Steiner Trees, Partial 2-Trees, and Minimum IFI Networks

Joseph A. Wald and Charles J. Colbourn
Department of Computational Science, University of Saskatchewan, Saskatoon, Saskatchewan, S7N 0W0, Canada

Minimum isolated failure immune networks are shown to be 2-trees. Further, subgraphs of 2-trees are shown to be exactly those graphs which contain no subgraph homeomorphic to the four-vertex complete graph. Together, these two characterizations yield a linear time algorithm for adding lines to a network to produce a minimum isolated failure immune network, whenever this is possible. This same algorithm, in conjunction with a linear time Steiner tree algorithm for 2-trees, yields a linear time Steiner tree algorithm for partial 2-trees. This contrasts with the known NP-completeness of the Steiner tree problem for planar graphs.

I. INTRODUCTION

For our purposes, a network is represented as an undirected graph $G = (V, E)$; we assume standard graph-theoretic terminology (see, for example, [2, 10]). The vertices in V represent sites or nodes in the network, and the edges represent communication lines. Since lines are typically costly, communication between distant sites is accomplished by using many lines to form a communication path. The correct functioning of the network requires the capability of communication between any two sites. This can be easily accomplished by any tree network; however, a model of a real communications network must reflect the possibility that sites and/or lines may become unavailable, owing either to error or to scheduled maintenance. These are termed *site failures* or *line failures.*

For tree networks, both site failures and line failures are catastrophic. Only one is needed to cause the network to fail in its primary task—enabling complete communication. Many methods have been proposed for designing networks which are immune to certain types of site and line failures; most methods try to overcome failure by having a network with large connectivity (see, for example, [9]).

Farley [3] proposes another type of network, which can survive many site and line failures, as long as they are *isolated.* To be precise, two site failures are isolated if the sites are not adjacent. Two line failures are isolated if the lines do not meet at a site. A site failure and a line failure are isolated if the line failure is not incident to a site which is adjacent to the site failure. A network is *isolated failure immune* (IFI) if and

Networks, Vol. 13 (1983) 159-167

CCC 0028-3045/83/020159-09$1.90

only if all pairs of operative sites can communicate as long as the failures are (pairwise) isolated.

Farley [3] gives a more detailed discussion of both the merits and the usage of IFI networks. In addition, he shows that IFI networks on $|V|$ sites have at least $2|V| - 3$ lines. He further demonstrates that 2-trees are IFI networks. A *2-tree* can be defined recursively as follows, and all 2-trees may be obtained in this way. A triangle is a 2-tree. Further, given a 2-tree and an edge (x, y) of the 2-tree, we can add a new vertex z adjacent to both x and y; the result is a 2-tree.

A 2-tree on vertex set V has $2|V| - 3$ edges, the fewest possible for an IFI network; hence 2-trees are *minimum IFI networks*, IFI networks with the fewest possible edges. All minimum IFI networks are, in fact, 2-trees; we establish this in Section II. Subsequently, we characterize the networks which can be completed to minimum IFI networks by adding new lines; in Section III, these partial 2-trees are shown to be exactly those graphs containing no subgraph which is homeomorphic to the four-vertex complete graph K_4. The latter equivalence underlies a linear time algorithm for taking an arbitrary partial 2-tree and making it a minimum IFI network; this is the topic of Section IV.

In Section V, we introduce the Steiner tree problem, and describe a linear time algorithm for solving it on 2-trees. Together with the results of section IV, this yields a linear time algorithm for finding Steiner trees in partial 2-trees. Outerplanar graphs are all partial 2-trees, and partial 2-trees are all planar graphs. The Steiner tree problem is solvable in linear time on outerplanar graphs [17], but is NP-complete on planar graphs [6, 7]; thus the result on partial 2-trees settles an interesting intermediate case.

II. IFI NETWORKS AND 2-TREES

All 2-trees are minimum IFI networks [3]. We prove the converse here, which enables us to transform network problems into purely graph-theoretic ones. This was first shown by Farley and Proskurowski [5]; we give a somewhat different proof here.

Theorem 2.1. A minimum IFI network is a 2-tree.

Proof. We prove the result by contradiction. Suppose $G = (V, E)$ is a minimum IFI networks, but not a 2-tree; further, let G have the fewest vertices of all such graphs. Since G is minimum, $|E| = 2|V| - 3$; since G is IFI, it has no vertices of degrees 0 or 1. Thus G has a vertex of degree 2 or a vertex of degree 3. If G has a vertex v of degree 2, the neighbors x and y of v are adjacent, since otherwise the failures of x and y would be isolated, yet separate v from the rest of the network. But then $G - v$ is a minimum IFI network; moreover, since G is assumed to be a smallest counterexample, $G - v$ is therefore a 2-tree. Then G also satisfies our definition of a 2-tree; hence G may have no vertices of degree 2.

Thus the minimal counterexample G must contain a vertex of degree 3. Let v be a vertex of degree 3 with neighbors x, y, and z. In the proof of his Lemma 4, Farley [3] shows that without loss of generality, y is adjacent to x and z, but x and z are nonadjacent. We produce a smaller network G' by adding the edge (x, z) and eliminating the vertex v. We claim that G' is a minimum IFI network. If a set of isolated failures existed in G', a set of isolated failures would also exist in G, contradicting our

assumption. Thus G' is a minimum IFI network; moreover, since G' is smaller than G, and G is assumed to be a smallest counterexample, G' must be a 2-tree.

Since G' is a 2-tree, it has at least two vertices of degree 2 (an easy consequence of the recursive definition). The transformation from G to G' could introduce at most one new vertex of degree 2, namely y, and hence G must contain a vertex of degree 2; this provides the required contradiction. ∎

This theorem establishes that the study of minimum IFI networks is essentially the graph-theoretic study of 2-trees.

III. SUBGRAPHS OF 2-TREES

A typical network design problem is the construction of a network to connect a set of sites. Often this takes the form of equipping an existing network with additional lines to obtain the desired reliability. In the context of IFI networks, we ask: when can we add lines to a network to create a minimum IFI network? In graph-theoretic terms, we are simply asking when a graph is a spanning subgraph of a 2-tree, or *partial 2-tree.* We establish here that

Theorem 3.1. Partial 2-trees are precisely those graphs which contain no subgraph homeomorphic to K_4.

Proof. Partial 2-trees have no subgraph homeomorphic to K_4; to see this, it suffices to show that 2-trees do not. If there were a 2-tree G containing a subgraph homeomorphic to K_4, for any degree-2 vertex v in G, $G - v$ would also contain a subgraph homeomorphic to K_4. Repetition of this process would ultimately require that K_3 contain a subgraph homeomorphic to K_4, and this cannot be. Thus partial 2-trees do not contain subgraphs homeomorphic to K_4.

In the other direction, suppose G is a "minimal counterexample," that is, a graph with the fewest vertices which is not a partial 2-tree, yet contains no homeomorph of K_4. If G is disconnected, each component of G is a partial 2-tree; selecting a representative vertex for each connected component, we connect the representative of component 1 to the representatives of each other component. This produces a connected counterexample with the same number of vertices.

Similarly, if G has a cutvertex v, this induces at least two biconnected components. Connecting a neighbor of v in one to a neighbor of v in the other does not introduce a homeomorph of K_4, and so produces a counterexample of the same size with fewer biconnected components. Repeating this process yields a biconnected counterexample.

Further, if G contains a 2-separator $\{x,y\}$ in which (x,y) is not an edge, we can add the edge without introducing a subgraph homeomorphic to K_4. In this way, we produce a biconnected counterexample in which every 2-separator is connected. If this counterexample contains a 2-separator, say $\{x,y\}$, two graphs G_1 and G_2 are defined whose union is G and whose intersection is $\{x,y\}$. Both of these graphs are partial 2-trees. But an easy consequence of the recursive definition is that whenever two 2-trees are joined by identifying an edge, the result is a 2-tree. Thus G is a partial 2-tree, and hence not a counterexample.

The only remaining possibility is that G has no 2-separators, i.e., is 3-connected. Consider two nonadjacent vertices u and v, and three disjoint paths between them;

Menger's theorem ensures the existence of these paths. Now consider $G - \{u, v\}$; since G is 3-connected, this graph is connected. Thus there are two vertices x and y which are on different u-v paths, and moreover, there is an x-y path which is disjoint from all of the u-v paths. Then u, v, x, and y form the four "corners" of a subgraph homeomorphic to K_4. Thus the supposed counterexample cannot be 3-connected, and this furnishes the needed contradiction. ∎

Theorem 3.1 is a valuable tool, since it gives us a completely graph-theoretic characterization of partial 2-trees, and thus provides us the wherewithal to recognize networks which can be completed to minimum IFI networks.

It should be noted that Liu and Geldmacher [13] examine the class of multigraphs containing no subgraph homeomorphic to K_4. They show that these multigraphs can be recursively constructed with four formation rules. Using this characterization, they devised a linear time algorithm for deciding whether a graph has a subgraph homeomorphic to K_4 [14]. Our Theorem 3.1 shows that partial 2-trees are precisely the multigraphs defined by Liu and Geldmacher's rules, which have neither loops nor parallel edges.

IV. COMPLETING PARTIAL 2-TREES

We next consider the problem of adding lines to a network to construct a minimum IFI network, whenever this is possible; we describe a linear time algorithm for this problem, employing both characterization theorems. Farley [3] considered a restricted version of this problem, giving an algorithm for completing a tree network to a minimum IFI network.

We employ Theorem 2.1 to note that completing a network to a minimum IFI network is equivalent to completing a partial 2-tree to a 2-tree. We first transform an arbitrary partial 2-tree to a connected one. In linear time, we find connected components using, for example, depth-first search [1, 16]. In each component, we arbitrarily choose a vertex as representative. We then add new edges to connect the representative of the first component to the representatives of all other components. No homeomorph of K_4 is introduced, and so Theorem 3.1 ensures that the result is a partial 2-tree if and only if the original was.

We next transform connected partial 2-trees to biconnected ones in linear time. To do this, we explore the vertices of the graph using standard depth-first search. Whenever we locate a cutvertex v, we retain a representative of each biconnected component among the descendants of v; this representative is one of v's neighbors in the biconnected component. Once all biconnected components among v's descendants are found, we add an edge from v's parent in the DFS tree to each of the representative. In the event that v is the root, one of the representatives is chosen arbitrarily to serve the role of v's parent. Once again, no homeomorph of K_4 is introduced, as long as one was not already present.

Given a biconnected partial 2-tree G, we transform it to a 2-tree in linear time as follows. Form a queue containing all vertices of degree 2 in G. Then repeatedly perform the following operations, until G is a triangle. Remove the first vertex v from the queue; if there is no such queue element, declare that G is not a partial 2-tree and stop. Otherwise, locate the neighbors x and y of v in G. If (x, y) is not an edge, add it. Delete the vertex v; if either x or y has its degree decreased to 2, add it to the queue.

Return to process the next queue element. Using this algorithm, we have the following theorem.

Theorem 4.1. In linear time, edges can be added to a partial 2-tree to construct a 2-tree whenever this is possible.

Proof. Our earlier remarks show that it suffices to start with a biconnected partial 2-tree. In this case, we claim that the algorithm described above performs the required completion and can be implemented in linear time. Timing follows from the observation that in each step, a constant number of operations is required. Since each step deletes a vertex, the number of steps is linear in the number of vertices, and so the overall time is linear.

Correctness follows from the remarks that if there is a homeomorph of K_4 involving a degree-2 vertex v with neighbors x and y, one can excise v and use the edge (x, y) instead. Conversely, if there was not a homeomorph of K_4, the deletion of v and addition of (x, y) cannot create one. Thus at every step, we obtain a partial 2-tree if and only if we began with one; correctness follows directly from this. ∎

V. STEINER TREES

The Steiner tree problem in graphs has been extensively studied, in part because of its practical importance in network design [8] and in circuit layout (see [6] and references cited therein). Karp [11] showed that the Steiner tree problem for arbitrary graphs is NP-complete, and Garey and Johnson [6, 7] have shown that the problem remains NP-complete for planar graphs. We have previously shown that Steiner trees in outerplanar graphs can be found in linear time [17]. We here consider the Steiner tree problem for partial 2-trees, which are intermediate between outerplanar and planar graphs.

Given a graph $G = (V, E)$ with positive integer edge costs, and a set of *targets* $T \subseteq V$, a *Steiner tree*, denoted $\mathrm{ST}(G, T)$, is a subtree $G' = (V', E')$ satisfying

1. $T \subseteq V' \subseteq V$,
2. the sum of edge costs in E' is minimal over all subtrees satisfying (1).

The *Steiner tree problem* asks whether a Steiner tree $\mathrm{ST}(G, T)$ has cost at most k, for given G, T, and k.

In this section, we describe an $O(n)$ algorithm for finding a Steiner tree for a given n-vertex partial 2-tree G with target vertex set T. The graph-theoretic motivation for this result is to study the gap between the known complexities on outerplanar and planar graphs. Perhaps more compelling is the motivation from network design. Since 2-trees are a useful network topology, the solution of network location problems for them merits attention.

Our Steiner tree algorithm for partial 2-trees operates in two phases. The first completes the graph to a 2-tree, using the method of Section IV. It is essential that we do not change the Steiner trees in the process. In order to ensure this, every added edge is given a very large edge cost BIG, which exceeds the sum of all other edge costs. Thus these added edges will never be chosen in a Steiner tree.

The second phase involves finding Steiner trees in 2-trees. Our algorithm to do this is based on a separation property of 2-trees [15]: every edge $\{x,y\}$ is a 2-separator which partitions the 2-tree into one or more components, which pairwise intersect at $\{x,y\}$, whose union is the entire 2-tree; moreover, each component so obtained is a 2-tree. This enables the development of a recursive method to produce a Steiner tree in the 2-tree by finding Steiner trees in each of the component 2-trees. We develop this idea formally here; the method developed owes much to techniques used previously [4, 17].

Our strategy is to reverse the recursive construction process of the 2-tree—that is, repeatedly eliminate vertices of degree 2 until the graph which remains is a single edge. During this vertex elimination procedure, we summarize information about the triangle $\{x,y,z\}$, where y has degree 2, on the arcs (x,z) and (z,x), prior to deleting y. This summary information encodes information about Steiner trees in the subgraph of the 2-tree which has thus far been reduced onto the edge $\{x,z\}$.

We are given a 2-tree G with target set T. With each arc $\alpha = (x,y)$ corresponding to an edge $\{x,y\}$ of G, we will associate six cost measures, which summarize the cost incurred so far in the subgraph S which has been reduced onto the edge $\{x,y\}$:

1. $st(\alpha)$ is the minimum cost of a Steiner tree for S, in which x and y appear in the same tree.
2. $dt(\alpha)$ is the minimum cost of two disjoint trees for S including all targets, one tree involving x and the other y.
3. $yn(\alpha)$ is the minimum cost of a Steiner tree for S, which includes x but not y.
4. $ny(\alpha)$ is the minimum cost of a Steiner tree for S, which includes y but not x.
5. $nn(\alpha)$ is the minimum cost of a Steiner tree for S, which includes neither x nor y.
6. none(α) is the cost of omitting all vertices of S from a Steiner tree.

Initially, no reduction of the graph has been done. As initial measures of costs, we set the measures for each arc $\alpha = (x,y)$ corresponding to an edge $e = \{x,y\}$ as follows:

1. $st(\alpha) = \text{cost}(e)$.
2. $dt(\alpha) = 0$.
3. $yn(\alpha) = \text{BIG}$ if $y \in T$, 0 otherwise.
4. $ny(\alpha) = \text{BIG}$ if $x \in T$, 0 otherwise.
5. $nn(\alpha) = \text{BIG}$ if $x \in T$ or $y \in T$, 0 otherwise.
6. none$(\alpha) = \text{BIG}$ if $x \in T$ or $y \in T$, 0 otherwise.

A cost of BIG is incurred whenever an attempt is made to omit a target vertex.

These initial arc costs are used to recalculate arc costs as the graph is reduced. The graph is reduced by repeated deletion of degree-2 vertices, so suppose at some point there is a triangle $\{x,y,z\}$ in which y is a degree-2 vertex. Costs have been computed for (x,y) and (y,z); we use these to recalculate the costs for (x,z) prior to deleting y, as follows. Let $L = (x,y)$, $R = (y,z)$, and $M = (x,z)$. The costs are updated for M using six recurrences:

1. $st(M) = \min [st(M) + \min [dt(L) + st(R), st(L) + dt(R), yn(L) + ny(R)],$
   $dt(M) + st(L) + st(R)],$

2. $dt(M) = dt(M) + \min [dt(L) + st(R), st(L) + dt(R), yn(L) + ny(R)]$,
3. $ny(M) = ny(M) + \min [ny(L) + st(R), \text{none}(L) + ny(R)]$,
4. $yn(M) = yn(M) + \min [yn(L) + \text{none}(R), st(L) + yn(R)]$,
5. $nn(M) = \min [nn(M) + \text{none}(L) + \text{none}(R), \text{none}(M) + nn(L) + \text{none}(R),$
   $\text{none}(M) + \text{none}(L) + nn(R), \text{none}(M) + ny(L) + yn(R)]$,
6. $\text{none}(M) = \text{none}(M) + \text{none}(L) + \text{none}(R)$.

These measures are computed symmetrically for (z, x), after which y is deleted. Repeating this process, ultimately G is a single edge $\{x, y\}$ with arcs $\alpha = (x, y)$ and its reverse. At this point the cost of the minimal Steiner tree for G, mincost(G, T) is taken to be $\min [st(\alpha), yn(\alpha), ny(\alpha), nn(\alpha)]$. This completes the algorithm for finding the cost of a Steiner tree for G.

We establish timing, and then correctness, for this algorithm.

Lemma 5.1. For an n-vertex 2-tree G and a target set T, mincost(G, T) can be computed in time which is linear in n.

Proof. The algorithm proceeds as follows. Given G, enter all degree-2 vertices of G on a queue. For every arc (both directions of every edge) of G, compute the initial cost measures. There are $O(n)$ arcs and $O(1)$ measures per arc; since membership in T can be tested in $O(1)$ time, the initialization is $O(n)$.

Repeat the following steps until the remaining graph is a single edge. Take the first vertex y from the queue; locate y's two neighbors x and z. Apply the six recurrences to recompute the six cost measures for (x, z) and (z, x). Delete y; if either x or z now has degree 2, enter it on the queue. Now return to process the next queue element. Each iteration of this process is $O(1)$; since a vertex is eliminated in each iteration, the total time is $O(n)$.

Obtaining the overall cost mincost (G, T) from the last edge is $O(1)$, and thus mincost(G, T) is computed in $O(n)$ time. ∎

Lemma 5.2. For an n-vertex 2-tree G and a target set T, mincost(G, T) is the cost of ST(G, T).

Proof. ST(G, T) has cost at most mincost(G, T), since in computing mincost, a cost of BIG is incurred whenever a target vertex is omitted. That ST(G, T) has cost at least mincost(G, T) depends on the following observations. Mincost(G, T) is computed as the minimum, for some arc $\alpha = (x, y)$, of $st(\alpha)$, $yn(\alpha)$, $ny(\alpha)$, and $nn(\alpha)$. A Steiner tree must contain some subset of $\{x, y\}$, and all four possibilities are considered. The arc a separates G into components each of which is a 2-tree; their union is G, and their pairwise intersection is $\{x, y\}$. A Steiner tree in G induces forests in each of these components; each is minimum cost subject to the constraint that x and y are used similarly in all. This observation validates the updating process which sums information obtained from each component in turn.

Finally, in a component induced by an arc $\alpha = (x, y)$, the edge $\{x, y\}$ forms the base of a triangle $\{x, z, y\}$; the costs for the entire component are properly computed by the recurrence relations given. In order to show this, we need only establish that all

possible ways of constructing each type of tree are included; this in turn is an easy application of the principle of inclusion/exclusion. ∎

Lemmas 5.1 and 5.2, together with Theorem 4.1, establish that

Theorem 5.3. Steiner trees in n-vertex partial 2-trees can be found in time which is linear in n. ∎

VI. CONCLUSIONS

We have given a significant generalization of Farley's method of constructing minimum IFI networks. In the process, we have established useful characterization theorems. Future research on network applications in this area should consider immunity whenever fewer than m nonisolated failures occur. Farley [3] notes that $(m+1)$-trees are such networks; this suggests as an initial problem the recognition and completion of partial k-trees. Even for $k = 3$, the techniques of this paper do not appear applicable, at least in their current form. Thus the recognition of partial 3-trees is a fine area for further research.

Future research on graph-theoretic extensions of this work should consider the use of our results in solving subgraph homeomorphism problems. Except for small fixed graphs, subgraph homeomorphism problems are open [12]. It would be of interest to ascertain the usefulness of our techniques in their solution.

We have also shown that Steiner trees in partial 2-trees can be found in linear time, filling the gap between outerplanar graphs and planar graphs. This method will likely prove useful for many classes of sparse networks. In a practical setting, many improvements to the algorithm could be effected, such as avoiding the necessity of "virtual" edges of cost BIG; such practical considerations would, we believe, make the algorithm quite appealing computationally.

We thank Art Farley and Andrzej Proskurowski for pointing out related work, and for their helpful comments. One of us (C.J.C.) was partially supported by NSERC Canada Grant A5047.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA (1974).

[2] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications.* Macmillan, London (1976).

[3] A. M. Farley, Networks immune to isolated failures. *Networks* 11 (1981) 255–268.

[4] A. M. Farley and A. Proskurowski, Computation of the center and diameter of outerplanar graphs. *Discrete Appl. Math.* 2 (1980) 185–191.

[5] A. M. Farley and A. Proskurowski, Extremal graphs with no disconnecting independent vertex set or matching. University of Oregon Computer and Information Science, Technical Report CIS-TR-80-21 (1980).

[6] M. R. Garey and D. S. Johnson, The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math.* 32 (1977) 826–834.

[7] M. R. Garey, D. S. Johnson, and L. Stockmeyer, Some simplified NP-complete graph problems. *Theor. Comput. Sci.* 1 (1976) 237–267.

[8] S. L. Hakimi, Steiner's problem in graphs and its implications. *Networks* 1 (1972) 113–145.
[9] S. L. Hakimi and A. T. Amin, On the design of reliable networks. *Networks* 3 (1973) 241–260.
[10] F. Harary, *Graph Theory*. Addison-Wesley, Reading, MA (1969).
[11] R. M. Karp, Reducibility among combinatorial problems. In *Complexity of Computer Computations*. R. E. Miller and J. W. Thatcher, Eds. Plenum, New New York 1972, pp. 85–104.
[12] A. LaPaugh and R. L. Rivest, The subgraph homeomorphism problem. In *Proceedings, Tenth ACM Symposium on the Theory of Computing*, 1978, pp. 40–50.
[13] P. C. Liu and R. C. Geldmacher, Graph reducibility. In *Proceedings, Seventh Southeastern Conference on Combinatorics, Graph Theory, and Computing*, 1976, pp. 433–455.
[14] P. C. Liu and R. C. Geldmacher, An $O(\max(m, n))$ algorithm for finding a subgraph homeomorphic to K_4. *Proceeding Eleventh Southeastern Conference on Combinatorics, Graph Theory, and Computing*, 1980, pp. 597–609.
[15] D. J. Rose, On simple characterizations of k-trees. *Discrete Math.* 7 (1974) 317–322.
[16] R. E. Tarjan, Depth-first search and linear graph algorithms. *SIAM J. Comput.* 1 (1972) 146–160.
[17] J. A. Wald and C. J. Colbourn, Steiner trees in outerplanar graphs. In *Proceedings, Thirteenth Southeastern Conference on Combinatorics, Graph Theory, and Computing*, 1982, pp. 15–22.

Received May 10, 1982.
Accepted September 30, 1982.