be successfully connected must be greater than 95%, even with up to half of the network nodes inoperable. Consequently the network must be designed to spread the load and so reduce the impact of any outages caused through attack or simple failures. This differs from commercial networks, which operate to generate a profit for their shareholders, and must, therefore, optimise their networks in terms of minimising cost.

Clearly, in a military network, even in peacetime, technical failures of links cannot be ignored. Moreover, in peacetime, when the likelihood of node destruction is extremely small, the routing tables devised to be survivable would also effectively re-route calls through the network following simple failures or accidentally dug-up cable. However, the resultant routing would not necessarily be optimum in terms of efficient use of the network, as it could use more links than strictly necessary. Given that the capacity provided in a military network could be less than 50% utilised, owing to the requirement to provide increased levels of redundancy, inefficient use of capacity in a fully operating peacetime network is not an overriding concern.

4 Network design

There are many stages in the design of a telecommunications network, including: topology design, traffic routing, and link dimensioning. Once the topology has been determined, the routing of the network could then be addressed, reflecting in more detail the expected call patterns, and relying on the links provided. The final stage could then be to identify the capacity required for each of the links. The whole design process may then go through several iterations to obtain further optimisation benefits or to improve areas that have been demonstrated to be vulnerable. However, here the network topology is given, and the focus is on routing for survivability.

4.1 Circuit-switched routing

For a *static* routing scheme the routing tables remain fixed, regardless of the current state of the users or the network. The tables are based on expected, rather than actual, user and network behaviour. In most static routing systems, routing is an integral part of the network design and hence is changed infrequently. Static routing involves virtually no real-time activities other than table interrogation and traffic forwarding. Hence static routing requires minimal computational resources within the network nodes. Any changes to the routing tables can even be carried out manually. Thus by keeping the computational aspects central, and minimising the computation required within the network itself, network capital costs are reduced. The static tables can be calculated on systems that are separate from the network, in an off-line manner. The advantages of computing routes off-line include complete control over the routes selected (ensuring compliance with network design assumptions and thus predictable routing performance) and minimal processing requirements for switches. This is beneficial for military networks as it allows the performance of routing tables to be determined in advance of their deployment, which gives confidence in the system. Moreover, military networks have a lower user-to-node ratio than commercial networks, and hence more nodes for a given traffic loading, and it is therefore advantageous to reduce the cost and complexity of each of the nodes. The disadvantages, however, include slow adaptation to unpredicted events (perhaps even requiring human intervention), and inability to optimise routes within the network [11] for efficiency.

By way of contrast, *dynamic* routing in a circuit-switched network is a way of providing flexibility at the switching level. It allows the path that each call takes between any source-destination pair to vary with time or the state of the network. This enables the network to be used more efficiently and provides robustness [11]. However, although a dynamic routing scheme increases an individual call's chance of success, this very success could prevent the success of other calls, owing to local uncoordinated routing changes.

4.2 Routing tables

A static routing strategy is completely determined by a routing table and a call control rule, where the latter determines how the routing table is to be used. For a network of n nodes, the routing table can be expressed in the form of a matrix of $n \times n$ entries, with the (falling) diagonal entries unused. Each entry, as shown in Table 1, contains an ordered set of nodes. For the *successive office control* rule [12], each such entry indicates the sequence in which the nodes should be addressed when attempting to set up calls across the corresponding net-

From/To	1	2	3	4	5
1	—	2	3,2	4,3,2	5,2
2	1	—	3,4	4,3	5
3	1,2	2,1	—	4	1,2
4	1,2,3	2,3	3	—	2,1
5	1,2	2,1	1,2	1,2	—

Table 1: Example routing table (after [12])

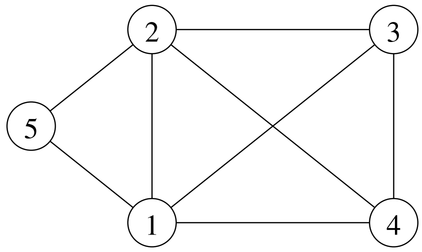

Figure 1: Example network (after [12])

work (Fig. 1). The entry in row 1, column 4 (*from* node 1 *to* node 4) has three entries, in the order 4, 3, 2. This means that a call reaching node 1 (regardless of the origin of the call) and destined for node 4, is to be routed through the links, in order of preference, 1–4, 1–3, and 1–2. If a particular link is unavailable owing to congestion, link failure, or more relevant to this work, node failure or loss, then the next link is attempted from the list given. If all of the links in the set are unavailable, then the call is blocked and cannot be set up. This system can be regarded as 'fire and forget' since, if a node can pass on the call-set-up request signalling, then it will do so; if not, it will terminate the call.

If the routing table is not properly designed, then call-set-up signalling could continually be passed around the network. Under such conditions the network could enter an infinite-routing loop, where the network makes a (potentially) never-ending attempt to establish a path between the source and the destination. This would happen, for example, if a call from 5, went to 2, which passed it to 1 which in turn passed it back to 5 and so on. However, in practice, the number of links (hops) used to route a call is counted as routing proceeds, and when this hop count exceeds a given number then the call-set-up process is abandoned; this prevents routing loops tying up network resources indefinitely.

4.3 Survivable network design

A simple method that can be employed to reduce the impact of losing a node is to put less traffic through each one. Therefore, a lower user-to-node ratio is required in a military network than with a commercial network. Consequently the practise of concentrating traffic in higher-level centralised switches, to reduce the costs associated with long interconnecting links, is not followed in military networks, as the impact of losing such an exchange would be highly significant. Schroeder & Newport [13] have stated that survivability can be equated to the inability of an enemy to identify critical node resources. Whilst this is true, there are also physical methods that can be employed to enhance survivability such as hardening (physical protection against explosives) of nodes or introducing transportable nodes. However, in line with Schroeder & Newport's observation, military networks are designed to be functionally dispersed and physically distributed, with all nodes having as near equal importance to the network's functionality as possible. However, while survivability depends heavily on network configuration, for an existing network it would be preferable, in the first instance, to change the routing, held in software, rather than physically modifying the network. Schroeder & Newport's study of survivability is extensive, but it concentrates on topology design without mention of routing methods, which are also important. Routing can be used to reduce the vulnerability of network components by equalising the traffic loading.

5 Problem description

The topology of the actual military network, on which the model network used in this paper is based, was determined from predicted traffic levels. Nodes were placed so that the vulnerability of each node was nearly equal [13]. Consequently, areas of high traffic concentration, or of high tactical importance, would have proportionally more nodes per unit area and less users per node than a commercial network. The whole network topology consists of many local access switches which feed into $n = 18$ nodes, which are highly mesh-connected. To fully interconnect all of the nodes across a nation-wide network would be prohibitively expensive; therefore, only as many links as can be justified, to give the required level of survivability, are provided. It is this core network of 18 nodes which is investigated here.

As in the actual network, the model network uses a static routing scheme and successive office control [12]. Each node uses a small measure of distributed processing to route calls according to the static routing tables. The tables are calculated centrally and off-line. The advantage of this is that the network should perform predictably, as a whole, for any failure state. The general acceptance that dynamic routing schemes are preferable in terms of survivability and reliability is questionable, because dynamic schemes are constrained by time and processing power in finding an optimal or near-optimal solution. Moreover, whilst they may solve a local problem, when the network is up to 50% damaged, confidence cannot be high that the whole network will continue to function. With static routing tables that can be tested exhaustively against all failure combinations, confidence is higher that the network will continue to function even with up to 50% of the nodes destroyed. An added benefit, although not a deciding factor, is that centralised processing of the static routing tables is cheaper in terms of both hardware and software than distributed processing, as each node is less complex.

Node	Choices
1	`2,3,6,4`
2	`3,4,1`
3	`4,7,1`
4	
⋮	
n	

Table 2: Single-destination routing table

5.1 Single-destination routing tables

The routing tables for the actual network are stored in the memory of each physical network node, and indicate the order that links should be used for routing a call onwards, for all destinations. However, the routing tables used in this paper are network (rather than node) routing tables, and they do not reflect how the information is actually stored in the memory of the nodes. Each of these network routing tables is constructed as indicated in Table 2 and refers to just one destination. It corresponds to an entry taken from every node routing table, all referring to the same destination. Optimising such a table is equivalent to optimising the routing through the whole network for one given destination node at a time, as routing for any one destination is independent of that for any other. To derive an overall network table for n nodes, each of the n separate single-destination routing tables would have to be separately determined. Nevertheless, by using single-destination tables in this manner, the search space for individual runs is divided by a factor of n.

5.2 Routing-table assessment

Introducing a limit on the number of hops taken through the routing table helps to prevent infinite-routing loops occurring, as well as reducing the computational effort involved in determining near-optimal tables. This also improves the network's use of resources [14]. However, some additional work has to be done to determine an appropriate hop limit. In addition, the objective (fitness function) used to assess a table must discourage the generation of routing tables with infinite-routing loops.

Consequently, routing tables are evaluated against several different objectives simultaneously: shortest-path routing, the absence of infinite-routing loops, and ability to reach the destination. Connections between each working source node i and the fixed destination node j ($i \neq j$) are tested for each network failure state s (from the set of all valid failure states, S_j for this routing table *i.e.* those failure states for which the destination node j is working). Assessing the routing tables simply in terms of connections, in this way, makes no allowance for possible weighting by offered traffic load. However, in a survivable military network, with considerable over capacity, it is the ability to communicate between node pairs that is of prime importance. For each such connection tested, the assessment penalises unwanted traits in the routing table by adding a weight, P_{ijs}, to its overall fitness value. The fitness of an idealised table attracting no penalties at all would simply be zero. The penalties incurred are as follows.

Direct routes: It is desirable that the routing tables choose the shortest route whenever possible to minimise the number of nodes and links utilised for any single connection. If the table does not provide direct routes whenever possible, then the network will be inefficient and oversized in terms of link capacity. If a path is found, then the value added to the final fitness, to reflect the directness of a route, is directly proportional to the number of hops used:

$$P_{ijs} = K_p \cdot h_{ijs} \tag{1}$$

where K_p is a constant (taken to be 1) and h_{ijs} is the number of hops in the path from node i to node j in failure state s. For example, a routing table which uses only one hop to connect two nodes would have the value 1 added to its fitness assessment. A table which uses four hops to connect the same node pair would have the value 4 added.

Absence of infinite-routing loops: It is beneficial to produce routing tables that do not have loops within them. However, given the highly interconnected nature of the network, and the high number of nodes that could be destroyed, it may well be impossible to produce a network with no routing loops without discarding acceptable solutions which achieve the requirement to pass 95% of traffic. The penalty function attached to infinite-routing loops is inversely proportional to the hop count:

$$P_{ijs} = K_i / h_{ijs} \tag{2}$$

where K_i is a constant (taken to be 100) and h_{ijs} is the number of links in the path (from node i to node j in failure state s) at the point the loop is detected. Thus, a loop detected after four hops will have a higher penalty associated with it than a loop that was detected after six hops. This reflects the observation that large hop counts can realistically be expected in states with a large number of failures, as the network is expected to choose direct and short routes, *i.e.* routes with minimal hops. As the network starts to suffer failures, then the direct routes will no longer be available and longer routes, with more hops, will be taken instead.

Reaching the destination: If the routing table does not manage to connect a particular source node to the destination, then this is a highly undesirable outcome. Consequently, the resulting penalty is a high one. The reasons why a routing table would be unable to connect a call are either running out of alternative routes at a node or exceeding the hop limit. Thus:

$$P_{ijs} = \begin{cases} K_d \\ K_h \end{cases} \tag{3}$$

where K_d (a constant, taken to be 200) is applied on running out of alternative routes and K_h (a constant, taken to be 150) when the hop limit is exceeded.

The overall fitness, f_j, of the routing table for destination node j is taken to be:

$$f_j = \sum_{\forall s \in S_j} \sum_{\forall i \in W_s} P_{ijs}, \quad i \neq j \tag{4}$$

where W_s is the set of all working nodes in network failure state s.

5.3 Failure-state generation

The method chosen to generate the failure states uses an array of integers $(a_1, a_2, \ldots, a_m)$, representing node identification numbers (ids), whose length depends on the number of node failures required, m. Each integer in the array is limited to the range 1–n, and indicates the id of a node which has failed. To step through each of the failure states the least significant array position is incremented; then, when it is about to exceed its limit, it is reset to 1 and the next number in the array is incremented (modulo n). Clearly, each failure state must only appear once. However, using the above method, it would be possible for each node combination to be repeated, giving permutations rather than combinations, since it is possible that, say, $a_1 = 12$ and $a_2 = 13$ could appear as well as $a_1 = 13$ and $a_2 = 12$. To prevent this happening the following rule was adopted: $a_{r-1} < a_r$ for all values of r.

Considering the routing table for destination j, and a particular number of node failures, m, the number of valid network failure states $|S_j^m|$ (*i.e.* those failure states for which the destination node j is working) is:

$$|S_j^m| = \binom{n-1}{m} = \frac{(n-1)!}{m!(n-m-1)!} \tag{5}$$

In any such state, the number of working source nodes (*i.e.* **not** including destination node j) is $n - m - 1$. Consequently, the total number of connections that must be tested, C_j, in assessing the table, and for the number of node failures ranging from 0 to M, is:

$$\begin{aligned} C_j &= \sum_{m=0}^{M} (n-m-1) \cdot |S_j^m| \\ &= \sum_{m=0}^{M} \frac{(n-1)!}{m!(n-m-2)!} \end{aligned} \tag{6}$$

For the model network, with $n = 18$, the maximum number of failures used in assessing a single-destination routing table was fixed at $M = 8$, equivalent to (almost) 50% of the remaining 17 nodes failed. Using these values in Eq. 6 gives the number of connections to test as 666,451: a large number for a single fitness assessment. This is an important figure as it allows the percentage of successful connections made by a particular routing table to be calculated. For example, 1% failure would correspond to 66,645 unsuccessful connections during assessment.

Elitist	
Population size	100
Maximum trials	40,000
Crossover rate	0.6
Mutation rate	0.001
Generation gap	1.0
Scaling window	5

Table 3: GA parameters

6 Solution approaches

Two solution approaches were adopted to tackle the problem described in §5: a bit-string GA and an ISHC.

6.1 The genetic algorithm

For the GA, binary encoding of the routing table was used, but rather than directly encoding the node ids in each table entry, instead indices into a look-up table were employed [4]. For a particular node, the look-up table lists only those nodes to which it is directly connected, reflecting the actual topology of the model network. The index values thus allow routing selection, for a particular node, from just those nodes to which it is actually connected. In addition, once a next node has been identified and used as a choice in the routing table, clearly it cannot be used again. Therefore, subsequent indices refer to fewer choices and, therefore, less bits are needed to code them. For example, with 8 choices of outgoing route available from a switch, the first 'cell' (*i.e.* group of bits encoding a single index value) will contain 3 bits, to identify which of the 8 links is the first choice. The second cell must also be 3 bits long as it must refer to one of 7 nodes. Likewise the third and fourth cells. However, the fifth and sixth cells have only 4 and 3 options respectively, so they can be coded in just 2 bits. The seventh cell has only 2 options and therefore can be coded in just 1 bit. The eighth cell has an option of only one node, and therefore does not need to be coded at all. For the model network, the encoding for the routing table to node 9 required a total of 220 bits. (In decoding a cell from binary, if an invalid, *i.e.* over-large, cell value was encountered, it was simply reduced to the largest allowable index.)

The GA was implemented using GENESIS v5.0 [15], with fitness-proportional selection (using window scaling), and parameters as given in Table 3. These were selected from a few trial runs to obtain good results using 40,000 trials, the longest run length that could be readily accommodated with available computational resources. Both higher mutation levels (up to 0.01) and larger population sizes (up to 250) provided inferior results, even with runs up to 100,000 trials in some cases. In addition, using restricted fitness assessments (*e.g.* only 0–3 node failures) for early trials, and so hastening execution, appeared to produce a slight reduction in final solution quality.

Node	Choices
1	9,14,17,18
2	9, 4,15, 3,11
3	2, 4, 7,10, 5,13
4	9,14, 7, 2,11,18,17, 8, 3, 5,13,15,16
5	14, 4, 7, 6, 3,12
6	9,14,12, 5,13
7	9, 4,16,14, 3, 5,17
8	16, 4,14,12,17,11
10	14,16,11,18,17, 3
11	2, 4,14,16,12, 8,10,13
12	6,14, 8,11, 5
13	4,15,11, 6, 3
14	9, 4, 7,16, 6, 1, 5, 8,10,11,12,18
15	2, 4,16,13
16	9, 7,14,15,11,10, 8
17	4, 7, 1,16,10, 8
18	14, 4,10,17, 1

Table 4: Existing routing table to node 9

Metrics	Hop limit		
	4	6	8
f_9 ($\times 10^6$)	1.864	1.767	1.767
c_i	6,843	6,997	6,998
c_d	2,461	1,776	1,771
c_t	9,304	8,773	8,769
g	98.60%	98.68%	98.68%

Table 5: Varying hop count for existing routing table

Alg.	Median	Best Run				
	f_9 ($\times 10^6$)	f_9 ($\times 10^6$)	c_i	c_d	c_t	g
GA1	1.507	1.501	1,519	1,776	3,295	99.51%
ISHC1	1.499	1.499	1,451	1,775	3,226	99.52%
GA2	1.502	1.499	1,451	1,775	3,226	99.52%
ISHC2	1.499	1.499	1,451	1,775	3,226	99.52%

Table 6: Fitness summary results

6.2 Iterative stochastic hill climber

The ISHC does not encode the routing table, but works directly on it using a swap operator. This selects a source node at random and then, for that node, chooses two routing choices (next nodes) at random. These are exchanged, and the table re-assessed. If the swap results in an improvement, it is retained, otherwise the original choices are restored. However, in the latter case, the identity of the node and pair of choices tried are remembered. This allows the ISHC to avoid re-trying a previously failed swap. As soon as an improvement is found, though, the memory of all unsuccessful swaps is cleared. The ISHC is initialised either with a user-supplied routing table, or a randomly-generated one. On detecting a local optimum (all of the possible swaps tried unsuccessfully since the last improvement), the ISHC restarts using either the supplied routing table, or a new randomly-generated one, as appropriate.

7 Experimental results

The existing routing table for destination node 9 is shown in Table 4. This was assessed for all node failures 0–8, with hop limit of 4, 6 and 8. Table 5 shows the fitness, f_9; the count of the number of infinite-routing loops, c_i; the count of the number of failures to reach the destination, c_d; the total number of connection failures, $c_t = c_i + c_d$; and the success level, $g = 1 - (c_t/666,451)$. Clearly, a hop limit of 6 is an appropriate choice, as it represents the lowest value beyond which further improvement is unlikely. Consequently, this value was adopted for all the additional experimental work reported here.

Before considering the results obtained by the GA and ISHC, it should be noted that the good performance of the existing routing table (98.68%) against the target of 95% has dispelled the suspicion that the existing routing tables were not meeting the network requirement, at least for destination node 9. However, the algorithm used to generate the existing tables was implemented in hardware and unfortunately, the original documentation for the algorithm could not be located by either the first author or his colleagues. Moreover, the initial hardware provided is some 10 years old, does not reflect current technology and more importantly, no longer works.

Consequently, two different overall scenarios were considered for the experimental work. For the first, the existing routing table was taken to be unknown, and thus both the GA and the ISHC were initialised with random tables (GA1 and ISHC1). Acting in this way, the algorithms are to be considered as potential replacements for the (non-working) current approach. In the second scenario, the existing table was assumed known, and consequently both a single instance was placed in the otherwise random initial population of GA2, and it was also used as the (re-)starting point of ISHC2. Thus the algorithms are being applied in this case as an attempt to improve on the existing table.

For the first scenario (existing table unknown), ten runs each were performed, with different pseudo-random number generator seeds, for GA1 and ISHC1. For comparison with GA1, ISHC1 was also allowed runs of 40,000 trials. The experimental work was carried out on a Sun UltraSPARC 5, with each individual run taking between 4 and 5 hours. While there was some variation in the GA1 runs, the outcome of all ten ISHC1 runs was identical (in minimum 1,063, median 13,302 trials). Table 6 summarises the best individual run, in fitness terms, and median of ten for the two algorithms.

Clearly, both algorithms provide a substantial improvement over the existing routing table: 15% in fitness terms, 63% for success level. These represent a considerable improvement in communication capability in the face of a hostile aggressor. However, of the two algorithms, it is ISHC1 which consistently provides marginally better results. A non-parametric pairwise sign test [16], applied to determine if the difference in the two medians was statistically significant, established a 0.1% significance level (*i.e.* a very-highly signifi-

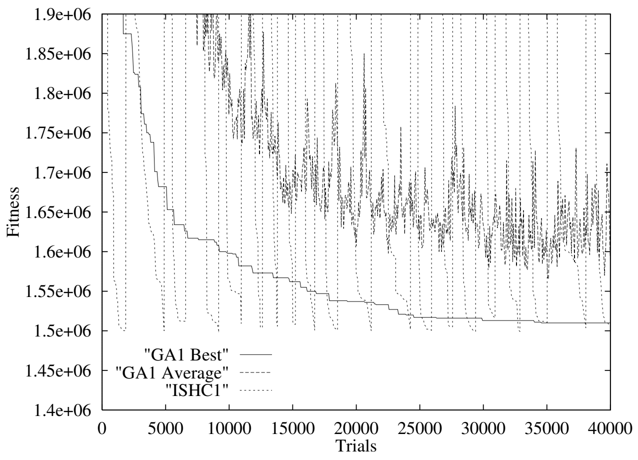

Figure 2: Example run for GA1/ISHC1

Node	Choices
1	9,14,17,18
2	9, 4,15, 3,11
3	2, 4, 7,10, 5,13
4	9,14, 7, 2,11,18,17, 8, 3, 5,13,15,16
5	14, 4, 7, 6, 3,12
6	9,14,12, 5,13
7	9, 4,16,14, 3, 5,17
8	16, 4,14,17,12,11
10	14,16,11, 3,18,17
11	2, 4,14,16,13, 8,12,10
12	6,14,11, 8, 5
13	6, 4, 3,11,15
14	9, 4, 7,16, 6, 1, 5, 8,10,11,12,18
15	2, 4,16,13
16	9, 7,14,15,11,10, 8
17	4, 7, 1,10, 8,18
18	14, 1, 4,17,10

Table 7: Best routing table to node 9

cant difference).

In considering the absolute quality of the results obtained, it should be noted that, due to limitations in the network topology, 1,771 connections cannot reach the destination because the source node is disconnected (*i.e.* isolated) by node failures. This therefore represents a lower limit on c_d. Similarly, 351 connections must inevitably result in infinite-routing loops (or exceed the hop limit) because: the source is part of a two-node component that does not include the destination (244); the destination is isolated (99); or the destination is part of a two-node component that does not include the source (8). Additional, more complex, network disconnections must also be responsible for further infinite-routing loops, regardless of routing-table design. Consequently, the results obtained by GA1/ISHC1, particularly for c_d, closely approach the limit of what can be achieved with the existing network topology.

The progress of typical runs for GA1/ISHC1 are illustrated in Fig. 2, which shows the average and best fitness for GA1, and the current fitness for ISHC1, against the number of trials. Through the run, GA1 shows steady improvement in the best individual, passing the fitness of the existing table in less than 3,500 trials. However, ISHC1 is able to find good quality results at the 'top' of every 'hill'; its full run, including restarts, simply ensuring the excellence of the final result.

Similarly, for the second scenario (existing table known), ten runs each were carried out using GA2 and ISHC2. The number of trials allocated was identical to those allowed for GA1/ISHC1, *i.e.* 40,000 trials. The results are summarised in Table 6. Once again, the outcome of all ten ISHC2 runs was identical (in minimum 1,848, median 6,047 trials).

As before, substantial gains over the existing table are seen using both algorithms for this problem scenario. In addition, while best individual GA2 run was able to match the consistent outcome of ISHC2, clearly the latter provided the better median. The same non-parametric test as for GA1/ISHC1 established a 0.8% significance level (*i.e.* a highly significant difference between the two medians).

The best routing table overall (in fitness terms), found by ISHC1, is given in Table 7 (although tables of the same fitness were also found by GA2 and ISHC2).

7.1 Limitations of network topology

All of the research reported here was carried out with the topology of the model network fixed. However, it was suggested in §4 that the design of networks may be tackled as an iterative process, with each of the various design stages (topology, routing and sizing) being revisited several times. Given the limitations on survivability imposed by the existing topology, some of which were enumerated above, alteration of the real-network topology would allow further improvement in survivability over what can be achieved with routing tables alone. Clearly, when a network must cater for the possibility that up to eight nodes might be destroyed, it is reasonable to assume that at least eight links be provided to each node. If nine links to nine different nodes were provided then, even in the worst failure states, with eight nodes failed, there would be one link left that could pass traffic from any node. When looking at node 9 as the destination, the current network has fourteen nodes with less than eight links. Moreover, contrary to Schroeder & Newport's recommendations [13] the remaining three nodes with more than eight links represent areas of increased vulnerability.

8 Conclusions & further work

The aim of this work was to demonstrate the use of both a GA and an ISHC in optimising a military circuit-switched network for survivability. The authors, to the best of their knowledge, are the first to apply GA techniques specifically to the optimisation of routing for military network survivability.

In the experimental work, both the genetic algorithm (GA) and iterative stochastic hill climber (ISHC) approaches provided substantial improvement over the existing routing table for a single destination, whether the knowledge of the existing table was utilised or not. Indeed, if the best routing table were to be used in the real network, it would reduce the number of potential failed connections by more than half, representing a notable enhancement in communications under hostile attack. Clearly, especially given the lack of a current working method for routing table design, either approach would be suitable for adoption for future routing table design. The only note of caution may be that further experimental work should be carried out with tables for different destinations, as well as different model networks, to ensure the results have not been biased by the particular table adopted for study.

Of the two algorithms, however, the ISHC was found to be both more consistent, as well as producing marginally better results. In addition, these were often obtained with far less than the full allocation of 40,000 trials. The authors were disappointed by this relatively poor performance from the GA. However, this may have been due to the choice of both a bit-string representation and standard genetic operators. In future work, it may be beneficial to develop a hybrid GA (*cf.* [8]) combining both direct representation of the routing tables with problem-specific operators, including incorporation of the swap operator from the ISHC. Further investigation of the problem may also yield additional heuristic procedures that can be utilised in the hybrid GA. Another area which would benefit from further investigation is the assessment of survivability when more than eight nodes fail. It may well be possible that the network could function according to its specification with nine nodes destroyed.

Acknowledgements

The research described in this paper was developed from a project undertaken by the first author (supervised by the second) as part of an M.Sc. in Telecommunications and Information Systems at the Department of Electronics Systems Engineering, University of Essex. The first author was supported by the RAF. The first author would also like to thank Squadron Leaders Jim Allan and Huw Roberts for their extensive proof reading of his M.Sc. project report, and Ian Kirkwood for the original inspiration.

Bibliography

[1] Cox, L.A., Jr., Davis, L. & Qiu, Y. Dynamic anticipatory routing in circuit-switched telecommunications networks, in Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, Ch. 11, 124–143, 1991.

[2] Davis, L., Orvosh, D., Cox, A. & Qiu, Y. A genetic algorithm for survivable network design, *Proc. 5th Intl. Conf. on Genetic Algorithms (ICGA'93)*, University of Illinois at Urbana-Champaign, USA, 408–415, Jul 1993.

[3] Pan, H. & Wang, I.Y. The bandwidth allocation of ATM through genetic algorithms, *IEEE Global Telecom. Conf. (GLOBECOM'91)*, 1:125–129, 1991.

[4] Sinclair, M.C. The application of a genetic algorithm to trunk network routing table optimisation, *Proc. 10th UK Teletraffic Symposium*, Martlesham Heath, UK, 2/1–2/6, Apr 1993.

[5] Bentall, M., Turton, B.C.H. & Hobbs, C.W.L. Benchmarking the restoration of heavily loaded networks using a two dimensional order-based genetic algorithm, *Proc. 2nd IEE/IEEE Intl. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97)*, Glasgow, 151–156, Sep 1997.

[6] Shami, S.H., Kirkwood, I.M.A., & Sinclair, M.C. Evolving simple fault-tolerant routing rules using genetic programming, *Electronics Letters*, 33(17):1440–1441, Aug 1997.

[7] Munetomo, M., Takai, Y. & Sato, Y. An adaptive network routing algorithm employing genetic operators, *Proc. 7th Intl. Conf. on Genetic Algorithms (ICGA'97)*, Michigan State University, East Lansing, Michigan, USA, 643–649, Jul 1997.

[8] Sinclair, M.C. Minimum cost wavelength-path routing and wavelength allocation using a genetic-algorithm/heuristic hybrid approach, *IEE Proc. Commun.*, 146, 1999.

[9] Zhu, L., Wainwright, R.L. & Schoenefeld, D.A. A genetic algorithm for the point to multipoint routing problem with varying number of requests, *Proc. 1998 IEEE Intl. Conf. on Evolutionary Computation (ICEC'98)*, Anchorage, Alaska, USA, 171–176, May 1998.

[10] Defence Communications Information Systems Board, Decision 50/94, 1994.

[11] M. Steenstrup. *Routing in Communication Networks*, Prentice Hall, 1995.

[12] P.M. Lin. Analysis of circuit-switched networks employing originating-office control with spill-forward, *IEEE Trans. Commun.*, 26(6):754–765, Jun 1978.

[13] M.A. Schroeder & K.T. Newport. Enhanced network survivability through balanced resource criticality, *Proc. IEEE Military Communications Conf. (MILCOM'89)*, Boston, MA, USA, 682–687, Oct 1989.

[14] M. Herzberg, S.J. Bye & A. Utano. The hop-limit approach for spare-capacity assignment in survivable networks, *IEEE/ACM Trans. Networking*, 3(6):775–784, 1995.

[15] J.J. Grefenstette. *A User's Guide to GENESIS Version 5.0*, 1990.

[16] Sprent, P. *Applied Nonparametric Statistical Methods, 2nd Ed.*, Chapman & Hall, 1992.