Discovering simple fault-tolerant routing rules using genetic programming

I.M.A. Kirkwood, S.H. Shami and M.C. Sinclair

Dept. of Electronic Systems Engineering, University of Essex,
Wivenhoe Park, Colchester, Essex CO4 3SQ.
Tel: 01206-872477; Fax: 01206-872900; Email: mcs@essex.ac.uk

Abstract

A novel approach to solving network routing and restoration problems using the genetic programming (GP) paradigm is presented, in which a single robust and fault-tolerant program is evolved which determines the near-shortest paths through a network subject to link failures. The approach is then applied to five different test networks. In addition, two multi-population GP techniques are tried and the results compared to simple GP.

1 Introduction

The aims of this paper are to demonstrate the principle of applying genetic programming (GP) [1] to find the shortest or near-shortest path route through simple networks subject to link failures, to assess the approach on a number of test networks, and to explore whether multi-population GP could provide improved results. Traditional centralised methods, such as the simple Dijkstra's Algorithm [2] can be used to identify the shortest path through a static network. Should links fail, Dijkstra's Algorithm could be re-applied to the faulty network and new shortest paths obtained. Alternatively, a distributed restoration algorithm, such as Grover's [3], could be applied. However, the single centralised program described in this paper, found by GP and composed of problem-specific functions, is not in competition with Dijkstra's Algorithm or any other traditional centralised routing algorithms, but rather was 'blindly' evolved to find the near-shortest paths in the given network subject to link failures, and thus is fault-tolerant and robust.

In Figure 1 a notation of the form `p(q)` is used to identify the links in a network where `p` is the link number and `q` is the length of that link. For example in Network 1 of Figure 1, Node 3 is connected to Node 4 via Link 6 whose length is 2 units. Now

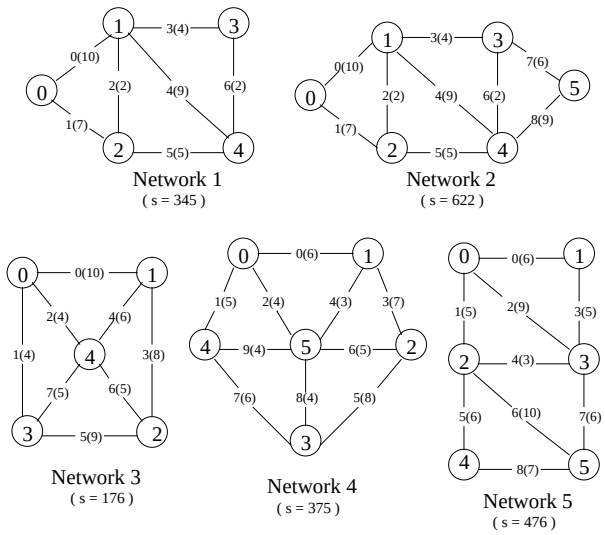


Figure 1: The test networks

consider the application of GP to, say, Network 1. Our problem then is to evolve a single program that can route information from any node back to Node 0, when there can be up to one link failure. The routes found will be the shortest or near-shortest paths. In other words, a single program is to be able to find simultaneously the shortest or near-shortest path trees through eight networks: the network when all links are fault-free, and seven networks when each of the seven links, in turn, is broken.

This is the first known application of GP to either a telecommunications network routing or restoration problem.

2 Problem Representation

GP usually describes its solution by means of LISP *s*-expressions which can be re-drawn as parse trees containing the problem-specific functions and terminals used. Our novel problem-specific function, used to

evolve a solution to the problem, is as follows:

```
(IF-CUR-GO W X Y Z) =
  X, if the Current Node is Node W,
     Node W and Node X are directly
     connected, and Node X has not been
     visited twice before.
  Y, if the Current Node is Node W,
     Node X and Node W are not directly
     connected, or Node X has been
     visited twice before.
  Z, if the Current Node is not Node W.
```

For example, in Network 1, if the information to be routed was at Node 3, the `Current Node`, the function `(IF-CUR-GO 3 1 4 2)` would route the information from Node 3 to Node 1 provided Node 1 has not been visited twice before, else it would be routed to Node 4. This next node then becomes the `Current Node`. Our Function Set [1] is then {`IF-CUR-GO`} and the Terminal Set comprises the nodes {`0,1,2,3,4`}. For our initial experiments on Network 1, a population of 500 programs and Koza's default parameters [1] were used, except that the maximum number of generations was 31. The driving force behind all evolutionary algorithms is the problem-specific fitness function. In this application, the fitness of a potential solution was obtained by calculating the lengths of the paths from Nodes 1, 2, 3 and 4 back to Node 0 in each of the eight possible networks (Network 1, and the seven variants of Network 1 with one link broken); clearly, in this case, the smaller the fitness measure, the better is the solution.

3 Initial Results

For our initial experiments on Network 1, different random seeds were used to obtain a different initial population on each occasion. At Generation 30, the fitness of the best individual was 356; this compares with 345 for the optimum solution.

To assess the results, we define a new fitness measure, *Dijkstra fitness* (f_D) which is the difference between the fitness of the solution found by our GP run (f) and the true optimum (found by Dijkstra's Algorithm [2]), which we call s, *i.e.* $f_D = f - s$. Thus zero Dijkstra fitness corresponds to the true optimum, indicating that the value found by that GP run is equal to s. In order to have a fitness measure which is more representative for comparing several networks, we use another measure which we call P_A (percent above the optimum) and define it as $P_A = (f_D/s) \times 100$. As an example, in our initial experiments on Network 1, the Dijkstra fitness of the best run was 11 and P_A was 3.2.

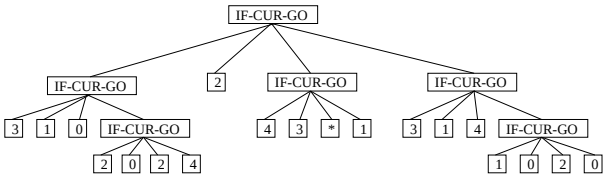


Figure 2: Diagram of fault-tolerant routing parse tree for Network 1

Applying some judicious manual editing to the LISP code, superfluous genetic material was removed from the parse tree of the best evolved individual from Generation 30. The resulting simplified program is at Figure 2, where the asterisk indicates an element of the parse tree that is never reached and so can equal any value.

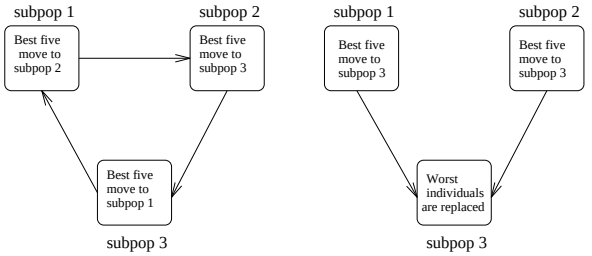
4 Robustness of Solution

The robustness of the best evolved solution can be examined by testing its ability to find near-shortest path routings in the same network, but this time with more than one link failure. In a seven-link network there are twenty-one possible networks resulting from breaking two links. Of these, two are discarded as invalid because they isolate at least one node; clearly, no program can solve that network problem. On examination, the best evolved program finds near-shortest paths for seventeen of the remaining nineteen valid networks. Here, the program has a success rate of 89.5% when almost 30% of the links have failed. Considering three link failures, there are twenty-one valid networks out of a possible thirty-five networks. The program finds near-shortest path routes for fourteen networks, a success rate of 66.7% when 43% of the network's links have failed. Clearly for Network 1, there are no valid networks when four or more links have failed.

5 Use of Multiple Populations

Having established proof of principle with our initial experiments, we decided to test our approach on a larger set of networks, as well as explore the use of multi-population GP. Our initial experiments were implemented using `sgpc` (v1.1) [4], a poorly documented and supported GP package. For our subsequent work we moved to the far superior `lilgp` (v1.02) [5]. This package is clearly documented, regularly upgraded, and supports multiple populations with arbitrary exchange topologies.

Premature convergence [6] is a problem that is



(a) Ring Architecture (three subpops) (b) Injection Architecture (three subpops)

Figure 3: Multi-population (two distinct techniques)

often faced in the standard uni-population GP (UGP) runs. One effective approach to deal with this problem is to employ multiple populations (MP) within each generation. This relates to a more realistic model of nature than a single large population. Multiple populations of genetic programs have been found to reduce processing time and also explore the search space better, although recent results indicate this is not always the case [6]. The uni-population GP technique is essentially sequential and entails high computational cost for maintaining genetic diversity based on similarity comparison. However, the MP technique maintains separate sub-populations which are allowed to evolve independently. This enables each sub-population to explore different parts of the search space, retain high fitness individuals and control how mixing occurs with other sub-populations. The MP technique can easily be ported to a multi-processor or distributed computing environment.

Two distinct MP techniques called MP-Ring Architecture (MP-R) and MP-Injection Architecture (MP-I) used here are explained with reference to Figure 3 [6]. In the ring architecture each sub-population chooses its five best individuals and sends it to the next sub-population in the ring. Each sub-population takes the individuals sent to it, and uses them to replace its five worst members. This is done after a specified number of generations. On the other hand, the injection architecture is a hierarchical arrangement. In the three sub-population example shown, sub-populations 1 and 2 both send their five best solutions (total ten) to sub-population 3. Sub-pupulation 3 replaces its ten worst individuals with those received from the donors.

6 Additional Test Networks

Four additional networks with varying topologies were arbitrarily constructed in order to further test the ability of GP to discover routing rules for near-

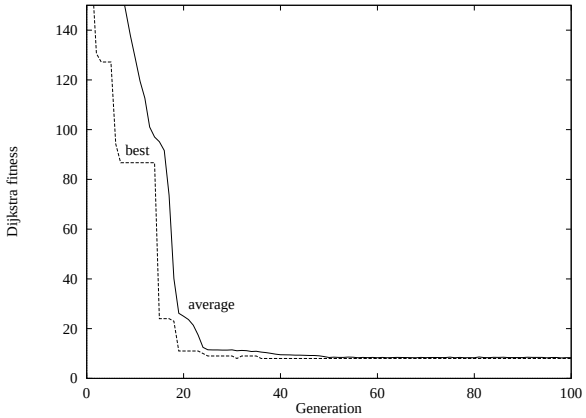


Figure 4: Best and average f_D vs. generations for Network 1

shortest paths. The objective is the same, that is to obtain near-shortest paths from every other node to Node 0 in each test network, which can be subject to one link failure. All five networks are shown in Figure 1. Beneath each network s refers to the actual sum of shortest path lengths for that network. The Function Set for all five networks remained {`IF-CUR-GO`} and the Terminal Set comprised the node numbers as before: either {`0,1,2,3,4`} or {`0,1,2,3,4,5`}, as appropriate.

7 Additional Results

For each network fifteen runs were carried out, five for each of the three techniques: namely UGP, MP-R and MP-I. Each run was given 100 generations for uniformity. A population size of 600 was used for all runs. All other control parameters used were Koza's default parameters [1].

Figure 4 shows a plot obtained for Network 1 using UGP and shows the best and average Dijkstra fitness (f_D) for each generation. Note that these additional runs, with a slightly larger population, but many more generations, have improved f_D to 8 and P_A to 2.3. Figure 5 shows the performance of the best runs using the three techniques on Network 2 with the same random seed. The delay in convergence using MP can be clearly seen.

Table 1 shows the performance of GP on the five networks. The paradigm performed extremely well on Network 3 with the GP result being only 0.57% above the optimum. It was under 3% for Network 1, and under 10% for Network 2. For Networks 4 and 5 the performance of GP is certainly poor.

Table 2 lists the comparative performance of the three techniques for each of the five test cases. A

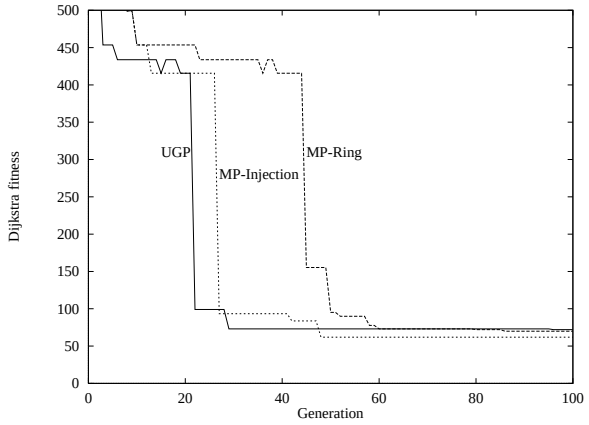


Figure 5: The three techniques on Network 2

Network Number	Target optimum (s)	Best Fitness obtained by GP	Best percent above the optimum (P_A)	Technique responsible
1	345	353	2.32	all three
2	622	684	9.97	MP-I
3	176	177	0.57	all three
4	375	603	60.8	all three
5	476	581	22.1	all three

Table 1: GP results for the five networks

95% confidence interval to cover the five runs within each technique is also computed for better comparison. It can be seen that MP-I has performed better in two networks (1 and 2), but there was little to distinguish the three techniques on the other networks. Overall, the results for Network 3 are nearly ideal, for Networks 1, 2 and 5 are reasonable, but for Network 4, none of the techniques brought the fitness within useful limits.

8 Conclusions & Further Work

The above results demonstrate that genetic programming (GP) has some limited potential, for small networks, to evolve near-optimal fault-tolerant routing rules which are robust enough to be able to solve a high proportion of multiple link failures. Overall, though, this approach lacks adequate performance even for modest-sized networks. In addition, our comparison of uni-population and multi-population GP is arguably inconclusive (*cf.* [6]).

For the future, the use of a context-free grammar [7] to bias the evolution of GP programs for routing holds promise and needs to be explored. Currently, however, the second and third authors are investigating evolving distributed software agents for telecommunications network routing and restoration, rather than centralised routing rules, in an effort to obtain more scalable results.

Network Number	Uni-population GP			Multipop-Ring			Multipop-Injection		
	Best Dijkstra fitness	Best P_A	Five run average P_A (95% C.I.)	Best Dijkstra fitness	Best P_A	Five run average P_A (95% C.I.)	Best Dijkstra fitness	Best P_A	Five run average P_A (95% C.I.)
1	8	2.32	2.41—3.16	8	2.32	2.31— 3.15	8	2.32	2.19— 2.91
2	70	11.25	11.19—11.64	70	11.25	11.14— 11.56	62	9.97	10.10—11.90
3	1	0.57	0.57	1	0.57	0.57	1	0.57	0.57
4	228	60.8	60.8	228	60.8	60.8	228	60.8	60.8
5	105	22.1	22.0 —22.8	105	22.1	22.0— 23.2	105	22.1	21.9— 22.8

Table 2: Comparison of the three techniques

References

[1] Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection* MIT Press, 1992.

[2] Gondran, M. & Minoux, M.: *Graphs and Algorithms* John Wiley & Sons, 1994.

[3] Grover, W.D.: *The Self-healing Network: A fast distributed restoration technique for networks using digital cross-connect machines* IEEE Global Conference on Communications 1987, pp.1090–1095.

[4] Tackett, W.A. & Carmi, A.: *Simple Genetic Programming in C* 1993.

[5] Zongker, D. & Punch, B.: *lil-gp 1.0 User's Manual* Michigan State University, 1995.

[6] Punch B., Zongker D. & Goodman E.: *The Royal Tree Problem, a Benchmark for Single and Multiple Population Genetic Programming* in Angeline, P.J. & Kinnear, K.E., Jr. (Eds) *Advances in Genetic Programming Volume 2* MIT Press, 1996

[7] Whigham, P.A.: *Inductive Bias and Genetic Programming* GALESIA'95, London, pp.461-466.

Acknowledgements

The initial experiments on Network 1 were undertaken by the first author as part of an MSc in Telecommunication and Information Systems and subsequently continued by the second author as part of a PhD project, both supervised by the third author. The first author was sponsored by the RAF, and the second by the Government of Pakistan.