# Analysis of fault location in a network

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, USA*
E-mail: schoend@utulsa.edu

We review and analyze a formal problem of fault location in point-to-point telecommunication networks which are modeled as undirected graphs. For a network with no bridge edges, we present a technique for constructing a tour that is able to locate a single fault. We also develop theory that provides the necessary and sufficient conditions for locating and detecting a single fault.

**Keywords**: fault location, telecommunication networks, graph theory, complexity theory, heuristic techniques

## 1. Introduction

Fault location is an important aspect of network management in which a network manager tries to determine the location of a fault in a network, if one exists. If no fault exists in the network, the network manager should report that the network is functioning properly. Recently, many problems related to network management have been formally stated and partially solved [1,4,7,8,10,11].

We model a network as an undirected graph $G(V, E)$ with a set of nodes $V$, which represents the active elements of the network, and a set of edges $E$, which represents the passive elements interconnecting the active elements. An active element has the ability to report its status and to receive information about the status of other active elements. Some examples of active elements are hosts, bridges and routers. Passive components, such as network cables, 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 [7]. They also analyze the performance of their schemes.

In the incremental multicasting scheme (IMS), which is discussed by Paul and Miller [7], the network manager uses a set of spanning trees, each rooted at the network

manager. The network manager (NM) selects one of these trees. Then, it multicasts a status request message to all of the nodes at distance $i$, where $i$ ranges, in turn, 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 can determine from the replies it receives whether a node is alive or if it is unreachable in the current spanning tree. In the former case, the NM knows that the node is alive and so are the links along the path in the tree from the NM to the node in question. However, in the latter case, either the node in question is down or the last link on the path from the NM to the node is down. In this case, the NM uses an alternative spanning tree to determine which element is faulty, either the node or the link. If an undirected graph has no bridge edges and has a single fault (either at a node or an edge), the IMS from [7], with a suitable collection of spanning trees, is guaranteed to locate the fault. An edge in a graph is a bridge edge if its removal disconnects the graph.

In the Investigator Propagation Scheme (IPS) described by Paul and Miller [7], the NM sends a special message, called an investigator, along a precomputed route beginning and ending at the NM. At each node in the precomputed route, the investigator instructs the node to collect the status of all incident 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 at most one fault and no bridge edges, Paul and Miller state that the ambiguity can be resolved [7]. If all replies are received, the investigator continues to the next node and repeats the process. The precomputed tour is called an Investigator Tour and to find such a tour, one must solve the Investigator Tour Problem.

Paul and Miller state that the advantage of the IMS is that it has a response time of $O(\log_2 n)$, where $n$ is the number of nodes in the network. The disadvantage of the IMS is the amount of network traffic that it generates (see [7] for a detailed explanation). The IPS generates considerably less network traffic than the IMS. However, its response time is $O(m)$, where $m$ is the number of nodes in the tour. This value $m$ is usually close to the number of nodes in the network $n$. Therefore, the IMS typically has a better response time than the IPS but at the cost of generating more traffic [7].

With the IPS, Paul and Miller [7] introduced the concept of using an investigator tour to locate a single fault in a network. Their networks were modeled as undirected graphs without bridge edges. The nodes in an investigator tour form a vertex cover for the graph. Using this fact, Paul and Miller discuss how to locate a single fault in the network. However, in our previous work [4] we discovered that if the tour was not simple, i.e., if edges were used more than once, then locating a single fault is not guaranteed. This issue was not discussed by Paul and Miller [7].

Initially, we wanted to show that every graph without bridge edges had an investigator tour that was simple [4]. However, we subsequently found several graphs without bridge edges that do not have simple investigator tours. Nevertheless, each of our example graphs has an investigator tour that, although not simple, is able to locate a single fault. In this research, we expand on the initial work of Paul and Miller. A

fundamental result of our work is as follows. Any graph $G$ without bridge edges contains one or more subgraphs with the following two properties: (1) the subgraph has no bridge edges, and (2) the nodes of the subgraph are a vertex cover of $G$. We show that we can construct a tour in any of these subgraphs that is an investigator tour of $G$ able to locate a single fault in $G$. Thus, determining the location of a single fault is possible in any network without bridge edges.

A second objective of this research is to ensure that a heuristic can find an optimal or near-optimal fault locating tour in a network without bridge edges. We also develop a taxonomy of the different types of investigator tours that exist. In addition, the existence of each of these types of tours in an arbitrary graph is discussed.

## 2. The Investigator Tour Problem (ITP)

Using the terminology presented by Liu [5], we let $G(V, E)$ be a connected, 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 $(v_{i_1}, v_{i_2}, \ldots, v_{i_k})$, 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 $V'$ [2]. As noted above, an edge in a graph is a *bridge edge* if its removal disconnects the graph.

The Investigator Tour Problem, given a connected, undirected graph $G$ with edge weights, is to find a minimum length tour that covers each edge of $G$. Note that such a tour does not necessarily include every edge and does not necessarily include every node. All of our research assumes each edge in $G$ has weight equal to one, corresponding to the notion of a "hop" in a telecommunication network, and that $G$ has no bridge edges. These assumptions do not detract from the complexity of the problem. The ITP was shown to be NP-hard in [7].

A deterministic heuristic for the ITP was proposed in [7]. The steps of the deterministic heuristic are as follows:

### Deterministic heuristic for the ITP

**Step 1.** Find a vertex cover $V'$ of $G(V, E)$.

**Step 2.** Build a new complete graph $G'(V', E')$ with edge weights. $V'$ is the set of nodes in the vertex cover of $G$ (which was computed in step 1). The weight of any edge, say $e' = (u', v')$, in $G'$ is the length of the shortest path from $u'$ to $v'$ in $G$.

**Step 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, $G'$.

**Step 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.

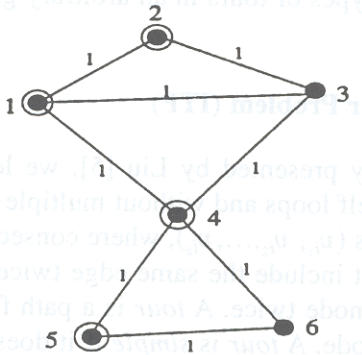


Figure 1. Step 1 of the VCNN heuristic.

Most graphs contain many vertex covers. 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. However, finding a vertex cover of minimum cardinality is itself NP-complete [2]. 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 constant ratio bound of 2 [2]. Although our heuristic does not have a constant ratio bound of 2, the degree considerations seem to be advantageous in our context. 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 weight 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 a canonical NP-complete problem [3]. We use the classical, deterministic nearest-neighbor heuristic to find a near-optimal solution to the TSP. When applied to figure 2 with node 1 as the initial node, the nearest-neighbor heuristic produces the TSP tour given by (1, 2, 4, 5, 1), as illustrated in figure 3.

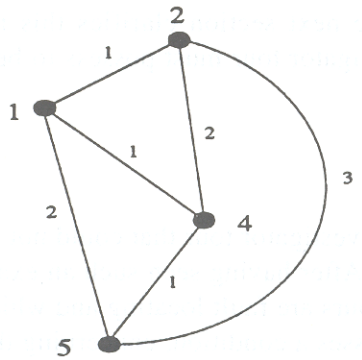


Figure 2. Step 2 in the VCNN heuristic.

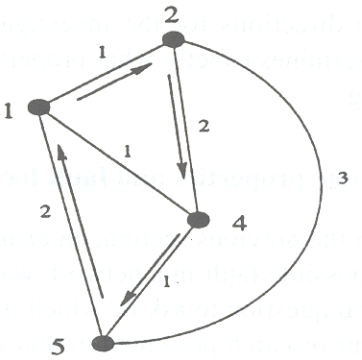


Figure 3. Step 3 in the VCNN heuristic.

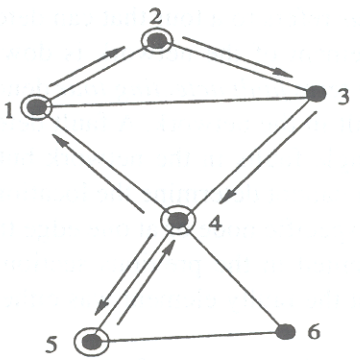


Figure 4. An *FD* investigator tour found by the VCNN heuristic.

The TSP tour in $G'$, illustrated in figure 3, maps, at least for one strategy of tie breaking, 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).

Notice, this tour is not a simple tour. Therefore, if node 1 is the NM and a fault occurs at node 5 or the edge between node 4 and node 5, the investigator will be unable to determine the exact location of the fault. To illustrate, the investigator will proceed from node 1 to node 4 via nodes 2 and 3 without difficulty. At this point, the investigator will not receive a response from node 5. So, the investigator concludes that either node 5 is down or the link between node 4 and node 5 is down. To determine the location of the fault, the investigator tour will reverse its direction of travel along the tour. When it reaches the NM, it will report that a fault exists and continue along the tour in the reverse direction to determine the exact location of the fault. It will travel from node 1 to node 4. At this point, the investigator is on the "same" side of the fault as it was before. Thus, it cannot determine the exact location of the fault. This ambiguity is due to the fact that the edge between node 4 and node 5 is used

in both directions for the investigator tour. The next section clarifies this notion and determines exactly what properties an investigator tour must possess to be fault locating.

## 3. Edge properties and fault location

In the previous section, an example of an investigator tour that could not locate every possible fault in a network was presented. After having seen such an example, a natural question to ask is: which investigator tours are fault locating and which are not? Our research presented in this section discusses a condition, concerning the use and reuse of edges in an investigator tour, which classifies all investigator tours as either fault locating or fault detecting.

All networks considered in this research are assumed to have at most one fault. The term *fault locating tour* refers to a tour that can determine the exact location of a fault, i.e. report which element of the network is down, no matter where the fault occurs in the network. The term *fault detecting tour* denotes a tour that recognizes the presence of any single fault in the network. A fault detecting tour can determine the exact location of some single faults in the network but not all of them. In the case when a fault detecting tour cannot determine the location of a fault, it can specify that the fault occurs at either a specific node or at one edge that is incident to the node. An example of this was presented in the previous section, where the investigator was only able to determine that the faulty element was either node 5 or the edge between node 4 and node 5.

To determine if an investigator tour is fault locating, one must look at each edge that is used in the investigator tour. If an edge is used only once, it will not cause a problem during tour reversal. However, if an edge is used more than once in an investigator tour (as in figure 4), it must be used in a specific way to ensure that the tour is fault locating for any single fault. For a tour with repeated edges to be fault locating for any single fault, the first and last uses of all of the repeated edges must be in the same direction. Note: The first and last uses of a repeated edge are relative to the starting node of the tour, i.e., the NM. Hence, a node must be selected as the NM before the tour is constructed.

To see why this is true, consider any repeated edge, say the edge between node $A$ and node $B$ in some graph. Assume that the first occurrence of this edge in the investigator tour is in the direction from node $A$ to node $B$. If a fault occurs either at node $B$ or at the edge itself, the investigator will be unable to locate the fault when the fault is first encountered (this is the same situation that was presented in the previous section). If the last use of the edge between node $A$ and node $B$ is in the opposite direction as the first use, i.e., from node $B$ to node $A$, then, since during reversal the investigator will be traversing the end of the tour in reverse, it will use this edge in the direction from node $A$ to node $B$. Thus, the investigator will once again be on the "same" side of the fault and will be unable to resolve the ambiguity, see figure 4 with

$A$ as node 4 and $B$ as node 5. However, if the last use of the edge from node $A$ to node $B$ is in the direction from $A$ to $B$, then reversing direction along the end of the tour will bring the investigator to node $B$ before node $A$. Hence, the investigator can determine if node $B$ is faulty or if the link is faulty. To illustrate, see figure 5, given that node 1 is the NM, node 2 is $A$, and node 8 is $B$.

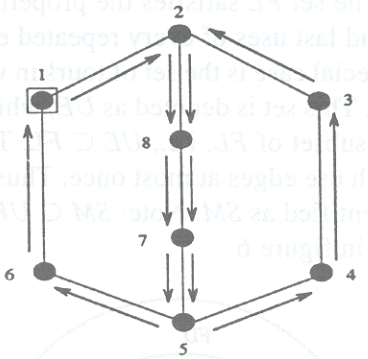


Figure 5. An *UE* investigator tour.

A similar example demonstrates why it is difficult to extend the current framework to the case of multiple faults. If we assume that we are still only able to collect information about the active elements (the nodes) and not the passive elements (the edges), then, without any additional conditions upon the type of graph or the location of the faults in the network, an example can be constructed where the IPS, using an investigator tour which satisfies the conditions on the reuse of edges, cannot locate all the faults in the network. For example, consider the graph in figure 5. Now consider three cases where the network contains two faults: (1) where the first fault occurs at the edge between node 2 and node 8 and the second fault occurs at node 8, (2) where the first fault occurs at the edge between node 2 and node 8 and the second fault occurs at the edge between node 8 and node 7, and (3) where the first fault occurs at node 8 and the second fault occurs at the edge between node 8 and node 7. Applying the IPS with the investigator tour in figure 5 will yield the same information for all three of these cases, i.e., there is no response when the investigator tries to ping node 8 from node 2 and there is no response when the investigator tries to ping node 8 from node 7 after it reverses its direction along the tour. Therefore, we can conclude that without further constraints the IPS may not be able to locate multiple faults.

## 4. Investigator tour taxonomy

From our discussion of edge reuse in the previous section, it is possible to develop a classification for investigator tours. As was originally stated in [7], the VCNN heuristic is only guaranteed to generate fault detecting tours. However, at times it can

generate fault locating tours. This is due to the fact that all fault locating tours are special cases of fault detecting tours where there are no elements in the network at which the tour cannot locate a fault. Therefore, the set of all fault locating tours, say *FL*, is a proper subset of the set of fault detecting tours, say *FD*, i.e., $FL \subset FD$.

The set of fault locating tours *FL* contains two types of special cases, one of which contains the other. The set *FL* satisfies the property presented in the previous section; namely, the first and last uses of every repeated edge in the tour occur in the same direction. The first special case is the set of tours in which the use(s) of all edges occur in the same direction. This set is denoted as *UE*, which stands for unidirectional edges. This set is a proper subset of *FL*, i.e., $UE \subset FL$. The second special case is a set that contains tours which use edges at most once. Thus, all the tours in this set are simple tours. This set is identified as *SM*. Note: $SM \subset UE$. A pictorial representation of this taxonomy is shown in figure 6.

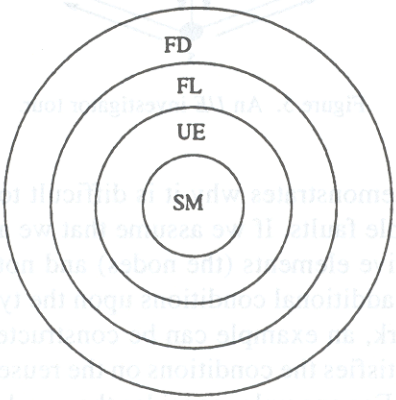


Figure 6. The investigator tour taxonomy.

It is helpful to be able to identify a concrete example with each of the classes in the taxonomy. An example of a tour, say $T_1$, that is only fault detecting, i.e., $T_1 \in FD \wedge T_1 \notin FL$, is shown in figure 4. A tour, say $T_2$, that uses repeated edges in the same direction, i.e., $T_2 \in UE \wedge T_2 \notin SM$, is presented in figure 5. An illustration of a tour, say $T_3$, that is simple, i.e., $T_3 \in SM$, can be found in figure 7. A tour, say $T_4$, is the most general type of fault locating, if $T_4 \in FL \wedge T_4 \notin UE$. An example of this type of tour is shown in figure 8. The numbered arrows in figure 8 trace the path of the tour through the graph. This tour is not a *UE* tour because there is an edge which is used in both directions, namely the edge which is used as the second, seventh, and fourteenth edges in the tour.

Now that the taxonomy and examples for each type of tour in the taxonomy have been presented, we next address the question of existence of each type of tour in an arbitrary graph that is connected, undirected, has no bridge edges, and has at most one fault. Graphs of this type will be referred to as *dn-graphs*, for data network graphs. A

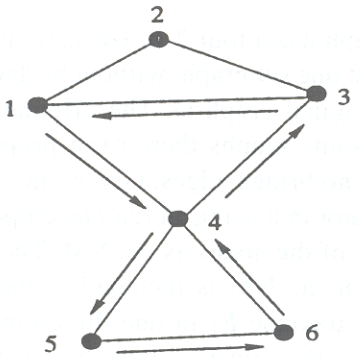


Figure 7. An *SM* investigator tour.

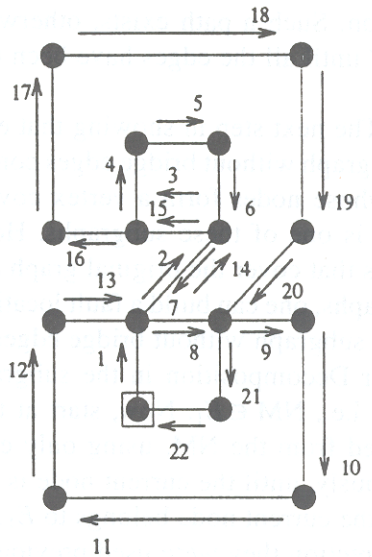


Figure 8. An *FL* investigator tour.

sufficient condition for a tour to be fault detecting is that the nodes of the tour must form a vertex cover of the graph. Thus, all dn-graphs have at least one investigator tour that is fault detecting, say $T$, i.e., $T \in FD$. On the other end of the spectrum is the set of simple investigator tours, *SM*. Not every dn-graph has a simple investigator tour. An example is the graph shown in figure 5.

Every dn-graph has a tour, say $T$, that uses all edges in the same direction, i.e., $T \in UE$. The following argument will prove this claim. The first step in this argument is a theorem from graph theory concerning graphs without bridge edges. We present the proof of this theorem because it does not appear in the cited reference.

**Theorem 1** (Ear Decomposition Theorem) [6, p. 34]. For every connected, undirected graph $G(V, E)$ without bridge edges, the set $V$ can be partitioned into disjoint sets $E_1, E_2, \ldots, E_k$, such that $E_1$ is a simple cycle and, for each $i$, $1 < i \leq k$, $E_i$ is a simple path whose endpoints are nodes that already appear in a previous $E_j$, $j < i$, and its other nodes (if any) have not appeared in previous $E_j$'s. (The path may be a closed one, in which case it includes only one previous vertex.)

*Proof.* Since $G$ does not contain a bridge edge, it contains at least one simple cycle. Let this cycle be $E_1$. If $E_1 = E$, then the decomposition is finished. Else, there is at least one edge incident with $E_1$ that is not in the decomposition. Start the first "ear", i.e. $E_2$, with this edge. Follow a path that starts with this edge until the path's last edge is incident to a vertex that has another incident edge that is already in the decom-

position. Such a path exists, otherwise $G$ contains a bridge edge. Continue adding "ears" until all the edges have been used. □

The next step in showing that every dn-graph has a tour $T$ in $UE$ is to note that every graph without bridge edges contains at least one subgraph, without bridge edges and whose nodes form a vertex cover of the original graph $G$. The original graph, itself, is one of these subgraphs. However, in some graphs there exist proper subgraphs that cover the original graph and contain no bridge edges. Given one of these subgraphs, one can build a fault locating investigator in it using an Ear Decomposition. In the subgraph without bridge edges, select one of the nodes as the NM. Then build an Ear Decomposition in the subgraph such that the NM is included in the initial cycle, i.e., NM $\in E_1$. Now, start at the NM and traverse $E_1$ in one direction. Then, proceed from the NM, using only edges in $E_1$ in the direction that they were used previously, until the current node is an endpoint of $E_2$. Traverse $E_2$, which is a path. Now the current node belongs to $E_1$. So, return to the NM, using only edges in $E_1$ in the direction they were used previously. At this point, the tour leaves the NM, using edges in $E_1$ and $E_2$ in the direction they were used previously, until it encounters an endpoint of $E_3$. The path that $E_3$ constitutes is traversed, and the tour returns to the NM using edges in $E_1$ and $E_2$ in the direction they were used previously. This procedure continues until all elements of the subgraph have been included in the tour and the tour has returned to the NM. Thus, all graphs without bridge edges contain an investigator tour $T$ where $T \in UE$. Note: this is a constructive argument, but the result is generally a long tour. Thus, this constructive argument is not efficient as a heuristic for finding near-optimal investigator tours. However, it does show that in a subgraph without bridge edges, whose nodes form a vertex cover of the original graph, there exists at least one fault locating investigator tour.

## 5. The GITP

If the constructive argument, from the previous section, were used as a heuristic, most of the time it would not generate efficient investigator tours. Hence, it would be advantageous to find another heuristic that could generate efficient investigator tours. Another property that this heuristic should possess is some guarantee about which class of the taxonomy the resulting tours will belong to. A reasonable heuristic for the ITP already exists. It is the VCNN heuristic which was presented earlier [7]. However, solving the ITP only guarantees that the resulting tour will be a member of $FD$. A feasible course of action is to modify the existing heuristic for the ITP to produce tours that are required to be in other classes of the taxonomy. To provide a theoretical framework for this modification, a generalized version of the ITP will be presented. Next, techniques for modifying the heuristic so it can be applied to this new problem will be discussed.

Using the symbol $\mathcal{T}$ to represent the set of all tours in a graph, the Generalized Investigator Tour Problem (GITP) is as follows:

**Definition 2** (Generalized Investigator Tour Problem – GITP). Given a connected, undirected graph $G(V, E)$ and a Boolean function $\Psi_0 \in \{\Psi | \Psi : E \times \mathcal{T} \rightarrow \{true, false\}\}$, find a minimum length tour $T \in \mathcal{T}$, $\exists T$ covers $E$ and $\Psi_0(e, T) = true$, $\forall e \in T$.

Notice, the main difference between the ITP and the GITP is the addition of the Boolean function $\Psi_0$ which is applied to an edge and a tour. For each class of our taxonomy, there is an appropriate choice for this function that forces the solutions of the problem to be in the corresponding desired class. After defining $\Psi_{FD} \equiv true$ and selecting $\Psi_0$ to be $\Psi_{FD}$, the resulting special case of the GITP is the ITP. In fact, this choice for $\Psi_0$ allows for a polynomial-time reduction of the TSP to the GITP, which is similar to the reduction of the TSP to the ITP presented in [7]. Thus, the GITP where $\Psi_0 = \Psi_{FD}$ is NP-hard.

Letting $\Psi_0 = \Psi_{SM}$, where $\Psi_{SM}(e, T) = true$ iff edge $e$ is used in tour $T$ once, we obtain the special case of the GITP for which a solution is guaranteed to be in $SM$. This special case of the GITP is also NP-hard. A sketch of the reduction of the ITP to the GITP with $\Psi_0 = \Psi_{SM}$ is as follows. Given an instance of the ITP (assume the graph in this problem is denoted by $G(V, E)$), construct a new graph $G^*$ consisting of $|V|$ copies of $G$. The weight on each of the edges in each of these copies is the same as it was on the corresponding edge in $G$. To complete $G^*$, we take each vertex in the first copy of $G$ and connect it to the corresponding vertices in each of the other $|V| - 1$ copies of $G$ by two paths. Each of these paths consist of two edges and one vertex. The edges in these paths have weight zero. The reason for adding the $|V|$ copies of $G$ is that an edge is used at most $|V|$ times in a solution to the ITP, which was generated by a reasonable algorithm. A reasonable algorithm terminates after all edges have been covered and the tour has returned to the NM. To see why an edge is used at most $|V|$ times, consider any heuristic for the ITP. In the worst case, the heuristic will use a vertex cover of size $|V|$. Then it will find $|V|$ paths: one from the first vertex in the cover to the second, one from the second vertex to the third, and finally, one from the $|V|$th vertex back to the first. A reasonable algorithm uses an edge at most once in each of these paths. Thus, a reasonable algorithm uses an edge no more than $|V|$ times. The first path of weight zero that was added between corresponding vertices allows the tour to "jump" to another copy of $G$ if it needs to reuse an edge. If the tour does not need to use an edge in one or more of the additional copies of $G$, then the second path of weight zero allows the tour to come back to the first copy of $G$ after having visited the additional copy or copies of $G$ and thus covering the same edges in the additional copy or copies of $G$ that were covered in the first copy. We conclude that an instance of the GITP on the graph $G^*$ with $\Psi_0 = \Psi_{SM}$ is polynomially equivalent to the original ITP on $G$. Since the ITP is NP-hard, we conclude that the GITP with $\Psi_0 = \Psi_{SM}$ is NP-hard. An example of this reduction is shown for a graph $H$, which is a ring of four vertices, in figures 9 and 10.

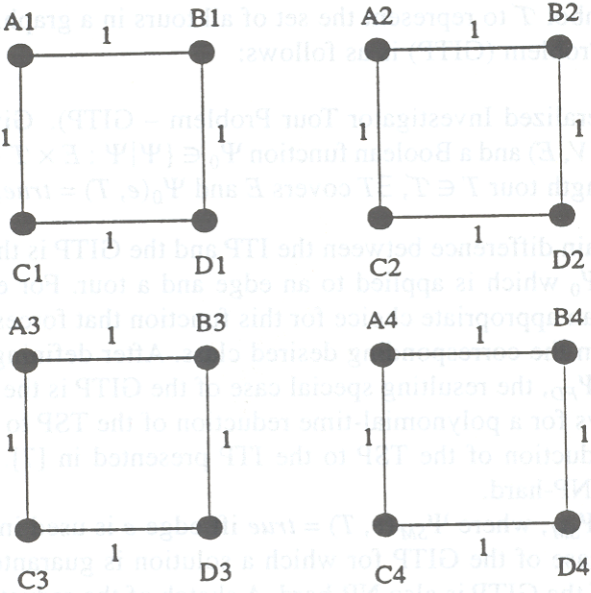


Figure 9. The $|V|$ copies of $H$ in the $\Psi_{SM}$ reduction.

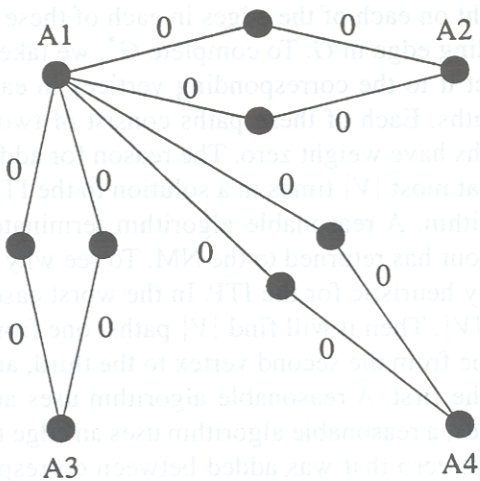


Figure 10. The paths of weight zero for the $A_i$ vertices of $H^*$ in the $\Psi_{SM}$ reduction.

Letting $\Psi_0 = \Psi_{UE}$, where $\Psi_{UE}(e) = \textit{true}$ iff edge $e$ is traversed in the same direction every time it is used in the tour, we obtain the special case of the GITP for which solutions are guaranteed to be in $UE$. Next, we will show that this special case of the GITP is NP-hard. A sketch of the reduction of the ITP to the GITP with $\Psi_0 = \Psi_{UE}$ is as follows. Given an instance of the ITP (assume the graph in this problem is denoted by $G(V, E)$), construct a new graph $G^*$ consisting of two copies of $G$. Each edge in this

part of $G^*$ has the same weight as the corresponding edge in $G$. To complete $G^*$, we take each vertex in the first copy of $G$ and connect it to the corresponding vertex in the second copy of $G$ by two paths. Each of these paths consists of two edges and one vertex. The edges in these paths have weight zero. The reason for the two copies of $G$ is that each edge in $G$ can be used in at most two directions in a solution to the ITP, i.e. for endpoint A to endpoint B and vice versa. The paths of weight zero are added for the same reason they were added in the previous reduction. We conclude than an instance of the GITP on the graph $G^*$ with $\Psi_0 = \Psi_{UE}$ is polynomially equivalent to the original ITP on $G$. Since the ITP is NP-hard, we conclude that the GITP with $\Psi_0 = \Psi_{UE}$ is NP-hard.

Finally, letting $\Psi_0 = \Psi_{FL}$, where $\Psi_{FL}(e) = \mathit{true}$ iff the first and last uses of edge $e$ in the tour are in the same direction, we obtain a special case of the GITP for which solutions are guaranteed to be in $FL$. We have not constructed a reduction to show that this special case of the GITP is also NP-hard, although we are reasonably sure that it is, nor have we extensively investigated the properties of tours in $FL$.

We now discuss techniques for modifying any ITP heuristic so it will produce solutions to the GITP for any of the four $\Psi_0$'s presented in the previous paragraph. For $\Psi_0 = \Psi_{FD}$, no modification is required, because this special case of the GITP is equivalent to the ITP. For $\Psi_{SM}$, the required change is to remove an edge, from the data structure for $G$, once it is used in the tour. With this modification, a heuristic may be unable to find a tour in all graphs. An example of a graph for which a heuristic, modified in this fashion, will fail is the graph shown in figure 5. For $\Psi_0 = \Psi_{UE}$, the first time an edge is used in the tour it must be changed to a directed edge. However, when a heuristic makes edges directed after they have been traversed brings up the question of whether the heuristic will be able to complete the tour, by returning to the NM, or not? The following theorem answers this question:

**Theorem 3** (Completion Theorem). Given a connected, undirected graph $G$ without bridge edges, and a partial investigator tour where all edges in the tour are only used in one direction, then $\exists$ a path that completes the tour so that the tour is in $UE$.

*Proof.* A constructive proof can be written using techniques similar to those used for the Ear Decomposition Theorem. For example, the general procedure for taking any of the three partial tours and creating the corresponding complete tour in figure 11 is as follows. Given a partially completed cycle (as in case 1 of figure 11), we can always finish the cycle because the graph does not contain a bridge edge. Given one or more partially completed "ears" (as in cases 2 and 3 of figure 11), we know the ears can be completed because the graph has no bridge edges. Hence, we can always continue constructing the investigator tour, either by completing the initial cycle of an ear decomposition, by completing an incomplete ear of an ear decomposition, or by traversing an edge which has already been used in the tour (provided we traverse it in the same direction it was originally traversed). □

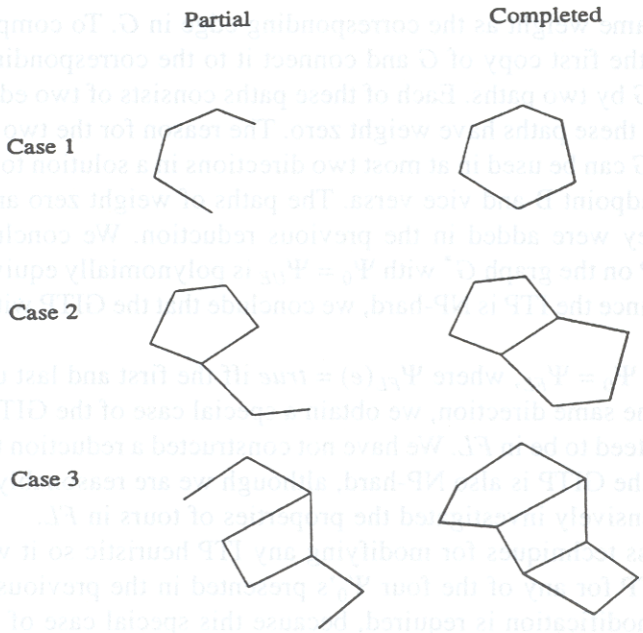


Figure 11. Examples of tour completion.

Thus, a heuristic that uses this technique will always be able to finish the tour. This is a nice result because it has been shown that such a tour exists in every graph without bridge edges. We have shown that a simple modification to any heuristic for the ITP will efficiently produce reasonable tours of this type. For $\Psi_{FL}$, the final step in the heuristic must check that all of the edges satisfy $\Psi_{FL}$. If not, the heuristic must change its solution using some form of backtracking. Since all heuristics trying to produce tours in *FL* must use backtracking, their order will be exponential.

## 6. Experimental results

Two test suites were used to demonstrate the theory developed in this research. The first test suite consists of six graphs. These graphs were designed by hand to ensure that they contained no bridge edges. The 25-node graph from this test suite is shown in figure 12. This first test suite will be referred to as the set of contrived graphs from this point on. The results obtained by applying the VCNN to this set of graphs are shown in table 1. Each graph has two entries. One in the *FD* column and the other in the *UE* column. Each number in table 1 represents the length of the tour obtained by the VCNN. The tours represented in the *FD* column are solutions to the ITP, i.e. they are members of the set *FD*. Note: since *FL*, *UE*, and *SM* are subsets of *FD*, the tours represented in the *FD* column may also be members of some of these subsets. However, they are only guaranteed to be members of *FD*. The tours repre-

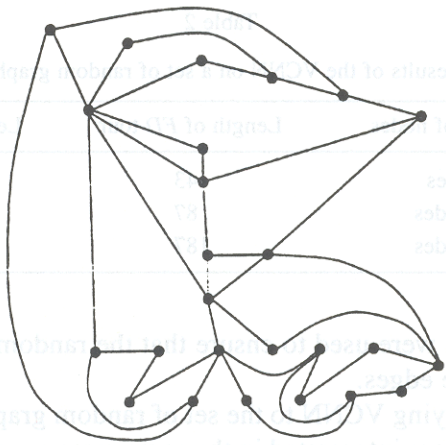

Figure 12. The 25-node graph from the first test suite.

Table 1

Results of the VCNN on a set of contrived graphs.

| Number of nodes | Length of *FD* tour | Length of *UE* tour |
|---|---|---|
| 6 nodes | 6 | 7 |
| 12 nodes | 9 | 9 |
| 25 nodes | 25 | 28 |
| 50 nodes | 53 | 63 |
| 100 nodes | 110 | 14 |
| 200 nodes | 222 | 281 |

sented in the *UE* column are solutions to the GITP, for the Boolean function $\Psi_{UE}$. Some of these tours may also be members of *SM*, but not necessarily.

Since $UE \subset FD$, it follows that the length of the optimal tour in *FD* is less than or equal to the length of the optimal tour in *UE*. Even though the tours in table 1 are not guaranteed to be optimal, this relationship still holds in all cases. Intuitively, this is true because representative tours from *FD* will generally be shorter than representative tours from *UE*. However, the difference between the lengths of the two tours is not outlandish given the more demanding properties that the *UE* tours are required to satisfy.

The second test suite consists of three graphs that were randomly generated using the *Combinatorica* package [9] in *Mathematica* [12]. Each of the graphs was generated with edge probability of 0.2, i.e. each graph contains approximately 20% of the edges that a complete graph with the same number of nodes contains. The contrived graphs tended to be a bit more sparse than the random graphs. Built-in functions of the

Table 2

Results of the VCNN on a set of random graphs.

| Number of nodes | Length of *FD* tour | Length of *UE* tour |
|---|---|---|
| 50 nodes | 43 | 44 |
| 100 nodes | 87 | 87 |
| 200 nodes | 187 | 187 |

*Combinatorica* package were used to ensure that the random graphs were connected and contained no bridge edges.

The results of applying VCNN to the set of random graphs are shown in table 2. The entries in this table are interpreted in the same manner as those in table 1, i.e. the tours in the first column are members of *FD* and the those in the second column are members of *UE*. It is interesting to note that the same relationship between the tour lengths applies for the random graphs as it did for the contrived graphs, i.e. the lengths of tours in *FD* are less than or equal to the tours in *UE*. However, the difference between the two is smaller than with the contrived graphs.

## 7. Conclusions and future work

This research has expanded the notion of using investigator tours to locate a single fault in a network without bridge edges. Specifically, the terms fault locating and fault detecting were formalized, edge conditions used to classify investigator tours were stated, a taxonomy of the different types of investigator tours was developed, and a generalized version of the ITP with its relationship to the types of tours produced by heuristics was presented.

Two issues that will be addressed in the future are the following. First, a more complete exposition of the theoretical facets of the set *FL–UE*. This will include a discussion about instances when the less restricted tours in *FL–UE* are shorter than tours in *UE*. Second, the use of an extensive empirical study of the expected cost of using a tour in *FL* for fault location versus using a pair of tours, one from *FD* and one from *FL*, for fault location to determine which approach is better.

## References

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

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

[3] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

[4] C.J. Hoelting, D.A. Schoenefeld and R.L. Wainwright, Finding investigator tours in telecommunication networks using genetic algorithms, *Proceedings of the1996 ACM Symposium on Applied Computing*, ACM Press, 1996, pp. 82–87.

[5] C.L. Liu, *Elements of Discrete Mathematics*, 2nd ed., McGraw-Hill, 1985.

[6] U. Manber, *Introduction to Algorithms*, Addison-Wesley, Reading, MA, 1989.

[7] S. Paul and R. Miller, Locating faults in a systematic manner in a large heterogeneous network, *Proceedings of INFOCOM 1995*.

[8] I. Rouvellou and G. Hart, Topology identification for traffic and configuration management in dynamic networks, *Proceedings of INFOCOM 1992*, pp. 2197–2204.

[9] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Addison-Wesley, Redwood City, CA, 1990.

[10] C. Wang and M. Schwartz, Fault detection with multipleobservers, IEEE/ACM Transactions on Networking 1(1993)48–55.

[11] C. Wang and M. Schwartz, Fault diagnosis of network connectivity problems by probabilistic reasoning, in: *Proceedings of the 2nd IEEE Network Management and Control Workshop*, Sept., 1993.

[12] S. Wolfram, *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley, Redwood City, CA, 1991.