Microelectron. Reliab., Vol. 23, No. 5, pp. 837–840, 1983.
Printed in Great Britain.

0026–2714/83$3.00+.00
© 1983 Pergamon Press Ltd.

STEINER TREES IN PROBABILISTIC NETWORKS†

JOSEPH A. WALD and CHARLES J. COLBOURN
Department of Computational Science, University of Saskatchewan,
Saskatoon S7N 0W0, Canada

(*Received for publication* 16 *December* 1982)

Abstract—Various network reliability problems are $\#P$-complete, however, certain classes of networks such as series-parallel networks, admit polynomial time algorithms. We extend these efficient methods to a superclass of series-parallel networks, the partial 2-trees. In fact, we solve a more general problem: given a probabilistic partial 2-tree and a set T of target nodes, we compute in linear time the probability of obtaining a subgraph connecting all of the target nodes. Equivalently, this is the probability of obtaining a Steiner tree for T. The algorithm exploits a characterization of partial 2-trees as graphs with no subgraph homeomorphic to K_4.

1. INTRODUCTION

Transmission of data in communications networks is a challenging problem with many important applications. One essential aspect of this is *network reliability*; the ability of the network to survive random or purposeful attack. Many different notions of network reliability are extant [1,2,3]; roughly, they divide into the *deterministic* and the *probabilistic* models. We are concerned here with the latter.

A network can be modelled as a *probabilistic graph*, a graph in which each edge has an associated probability of appearance (see [4] for graph theoretic terminology). This edge probability describes the likelihood that the edge (or associated communication link) is available. These probabilities are assumed to be statistically independent. This probabilistic network model has been used in many contexts [1,3], notably in the analysis of telephone switching networks [5].

The primary reliability problem here is to determine *terminal-pair connection probabilities* for a network N, the probability $p(N;s,t)$ that communication is enabled between a source s and a destination t. The second major reliability problem arises in broadcasting: what is the probability that all vertices are connected? This is termed *probabilistic connectedness*.

Many algorithms to solve these reliability problems have been proposed (see [2] and references therein). One major class of methods exploits a "factoring theorem" which expresses the reliability of a network in terms of the reliabilities of two smaller networks [6,7]. A second major class relies on enumeration of paths from source to destination [8]. All known exact methods require exponential time in the worst case; this is not at all surprising in the light of Valiant's result that terminal-pair connection and probabilistic connectedness are $\#P$-complete [9].

In the restricted case of series-parallel networks, however, there is an efficient solution for terminal-pair connection. A network N is *series-parallel* with source s and destination t if it satisfies one of the following.

1. N contains a single edge, $\{s,t\}$.
2. There are two series-parallel networks, N' and N'', whose union is N. N' connects s to r, N'' connects r to t, and N' and N'' intersect only at r. This is *series connection*.
3. There are two series-parallel networks N' and N'', whose union is N. Both connect s to t, and they intersect only at s and t. This is *parallel connection*.

The reader should beware that there are other definitions of "series-parallel" in the literature. For our series-parallel networks, terminal-pair connection from s to t is computed by noting that:

1. For a single edge $\{s,t\}$, terminal-pair connection probability is just the probability of the edge $\{s,t\}$.
2. For the series connection given above, $p(N;s,t) = p(N';s,r)\cdot p(N'';r,t)$.
3. For the parallel connection given above, $p(N;s,t) = p(N';s,t)+p(N'';s,t)-p(N';s,t)\cdot p(N'';s,t)$.

This method is implicit in the Kittredge–Molina formulas given by Lee [5] and is explicitly given by Moskowitz [7]. This method can be used to compute terminal-pair connection probabilities in series-parallel networks in polynomial time. In conjunction with the factoring theorem, it provides an often-used practical method for sparse networks [10,11]. These remarks suggest the practical and theoretical importance of extending the results beyond series-parallel networks.

We extend the result in two ways—by generalizing the problem and by broadening the class of graphs to which it applies. The generalized problem is a *probabilistic Steiner tree* (PST) problem: given a probabilistic graph $G = (V,E)$ and a subset T of its vertices V, what is the probability $P(G,T)$ that all vertices in T are connected? This generalizes terminal-pair connection (where $T = \{s,t\}$) and probabilistic connectedness (where $T = V$).

† Research partially supported by NSERC Canada Grant A5047.

The expanded class of graphs is the class of partial 2-trees. A *2-tree* is defined recursively as follows.

1. K_2 (a single edge) is a 2-tree.
2. Given any 2-tree G, let $\{x, y\}$ be an edge of G. Adding a new vertex z, and the two edges $\{x, z)$ and $\{y, z\}$ produces a 2-tree.

Partial 2-trees are simply subgraphs of 2-trees.

We develop a linear time algorithm for the PST problem on partial 2-trees. This employs the following equivalence.

Theorem 1.1. [12] Partial 2-trees are exactly those graphs with no subgraph homeomorphic to K_4.

This equivalence demonstrates that series-parallel networks are partial 2-trees and therefore that our algorithm indeed generalizes known reliability methods.

Section 2 introduces a linear time algorithm for adding edges to a partial 2-tree to construct a 2-tree; added edges are "virtual" in the sense that they have probability 0. Finally, Section 3 combines this completion method with a linear time algorithm to solve PST on 2-trees.

2. COMPLETING PARTIAL 2-TREES

We consider here the problem of adding edges to a graph to construct a 2-tree, whenever this is possible; we describe a linear time algorithm for this problem. 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 [13, 14]. 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 1.1 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 cut vertex v, we retain a representative of each biconnected component among the descendants of v; this representative is one of v's neighbours 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 representatives. 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 two 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 neighbours 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 now has degree two, add it to the queue. Return to process the next queue element.

Using this algorithm, we have theorem 2.1.

Theorem 2.1. 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 two vertex v with neighbours 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. *Q.E.D.*

3. PROBABILISTIC STEINER TREES

Our PST algorithm for partial 2-trees operates in two phases. The first completes the graph to a 2-tree, using the method of Section 2. It is essential that we do not change the reliabilities in the process. In order to ensure this, every added edge is given an availability probability of 0. Thus these added edges will never be chosen.

The second phase involves solving PST for 2-trees. Our algorithm to do this is based on a separation property of 2-trees [15] which has been exploited previously [16, 17]: 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 reliabilities for the 2-tree by finding reliabilities for each of the component 2-trees, as outlined in [8]. We develop this idea formally here.

Our strategy is to reverse the recursive construction process of the 2-tree—i.e. repeatedly eliminate vertices of degree two 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 two, on the arcs (x, z) and (z, x), prior to deleting y. This summary information encodes information about reliabilities 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 nonempty target set T;

every edge e of G has an associated probability p_e. With each arc $\alpha = (x, y)$ corresponding to an edge $\{x, y\}$ of G we will associate six probability measures, which summarize the probabilities examined in the subgraph S which has so far been reduced onto the edge $\{x, y\}$:

1. $sc(\alpha)$ is the probability of obtaining a subgraph of S connecting all target vertices in S in addition to both x and y;
2. $dc(\alpha)$ is the probability of obtaining a subgraph of S in which each target vertex is connected to x or to y, but there is no path from x to y;
3. $yn(\alpha)$ is the probability of obtaining a subgraph of S not containing y in which all target vertices are connected to x;
4. $ny(\alpha)$ is the probability of obtaining a subgraph of S not containing x in which all target vertices are connected to y;
5. $nn(\alpha)$ is the probability of obtaining a subgraph of S not containing x or y in which all target vertices are connected to one another;
6. $none(\alpha)$ is the probability that S contains no target vertices.

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. $sc(\alpha) = p_e$.
2. $dc(\alpha) = 1 - p_e$.
3. $yn(\alpha) = 0$ if $y \in T$, $1 - p_e$ otherwise.
4. $ny(\alpha) = 0$ if $x \in T$, $1 - p_e$ otherwise.
5. $nn(\alpha) = 0$.
6. $none(\alpha) = 0$ if $x \in T$ or $y \in T$, 1 otherwise.

Probabilities of 0 arise whenever an attempt is made to omit a target vertex.

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

1. $sc(M) = sc(M) \cdot (sc(L) \cdot sc(R) + sc(L) \cdot dc(R) + dc(L) \cdot sc(R) + yn(L) \cdot ny(R)) + dc(M) \cdot sc(L) \cdot sc(R)$.
2. $dc(M) = dc(M) \cdot (sc(L) \cdot dc(R) + dc(L) \cdot sc(R) + yn(L) \cdot ny(R))$.
3. $yn(M) = yn(M) \cdot (sc(L) \cdot yn(R) + yn(L) \cdot none(R))$.
4. $ny(M) = ny(M) \cdot (ny(L) \cdot sc(R) + none(L) \cdot ny(R))$.
5. $nn(M) = nn(M) \cdot none(L) \cdot none(R) + none(M) \cdot (nn(L) \cdot none(R) + none(L) \cdot \mathrm{nn}(R) + ny(L) \cdot yn(R))$.
6. $none(M) = none(M) \cdot none(L) \cdot none(R)$.

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 overall probability of obtaining a subgraph connecting all target vertices, $P(G, T)$, is taken to be $sc(\alpha) + yn(\alpha) + ny(\alpha) + nn(\alpha)$. This completes the algorithm for solving PST.

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

Lemma 3.1. For an n-vertex 2-tree G and a target set T, $P(G, T)$ can be computed in time which is linear in n.

Proof. The algorithm proceeds as follows. Given G, enter all degree two vertices of G on a queue. For every arc (both directions of every edge) of G, compute the initial probability measures. There are $0(n)$ arcs and $0(1)$ measures per arc; since membership in T can be tested in $0(1)$ time, the initialization is $0(n)$.

Repeat the following steps until the remaining graph is a single edge. Take the first vertex y from the queue and locate y's two neighbours x and z. Apply the six recurrences to recompute the six probability measures for (x, z) and (z, x). Delete y. If either x or z now has degree two, enter it on the queue. Now return to process the next queue element. Each iteration of this process is $0(1)$. Since a vertex is eliminated in each iteration, the total time is $0(n)$.

Obtaining the overall probability $P(G, T)$ from the last edge is $0(1)$, and thus $P(G, T)$ is computed in $0(n)$ time. *Q.E.D.*

Lemma 3.2. For an n-vertex 2-tree G and a target set T, $P(G, T)$ is the probability of connection for the target vertices in T.

Proof. $P(G, T)$ is computed as the sum for some arc $\alpha = (x, y)$, of $st(\alpha)$, $yn(\alpha)$, $ny(\alpha)$ and $nn(\alpha)$. A subgraph must contain some subset of $\{x, y\}$ and all four possibilities are considered. The arc α separates G into components, each of which is a 2-tree; their union is G and their pairwise intersection is $\{x, y\}$. A subgraph in G induces subgraphs in each of these components, subject to the constraint that x and y are used similarly in all. This observation validates the updating process which combines 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 probabilities 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 subgraph are included. This in turn in an easy application of the principle of inclusion/exclusion.

Lemmas 3.1 and 3.2, together with theorem 2.1, establish theorem 3.3.

Theorem 3.3. The probabilistic Steiner tree problem for n-vertex partial 2-trees can be solved in time which is linear in n. *Q.E.D.*

Theorem 3.3 is a substantial generalization of known techniques for computing network reliability and is especially valuable in the light of the $\# P$-completeness of network reliability in general. One area for future research would be the solution of reliability and Steiner tree problems on k-trees. However, the usefulness of this depends highly on first being able to characterize and complete partial k-trees. This appears to be an excellent problem for further study.

REFERENCES

1. H. Frank and I. T. Frisch, *Communications, Transmission, and Transportation Networks.* Addison–Wesley (1971).
2. C. L. Hwang, F. A. Tillman and M. H. Lee, System-reliability evaluation: techniques for complex/large systems—a review, *IEEE Trans. Reliab.* **R-30**, 416–423 (1981).
3. R. S. Wilkov, Analysis and design of reliable computer networks, *IEEE Trans. Commun.* **COM-20**, 660–678 (1972).
4. J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications.* Macmillan (1976).
5. C. Y. Lee, Analysis of switching networks, *Bell Syst. Tech. J.* **29**, 1287–1315 (1955).
6. H. Mine, Reliability of physical systems, *I.R.E. Trans. Circuit Theory* **CT-6**, 138–151 (1959).
7. F. Moskowitz, The analysis of redundancy networks, *A.I.E.E. Trans. Commun. Electron.* **39**, 627–632 (1958).
8. Y. Fu and S. S. Yau, A note on the reliability of communication networks, *J. SIAM* **10**, 469–474 (1962).
9. L. G. Valiant, The complexity of enumeration and reliability problems, *SIAM J. Comput.* **8**, 410–421 (1979).
10. K. B. Misra, An algorithm for the reliability evaluation of physical systems, *IEEE Trans. Reliab.* **R-19**, 146–151 (1970).
11. K. B. Misra and T. S. M. Rao, Reliability analysis of redundant networks using flow graphs, *IEEE Trans. Reliab.* **R-19**, 19–24 (1970).
12. J. A. Wald and C. J. Colbourn, Steiner trees, partial 2-trees and minimum IFI networks, *Networks* **13**, 159–167 (1983).
13. A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Addison-Wesley (1974).
14. R. E. Tarjan, Depth-first search and linear graph algorithms, *SIAM J. Comput.* **1**, 146–160 (1972).
15. D. J. Rose, On simple characterizations of k-trees, *Discrete Math.* **7**, 317–322 (1974).
16. A. M. Farley and A. Proskurowski, Computation of the center and diameter of outerplanar graphs, *Discrete appl. Math.* **2**, 185–191 (1980).
17. J. A. Wald and C. J. Colbourn, Steiner trees in outerplanar graphs, *Proc. Thirteenth Southeastern Conf. Combinatorics, Graph Theory and Computing*, pp. 15–22. Boca Raton, Fl (1982).
18. J. A. Wald and C. J. Colbourn, Computing reliability for a generalization of series-parallel networks, *Proc. Twentieth Allerton Conf. Communications, Control, Computing*, pp. 25–26. Allerton, Il (1982).