Optical Mesh Network Topology Design using Node-Pair Encoding Genetic Programming

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

Abstract

Two variants of a new node-pair encoding genetic programming (GP) approach for optical mesh network topology design are presented. In addition, a new variant of the earlier connected-nodes encoding is described. Experimental work on 15- and 20-node networks demonstrates that both GP approaches can almost match the design quality of bit-string genetic algorithms, particularly for the larger network size investigated.

1 INTRODUCTION

Telecommunications is a vital and growing area, important not only in its own right, but also for the service it provides to other areas of human endeavour. Moreover, there currently seems to be a demand for an ever-expanding set of telecommunication services of ever-increasing bandwidth. One particular technology that has the potential to provide the huge bandwidths necessary if such broadband services are to be widely adopted, is multi-wavelength all-optical transport networks. However, the development of such networks presents a challenging range of difficult design and optimisation problems.

One such problem is mesh network topology design. In the general case, this starts with a set of node locations and a traffic matrix, and determines which of the node pairs are to be directly connected by a link. The design is guided by an objective function, often cost-based, which allows the 'fitness' of candidate networks to be evaluated. In the more specific problem of the topology design of multi-wavelength all-optical transport networks, the nodes would be optical cross-connects, the links optical fibres, and the traffic static. Suitable routing and dimensioning algorithms must be selected, with sufficient allowance for restoration paths, to ensure that the network would at least survive the failure of any single component (node or link).

In previous papers, Sinclair has applied a simple bit-string genetic algorithm (GA) (1995), a hybrid GA (1997) and, with Aiyarak & Saket, three different genetic programming (GP) approaches (1997) to this problem. In this paper, a new GP approach, inspired by edge encoding (Luke, 1996), is presented. It was hoped that this would provide better results than the best previous GP approach, and perhaps even prove competitive with GAs. The eventual aim of the author's current research into GP encoding schemes for mesh network topologies is to provide a more scalable approach than the inherently non-scalable encoding provided by bit-string GAs.

2 EC FOR TOPOLOGY DESIGN

Over the years, at least 30 papers have been published on evolutionary computation (EC) approaches to network topology design. Two of the earliest are by Michalewicz (1991) and Kumar *et al.* (1992). Michalewicz uses a two-dimensional binary adjacency matrix representation, and problem-specific versions of mutation and crossover, to evolve minimum-spanning-tree topologies for computer networks. Kumar *et al.* tackle three constrained computer network topology problems, aiming for maximum reliability, minimum network diameter or minimum average hop count. Their GA is bit-string, but uses problem-specific crossover, as well as a repair operator to correct for redundancy in their network representation.

For optical network topology design, in addition to the papers by Sinclair, mentioned above, there is the work of Paul *et al.* (1996) and Brittain *et al.* (1997). Both these groups of authors have addressed constrained minimum-cost passive optical network (PON) topology design for local access. However, while problem-specific representations are used by both, as well as problem-specific genetic operators by Paul *et al.*, only Brittain *et al.* provide full details of their algorithm. This employs a two-part chromosome comprising a bit-string and a permutation, with each part manipulated using appropriate standard genetic operators.

Other recent work of interest includes papers by Dengiz *et al.* on a hybrid GA for maximum all-terminal network reliability (1997); Ko *et al.*, who use a three-stage GA for minimum-cost design of computer network topology, routing and capacity assignment (1997); and Pierre & Legault, who employ a bit-string GA for computer mesh network design (1998).

3 PROBLEM DESCRIPTION

Given the locations of the n nodes (optical cross connects) and the static traffic requirements between them, the problem is to determine which of the $n(n-1)/2$ possible bi-directional links (optical fibres) should be used to construct the network. The number of possible topologies is thus $2^{n(n-1)/2}$.

The cost model used to guide the design was first developed by Sinclair (1995) for minimum-cost topology design of the European Optical Network (EON) as part of COST 239. It assumes static two-shortest-node-disjoint-path routing is used between node pairs, and that a reliability constraint is employed. The latter is to ensure that there are two, usually fully-resourced, node-disjoint routes between node pairs, thereby guaranteeing that the network will survive the failure of any single component. The intention is to approximate the relative contribution to purchase, installation and maintenance costs of the different network elements, while ensuring that the model is not too dependent on the details of the element designs, nor too complex for use in the 'inner loop' of a design procedure.

4 PREVIOUS APPROACHES

Two previous attempts at optical mesh topology design using the same cost model are outlined below. An additional attempt with a hybrid GA (Sinclair, 1997) has been excluded from consideration due to the highly problem-specific nature of its encoding and operators.

In 1995, Sinclair used a bit-string GA to tackle two variants of the EON: a small illustrative problem consisting of just the central 9 nodes, as well as the full network of 20 nodes. The encoding simply consisted of a bit for each of the $n(n-1)/2$ possible links. Clearly, this representation scales poorly as problem size increases, as the bit string grows $O(n^2)$, rather than only with the number of links actually required, m, *i.e.* $O(m) \approx O(n)$ provided node degree remains approximately constant with network size. The genetic operators were single-point crossover (probability 0.6) and mutation (probability 0.001); and fitness-proportionate selection (window-scaling, window size 5) was used. The GA was generational, although an elitist strategy was employed. For the full EON, with a population of 100 and ten runs of less than 48,000 trials each, a network design was obtained with a cost (at 6.851×10^6) which was some 5.1% lower than an earlier hand-crafted design. In addition, the GA network design was of superior reliability, at least in terms of the reliability constraint.

More recently, Aiyarak, Saket & Sinclair (1997) described three different approaches to applying GP to the problem. The most successful of these, connected nodes (CN), encodes the design as a program describing how the network should be connected. Only one terminal and one function are required. The terminal is the ephemeral random integer constant ($\mathcal{R}$), over the range of n node identification numbers (ids). The function is `con`, which takes two arguments, representing the ids of two nodes that are to be connected in the network topology represented by the program[1]. As each `con` function must also provide a return value, it simply returns its first argument. However, if its two id arguments are equal, it does nothing apart from returning their value. The program tree is evaluated depth-first: children **before** parents. For example, Fig. 1 shows a small target network and Fig. 2 a corresponding 'hand-crafted' connected-nodes GP tree. Executing the tree would connect those node pairs indicated above each of the `con` functions: `(4,3)`, `(1,4)`, `(1,3)`, *etc.*, resulting in the desired network. Clearly, with this representation, minimum program size grows only with the number of links ($O(m) \approx O(n)$), as only a single `con` and at most two terminals are required for each node pair connected. In (Aiyarak, 1997), the only genetic operator used was crossover, and tournament selection was employed. Experimental results were obtained for both the 9 central nodes, as well as the full EON, establishing the superiority of the CN approach over the two other GP encoding methods. In addition, the network design cost obtained with CN (6.939×10^6) was only some 1% above Sinclair's earlier GA result (1995). However, the computational burden of CN was far greater: the best design was obtained using two runs of 500,000 trials each on a population of 1,000 individuals.

[1] In (Aiyarak, 1997), this function was called `connect2`

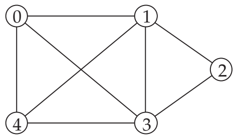

Figure 1: Target Network

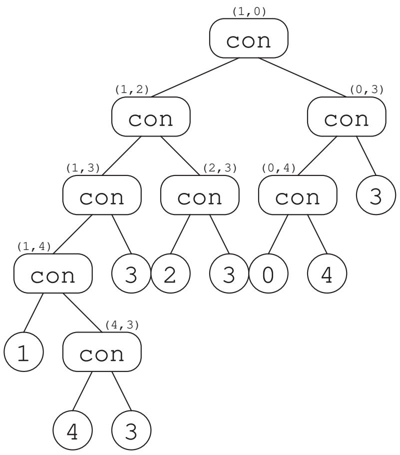

Figure 2: CN Program for Target Network

5 NODE-PAIR ENCODING GP

The two node-pair encoding (NP) GP approaches proposed in this paper are based on an earlier edge encoding for graphs described by Luke & Spector (1996). Their approach evolved a location-independent topology, with both the number of nodes and their interconnections specified by the GP program. Here, however, the number of nodes is fixed in advance, and the GP program is only required to specify the links.

As in the CN approach, the program again describes how the network should be connected. However, for NP, the program tree is evaluated top-down: children **after** parents. The functions operate on a node pair (represented by two node ids, `a` and `b`), then pass the possibly-modified node-pair to each of their children. In a similar way, the terminals simply operate on the node pair passed to them. Overall execution of the program tree commences with `(a,b)` = `(0,0)`. It should be noted that this use of a current node pair is similar to the concept of the current edge in Luke & Spector's work (1996), but without any structure equivalent to their node stack.

Two different variants of node-pair encoding are proposed here (NP1 and NP2); the difference is due to the arities of their functions/terminals. These are given, for both variants, in Table 1.

The function/terminal `add` adds a link between the

Table 1: NP Function/Terminal Sets

	ARITY		
ABBR.	NP1	NP2	DESCRIPTION
`rev`	1	1	reverse: `(a,b)` = `(b,a)`
`da`	1	2	(decrement `a`) mod n
`ia`	1	2	(increment `a`) mod n
`ia2`	1	2	(increase `a` by 2) mod n
`ia4`	1	2	(increase `a` by 4) mod n
`ia8`	1	2	(increase `a` by 8) mod n
`dbl`	2	2	double: pass current node pair to both children
`tpl`	3	3	triple: pass current node pair to all three children
`add`	1	0	add link `(a,b)`
`cut`	0	0	cut link `(a,b)`
`nop`	0	0	do nothing

current node pair `(a,b)`, provided `a` $\neq$ `b`; whereas terminal `cut` removes the link between the current node pair, if there is one. To allow the current node pair to change, thus moving the focus of program execution to a different point in the network, five of the functions (`da`, `ia`, `ia2`, `ia4`, `ia8`) modify the value of variable `a`. The choice of 1, 2, 4 and 8 for the values by which `a` may be increased was motivated by minimum-description length considerations. By providing the reverse function (`rev`), similar modifications can be made, in effect, to variable `b`. The double (`dbl`) and triple (`tpl`) functions allow operations on node pairs that are numerically close together to be accomplished using a smaller depth of tree. For example, in Fig. 3, the `tpl` in the lower left of the diagram, which refers to node pair `(2,1)`, enables links `(2,1)`, `(3,1)` and `(4,1)` to be added by its subtrees. Without this `tpl`, an equivalent subtree based on just `add`s, `ia?`s and `nop`s would be two levels deeper. Finally, terminal `nop` does nothing to its node pair, thus allowing a program branch to terminate without either adding or removing a link.

In NP1, `add` is a function and has a single child, whereas in NP2 it is a terminal. Further, in NP1, the functions that modify variable `a` all have just one child, whereas in NP2, they have two. The effect of these differences is to encourage taller, narrower program trees in NP1, with further operations following the addition of a link, and shallower, broader trees in NP2, with link addition ending a program branch. This is illustrated by the 'hand-crafted' program trees, for the target network of Fig. 1, given in Fig. 3 and 4 using NP1 and NP2 respectively. In both diagrams, the current node pair is indicated above each of the `add` function/terminals. It should be noted that the

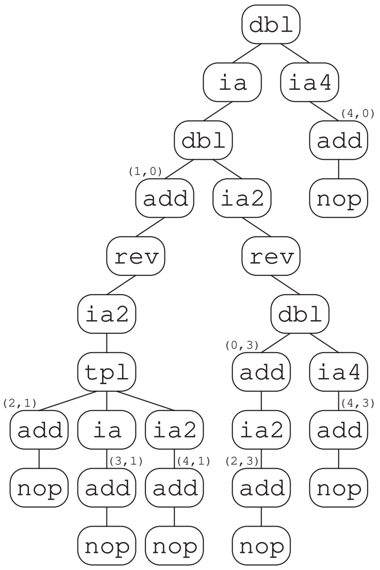

Figure 3: NP1 Program for Target Network

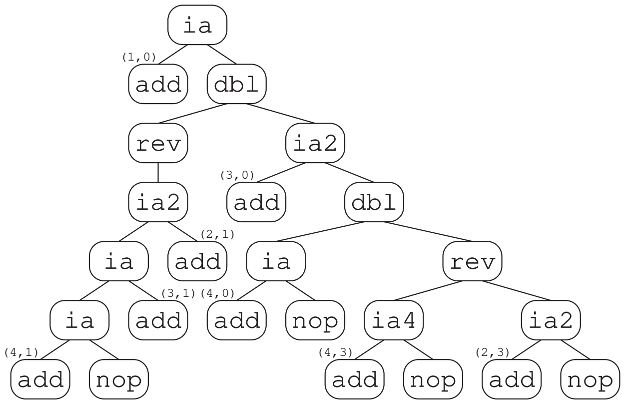

Figure 4: NP2 Program for Target Network

tree in Fig. 3 has a depth of 10 and uses 28 program nodes, whereas that in Fig. 3 is only 7 levels deep and uses just 24 function/terminals.

6 EXPERIMENTAL RESULTS

For the relative assessment of the different approaches to optical mesh network topology design, it was decided to use the full 20-node EON (Sinclair, 1995) and five additional network design problems. For the latter, the initial node locations and traffic requirements were generated using the approach described by Griffith *et al.* (1996), although further modified to ensure reasonable node separations. Each network has 15 nodes, covers a 1,000 km × 1,000 km area and carries an overall traffic of 1,500 Gbit/s.

The bit-string GA developed by Sinclair (1995), and described in §4 above, is referred to here as GA1 (with a maximum of 15,000 trials) when used on the five 15-node problems, and as GA4 (with a maximum of 50,000 trials) on the EON (Table 2). In addition, in an attempt to improve on Sinclair's results, both a higher mutation rate and a larger population size were also tried for both the 15-node problems and the EON. From a few trials runs, reasonably successful algorithms with a higher mutation rate (GA2) and a larger population size (GA3) were discovered for the 15-node problems. Also, for the EON, algorithms with an increased mutation probability of 0.01, and increased population sizes of 200, 500 and 1,000 (the latter three still with mutation probability 0.001) were tried, without good results, for runs of up to 1,000,000 trials. However, a reasonable algorithm was found with mutation probability 0.002 (GA5).

Table 2: GA Parameters

ALG.	POPULATION SIZE	MAXIMUM TRIALS	MUTATION PROB.
GA1	100	15,000	0.001
GA2	100	25,000	0.01
GA3	700	210,000	0.001
GA4	100	50,000	0.001
GA5	100	100,000	0.002

Table 3: Results for EON

ALG.	BEST ($\times 10^6$)	MEDIAN ($\times 10^6$)	SIGN. LEVEL	(%)
GA4	6.851	6.926	GA5<GA4	—
GA5	6.856	6.891	CN2<CN1	1.15
CN1	6.961	7.002	NP2<NP1	1.15
CN2	6.888	6.930	GA5<CN2	—
NP1	6.898	6.962	GA5<NP2	—
NP2	6.862	6.900	NP2<CN2	—

To allow a statistical comparison of the genetic algorithms, GA1, GA2 and GA3 were applied to all five 15-node test problems, plus GA4 and GA5 to the EON. In every case, ten runs were made with different pseudo-random number seeds. GENESIS v5.0 (Grefenstette, 1990) was used for the implementation. A non-parametric median test (Sprent, 1992) was applied to establish if there were significant differences in the medians from the different GAs. The results for the EON are given in Table 3; both the best and median of each set of runs is recorded, as well as the significance levels for the median differences.

For the 15-node problems, GA3 is the best algorithm, providing not only all five of the best individual runs, but also the best median for four of the problems (with

Table 4: Comparative Results for Problems 1–5

	GA3		CN2		NP2		SIGN. LEVEL (%)		
	BEST	MEDIAN	BEST	MEDIAN	BEST	MEDIAN			
PROB.	($\times 10^6$)	($\times 10^6$)	($\times 10^6$)	($\times 10^6$)	($\times 10^6$)	($\times 10^6$)	GA3<CN2	GA3<NP2	NP2<CN2
1	4.935	4.940	4.944	4.959	4.937	4.950	0.05	—	—
2	4.737	4.744	4.743	4.750	4.743	4.752	1.15	1.15	—
3	4.404	4.407	4.404	4.416	4.404	4.414	—	—	—
4	4.587	4.589	4.589	4.597	4.590	4.595	1.15	0.05	—
5	4.416	4.418	4.417	4.428	4.417	4.436	1.15	0.00	—

very highly significant differences). In addition, for the EON (Table 3), GA5 provided the better median (not a significant difference) although the best individual run still used GA4. However, it should be noted that both these improvements over Sinclair's GA1/GA4 (1995) were achieved at the cost of a larger number of trials for both network sizes (Table 2).

For the connected-nodes GP approach, the decision was taken to use a tournament size of 4, rather than the over-large value of 30 used in previous work (Aiyarak, 1997). In addition, as well as Aiyarak *et al.*'s original function set, here designated CN1, a second variant was introduced, CN2. The latter, in addition to the `con` function, also provides a `dis` function. This takes two node id arguments, and removes the link between them, if there is one. For the 15-node problems, the population size was 700 and the maximum number of trials 210,000; for the EON, 1,000 and 500,000, respectively (after Aiyarak, 1997). GP parameters are given in §4 above or followed Koza (1992).

Both CN1 and CN2 were applied to all five 15-node test problems and the EON. As with the GAs, ten runs were made with different seeds. The implementation used `lil-gp` v1.02 (Zongker, 1995). The results for the EON are again given in Table 3. On the 15-node problems, the CN2 connected-nodes encoding provided the best individual runs in three of the five networks, and the best median in three (very highly and highly significant differences). For the EON, CN2 also provided both the best individual run and the best median (highly significant difference). Thus including the new `dis` function, whose role in **removing** links may seem counter-intuitive, has resulted in a marked improvement in the results. This can perhaps be attributed to increased redundancy in the encoding, allowing greater freedom in program tree composition.

For the node-pair encoding GP approach, exactly the same parameters were used as for the corresponding CN runs. The results of applying both NP1 and NP2 encodings to the EON are recorded in Table 3. On the 15-node problems, the NP2 node-pair encoding provided the best individual runs in four of the five networks, and the best median in one (highly significant difference). For the EON, NP2 also provided both the best individual run and the best median (highly significant difference). These differences perhaps arise from the NP2 encoding's ability to use shallower trees, giving an advantage in tree composition, as the usual maximum tree depth of 17 was imposed on all runs (Koza, 1992).

The results of the three leading approaches, GA3, CN2 and NP2, are summarised in Table 4 for the 15-node problems, and in Table 3 for the EON. For the smaller networks, GA3 has shown itself to be the best approach, both in terms of individual runs and median differences. Nevertheless, one of the five best individual runs were performed using NP2 (Network 3), and for the other four networks the NP2 results were very close to those of GA3. For the larger EON, however, there is no overall best algorithm in terms of median differences, although both GA5 and NP2 performed well. The best individual run is still that from (Sinclair, 1995), using GA4. However, execution time for all the different approaches was almost entirely determined by the number of trials, due to the time-consuming fitness assessment. Consequently, while all the runs on the 15-node networks required approximately the same time, the GP runs on the EON took some five times longer than the GA.

7 CONCLUSIONS

In this paper two variants (NP1 and NP2) of node-pair encoding genetic programming (GP) for optical mesh network topology design have been proposed. In addition, a new variant (CN2) of the earlier connected-nodes encoding has been described. Experimental results have shown that both NP2 and CN2 are comparable in design quality to the best bit-string genetic algorithms (GA) developed by the author, particularly for the larger network size examined, although in that case requiring much greater computational effort. While node-pair encoding has only been applied to optical network design here, it may also be useful for graph construction in other applications, such as the interconnection of artificial neural networks.

Future work with node-pair encoding could include investigating its sensitivity to the choice of cost model, starting node pair or GP parameters, such as tree depth. In addition, the author hopes to develop further encodings that are both computationally efficient and scale well with network size. In particular, it is anticipated that removing or reducing dependency on explicit node ids will result in more powerful encodings. Also, for regular or near-regular networks, ADFs (Koza, 1994) may well provide a mechanism to capture and succinctly express network regularities.

Acknowledgements

The node-pair encodings used in this paper are based on an earlier encoding developed by Christos Dimitrakakis as part of an MSc in Telecommunication & Information Systems, supervised by the author.

The author is grateful to Rob Smith (University of the West of England) and Brian Turton (Cardiff University) for their encouragement to further improve the bit-string GA results obtained using GA1/GA4.

References

Aiyarak, P., Saket, A.S. & Sinclair, M.C. (1997), "Genetic programming approaches for minimum cost topology optimisation of optical telecommunication networks", Proc. 2nd IEE/IEEE Intl. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97), Glasgow, UK, September 1997, pp. 415–420

Brittain, D., Williams, J.S. & McMahon, C. (1997), "A genetic algorithm approach to planning the telecommunications access network", Proc. 7th Intl. Conf. on Genetic Algorithms (ICGA'97), July 1997, Michigan State University, East Lansing, Michigan, USA, pp. 623–628

Dengiz, B., Altiparmak, F. & Smith, A.E. (1997), "Local search genetic algorithm for optimal design of reliable networks", *IEEE Transactions on Evolutionary Computation* v1 n3, pp. 179–188

Grefenstette, J.J. (1990), *A User's Guide to GENESIS, Version 5.0*

Griffith, P.S., Proestaki, A. & Sinclair, M.C. (1996), "Heuristic topological design of low-cost optical telecommunication networks", Proc. 12th UK Performance Engineering Workshop, Edinburgh, UK, September 1996, pp. 129–140

Ko, K.-T., Tang, K.-S., Chan, C.-Y., Man, K.-F. & Kwong, S. (1997), "Using genetic algorithms to design mesh networks", *Computer* v30 n8, pp. 56–61

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

Koza, J.R. (1994), *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press

Kumar, A., Pathak, R.M., Gupta, M.C. & Gupta, Y.P. (1992), "Genetic algorithm based approach for designing computer network topology", Proc. 21st ACM Annual Comput. Sci. Conf., Indianapolis, USA, pp. 358–365

Luke, S. & Spector, L. (1996), "Evolving graphs and networks with edge encoding: Preliminary report", Late Breaking Papers at the Genetic Programming 1996 Conference, Stanford, USA, 1996, pp. 117–124.

Michalewicz, Z. (1991), "A step towards optimal topology of communications networks", Proc. Conf. on Data Structures and Target Classification, Orlando, Florida, USA, April 1991, pp. 112–122

Paul, H., Tindle, J. & Ryan, H.M. (1996), "Experiences with a genetic algorithm-based optimization system for passive optical network planning in the local access network", Proc. Broadband Superhighway (NOC'96-I), Heidelberg, Germany, June 1996, pp. 105–112

Pierre, S. & Legault, G. (1998), "A genetic algorithm for designing distributed computer network topologies", *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics* v28 n2, pp. 249–258

Sinclair, M.C. (1995), "Minimum cost topology optimisation of the COST 239 European optical network", Proc. 2nd Intl. Conf. on Artificial Neural Networks and Genetic Algorithms (ICANNGA'95), Alès, France, April 1995, pp. 26–29

Sinclair, M.C. (1997), "NOMaD: Applying a genetic-algorithm/heuristic hybrid approach to optical network topology design", Proc. 3rd Intl. Conf. on Artificial Neural Networks and Genetic Algorithms (ICANNGA'97), University of East Anglia, Norwich, UK, April 1997, pp. 299–303

Sprent, P. (1992), *Applied Nonparametric Statistical Methods, 2nd Ed.*, Chapman & Hall

Zongker, D. & Punch, B., (1995) *lil-gp 1.0 User's Manual*