Heuristics for Evolutionary Off-line Routing in Telecommunications Networks

Joshua Knowles and David Corne
Department of Computer Science, University of Reading, UK
J.D.Knowles@reading.ac.uk, D.W.Corne@reading.ac.uk
FAX: +44(0) 118 975 1994, TEL: +44 (0) 118 931 8983
http://www.rdg.ac.uk/~ssr97jdk

Abstract

Off-line routing in backbone telecommunications networks is a combinatorial optimization problem well-suited to iterative search methods. In this paper, a number of further heuristics applicable to the routing problem are introduced and evaluated. The presented results show that these methods significantly improve the search for solutions, particularly when on-line performance is considered. The use of delta-evaluation of solutions in order to reduce computation time further, is also investigated. Previously, simulated annealing has had a significant advantage over genetic algorithms on problems where delta-evaluation is applicable, because genetic algorithms employing standard crossover operators cannot make use of the technique. However, we introduce a specialized recombination operator which enables a genetic algorithm to exploit delta-evaluation effectively on the routing problem. The performance of the genetic algorithm employing the new recombination operator is found to be significantly better than that of a mutation-only genetic algorithm.

1 Introduction

Off-line routing has three main uses: First, it provides reference benchmark results for dynamic (on-line) routing strategies; second, it can be used in networks where bandwidth may be booked in advance; third, and most interestingly, off-line routing is becoming more and more investigated and employed in its own right as a way of quickly finding significantly improved routings for live networks which can then be imposed on the network to offer a net improvement in quality of service. The latter is applicable when calls are only partly known in advance, or not known at all: Given a collection of existing calls on a network, an off-line routing strategy can run in the background, and potentially find a better routing for the existing calls, which can then be imposed on the network via re-routing certain calls as necessary. Recent work investigating this idea in the context of adaptive distributed databases (intermittently re-routing clients to different servers based on changing access patterns and base network communications speed data) can be found in [14, 13]. In particular, [1] investigates the tradeoffs between the speed of off-line routing algorithm, quality of solution, and relevance of solution to the possibly changed network after the short delay. In the context of standard telecommunications routing scenarios, off-line routing has been investigated by Mann and Smith [10], and is also the topic of a major industrial project involving Nortel Communications Plc[1].

Off-line routing involves finding a set of routes for a given network/traffic combination such that communications costs and congestion are minimized. In this paper, we show how, by biasing the initialization and mutation operations used in the search process, significant improvements in performance on this problem can be made. We investigate three methods of biasing the search: First, towards solutions containing many short(est) paths; second, to favour paths which cause little congestion; and third, using a combination of the first and second methods. The latter is achieved through the use of a 'double' chromosome. The 'double' chromosome and its use has a similar flavour to the employment of multiploidy, in which a genotype consists of multiple 'sub'-chromosomes, and a 'mask' chromosome or a dominance mechanism is used to construct the chromosome to be interpreted by indicating

[1] See "hot emulation" at the Internet website, http://www.elec.qmw.ac.uk/telecom/industry.html

which sub-chromosome to use for each gene position. See [7] for an early discussion of work employing such ideas, and [3, 19] for more recent examples. A further variant on this theme is Dasgupta's Structured Genetic Algorithm (SGA) [4].

Next, we consider the use of 'delta-evaluation'. This is a way of very quickly calculating the fitness of a candidate solution s, by using the fitness of one of its parents, p, and knowledge of the genotypic difference between s and p. In the off-line routing problem delta-evaluation can be applied effectively; as has been noted previously [10], the computation per evaluation required by simulated annealing on this problem can be greatly reduced by employing delta-evaluation. When large problems are being considered this means that the solution evaluation can be accelerated significantly. This would seem to give the simulated annealing algorithm a major advantage in this problem domain. However, we introduce a specialized recombination operator which allows the effective use of delta-evaluation on this problem and show that through the use of this operator very fast evolution to good solutions can be achieved. We compare the delta-evaluated GA employing our new recombination operator with a mutation-only GA to verify its utility.

The new heuristics described above are compared with standard representations and operators in a steady state genetic algorithm (GA) [6]. Results are presented from multiple runs on eighteen different test problems of varying network sizes and traffic levels. Both the off-line and on-line performance of the GA is tested. A baseline shortest path routing algorithm is also used to place the performance of the genetic algorithms into context.

The organization of the rest of this paper is as follows: Section 2 summarizes the approach taken by Mann and Smith [10] to the routing problem, which involves an algorithm due to Yen [18] for calculating the K shortest paths in a graph. The heuristics that we employ and the rationale underlying them are then described. The incorporation of delta-evaluation is reported in section 3. Before presenting results, a brief outline of the test problems is given in Section 4. Results are then presented in Section 5 and the work is summarized in Section 6.

2 Approaches

The detailed formulation of the off-line routing problem adopted here, including the objective function, is described in [10] and [8] . Briefly, we are given a sparse graph consisting of a set of vertices, and a set of weighted edges connecting some pairs of vertices, each with an associated bandwidth capacity. We are also given a set of source-destination pairs and associated bandwidths, to represent connection requests. The task is then to assign paths through the network for each connection request such that:

1. No edge is over-capacitated;
2. calls are spread as evenly as possible over the graph;
3. the total cost of routing calls is minimized.

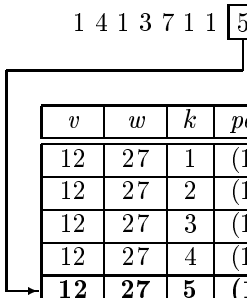
The problem can obviously be tackled as a (possibly constrained) multiobjective optimization task, but here we consider only the formulation used in [10], which uses an aggregating function.This allows us to focus on the effect of our heuristics, independent of the difficulties of interpreting multiobjective results. Elsewhere, we have considered this problem using a multiobjective formulation [9].

The space of feasible solutions to a routing task in a large network with a large number of calls to route is enormous. Only a very small fraction of these solutions meet the constraints on link utilization that we require. So, to reduce the search space, routes which we can predict will lead to a poor solution are excluded. One way to rule out such paths is to create a list of the K shortest paths between the $n(n-1)/2$ source-destination pairs. This is the approach taken by Smith and Mann and is the basis of all the work presented here. The *shortest* in "K shortest paths", applies to the cost of the path not the hop number but since the costs of using edges in our network are proportional to the Euclidean distance between the incident nodes, in general short paths are also ones with fewer hops.

Using the K shortest paths for each source-destination pair, as well as reducing the search space greatly, and the computation of routes, leads to a simple representation of solutions. Mann and Smith use a K-ary representation of length $n(n-1)/2$ to represent, without repetitions, all possible source-destination pairs. Our approach is to represent only the calls that need to be routed so that we have a vector(chromosome), also of K-ary elements (genes)[2], of length $l = \sum_{e \in P(v,w)} x(v,w)$, where $x(v,w) = 1$ if there is traffic to be routed over the path $P(v,w)$ between node v and node w, and $x(v,w) = 0$ otherwise.

[2]Empirically, we found $K = 15$ to be a sound choice for the problems considered here. Higher values lead to an increased search space. Lower values limit the choices of routes too much.

Each gene in the chromosome points to an entry in a look-up table containing the route that it refers to (see Figure 1). The routes in the look-up table, showing each of the nodes to be visited, are pre-computed using an algorithm due to Yen [18].



v	w	k	*path*
12	27	1	(12, 34, 27)
12	27	2	(12, 17, 34, 27)
12	27	3	(12, 16, 19, 27)
12	27	4	(12, 16, 28, 34, 27)
12	**27**	**5**	**(12, 16, 19, 28, 34, 27)**
12	27	6	(12, 21, 16, 28, 19, 27)

Figure 1: How a chromosome encodes for routes in the network. In the chromosome (top) each gene codes for the route to be taken between its corresponding source-destination pair v, w, via a look-up table (bottom). In this example, the eighth gene codes for the 5th shortest path ($k = 5$) between nodes 12 and 27.

Yen's algorithm is efficient, requiring only a few seconds to calculate all the required paths in the networks considered in this paper. It requires an upper bound of approximately $1/2Kn^3$ operations for each of the source-destination pairs, where n is the number of nodes in the network. The algorithm works by calculating the first shortest path (using some suitable method, see [17]) and then successively setting edge costs in the path to ∞, each time applying a standard shortest path algorithm to find a new path. Guarantees exist that the paths generated are the K shortest and no re-ordering is required.

2.1 Biasing by Path Length

Under the representation scheme described above, each chromosome consists of a list of genes, each having an allele value in the range 0 to K. Smaller allele values correspond to the selection of shorter paths. Our first heuristic is simply to bias the initialization and mutation functions used in the search process so that lower allele values, and hence shorter paths, are favoured. This is achieved by employing an exponential function on a random variable $x \in [0,1]$, having the form $int(e^{x^p.log(15.49)})$ (where p is a tuning parameter), in the random initialization and mutation functions of the search algorithm. The tuning of the exponential function was engineered to generate chromosomes with a distribution of allele values that resembled those chromosomes found (through previous optimizations) to represent good solutions. In addition, the connections represented by the chromosome are arranged in decreasing order of bandwidth. This allows that the amount of bias applied can be linked to the bandwidth of the connection request simply by linking the value of the tuning parameter p to position in the chromosome. This method is employed because it is more important for the high bandwidth calls to be routed by the shortest paths possible as they contribute more to congestion than do low bandwidth calls if they are routed along a round-about path.

2.2 Biasing by Congestion Contribution

We now seek to use the same biasing method but this time to apply it to the explicit congestion that a path contributes. This is achieved by re-ordering the list of K shortest paths according to their 'congestion cost'. For each path $P(v, w \in N)$ the congestion cost, $con(P(v,w))$ is calculated using :

$$con(P(v,w)) = \sum_{e \in P(v,w)} \{t(v,w)/b(e) \times 100\} \qquad (1)$$

where $t(v,w)$ is all the traffic to be routed from node v to node w, and $b(e)$ is the bandwidth capacity of edge e. Applying Equation 1 gives rise to high costs for long paths and/or paths that include edges with capacities that cannot accommodate the bandwidth of the traffic. After evaluation, ranking of the paths is carried out using a Shell sort.

2.3 Biasing Using a Multiploid Chromosome

The two schemes described above may be combined. One possible way to combine the two schemes is, of course, to weight the two previous scores and rank the path choices according to a linear combination of the scores. However, this method is unsatisfactory because the choice of weightings to use must be made in a rather arbitrary way. In addition, some paths in the K shortest lists are *much* longer than others and therefore it is difficult to obtain a ranking which is useful as a measure of both of the objectives. Thus, we take a different approach. We employ a 'double' chromosome for each solution. The primary chromosome, as before, represents the choice of path. The additional secondary chromosome is binary and represents whether to interpret the primary chromosome as a choice from the shortest path ranking or from the congestion score ranking. That is, each individual gene is used to look up the choice of path from the ranking established using *either* the measure of path length, *or* the measure of congestion, but as a whole

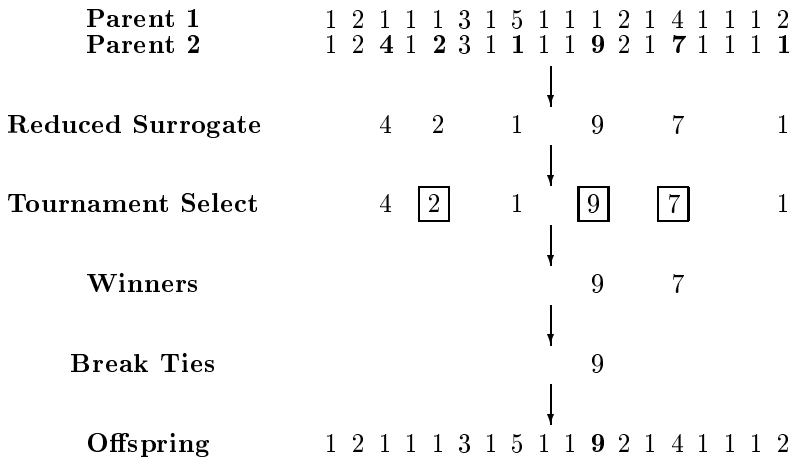


Figure 2: Selection of a gene from *parent 2* to generate offspring from *parent 1*.

the chromosome uses paths selected from both look-up tables.

Both chromosomes are subject to random initialization, mutation and, in the case of a genetic algorithm, recombination. Therefore the representation that is most useful will be chosen during the optimization process. The method has some similarities to Multiploid GAs [3] and structured GAs [4], both of which use genes to indicate whether other genes are active or not. However, these paradigms are aimed at improving the convergence properties of GAs in general, by keeping useful genetic material in the gene pool for longer. Our use of an extra chromosome is for wholly different reasons - to indicate the means of interpretation of genes - and we keep no redundant genetic material from one generation to the next.

3 Delta-Evaluation

We now consider the use of delta-evaluation of solutions on the routing problem in order to accelerate the optimization process. It is possible to evaluate, given the evaluation of a complete chromosome, and some other simple book-keeping the change in evaluation that results from a small change to the chromosome. Mann and Smith [10] note that this gives simulated annealing an advantage in this problem because delta-evaluation cannot be used in a genetic algorithm employing recombination. However, we introduce a new recombination operator which can be used with delta-evaluation, and which retains the utility of recombination to a great extent.

Empirically, we find that chromosomes which encode for good solutions to the problems considered here contain a small number of genes with allele values which lie in the middle to top of the allowed range. To combine these alleles through mutation alone requires a longer time than through recombination, and this is particularly true with the biased initialization and mutation used. Our approach in designing a recombination operator for use with delta-evaluation was to consider how we could retain the speed with which these unlikely allele values could be combined and retained. To allow efficient delta-evaluation, our operator selects two parents and crosses just one allele of one into the other. To ensure that this has a good chance of being a useful change, we use the 'reduced surrogates' [2] of the two parent chromosomes. We then look for elements containing values with the smallest absolute difference from 8, the median allele value which the genes can take. This is achieved by employing a tournament selection strategy between the genes remaining in the reduced surrogate. The genes randomly selected to take part in the tournament are evaluated according to $E = |K - 8|$, and the gene with the lowest value of E is selected. Any ties between the different genes competing in a tournament are broken randomly. A tournament size of three was used, or of size one (i.e. random choice) if the reduced surrogate contains only three elements or fewer. This method generates a new chromosome with only one gene changed but retains some of the effectiveness of crossover at combining rare allele values quickly in the early stages of the optimization process. The recombination operator is shown graphically in Figure 2.

4 Generation of Routing Problems

In order to test our new heuristics, a set of routing problems of different sizes and degrees of difficulty was used. A routing problem consists of a network topology, including a specification of the edge bandwidths, and a traffic matrix specifying the source-destination pairs of the connections to be routed and their bandwidth requirements. Real WAN telecommunications networks are designed to meet several criteria: They require that every node be accessible from every other node, they should be reliable with regard to node or link failure, and they should pay regard to parsimony and minimising delay. In addition, a constraint often exists on the degree of nodes. These considerations often lead to constructions based on minimum spanning trees or degree-constrained minimum spanning trees. To achieve reliability, extra edges are often added to ensure that every node is reachable by at least two paths from every other node, so that if a node or edge fails there is a good chance that all traffic can still be accommodated.

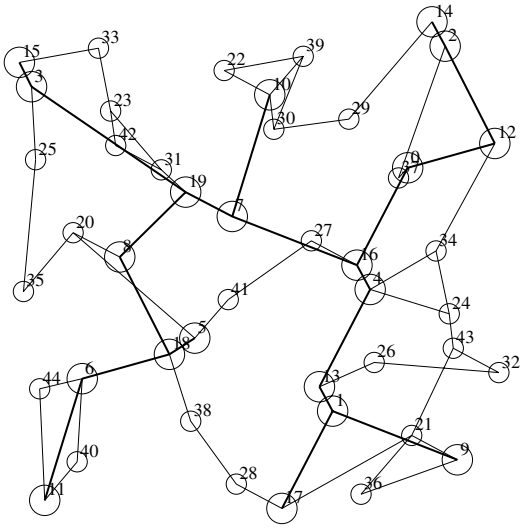


Figure 3: A randomly generated 45-node network.

To attend to the requirements above, a random network generator, based on a suboptimal degree-constrained minimum spanning tree construction algorithm due to Narula and Ho [12], was designed. Due to space restrictions we cannot give details here but a fuller specification is contained in [8]. The networks used in the experiments here have links of two bandwidths. Figure 3 shows one of the 45-node networks constructed using the generator. The larger circles represent nodes on the backbone, and are connected by bold edges, representing 64 unit bandwidth links. The smaller circles represent the remaining nodes, and are connected by feint edges representing 16 unit bandwidth links. The network possesses good, general communication properties: It offers numerous paths between most node pairs, is robust to node or link failure, and has an effective backbone spanning the entire network - all achieved with parsimonious connectivity.

To generate the traffic matrix simply requires randomly choosing source-destination pairs from the $n(n-1)/2$ that exist, and assigning them a random bandwidth requirement. We use bandwidth requirements $t(v,w) \in \{1,2,3,4\}$. Two networks at three different sizes, $n = 30$, $n = 45$ and $n = 60$, and nine different traffic matrices, three for each size of network, were generated. This gives eighteen problems in all: Six networks each of which could be used with any of three traffic matrices for that size. The traffic matrices were all challenging to an extent, but the first one (for each size of network) has slightly less traffic than the second, and the third has the most traffic, sometimes more than can be routed without violating the bandwidth constraint of at least one link.

5 Results

5.1 Implementation details

The basic genetic algorithm that we employ is a steady state GA [6] with a population size of 100. Tournament selection with tournament size 3 is used to select a chromosome for mutation and for selecting parents in crossover. In the former, the mutated chromosome replaces itself regardless of evaluation and in the latter, the offspring created always replaces the worst current member of the population. The mutate operator is applied to the bits of a selected chromosome with probability $1/chromosome\ length$ so that on average one gene is changed. The genes mutated are replaced with a new value taken from whatever distribution is being used. Recombination is achieved using standard uniform crossover [15], except in the delta-evaluated GA where reduced surrogates are used and a single gene is selected, as described earlier. The crossover operator is applied with probability p_c and mutation with probability $1 - p_c$ where p_c begins at 0.7 and is reduced to 0.3 over the first 4000 evaluations, where it remains until termination. The total number of solution evaluations is 20,000 including the initial population of 100. All parameter settings were derived empirically.

5.2 Baseline results

First, we show the validity of the cost function, and the algorithms presented here, by comparing the link

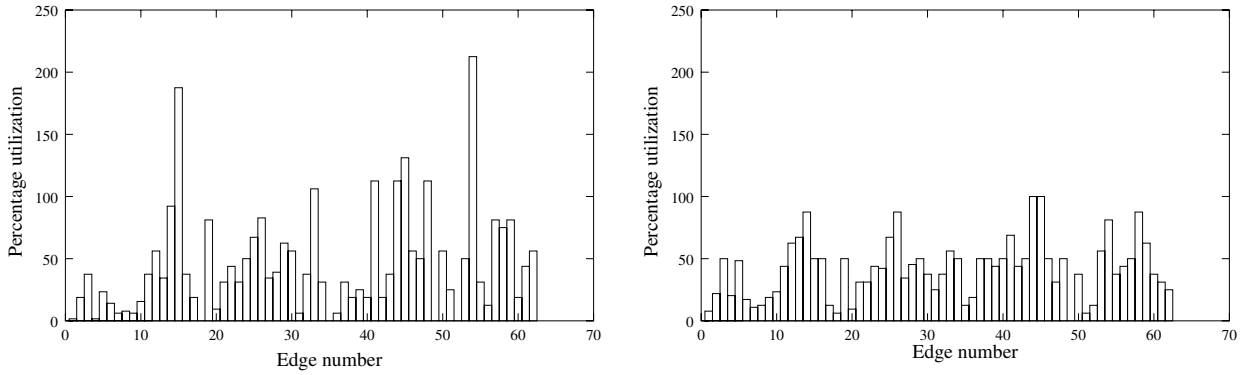


Figure 4: Utilization of links on a 45 node network: In (a) a solution generated using shortest path routing, and in (b) a solution generated using a genetic algorithm incorporating our double chromosome heuristics.

utilizations of a solution generated by shortest path routing [5] and by our genetic algorithm employing both congestion and cost ranking information. Figure 4 shows the percentage utilization of the links in a 45-node network problem. Histogram (a) was produced by the shortest path algorithm and histogram (b) was produced by a GA employing our double chromosome heuristic, after a run of 20,000 evaluations. In the GA solution, all links are at or below 100% utilization, whereas the shortest path algorithm results in link utilisations of over 200% in several links, and spreads the load far less efficiently. The cost of the solution generated using shortest path routing was 2408 and that generated using the GA was 1651, according to the objective function used.

5.3 Quality of final solutions

Next, we consider the performance of the four algorithms as judged by the cost of the final solutions generated. The different algorithms we are comparing are labelled as follows:

- **GA** The standard GA with no biasing of the initialization or mutation.
- **BIAS GA** The GA with initialization and mutation biased towards shorter paths.
- **CON GA** The GA with initialization and mutation biased towards paths which cause less congestion.
- **DOUBLE GA** The GA with a double chromosome.

The best heuristic in terms of final solution costs is, on average, the one employing a double chromosome, encoding both congestion and cost rankings. Table 1 presents these results. The results are the mean of 20 runs of 20,000 evaluations on six problems at each network size.

Algorithm	*Percentage Cost*		
	30-node	45-node	60-node
GA	103.8	105.5	104.4
BIAS GA	101.2	102.6	102.3
CON GA	101.1	102.1	101.9
DOUBLE GA	100.8	101.8	101.5

Table 1: Mean cost of solutions generated by the 4 algorithms at each of the 3 sizes of network. Costs are expressed as the percentage of the lowest cost ever found.

5.4 On-line performance

A plot indicating the typical on-line performance of the genetic algorithms is given in Figure 5. The results are the mean of 20 runs on one 60-node network problem. However, this data plot is representative of the results obtained with the other network sizes and traffic levels. The plot clearly indicates the success of our biasing heuristics at decreasing the number of evaluations required to reach solutions of a given level of quality. Much of the advantage can be seen to arise from the initialization process (observe the cost after 100 evaluations), but mutation must also be improved, since the gradient of the curve is steeper at a given cost level for the algorithms employing the heuristics. Statistical analysis indicates that DOUBLE GA reaches a cost of just 104% of the best cost found after 20000 evaluations, after only 4500 evaluations on average. This compares with a figure of 112% for the standard unbiased GA.

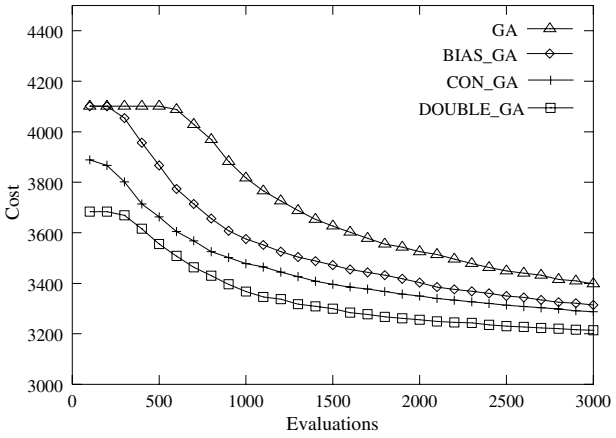


Figure 5: Mean on-line performance on a 60-node network.

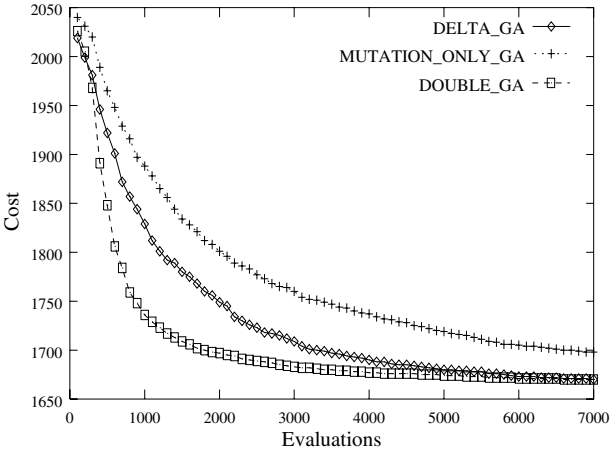


Figure 6: Mean on-line performance of the delta-evaluated GA on a 45 Node Network, compared with the standard DOUBLE GA and a mutation-only GA

5.5 Delta-evaluation of solutions

Next, we present results showing the performance of the delta-evaluated GA. Figure 6 shows how the delta GA compares with the fully-evaluated double GA, and a mutation-only GA in terms of cost against evaluation. As before, the results plotted are the mean from 20 runs. The delta GA is almost as effective as the standard GA here, despite the recombination operator only crossing over a single allele from one parent to the other. On 60-node problems, using the delta-evaluated GA, leads to a speed-up of nearly 30% when compared with the fully evaluated GA: The processor times for 20000 evaluations of an objective function in which 135 calls are routed over a 60-node network, on a 300 MHz SUN ULTRA 30 Workstation, were 72.09 seconds and 53.19 seconds for full and delta-evaluation respectively.

6 Summary and Conclusions

Finding routes for traffic in a congested network is a standard combinatorial optimization problem well suited to iterative search methods. In this paper we have shown that these methods can be further enhanced by biasing the initialization and mutation functions to better reflect the statistical distribution of allele values found in chromosomes which represent good solutions. By incorporating additional heuristics we were able to decrease the number of iterations required to reach a solution of a given level of quality. In particular, the use of a double chromosome encoding, for each gene, the choice of interpretation of the gene to be used, was found to be able to combine two separate and useful heuristics into a single superior heuristic.

In addition, the delta-evaluation of solutions was considered. In this problem domain and many others, the performance of simulated annealing and other search methods based on small change operators can be improved dramatically if solutions can be delta-evaluated. In contrast, genetic algorithms, with standard crossover operators cannot usually take advantage of delta-evaluation because of the large disruption caused to genetic material during recombination. However, we have introduced and tested a new recombination operator which selects a single allele from one parent to be crossed into the other parent to produce an offspring with a small change that can be delta-evaluated. We find that using this recombination operator is superior to mutation alone, at least on this problem, and is only slightly less efficient than uniform crossover at combining genetic material in the early stages of evolution. Thus, on this problem a genetic algorithm using the new recombination operator and delta-evaluation can conduct an effective search in comparable time to simulated annealing.

In further experiments reported by us in [8], the heuristics described here were also incorporated in a simulated annealing algorithm. Experimental results indicated that simulated annealing was still competitive with genetic algorithms on the routing problems, and in fact with a further heuristic for use in initialization, actually outperformed our GAs. However, investigations in a multiobjective formulation of the problem reported by us in [9], indicated that differences in performance between local search and population-based search on this problem were marginal. Further investigation of these matters, under a constrained mul-

tiobjective formulation of the routing problem is the subject of future work.

Acknowledgments

The first author would like to express his gratitude to BT Labs Plc., for their continuing sponsorship of his Ph.D.

References

[1] G. Bilchev and H. Olafsson. Comparing Genetic Algorithms and Greedy Heuristics for Adaptation Problems. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pages 458–463. IEEE Neural networks Council, 1998.

[2] L. Booker. Improving Search in Genetic Algorithms. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*. Lawrence Erlbaum, 1995.

[3] E. Collingwood, D. W. Corne, and P. Ross. Useful Diversity via Multiploidy. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pages 810–813. IEEE Neural Networks Council, 1996.

[4] D. Dasgupta and D. McGregor. A Structured Genetic Algorithm: The model and the first results. Technical Report IKBS-2-91, Dept. of Computer Science, University of Strathclyde, Glasgow, UK, 1992.

[5] E. Dijkstra. A note on two problems in connexion with graphs. *Numerischme Mathematik*, 1:269–271, 1959.

[6] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[7] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *Proceedings of the 2nd International Conference on Genetic Algorithms*. Morgan Kaufmann, 1987.

[8] J. D. Knowles and D. W. Corne. Evolutionary approaches to off-line routing in backbone communications networks. Technical Report RUCS/1999/TR/007/A, Department of Computer Science, University of Reading, Berkshire, UK, 1999.

[9] J. D. Knowles and D. W. Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Piscataway, NJ, July 1999. IEEE Service Center.

[10] J. W. Mann and G. D. Smith. A Comparison of Heuristics for Telecommunications Traffic Routing. In V. Rayward-Smith and I. Osman, editors, *Modern Heuristic Search Methods*. John Wiley and Sons Ltd., 1996.

[11] M. Munetomo, Y. Takai, and Y. Sato. An Adaptive Network Routing Algorithm Employing Path Genetic Operators. In *Proceedings of ICGA97*, pages 643–649, 1997.

[12] S. C. Narula and C. A. Ho. Degree-Constrained Minimum Spanning Tree. *Computers and Operations Research*, 7:39–49, 1980.

[13] M. J. Oates and D. W. Corne. Investigating Evolutionary Approaches to Adaptive Database Management Against Various Quality of Service Metrics. In T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature V*, pages 775–784. Springer, 1998.

[14] M. J. Oates and D. W. Corne. QoS-Based GA Parameter Selection for Autonomously Managed Distributed Information Systems. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence*, pages 670–674. Wiley, 1998.

[15] G. Syswerda. Uniform Crossover in Genetic Algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, 1989.

[16] A. Tanenbaum. *Computer Networks (third edition)*. Prentice-Hall Inc., 1996.

[17] R. J. Wilson and J. J. Watkins. *Graphs: an introductory approach: a first course in discrete mathematics*. John Wiley and Sons, Inc., 1990.

[18] J. Yen. Finding the K Shortest Loopless Paths in a Network. *Management Science*, 17, July 1971.

[19] Y. Yoshida and N. Adachi. A Diploid Genetic Algorithm for Preserving Population Diversity — pseudo-Meiosis GA. In Davidor, Schwefel, and Manner, editors, *Parallel Problem Solving from Nature III*, pages 36–45. Springer, 1994.