Proceedings of the 1996 ACM/SIGAPP Symposium on Applied Computing, February 18-20, 1996, Philadelphia, PA., Pages 82-87, ACM Press.

FINDING INVESTIGATOR TOURS IN TELECOMMUNICATION NETWORKS USING GENETIC ALGORITHMS *

Cory J. Hoelting, Dale A. Schoenefeld and **Roger L. Wainwright**

Department of Mathematical and Computer Sciences
The University of Tulsa
600 South College Avenue
Tulsa, OK 74104-3189
http://www.utulsa.edu
[hoeltc,dschoen,rogerw]@euler.mcs.utulsa.edu

Key Words: fault detection, telecommunication networks, genetic algorithms, algorithm complexity, graph theory

ABSTRACT

We review and analyze a formal problem of fault detection in point to point telecommunications networks that are modeled as undirected graphs. Two heuristics, one deterministic and the other an application of genetic algorithm techniques, are tested on several sample graphs. The performance of these heuristics is compared and interpreted. The genetic algorithm technique consistently outperforms the deterministic technique on our test data sets.

INTRODUCTION

Fault detection is an important aspect of fault management in which a network manager tries to determine whether there is a fault in a data network and, if so, to determine the location of the fault. Recently, many problems related to network management have been formally stated and partially solved [1, 9, 11, 12, 13].

We assume that a network is modeled as an undirected graph $G = (V,E)$ with a set of nodes, V, which corresponds to the active elements of the network, and a set of edges, E, which corresponds to the passive elements interconnecting the active elements. Active elements are those elements which have their own computing power and are able to send their own status. Examples of active elements include hosts, bridges and routers. Passive components, such as transmission lines, are those which do not have the ability to send their own status. One of the active elements is designated as the network manager. A fault in a network corresponds to the failure of a node or an edge in our model. Paul and Miller present two basic approaches, along with variations, for solving the fault location problem [9]. They also analyze the performance and effectiveness of their schemes.

In the incremental multicasting scheme (IMS) discussed in [9], the network manager sets up multicast spanning trees, each rooted at the network manager node and including all other nodes. After selecting one of the spanning trees, the network manager node (NM) multicasts a status request message to all of the nodes at distance i, where i ranges from one to the depth of the spanning tree. For each i, and for each node in the tree at a depth of i, the network manager receives a reply of 1 (the node is alive and a reply comes back) or a reply of 0 (no reply comes back). If the NM receives a 1, the NM knows that the node is alive and so are the links along the path in the multicast tree from the NM to the node in question. However, if the reply is 0, either the node in question is dead or the last link on the path from the NM to the node is dead. In either case, the NM uses an alternate spanning tree to probe the same node. If an undirected graph has no bridge edges and has a single fault (either a node or an edge), their IMS approach with a suitable

*Research partially supported by OCAST Grant AR2-004 and Sun Microsystems, Inc.

collection of multicast spanning trees is guaranteed to locate the fault [9]. An edge in a graph is a bridge edge if its removal disconnects the graph.

In the Investigator Propagation Scheme (IPS) described in [9], the NM sends a special message encapsulating a program, which is called the investigator, along a precomputed route beginning and ending at the NM. The function of the investigator, upon reaching a node along the precomputed route, is to activate the node to collect the status of all adjacent links and all adjacent nodes and to report to the NM if any of them are down. As with the IMS technique, the absence of a reply may indicate a problem with either the link or with the adjacent node. Again, assuming only one fault and no bridge edges, Paul and Miller state that the ambiguity can be resolved [9]. The investigator continues to the next node and repeats the process. The precomputed tour is called an Investigator Tour and determining the tour provides a solution to the Investigator Tour Problem.

Finally, hybrid and combined procedures are proposed in [9] that utilize notions from the IMS and the IPS as a way to overcome the poor response time of the IPS while reducing the network load generated by the IMS.

The objective of this research is to apply a genetic algorithm to find optimal or near optimal solutions to the Investigator Tour Problem (ITP).

THE ITP PROBLEM

Using the terminology presented by Liu [8], we let $G = (V,E)$ be an undirected graph without self loops and without multiple edges. A *path* in G from v_{i_1} to v_{i_k} is a sequence of nodes $(v_{i_1}, v_{i_2}, ..., v_{i_k})$ where consecutive nodes are adjacent. A *path* is *simple* if it does not include the same edge twice. A *path* is *elementary* if it does not include the same node twice. A *tour* is a path for which the terminal node coincides with the initial node. A *tour* is *simple* if it does not include the same edge twice. A *tour* is *elementary* if it does not include the same node twice (i.e., other than the initial node coinciding with the terminal node). A *tour* in G *covers an edge* if at least one of the endpoints of the edge is in the tour. A subset V' of V is a *vertex cover* if each edge in G has at least one of its incident nodes in V' [4]. As noted above, an edge in a graph is a *bridge edge* if its removal disconnects the graph.

The Investigator Tour Problem, assuming an undirected graph, G, with edge lengths, is to find a minimum length tour that covers each edge of G. Note that such a tour is not necessarily a postman tour because a postman tour must include each node. All of our research assumes that each edge in G has length equal to one, corresponding to the notion of a "hop" in a telecommunication network, and that G has no bridge edges. The ITP was shown to be NP-hard in [9].

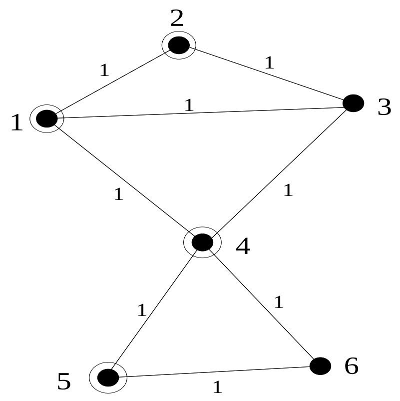

Figure 1:

A deterministic heuristic for the ITP is the technique proposed in [9]. The results obtained after implementing this heuristic are later compared to the results obtained by implementing our genetic algorithm technique. The steps of the deterministic heuristic follow:

1. Find a vertex cover, V', of $G = (V,E)$.
2. Build a new complete graph, $G' = (V', E')$, with edge lengths. V' is the set of nodes in the vertex cover of G. The length of any edge, say $e' = (u', v')$, in G' is the length of the shortest path from u' to v' in G.
3. Find a near optimal TSP tour of G'. That is, find a near optimal solution to the Traveling Salesman Problem (TSP) for the complete graph, G'.
4. Map the TSP tour in G' to a tour in G by letting each edge, e', of the TSP tour in G' expand to the shortest path in G between the corresponding end nodes of e'.

We illustrate the result of applying each of the above four steps of the deterministic ITP heuristic, as implemented in our research, to the graph $G = (V,E)$ presented in Figure 1.

There may be several vertex covers for a graph. For example, in Figure 1, $V' = \{1, 2, 4, 5\}$ is a vertex cover for G. $V' = V$ and $V' = \{2, 3, 4, 5\}$ are two other vertex covers. In our telecommunications application, it would seem to be advantageous to find a minimal vertex cover. Finding a vertex cover of minimum cardinality is itself NP- complete [4]. The specific heuristic that we use to find a minimal or near minimal vertex cover is to repeatedly select a node of highest degree and remove all of its incident edges. There are vertex cover heuristics that are known to have a ratio bound of 2 [4]. Although our heuristic does not have a ratio bound of 2, the degree considerations seem to be advantageous in our context.

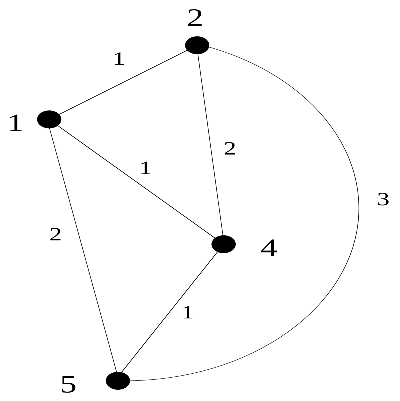

Figure 2:

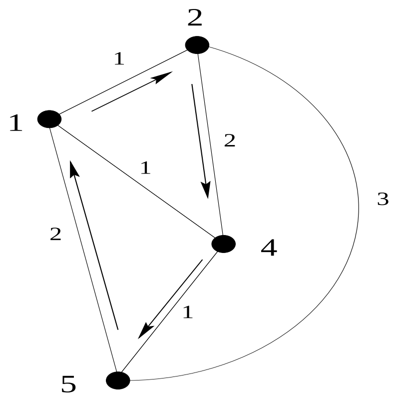

Figure 3:

Application of our heuristic to Figure 1 results in the selection sequence 4, 1, 2, and, then, 5, when ties are broken by selecting the node with the smaller label. The resulting vertex cover is $V' = \{1, 2, 4, 5\}$.

Given that the length of each edge in $G = (V,E)$ is assigned to be 1, the complete graph, $G' = (V', E')$, that results when the vertex cover, V', is selected to be $\{1, 2, 4, 5\}$ is presented in Figure 2.

The TSP is also NP-complete and has been used to benchmark many problem solving techniques in combinatorial optimization [2, 4]. We use the classical deterministic nearest neighbor heuristic and, when applied to Figure 2 with node 1 as the initial node, produces the TSP tour given by (1,2,4,5) as illustrated in Figure 3.

The TSP tour in G', illustrated in Figure 3, maps, at least for one strategy of breaking ties, to the investigator tour given by (1,2,3,4,5,4,1) and is illustrated in Figure 4.

We refer to the entire four step deterministic ITP heuristic as simply the *Vertex Cover Nearest Neighbor* technique (VCNN).

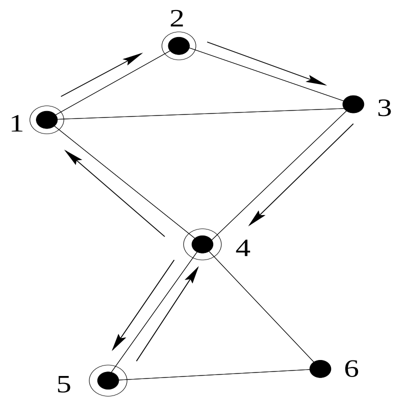

Figure 4:

THE MITP PROBLEM

For a graph with no bridge edge that contains at most one fault, either at an active node or along a passive edge, the objective of the IPS in Paul and Miller [9] is (a) to determine that there is no fault, or, otherwise, (b) to determine the location of the only fault. The IPS scheme presented in [9] for finding the location of a fault relies on the fact that one can reverse the direction of travel along the investigator tour. However, we can show, by using an example only slightly more complex than the example in Figure 1, that reversing the direction of travel is not necessarily successful in locating a fault if the investigator tour contains repeated edges. An example is an investigator tour that is made up of two simple cycles that are connected by a single edge. This edge must be repeated to form the investigator tour. Furthermore, if the investigator encounters a fault as it is attempting to cross the repeated edge from a node in one of the simple cycles, then reversing direction along the tour will bring the investigator back to the same location in the same cycle. The investigator will be unable to determine whether the fault is located along the repeated edge or at the incident node in the other simple cycle.

Hence, the Modified Investigator Tour Problem (MITP), still assuming an undirected graph, G, with edge lengths, is to find a minimal length tour that is simple and that covers each edge of G. Since a solution to the MITP problem requires a simple tour, the MITP problem is harder than the ITP problem. As with the ITP problem, all of our research with the MITP problem assumes that each edge in G has length equal to one and that G has no bridge edges.

The objective of a tour being minimal and the objective of a tour being simple often conflict with each other. Most of our research seeks a solution to the MITP problem. That is, we attempt to find a tour that is necessarily simple, and, among the simple tours, the one that has minimal length. However, our genetic algorithm is very flexible and provides reward and penalty mechanisms to attach either higher or lesser priority to each of the conflicting objectives.

GENETIC ALGORITHMS

Several researchers have investigated the benefits of solving combinatorial problems using genetic algorithms. Davis [5, 6], Goldberg [7], Rawlins [10], and Corcoran and Wainwright [3] provide an excellent indepth study of genetic algorithms.

The genetic algorithm operates on a fixed size population of chromosomes. In the context of combinatorial optimization, a chromosome is a string of genes that represents an encoding of a candidate solution. An allele is a value assumed by a gene. Associated with each chromosome is a fitness value which is found by evaluating a fitness function at the chromosome. The fitness function is designed so that chromosomes representing better candidate solutions will have better fitness. It is the fitness of a chromosome that determines its ability to survive and produce offspring. The genes of a chromosome may be bits, integers, floating point numbers, or instances of some other primitive data type. This research work uses an order based genetic algorithm where a chromosome is a permutation of integers. The genetic algorithm implementation used in this research is LibGA [3].

GENETIC ALGORITHM ENCODING

As stated previously, the ITP for a graph G is NP-hard. Noting that step 1 and step 3 in the VCNN are each NP-complete, one strategy for using a genetic algorithm to find near optimal solutions to the ITP problem is to apply the GA technique to determine a near optimal vertex cover in step 1 and, then, to apply a good deterministic heuristic to find a near optimal solution to the TSP problem in step 3. Alternately, one could apply a good deterministic heuristic to find a near optimal solution to the vertex cover problem in step 1 and, then, apply a GA technique to find a near optimal solution to the TSP problem in step 3. We did not investigate either of these approaches, but, rather, we designed a representation scheme that directly determines an investigator tour or a modified investigator tour from a permutation of the integers from 1 to N, where N is the number of nodes in G. The decoding algorithm for a chromosome is:

```
Let the first allele in the chromosome
be the current node.

Mark the edges incident to the current
node as being covered.

Repeat
  Let the next node be the next allele
  in the chromosome that has edges incident
  upon it that have not been covered;

  Proceed to this node using the shortest
  path (consistent with allowing or not
  allowing repeated edges), while marking
  the newly encountered edges as covered
  at each node along the way;

until all edges are covered.

Return to the initial node via the shortest
path.
```

We illustrate our representation scheme by decoding the chromosome given by (4,2,3,6,5,1) to obtain an investigator tour in the context of the graph in Figure 1. The NM node is assumed to be at the first allele in the chromosome, but could be relocated at any node in the resulting tour. We start at node 4 and record that four new edges are covered. Then the chromosome dictates that we traverse the graph to node 2 by the shortest path. We go from node 4 to node 1 along edge (4,1) and record that two new edges, namely (1,3) and (1,2), are covered. Next, we go from node 1 to node 2 along edge (1,2). This causes one new edge, namely (2,3), to be covered. At this point, all of the edges incident upon node 3 have been covered. Therefore, we skip node 3. From the chromosome, it is determined that the investigator must move to node 6. Using the shortest path, we go through nodes 3 and 4. Notice that at nodes 3 and 4 no new edges are covered, and, at node 6, one new edge, namely (6,5), is covered. At this point, all of the edges are covered and we can return to the initial node, in this case node 4. If we are enforcing no repeated edges, the shortest path between node 6 and node 4 is via node 5. Otherwise we would go back to node 4 directly from node 6. Thus, the tour that we obtain from this chromosome that requires no repeated edges is (4,1,2,3,4,6,5,4). The tour derived from this chromosome while allowing repeated edges is (4,1,2,3,4,6,4). Note that, when repeated edges are prohibited, not every chromosome represents a feasible tour. That is, the tour specified by the chromosome may not be able to return to the initial node or even cover all of the edges without traversing an edge twice.

GENETIC OPERATORS

The crossover operator used in this research is MX1, a member of the MX family of crossover operators introduced by Blanton and Wainwright [2]. Associated with a problem instance is a global precedence vector (GPV) which is used during the application of the MX1 operator. In our case, the global precedence vector consists of the nodes in decreasing order of degree. From Figure 1, it is clear that deg(1) = 3, deg(2) = 2, deg (3) = 3, deg(4) = 4, deg(5) = 2, and deg(6) = 2, where deg(i) is the degree of node i. Using this information, the global precedence vector (GPV) for this problem is GPV = (4,1,3,2,5,6). To illustrate the MX1 crossover operator, consider the two chromosomes, (1,5,3,6,4,2) and (4,6,5,3,1,2), as selected parents. The "*" indicates the allele currently under consideration.

```
                       *
Step 1.      Parent 1: 1  5  3  6  4  2
             Parent 2: 4  6  5  3  1  2

Consider the first allele in each parent.
Node 4 comes before node 1 in the GPV so 4
is selected.  Parent 1 is fixed by exchanging
alleles 1 and 4.  Now consider the second
allele.

                          *
Step 2.      Parent 1: 4  5  3  6  1  2
             Parent 2: 4  6  5  3  1  2
             Child:    4  x  x  x  x  x

Node 5 comes before node 6 in the GPV so 5
is selected.  Parent 2 is fixed by exchanging
alleles 5 and 6.  Now consider the third
allele.

                             *
Step 3.      Parent 1: 4  5  3  6  1  2
             Parent 2: 4  5  6  3  1  2
             Child:    4  5  x  x  x  x

Node 3 is before node 6 in the GPV so 3 is
selected.  Parent 2 is fixed by exchanging
alleles 3 and 6.  Consider the next allele.

                                *
Step 4.      Parent 1: 4  5  3  6  1  2
             Parent 2: 4  5  3  6  1  2
             Child:    4  5  3  x  x  x

Now the parents are identical and, so, the
child is determined:

             Child:    4  5  3  6  1  2
```

Note that the global precedence vector favors nodes with higher degree. Hence, there is a tendency to force the higher degree nodes toward the front of the chromosome and the lower degree nodes toward the end of the chromosome. Thus, the chromosome attempts to cover as many edges in as few steps as possible. Our mutation operator is swap (i.e. two allele values are exchanged). Our GA uses the generational model and the mutation rate varies from 0.01 to 0.10.

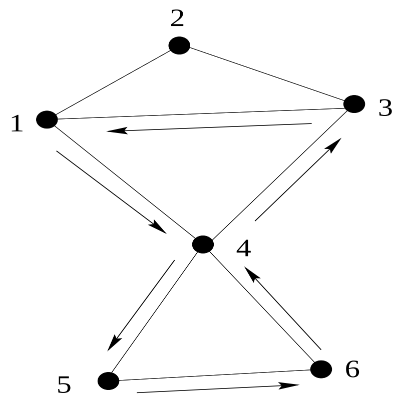

Figure 5:

CONCLUSIONS AND FUTURE WORK

The four graphs that were used to test the two heuristics were designed by hand. They contain no bridge edges in accordance with our assumptions. All of these graphs contain cycles with length 4 or greater. This is in response to the fact that first attempts to construct test cases produced graphs where the longest cycles were of length 3.

The best solutions found by the deterministic heuristic (VCNN) and the genetic algorithm (GA) for the six node graph presented as Figure 1 are depicted in Figure 4 and Figure 5, respectively. The chromosome that represents the tour shown in Figure 5 is (5,6,3,1,2,4). The 25 node graph is shown in Figure 6. The best results from our four test graphs are presented in Table 1.

No. of Nodes	VCNN Tour Length	VCNN Simple Tour	GA Tour Length	GA Simple Tour
6	6	No	6	Yes
12	9	No	7	Yes
25	25	No	23	Yes
50	53	No	51	No

Table 1: VCNN and GA Results for Various Graphs

The VCNN column gives the length of the ITP tour obtained by the deterministic heuristic. The GA column gives the tour length of the best individual obtained by the genetic algorithm. In each case, our first execution of the GA did not allow edges to be repeated. If a solution was not found, then the GA was restarted, this time allowing repeated edges. As shown in Table 1, the

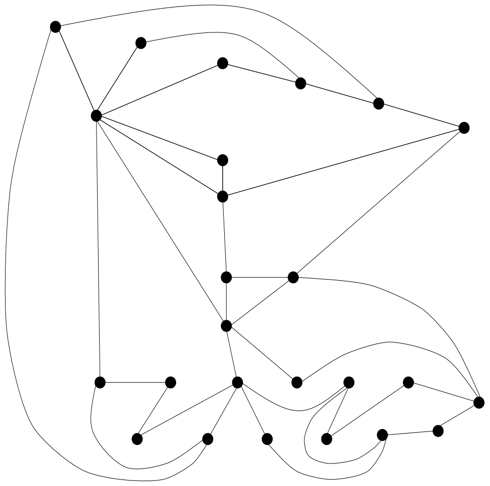

Figure 6:

tour length of the GA's solutions were always less than or equal to those of VCNN. It is also the case that the GA found a simple tour in all but the last test case. On the other hand, VCNN never found a simple tour.

Therefore, our results show that the GA can outperform VCNN on a consistent basis. However, the inability of the GA to find a simple tour in the test case of 50 nodes raises the question of scalability. It is unclear how the GA will perform on larger test cases. For this reason, alternate heuristics that attempt to solve the ITP in one step, as the GA does, will be considered in the future. The use of larger test cases will also be included in future work.

We also intend to consider the graph theoretic issues of the ITP more thoroughly. For example, does every graph without bridge edges have a simple investigator tour (i.e. one without repeated edges.) As another example, can some investigator tours with repeated edges still locate any single fault in a network. In a previous section, while defining the MITP problem, we presented an example where we were not able to locate certain faults with a particular tour containing a repeated edge. However, the example does not prove that every investigator tour with repeated edges is unable to locate every possible single fault in a network without bridge edges. We do not presently know whether having a simple tour is a necessary condition for locating a single fault. The theoretical result of our current work, however, shows that having a simple investigator tour is sufficient for locating any single fault in a network without bridge edges.

Acknowledgements

This research has been supported by OCAST Grant AR2-004. The authors also wish to acknowledge the support of Sun Microsystems, Inc.

References

[1] A. Bouloutas, G. Hart and M. Schwartz, "On the design of observers for failure detection of discrete event systems," Network Management and Control, A. Kershenbaum, M. Malek, M. Wall, Eds. New York: Plenum, 1990.

[2] J. Blanton and R. Wainwright, "Multiple Vehicle routing with time and capacity constraints using genetic algorithms," in S. Forrest (ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 452-459, Morgan Kaufman, 1993.

[3] A. Corcoran and R. Wainwright, "Using LibGA to Develop Genetic Algorithms for Solving Combinatorial Optimization Problems," in Practical Handbook of Genetic Algorithms, Applications, Volume 1, Editor Lance Chambers, CRC, 1995.

[4] T. Cormen, C. Leiserson and R. Rivest, Introduction to Algorithms, McGraw Hill, 1990.

[5] L. Davis, editor, Genetic Algorithms and Simulated Annealing, Morgan Kaufman, 1987.

[6] L. Davis, editor, Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.

[7] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[8] C. L. Liu, Elements of Discrete Mathematics, Second Edition, McGraw Hill, 1985.

[9] S. Paul and R. Miller, "Locating Faults in a Systematic Manner in a Large Heterogeneous Network," *Proceedings of INFOCOM 1995.*

[10] G. Rawlins, editor, Foundations of Genetic Algorithms, Morgan Kaufman, 1991.

[11] I. Rouvellou and G. Hart, "Topology Identification for Traffic and Configuration Management in Dynamic Networks," *Proceedings of INFOCOM 1992*, pp.2197-2204.

[12] C. Wang and M. Schwartz, "Fault Detection with Multiple Observers," IEEE/ACM Transactions on Networking, February 1993, Vol 1, No 1.

[13] C. Wang and M. Schwartz, "Fault Diagnosis of Network Connectivity Problems by Probabilistic Reasoning," *Proceedings of Second IEEE Network Management and Control Workshop*, Sept. 21-23, 1993.