Resilience of Partial k-tree Networks

Erick Mata-Montero

CIS-TR-89-09
October 20, 1989

Abstract

The resilience of a network is the expected number of pairs of nodes that can communicate. Computing the resilience of a network has been shown to be a #P-complete problem for planar networks and to take $\mathcal{O}(n^2)$ time for n-node partial 2-tree networks. We present an $\mathcal{O}(n)$ time algorithm to compute the resilience of partial 2-tree networks on n-nodes, and, for a fixed k, an $\mathcal{O}(n^2)$ algorithm to compute the resilience of n-node partial k-tree networks given with an embedding in a k-tree.

Department of Computer and Information Science
University of Oregon

Resilience of Partial k-tree Networks

Erick Mata-Montero *
Department of Computer and Information Science
University of Oregon, Eugene, Oregon 97403, USA

September 25, 1989

Abstract

The resilience of a network is the expected number of pairs of nodes that can communicate. Computing the resilience of a network has been shown to be a #P-complete problem for planar networks and to take $\mathcal{O}(n^2)$ time for n-node partial 2-tree networks. We present an $\mathcal{O}(n)$ time algorithm to compute the resilience of partial 2-tree networks on n-nodes, and, for a fixed k, an $\mathcal{O}(n^2)$ algorithm to compute the resilience of n-node partial k-tree networks given with an embedding in a k-tree.

1 Introduction

Computer communication networks perform communication tasks in an environment in which several kinds of failures may occur. Failures may arise at the software level (e.g. a routing algorithm that does not detect an operational path, although one exists) and at the topological level (e.g. natural catastrophes and component wear out). We use *probabilistic graphs* to model networks in which only failures due to random component wear out may occur. A probabilistic graph is a graph $G = (V, E)$ such that each edge has an associated *probability of operation*. The probability of operation of an edge e in E is a fixed precision real number p_e such that $0 \leq p_e \leq 1$. Edges are in either *operational (up)* or *failed (down)* state and their failures are statistically independent. A network is a probabilistic graph in which nodes represent communication sites and edges represent bidirectional communication lines.

Given a communication task, the reliability of a network has typically been defined as the probability that the network can perform such a task. For example, if the communication task is a broadcast, the reliability of the network may be defined as the probability that the network contains a spanning tree. If the communication task consists of sending information between two specified sites, the reliability of the network may be defined as the probability that the network contains a path between the two corresponding nodes. In the former example the reliability measure is called the *all-terminal reliability*, in the latter the measure is called the *two-terminal*

*Research supported in part by the Office of Naval Research contract N00014-86-K-0419.

reliability. These two measures, as well as others similarly defined (e.g. *k-terminal reliability*), have been the subject of extensive research (see [6] for an excellent survey).

Most of the traditional reliability problems have turned out to be intractable. For instance, computing the all-terminal reliability of a network is a #P-complete problem [16]. On the other hand, computing the two-terminal reliability of a network is a #P-complete problem even when the network is planar, acyclic, with bounded degree nodes, and with all edges failing with the same probability [15]. Whether the all-terminal reliability problem on planar graphs is #P-complete or not is an open problem [6].

The resilience of a network is an alternative measure of the reliability of the network. In some applications, the concern is not the probability that some collection of nodes remain connected but that the expected number of pairs of nodes that can exchange information be "high." The resilience of a network is the expected number of pairs of nodes that can communicate [7]. Computing the resilience of a network is a #P-complete problem, even when the network is planar [7].

The apparent intractability of the reliability problems has lead to the development of efficient approximation algorithms and of exact algorithms for restricted classes of graphs (see [6] for a comprehensive discussion). The class of partial k-trees has become the subject of a large body of research for two main reasons. First, the class of partial k-trees contains several important classes of graphs (e.g. series-parallel graphs and chordal graphs with bounded clique size, [5]). Secondly, some important NP-complete graph problems have polynomial time solutions when restricted to the class of partial k-trees [13]. For example, Arnborg and Proskurowski [3] prove that, for a fixed k, the all-terminal and two-terminal reliability of partial k-tree networks with fail-safe nodes can be computed in linear time (given a suitable embedding of the partial k-tree in a k-tree) . On the other hand, Colbourn [7] proves that the resilience of a partial 2-tree network with fail-safe nodes is computable in quadratic time.

This report is organized as follows. In section 2 we present some basic terminology. In section 3 we introduce the reduction paradigm defined in [3], and use it to develop an $\mathcal{O}(n^2)$ time algorithm to compute the resilience of partial k-tree networks (given a fixed k and an embedding in a k-tree). Finally, in section 4 we present a linear time algorithm to compute the resilience of partial 2-trees.

2 Terminology

Except for a few explicitly defined concepts, we use the basic graph theoretic terminology in [12]. Throughout this paper we assume that all graphs are probabilistic. Let $G = (V, E)$ be a graph. A *clique* of G is a (not necessarily maximal) complete subgraph of G. A *k-clique* is a clique that has exactly k nodes. A graph $H = (V_H, E_H)$ is a *partial graph* of G, denoted $H \subseteq G$, if H is a spanning subgraph of G. The *operational subgraph* of G is the partial graph of G whose edges are the operational edges in G. We use $P_G[H]$ to denote the probability that the operational subgraph of G is H. Clearly

$$P_G[H] = \prod_{e \in E_H} p_e \times \prod_{e \in E - E_H} (1 - p_e)$$

We extend the definition of P_G to the domain of sets of partial graphs of G in the natural way. Let S be a set of partial graphs of G. $P_G[S]$ denotes the probability that the operational edges of G induce a graph in S. Therefore $P_G[S] = \sum_{H \in S} P_G[H]$.

Let $H = (V_H, E_H)$ be a subgraph of G and u, v be two nodes in V_H. We say that u *is connected to* v *via* H (denoted $u \overset{H}{\sim} v$) iff there is a path, consisting of zero or more edges from E_H, that connects u to v. When $H = G$ we prefer the notation $u \sim v$ over $u \overset{G}{\sim} v$. We define sets of partial graphs of H by stating connectivity conditions for nodes of H. For example, if u, v are two distinguished nodes of H, $u \overset{H}{\sim} v$ denotes the set of partial graphs of H such that u is connected to v via H. So, $P_G[u \sim v]$ is the probability that u and v can communicate (i.e., the 2-terminal reliability of G). A *connectivity condition* over a network $G = (V, E)$ is a boolean expression with terms of the form $u \overset{H}{\sim} v$, where H is any subgraph of G, and u, v are nodes of G. Let CC be a connectivity condition over G. It is easy to prove that if all connections ($\sim$) in CC are of the form $\overset{H}{\sim}$, for a fixed $H = (V_H, E_H)$, then $P_G[CC] = P_H[CC]$. In such a case, edges in $E \backslash E_H$ are called *irrelevant*.

The set of all connected components of a graph $G = (V, E)$ define a partition π of V such that each block in π contains the nodes of one connected component of G. We further extend the function P_G so that $P_G[\pi]$ denotes the probability that the connected components of G correspond to π.

The resilience $Res(G)$ of a network $G = (V, E)$ is the expected number of (unordered) pairs of nodes of G that can communicate. Pairs of the form $\{u, u\}$ are not counted. The resilience problem consists of computing the resilience of a network. We can formulate $Res(G)$ as

$$Res(G) = \sum_{H \subseteq G} P_G[H] \times Pairs(H)$$

where $Pairs(H)$ is the number of pairs $\{u, v\}$ of nodes in V such that $u \overset{H}{\sim} v$ and $u \neq v$.

For any node x in V, let $E^x(G)$ be the expected number of nodes y in G (including x) such that $y \sim x$. Then

$$E^x(G) = \sum_{H \subseteq G} P_G[H] \times \sum_{y \in V} Connected(H, x, y)$$

where

$$Connected(H, x, y) = \begin{cases} 1 & \text{if } x \overset{H}{\sim} y \\ 0 & \text{otherwise} \end{cases}$$

Simple algebraic manipulation gives

$$E^x(G) = \sum_{y \in V} P_G[x \sim y]$$

So we can express $Res(G)$ in terms of $E^x(G)$ as follows

$$\begin{aligned} Res(G) &= \sum_{H \subseteq G} P_G[H] \times Pairs(H) \\ &= \sum_{H \subseteq G} P_G[H] \times \frac{1}{2} \sum_{x \in V} \sum_{\substack{y \in V \\ y \neq x}} Connected(H, x, y) \\ &= \frac{1}{2} \sum_{x \in V} \sum_{\substack{y \in V \\ y \neq x}} \sum_{H \subseteq G} P_G[H] \times Connected(H, x, y) \\ &= \frac{1}{2} \sum_{x \in V} \sum_{\substack{y \in V \\ y \neq x}} P_G[x \sim y] \end{aligned}$$

Therefore

$$Res(G) = \frac{1}{2} \sum_{x \in V} (E^x(G) - 1) \tag{1}$$

Colbourn [7] uses equation 1 to define an $\mathcal{O}(n^2)$ time algorithm to compute the resilience of a partial 2-tree network. The key of the solution in [7] is a linear time algorithm to compute $E^x(G)$. The algorithms presented in the remainder of this paper also use equation 1 as the starting point to compute the resilience of a network.

3 Resilience of Partial k-trees

Let k be a fixed positive integer. A graph is a *k-tree* iff it satisfies either of the following conditions:

(i) It is the complete graph on k nodes, K_k;

(ii) It has a node v of degree k with completely connected neighbors, and the graph obtained by removing v and its incident edges is a k-tree.

The recursive definition above was first given by Beineke and Pippert [4] as a generalization of the recursive definition of trees (1-trees). Since then, a large body of research has been devoted to the study the class of k-trees and other related classes of graphs (e.g., chordal graphs [11], and partial k-trees [3]). A graph is a *partial k-tree* if it is a partial graph of a k-tree. We refer the reader to [1, 2] for overviews of properties of partial k-trees and to [5, 13] for surveys of classes of graphs related to the class of partial k-trees.

The reduction paradigm

Arnborg and Proskurowski [3] have defined an algorithm design methodology, a *reduction paradigm*, for partial k-trees that leads to the development of efficient algorithms for a variety of NP-complete

problems restricted to partial k-trees. The reduction paradigm assumes that k is a fixed positive integer, and that the input partial k-tree is given with a suitable embedding in a k-tree. For the sake of simplicity we will discuss this reduction paradigm assuming that the input graph is a k-tree rather than a partial k-tree given with an embedding in a k-tree.

The reduction paradigm in [3] uses a dynamic programming approach to compute the solution to a problem X on a (partial) k-tree. First, we associate a state with each k-clique in the graph. The state of each k-clique contains some local information that will be combined with the information in other states to solve problem X. Once each k-clique has been assigned an initial state, we proceed to eliminate $n-k$ nodes of G in some convenient order $v_1, \ldots, v_{n-k}$. Each time we eliminate one node v we destroy a number of k-cliques whose states contain valuable information. So, before destroying those k-cliques we combine their states and save the result as the state of a specific k-clique that is not destroyed by the removal of v. When the $n-k$ nodes have been removed from G we are left with a *root* R of G. The root R is a k-clique whose state contains enough information to solve problem X on G. We need some notation to formalize these ideas.

A *perfect elimination ordering (peo)* of a graph G is an enumeration $v_1, \ldots, v_n$ of the nodes of G such that for each i $(i = 1, \ldots, n)$, the higher numbered neighbors of v_i form a clique. Clearly, we can always find a peo for a k-tree. Furthermore, we can guarantee that for any peo of a k-tree the higher numbered neighbors of each of the first $n-k$ nodes induce a k-clique. A node whose neighborhood induces a k-clique is called a *k-leaf.*

Algorithm 1 presents the reduction paradigm in detail. Let us suppose that we want to solve problem X on a k-tree G. The first step of the algorithm, the initialization step, finds the first $n-k$ nodes of a peo and initializes the state of each k-clique in the graph G. The initial state of each k-clique K is computed by a function $e(K)$. Each reduction step removes one of the $n-k$ nodes in the queue PEO. Upon removal of a node v, the algorithm performs two sub-steps. First it "combines" the states of $k+1$ k-cliques. We use f to denote the function that computes such a combination of states. The result of applying f to the states of the k k-cliques that will be destroyed and to the state of the neighborhood of v is called the "state" of $K^+(v)$ [1]. The second sub-step combines the effect of the edges that connect v to its neighborhood $(K(v))$ and the state of $K^+(v)$. Algorithm 1 represents this second combination of information as the computation of $g(state(K^+(v)), S(v))$. The termination step extracts the solution to problem X from the state of the root R and the effect of the edges in R.

The state information in each k-clique describes solutions to a problem (usually a generalization of the original problem) restricted to the subgraph induced by the nodes in the k-clique and by those removed nodes that the k-clique separates from from all non-removed nodes excluding all edges between nodes in the k-clique. The specification of an algorithm that uses the reduction paradigm described above consists of five main parts. First we define the information that is maintained in the state of each k-clique. Then we specify how to compute e, f, g, and h in

[1] The "state" of $K^+(v)$ is ephemeral; we compute it once and immediately use it to update the state of $K(v)$. Once the state of $K(v)$ has been updated, we destroy $K^+(v)$ by removing the node v. So, $state(K^+(v))$ is simply an intermediate value that we calculate to update the state of $K(v)$. We believe that the metaphor of having a state for $K^+(v)$ is useful in understanding and analyzing the functions f and g for specific problems that use this reduction paradigm.

Algorithm 1.

Algorithm 1

Reduction Paradigm
Input: $G = (V, E)$, *a* k*-tree (for a fixed* k*).*

1. *Initialization step.*
   $PEO \leftarrow$ *empty queue.*
   Do $n - k$ *times:*
      Let v *be a* k*-leaf of* $G - PEO$*.*
      Let $K(v)$ *be the (*k*-clique) neighborhood of* v *in* $G - PEO$*.*
      Let $K^+(v)$ *be the* $(k+1)$*-clique induced by* $V(K(v)) \cup \{v\}$*.*
      For all nodes u *in* $V(K(v))$ *do:*
         Let $K^u(v)$ *be the* k*-clique induced by* $V(K^+(v)) \setminus \{u\}$*.*
         $state(K^u(v)) \leftarrow e(K^u(v))$
      Append v *to* PEO*.*
   $state(R) \leftarrow e(R)$*.*
2. *Reduction steps.*
   For each node v *in* PEO*, in order, do:*
      $state(K^+(v)) \leftarrow f(\{state(K^u) \mid u \in V(K^+(v))\})$*.*
      Let $S(v)$ *be the star graph induced by the edges* $\{v, u\}$*,* $\forall u \in V(K(v))$*.*
      $state(K(v)) \leftarrow g(state(K^+(v)), S(v))$*.*
      Remove v *from* G*.*
3. *Termination step.*
   $Solution \leftarrow h(state(R),$ *edges in* $R)$*.*

We need to formalize some concepts before presenting our reduction algorithm to compute the resilience of partial k-tree networks. If K is a k-clique, $v \notin V(K)$ is a *descendant* of K in a peo iff each higher numbered neighbor of v is either a node in $V(K)$ or a descendant of K. The connected components of the subgraph induced by all descendants of K are *branches* on K.

Suppose that we have a peo defining a reduction process. We associate two subgraphs, $B(K)$ and $B'(K)$, with each k-clique K. These two subgraphs change as we execute the reduction process. We use $B(K)$ to denote the *removed branches* on K, i.e., the subgraph induced by the nodes in the branches on K that have been (completely) removed. $B'(K)$ denotes the subgraph induced by the nodes in $V(K \cup B(K))$ without the edges between nodes in $V(K)$. We call $B'(K)$ the *shell* of K. The state of a k-clique K describes solutions to problems restricted to the shell $B'(K)$. The following set of equations describes how $B(K)$ and $B'(K)$ change during the execution of Algorithm 1. Notice that these equations also define $B(K^+(v))$ and $B'(K^+(v))$.

Dynamic definition of $B(K)$ and $B'(K)$ *(annotations on Algorithm 1)*

1. *Initialization step.*
   $B(K(v)) = (\emptyset, \emptyset)$
   $B'(K(v)) = (V(K(v)), \emptyset)$
2. *Reduction steps.*
   Let $K = K(v)$, $K^+ = K^+(v)$, $K^u = K^u(v)$, *and* $S = S(v)$.
   $B(K^+) = \bigcup_{u \in V(K^+)} B(K^u)$.
   $B'(K^+) = \bigcup_{u \in V(K^+)} B'(K^u)$.
   $B(K) =$ *subgraph induced by* $V(B(K^+)) \cup \{v\}$.
   $B'(K) = B'(K^+) \cup S$.
3. *Termination step.*
   $B(R) = G - R$
   $B'(R) = G$ *without the edges in* R

Figure 3.1(a) depicts a 3-tree in which x, y, and z are the nodes of a 3-clique K. After the removal of v_1, v_2, and v_3 , the removed branches $B(K)$ is the shadowed subgraph in Figure 3.1(b) and the shell $B'(K)$ is the graph in Figure 3.1(c).

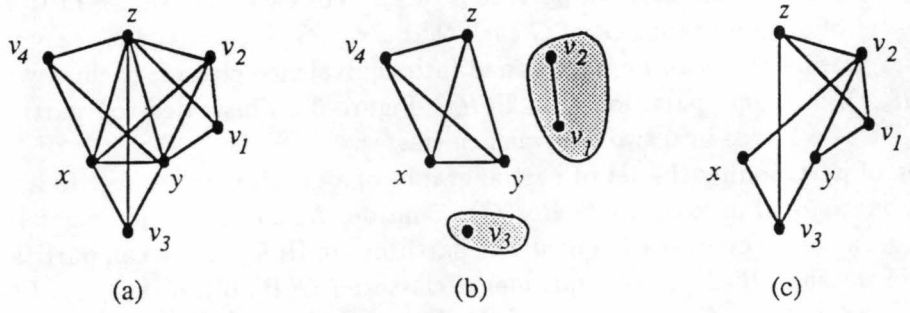


Fig. 3.1 (a) A 3-tree (b) $B(K)$ after three reductions (c) $B'(K)$ after three reductions.

Algorithm for the resilience problem on partial k-trees

We solve the resilience problem on partial k-trees in two steps. First we find an embedding of the n-node partial k-tree in a k-tree. Then we reduce the input graph to a root n times. Algorithm 2 describes the general structure of our solution.

Step 1 of Algorithm 2 can be performed using the algorithm given in [1] and assigning probability 0 to the added edges. Step 3 is justified in section 2. We therefore focus our attention on the body of the loop.

Given a k-tree G and a node x in V, we want to reduce G to R, a k-clique that contains x. As a result of the reduction process, we want to be able to compute $E^x(G)$ from the state of R. The first goal is easy to meet because all k-trees have either zero or more than one k-leaves. So we can always remove a k-leaf different from x. To achieve the second goal we need to define the information that will be maintained in each k-clique, and the functions e, f, g, and h (cf., Algorithm 1).

Algorithm 2

Resilience of Partial k-trees *(for a fixed k)*
Input: *$G' = (V, E')$, a partial k-tree.*

1. *Find an embedding of G' in a k-tree $G = (V, E)$.*
2. *For each node x in V do* $\{$*Compute $E^x(G)$*$\}$
   Initialize the state of each k-clique of G.
   Reduce G to R, a k-clique that contains x.
   Compute $E^x(G)$ from the state of R.
3. $Res(G) = \frac{1}{2}\sum_{x \in V}(E^x(G) - 1)$.

Some notation is in order. Let W be a subset of nodes of G. The *projection of the connected components of G onto W ($Proj(G, W)$)* is the partition of W defined by intersecting each connected component of G with W. Let us now consider $H = (V_H, E_H)$, a graph such that $V_H \subseteq V$. We use $\Pi(H)$ to denote the set of all partitions of V_H. For each partition π in $\Pi(H)$, $PG(G, H, \pi)$ denotes the set of partial graphs G' of G such that $Proj(G', V_H) = \pi$. It is easy to verify that the set of partial graphs of G can be partitioned into equivalence classes, each of which corresponds to $PG(G, H, \pi)$, for some partition π in $\Pi(H)$. Figure 3.2 illustrates the partition of the set of partial graphs of a 2-tree into two equivalence classes.

The idea of partitioning the set of partial graphs of a graph with respect to a fixed set of nodes is crucial in our algorithm to compute $Res(G)$. Consider K, a k-clique of a partially reduced k-tree G. Let $\pi_1, \ldots, \pi_q$ be an enumeration of all the partitions in $\Pi(K)$ [2]. We can partition the set of partial graphs of the shell $B'(K)$ into q equivalence classes: $PG(B'(K), K, \pi_1), \ldots, PG(B'(K), K, \pi_q)$. The state of the k-clique K contains the following statistical information about each equivalence class of partial graphs of the shell $B'(K)$:

$$\{s(\pi, K), E(\pi, K, C) \mid \pi \text{ is a partition in } \Pi(K) \text{ and } C \in \pi\}$$

where $s(\pi, K)$ denotes the probability that π is the projection onto K of the connected components of a partial graph of $B'(K)$. So

$$s(\pi, K) = \sum_{H \in PG(B'(K), K, \pi)} P_{B'(K)}[H] \tag{2}$$

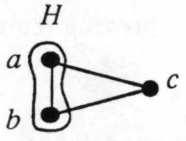
[2] Notice that, for a fixed value of k, q is constant (although exponential in k).

On the other hand, we define $E(\pi, K, C)$ as

$$E(\pi, K, C) = \sum_{H \in PG(B'(K), K, \pi)} P_{B'(K)}[H] \times BN(H, C) \tag{3}$$

where $BN(H, C)$ is the number of *branch nodes* (nodes in $B(K)$) connected to C via H, i.e., $BN(H, C) = |\{y \in B(K) \mid y \overset{H}{\sim} z \text{ for some } z \in C\}|$. It is easy to verify that if the probability $s(\pi, K) \neq 0$, then $E(\pi, K, C)/s(\pi, K)$ is a conditional expected value, namely, the expected number of nodes in $B(K)$ that are connected to C, via H, given that π is the projection onto K of the connected components of a partial graph H of $B'(K)$.

Equations 2 and 3 also define the ephemeral state of the $(k+1)$-clique $K^+(v)$ in Algorithm 1. The next four lemmata define the initialization, reduction, and termination steps of a quadratic time algorithm for the resilience problem.



(a)

Partition π	Graphs in $PG(G,H,\pi)$
{ {a,b} }	
{ {a} , {b} }	

(b)

Fig. 3.2 (a) A 2-tree G. (b) Equivalence classes induced by partitions in $\Pi(H)$.

Lemma 3.1 (initialization)*Let G be a k-tree, K a k-clique of G, π a partition of K, and C a block of π. Then*

(i) $s(\pi, K) = \begin{cases} 1 & \text{if } \pi \text{ consists of singletons only} \\ 0 & \text{otherwise} \end{cases}$

(ii) $E(\pi, K, C) = 0$

Proof: Immediate from the definition of $s(\pi, K)$ and $E(\pi, K, C)$ ∎

Let us now consider the reduction step. Let G be a (partially reduced) k-tree, and v be a k-leaf of G with neighborhood $K(v)$. In the following we use K to denote $K(v)$, K^+ to denote $K^+(v)$,

and K^u to denote $K^u(v)$. Let the nodes of K^+ be $u_1, \ldots, u_{k+1}$. The reduction step consists of two parts. First we compute the state of K^+ by combining the states of K^u, for all u in $V(K^+)$. Then we update the state of K by considering the state of K^+ and the effect of the edges that connect v to K.

We will introduce some additional notation. Let π be a partition. Following [3], we use π/u to denote the partition obtained by removing u from its block in π and then removing the block if it became empty. Moreover, for partitions π_1 and π_2 let their *join* $(\pi_1 \vee \pi_2)$ be the partition obtained by replacing pairs of intersecting blocks by their union until a partition of the union remains (e.g., $\{\{a,b\},\{c\},\{d\}\} \vee \{\{a,d\},\{b,c\}\} = \{\{a,b,c,d,\}\}$).

Let π^+ be a partition of the nodes in K^+. We want to consider all possible ways of obtaining π^+ as the join of $\pi_{u_1}, \ldots, \pi_{u_{k+1}}$, where π_{u_i} is a partition of the nodes in K^{u_i} $(1 \leq i \leq k+1)$. To that effect, we define the set $T(\pi^+, K^+)$ as follows:

$$T(\pi^+, K^+) = \{(\pi_{u_1}, \ldots, \pi_{u_{k+1}}) \mid \forall i = 1, \ldots, k+1,\ \pi_{u_i} \in \Pi(K^{u_i}) \text{ and } \bigvee_{i=1}^{k+1} \pi_{u_i} = \pi^+\}$$

The following observation is useful in proving lemma 3.2.

Observation 3.1

(i) For each partition π^+ in $\Pi(K^+)$ we can establish a one-to-one correspondence between partial graphs in $PG(B'(K^+), K^+, \pi^+)$ and some $(k+1)$-tuples $(H_1, \ldots, H_{k+1})$ of graphs such that H_i is a partial graph of the shell of K^{u_i}, $1 \leq i \leq k+1$. Formally, there is a bijection ϕ from $PG(B'(K^+), K^+, \pi^+)$ to

$$\bigcup_{\substack{(\pi_{u_1}, \ldots, \pi_{u_{k+1}}) \\ in\ T(\pi^+, K^+)}} PG(B'(K^{u_1}), K^{u_1}, \pi_{u_1}) \times \ldots \times PG(B'(K^{u_{k+1}}), K^{u_{k+1}}, \pi_{u_{k+1}})$$

such that $\phi(H) = (H_1, \ldots, H_{k+1})$ iff $\bigcup_{i=1}^{k+1} H_i = H$.

(ii) Given m finite sets $X_1, \ldots, X_m$ and m real functions $f_1, \ldots, f_m$ with domain $X_1, \ldots, X_m$ respectively,

$$\prod_{i=1}^{m} \sum_{x \in X_i} f_i(x) = \sum_{\substack{(x_1, \ldots, x_m) \\ in\ X_1 \times \ldots \times X_m}} \prod_{i=1}^{m} f_i(x_i)$$

Lemma 3.2 *Let G be a (partially reduced) k-tree network. Let v be the next k-leaf of G to be deleted according to some peo. Let K be the neighborhood of v, and K^+ be the $(k+1)$-clique induced by $V(K) \cup \{v\}$. Let π^+ be a partition in $\Pi(K^+)$, and C be a block in π^+. Then*

(i) $$s(\pi^+, K^+) = \sum_{\substack{(\pi_{u_1}, \ldots, \pi_{u_{k+1}}) \\ in\ T(\pi^+, K^+)}} \prod_{i=1}^{k+1} s(\pi_{u_i}, K^{u_i})$$

(ii) $$E(\pi^+, K^+, C) = \sum_{\substack{(\pi_{u_1},\ldots,\pi_{u_{k+1}}) \\ in\ T(\pi^+,K^+)}} \sum_{i=1}^{k+1} \prod_{\substack{j=1 \\ j\neq i}}^{k+1} s(\pi_{u_j}, K^{u_j}) \times \sum_{\substack{D\in\pi_{u_i} \\ D\subseteq C}} E(\pi_{u_i}, K^{u_i}, D)$$

Proof:

(i) Using the definition of $s(\pi^+, K^+)$ (equation 2) and observation 3.1 (i) we get

$$s(\pi^+, K^+) = \sum_{\substack{(\pi_{u_1},\ldots,\pi_{u_{k+1}}) \\ in\ T(\pi^+,K^+)}} \sum_{\substack{(H_1,\ldots,H_{k+1}) \\ in\ X_1\times\ldots\times X_{k+1}}} P_{B'(K^+)}[H_1 \cup \ldots \cup H_{k+1}]$$

where $X_j = PG(B'(K^{u_j}), K^{u_j}, \pi_{u_j})$, $1 \leq j \leq k+1$.

Notice that the graphs $H_1, \ldots, H_{k+1}$ are edge-disjoint. Besides, edge failures are statistically independent and irrelevant edges can be ignored. So,

$$s(\pi^+, K^+) = \sum_{\substack{(\pi_{u_1},\ldots,\pi_{u_{k+1}}) \\ in\ T(\pi^+,K^+)}} \sum_{\substack{(H_1,\ldots,H_{k+1}) \\ in\ X_1\times\ldots\times X_{k+1}}} \prod_{i=1}^{k+1} P_{B'(K^{u_i})}[H_i]$$

The result follows by observation 3.1 (ii).

(ii) Analogously, we can use the definition of $E(\pi^+, K^+, C)$ (equation 3), observation 3.1 (i), the statistical independence of edge failures, and ignore irrelevant edges to obtain

$$E(\pi^+, K^+, C) = \sum_{\substack{(\pi_{u_1},\ldots,\pi_{u_{k+1}}) \\ in\ T(\pi^+,K^+)}} \sum_{\substack{(H_1,\ldots,H_{k+1}) \\ in\ X_1\times\ldots\times X_{k+1}}} \prod_{j=1}^{k+1} P_{B'(K^{u_j})}[H_j] \times BN(H_1 \cup \ldots \cup H_{k+1}, C)$$

But

$$BN(H_1 \cup \ldots \cup H_{k+1}, C) = \sum_{i=1}^{k+1} \sum_{\substack{D\subseteq C \\ D\in\pi_i}} BN(H_i, D)$$

So, simple algebraic manipulation and observation 3.1 (ii) give the desired result. ∎

We now show how to update the state of K when v is removed. Let S be the star graph induced by the the k edges that link v to K. Let $\Pi'(S)$ denote the set of partitions of nodes in S that have only singletons except for possibly the set containing node v. The set $\Pi'(S)$ models the set of operational subgraphs of S. Edges of S that are operational may cause two or more connected components of the operational subgraph of $B'(K)$ to become connected. We update the state of K by considering the join of pairs of partitions (π_1, π_2) such that π_1 is a partition in $\Pi(K^+)$ and π_2 is a partition in $\Pi'(S)$. Let π be a partition of the nodes in K and

$$PP(\pi, K) = \{(\pi_1, \pi_2) \mid \pi_1 \in \Pi(K^+), \pi_2 \in \Pi'(S),\ and\ (\pi_1 \vee \pi_2)/v = \pi\}$$

The following observation is useful in proving Lemma 3.3.

Observation 3.2 *For each partition π in $\Pi(K)$ we can establish a one-to-one correspondence between the partial graphs in $PG(B'(K), K, \pi)$ [3] and pairs (H_1, H_2) of graphs such that H_1 is a partial graph of the shell $B'(K^+)$, H_2 is a partial graph of S, and $H = H_1 \cup H_2$. Formally, there is a bijection ψ such that*

$$\psi : PG(B'(K), K, \pi) \mapsto \bigcup_{\substack{(\pi_1,\pi_2) \\ in\ PP(\pi,K)}} PG(B'(K^+), K^+, \pi_1) \times PG(S, S, \pi_2)$$

and $\psi(H) = (H_1, H_2)$ iff $H = H_1 \cup H_2$. .

Lemma 3.3 *Let G be a (partially reduced) k-tree. Let v be the next k-leaf of G to be deleted according to some peo. Let K be the neighborhood of v, π be a partition in $\Pi(K)$, and C be a block in π. Then*

(i) $$s(\pi, K) = \sum_{(\pi_1,\pi_2) \in PP(\pi,K)} s(\pi_1, K^+) \times P_S[\pi_2]$$

(ii) $$E(\pi, K, C) = \sum_{(\pi_1,\pi_2) \in PP(\pi,K)} P_S[\pi_2] \times \Big(\sum_{\substack{D \in \pi_1 \\ D \setminus \{v\} \subseteq C}} E(\pi_1, K^+, D) \; + \; r(\pi_1, K^+) \Big)$$

$$\text{where } r(\pi_1, K^+) = \begin{cases} s(\pi_1, K^+) & \text{if there is a block } D \in \pi_1 \text{ such that} \\ & D \setminus \{v\} \subseteq C \text{ and } v \in D \\ 0 & \text{otherwise} \end{cases}$$

Proof: The proof follows from the definitions of $s(\pi, K)$, $E(\pi, K, C)$, the statistical independence of edge failures, and Observation 3.2. We present the proof of (i) only.

(i) By equation 2

$$s(\pi, K) = \sum_{\substack{H\ in \\ PG(B'(K),K,\pi)}} P_{B'(K)}[H]$$

and by observation 3.2

$$s(\pi, K) = \sum_{\substack{(\pi_1,\pi_2)\ in \\ PP(\pi,K)}} \; \sum_{\substack{H_1\ in \\ PG(B'(K^+),K^+,\pi_1)}} \; \sum_{\substack{H_2\ in \\ PG(S,S,\pi_2)}} P_{B'(K)}[H_1 \cup H_2]$$

But $P_{B'(K)}[H_1 \cup H_2] = P_{B'(K^+)}[H_1] \times P_S[H_2]$. So

$$s(\pi, K) = \sum_{\substack{(\pi_1,\pi_2)\ in \\ PP(\pi,K)}} \Big(\sum_{\substack{H_1\ in \\ PG(B'(K^+),K^+,\pi_1)}} P_{B'(K^+)}[H_1] \Big) \times \Big(\sum_{\substack{H_2\ in \\ PG(S,S,\pi_2)}} P_S[H_2] \Big)$$

[3] At this point, $B'(K)$ denotes the shell of K after node v has been removed, i.e., it includes v. K^+ and $B'(K^+)$ were computed before v was removed.

We can use lemmata 3.1, 3.2, and 3.3 to reduce any k-tree G to a k-clique R that contains a specific node x. At this point, we want to compute $E^x(G)$. Notice however that the state of R contains information about the shell $B'(R)$, i.e., we have not considered the effect of the edges between nodes in R. Therefore, before computing $E^x(G)$, we extend the statistics about $B'(R)$ to statistics about G. Let π be a partition in $\Pi(R)$, define

$$E'(\pi, R, x) = \sum_{H \in PG(G,R,\pi)} P_G[H] \times N(H, x)$$

where $N(H, x)$ is the number of nodes y in G (including x) that are connected to x via H. We can then state the following lemma.

Lemma 3.4 *(termination) Let G be a k-tree and R be a root of G obtained by using the reduction paradigm and lemmata 3.1-3.3 to G. Then*

(i) For ech node x in $V(R)$

$$E^x(G) = \sum_{\pi \in \Pi(R)} E'(\pi, R, x)$$

(ii) For each partition π in $\Pi(R)$ and each node x in $V(R)$

$$E'(\pi, R, x) = \sum_{\substack{(\pi_1,\pi_2) \\ \pi_1,\pi_2 \in \Pi(R) \\ \pi_1 \vee \pi_2 = \pi}} P_R[\pi_2] \times (s(\pi_1, R) \times |C| + \sum_{\substack{D \in \pi_1 \\ D \subseteq C}} E(\pi_1, R, D))$$

where C is the block of π that contains x.

Proof:

(i) Recall that we can partition the set of partial graphs of G into equivalence classes each of which is characterized by a partition of the nodes in $V(R)$. So,

$$\begin{aligned} E^x(G) &= \sum_{H \subseteq G} P_G[H] \times \sum_{y \in G} Connected(H, x, y) \quad \text{(by def.)} \\ &= \sum_{\pi \in \Pi(R)} \sum_{H \in PG(G,R,\pi)} P_G[H] \times \sum_{y \in G} Connected(H, x, y) \\ &= \sum_{\pi \text{ in } \Pi(R)} E'(\pi, R, x) \end{aligned}$$

(ii) It suffices to use the definition of $E'(\pi, R, x)$ and to observe that there is a one-to-one correspondence between partial graphs in $PG(G, R, \pi)$ and pairs of partial graphs in

$$\cup_{\substack{(\pi_1,\pi_2) \\ \pi_1,\pi_2 \in \Pi(R) \\ \pi_1 \vee \pi_2 = \pi}} PG(B'(R), R, \pi_1) \times PG(R, R, \pi_2)$$

Therefore, lemmata 3.1-3.4 and the reduction paradigm give us the following theorem.

Theorem 3.1 *Let k be a positive integer number, $G = (V, E)$ be a k-tree on n nodes, and x be a node in V. The expected number of nodes that can communicate with x can be computed in $\mathcal{O}(n)$ time.*

Proof: Correctness follows from lemmata 3.1-3.4. Timing can be verified as follows. Consider Algorithm 3. Algorithm 3 is an implementation of Algorithm 1 that makes explicit the amount of computation involved in the reduction paradigm. By lemmata 3.1-3.4 we know that the functions e, f, g, and h can be computed in constant time. Let us consider the initialization step. The amount of time spent in this step is proportional to the amount of time spent traversing adjacency lists. It is easy to prove that a k-tree has exactly $k \times (k-1)/2 + (n-k) \times k$ edges. In addition, the adjacency list of each node is traversed only two times. So, the initialization step takes $\mathcal{O}(n)$ time. Clearly the synthesis step takes linear time, and the termination step takes constant time. ∎

Algorithm 3

Reduction Paradigm *(detailed description)*
Input: *$G = (V, E)$, a k-tree (for a fixed k).*

1. *Initialization.*
   $PEO \leftarrow$ empty list.
   For all v in V do:
      Traverse Adjacency_List(v) to determine degree(v).
      If degree(v) $= k$ then push v onto stack L.
   Do $n - k$ times:
      $v \leftarrow$ Pop(L).
      Traverse Adjacency_List(v) to find $K(v)$, the neighborhood of v in G.
      $K^+(v) \leftarrow$ $(k+1)$-clique induced by $V(K(v)) \cup \{v\}$.
      $S(v) \leftarrow$ star graph induced by the edges $\{v, u\}$, $\forall u \in V(K(v))$.
      For all nodes u in $K(v)$ do:
         $K^u(v) \leftarrow$ k-clique induced by $V(K^+(v)) \backslash \{u\}$.
         $state(K^u(v)) \leftarrow$ initial information.
         $degree(u) \leftarrow degree(u) - 1$.
         If $degree(u) = k$ then push u onto L.
      Append v to PEO.
      $degree(v) \leftarrow 0$.
   $state(R) \leftarrow e(R)$.
2. *Synthesis.*
   For each v in PEO do:
      $state(K^+(v)) \leftarrow f(\{state(K^u) \mid u \in K^+(v)\})$.
      $state(K(v)) \leftarrow g(state(K^+(v)), S(v))$.

3. *Termination.*
   Solution $\leftarrow$ $h(state(R),$ *edges in* $R)$.

We can combine equation 1 in section 2 and theorem 3.1 to obtain the following corollary.

Corollary 3.1 *Given a fixed k, the resilience of a k-tree network G on n nodes can be computed in $\mathcal{O}(n^2)$ time.*

Algorithm 2 describes how to compute the resilience of a partial k-tree G'. Arnborg, Corneil and Proskurowski [1] give an $\mathcal{O}(n^{k+2})$ time algorithm to find an embedding of a partial k-tree in a k-tree, for a fixed k. Therefore, we can compute $Res(G')$ in $\mathcal{O}(n^{k+2})$ time. However, for $k \leq 3$ the embedding of a partial k-tree in a k-tree can be found in $\mathcal{O}(n)$ time ([18], [14]). We can therefore state the following corollary of theorem 3.1

Corollary 3.2 *Let k be a fixed positive integer. The resilience of a partial k-tree on n nodes can be computed in $\mathcal{O}(n^{k+2})$ time. If the input includes an embedding in a k-tree, or $k \leq 3$, the resilience of the partial k-tree can be computed in $\mathcal{O}(n^2)$ time.*

4 Resilience of partial 2-trees

A graph is a partial 2-tree iff it is a *series-parallel* graph [18]. Series-parallel graphs have been the object of extensive research since they were first defined by Duffin [8, 10, 9]. A graph is series-parallel iff it can be transformed into a forrest via a sequence of *series* and *parallel* reductions. A series reduction replaces a node v of degree two and its incident edges with and edge that links the two neighbors of v. A parallel reduction replaces all edges between two nodes with a single edge. Series-parallel graphs also correspond to the class of graphs that have no subgraph homeomorphic to the complete graph on four nodes K_4[18]. The class of 2-tree graphs is the class of *minimum IFI* graphs (minimum isolated failure immune graphs [10]).

The algorithm presented in this section differs from the one described in section 3 in the following aspects. First of all, we use a slight modification of the reduction paradigm that simplifies some intermediate computations. The modification consists of redefining the shell ($B'(K)$) of each 2-clique K so that $B'(K)$ includes the edge that connects the two nodes in K. This is the same approach used in [col87]. The crucial difference between this algorithm and the one presented in section 3 is that the state of each 2-clique (edge) contains some additional information. Thanks to this extra information, we need to perform only one reduction on the input graph; thus, the resulting algorithm runs in $\mathcal{O}(n)$ time. We also introduce some new notation to facilitate the understanding of the rather long expressions involved in the reduction steps.

Terminology

Let $G = (V, E)$ be a 2-tree and $e = \{x, y\}$ be and edge (2-clique) of G. Suppose we are applying a reduction process on G. The shell of e is the subgraph induced by the nodes x, y, and by those removed nodes that the 2-clique e separates from all non-removed nodes. Initially the shell of e

is e itself. When a node z with neighborhood $\{x, y\}$ is removed (see Figure 4.1) we update the shell of e to $L \cup R \cup M$, where

L = shell of {x,z} prior to the removal of node z.
R = shell of {y,z} prior to the removal of node z.
M = shell of {x,y} prior to the removal of node z.

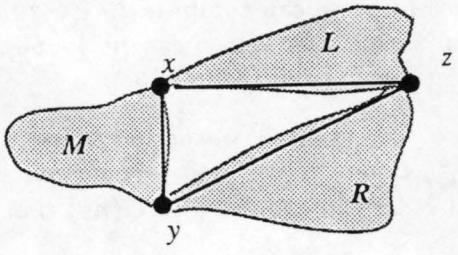


Fig. 4.1 Removing node z.

We need a few more definitions before presenting the statistics that define the state of an edge. Let $e = \{x, y\}$ be and edge of G, and H be the shell of e. Moreover, let $u_1, \ldots, u_l$ and $v_1, \ldots, v_m$ be two lists, not both empty, of nodes in H [4]. We use $E^{u_1 \ldots u_l}_{v_1 \ldots v_m}(H)$ to denote the expected number of nodes t in H such that $t \overset{H}{\sim} u_1, \ldots, t \overset{H}{\sim} u_l$, and $t \overset{H}{\not\sim} v1, \ldots, t \overset{H}{\not\sim} v_m$. Simple algebraic manipulation gives

$$E^{u_1 \ldots u_l}_{v_1 \ldots v_m}(H) = \sum_{t \in H} P_H[t \sim u_1 \wedge \ldots \wedge t \sim u_l \wedge t \not\sim v_1 \wedge \ldots \wedge t \not\sim v_m]$$

Also, let $S^{u_1 \ldots u_l}_{v_1 \ldots v_m}(H)$ be the expected number of ordered pairs (s, t) of nodes in H such that $s \overset{H}{\sim} t$, $s \overset{H}{\sim} u_1, \ldots, s \overset{H}{\sim} u_l$, and $t \overset{H}{\not\sim} v1, \ldots, t \overset{H}{\not\sim} v_m$. Thus

$$\begin{aligned} S^{u_1 \ldots u_l}_{v_1 \ldots v_m}(H) &= \sum_{s \in H} E^{u_1 \ldots u_l s}_{v_1 \ldots v_m}(H) \\ &= \sum_{s \in H} \sum_{t \in H} P_H[s \sim t \wedge t \sim u_1 \wedge \ldots \wedge t \sim u_l \wedge t \not\sim v_1 \wedge \ldots \wedge t \not\sim v_m] \end{aligned}$$

Finally, let $\overline{S}(H)$ be the expected number of ordered pairs (s, t) of nodes in H such that $s \overset{H}{\not\sim} t$, but $s \overset{H}{\sim} x$ *and* $t \overset{H}{\sim} y$ Thus,

$$\overline{S}(H) = \sum_{s \in H} \sum_{t \in H} P_H[s \not\sim t \ \wedge \ s \sim x \ \wedge t \sim y]$$

[4] Unless ambiguity arises, we use $v \in H$ to denote that node v is an element of $V(H)$.

Algorithm for the resilience problem on partial 2-trees

Let $\{x, y\}$ be an edge of G and H be the shell of edge $\{x, y\}$. The following statistics define the state of edge $\{x, y\}$:

1. $P_H[x \sim y]$, the probability that node x is connected to node y via H.
2. $E_y^x(H)$, the expected number of nodes t in H such that $t \overset{H}{\sim} x$ but $t \overset{H}{\not\sim} y$.
3. $E_x^y(H)$, the expected number of nodes t in H such that $t \overset{H}{\sim} y$ but $t \overset{H}{\not\sim} x$.
4. $E^{xy}(H)$, the expected number of nodes t in H such that $t \overset{H}{\sim} x$ and $t \overset{H}{\sim} y$.
5. $S_x(H)$, the expected number of pairs (s, t) of nodes in H such that $s \overset{H}{\sim} t$ but $t \overset{H}{\not\sim} x$.
6. $S_y(H)$, the expected number of pairs (s, t) of nodes in H such that $s \overset{H}{\sim} t$ but $t \overset{H}{\not\sim} y$.
7. $S^x(H)$, the expected number of pairs (s, t) of nodes in H such that $s \overset{H}{\sim} t$ and $t \overset{H}{\sim} x$.
8. $S^y(H)$, the expected number of pairs (s, t) of nodes in H such that $s \overset{H}{\sim} t$ and $t \overset{H}{\sim} y$.
9. $S_{xy}(H)$, the expected number of pairs (s, t) of nodes in H such that $s \overset{H}{\sim} t$ but $t \overset{H}{\not\sim} x$ and $t \overset{H}{\not\sim} y$.
10. $\overline{S}_H$, the expected number of pairs (s, t) of nodes in H such that $s \overset{H}{\not\sim} t$ but $s \overset{H}{\sim} x$ and $t \overset{H}{\sim} y$.

It is easy to verify that the first four items above correspond to the statistics maintained in each k-clique by the algorithm in section 3. The following observation guarantees that the values $E^x(H)$ (the expected number of nodes t in the graph H such that $t \overset{H}{\sim} x$) and $S_y^x(H)$ (the expected number of pairs (s, t) of nodes in H such that $s \overset{H}{\sim} t$, $t \overset{H}{\sim} x$, and $t \overset{H}{\not\sim} y$) can be computed from the state of e. We use Observation 4.1 in lemmata 4.2,4.3, and 4.4.

Observation 4.1

(i) $E^a(H) = \sum_{t \in H} P_H[t \sim a] = \sum_{t \in H} P_H[t \sim a \wedge (t \sim b \vee t \not\sim b)] = E^{ab}(H) + E_b^a(H)$.

(ii) $\forall s \in H \quad E_b^{as}(H) + E_{ab}^s(H) = \sum_{t \in H} P_H[t \sim a \wedge t \sim s \wedge t \not\sim b] + \sum_{t \in H} P_H[t \sim s \wedge t \not\sim a \wedge t \not\sim b]$
$= E_b^s(H)$.

(iii) $S_b^a(H) = S_b(H) - S_{ab}(H)$ *(by (ii))*.

The next four lemmata define the initialization, reduction, and termination steps of the linear time algorithm for the resilience problem.

Lemma 4.1 (initialization) *Let $H = e = \{x, y\}$ and p_e be the probability that e is operational. Then*

1. $P_H[x \sim y] = p_e$.
2. $E_y^x(H) = 1 - p_e$.
3. $E_x^y(H) = 1 - p_e$.
4. $E^{xy}(H) = 2 \times p_e$.
5. $S_x(H) = 1 - p_e$.
6. $S_y(H) = 1 - p_e$.
7. $S^x(H) = 3 \times p_e + 1$.
8. $S^y(H) = 3 \times p_e + 1$.
9. $S_{xy}(H) = 0$.
10. $\overline{S}_H = 1 - p_e$.

Proof: Just use the definition of each statistics. For example,
$S^y(H) = \sum_{s \in \{x,y\}} E^{ys}(H) = \sum_{s \in \{x,y\}} \sum_{t \in \{x,y\}} P_H[s \sim t \wedge t \sim y] = 3 \times p_e + 1.$ ∎

The reduction step involves the analysis of multiple cases. In the discussion that follows let us consider the scenario depicted in Figure 4.1. Let $B = L \cup M$ with identified nodes x and y, and $C = B \cup M$ with identified nodes x and y. For the sake of simplicity we first show how to compute the statistics about B (lemma 4.2). Finally, we show how to combine the current information in the state of $\{x, y\}$ (statistics about M) with the statistics about B to get the updated state of $\{x, y\}$ (lemma 4.3). This is the same approach used in [7] to solve the resilience problem in $\mathcal{O}(n^2)$ time on partial 2-tree networks.

Lemma 4.2 *Let G be a (partially reduced) 2-tree. Let z be a 2-leaf of G with neighbors x and y. Let L and R be the shell of $\{x, z\}$, and $\{y, z\}$, respectively. Let $B = L \cup R$, with identified nodes x and y. Then*

1. $P_B[x \sim y] = P_L[x \sim z] \times P_R[z \sim y]$
2. $E_y^x(B) = E_z^x(L) + E^{xz}(L) \times P_R[z \not\sim y] + E_y^z(R) \times P_L[x \sim z] - P_L[z \sim x] \times P_R[z \not\sim y]$
3. $E_x^y(B) = E_x^z(L) \times P_R[y \sim z] + E_z^y(R) + E^{yz}(R) \times P_L[z \not\sim x] - P_R[z \sim y] \times P_L[z \not\sim x]$

4 $E^{xy}(B) = E^{xz}(L) \times P_R[z \sim y] + E^{yz}(R) \times P_L[x \sim z] - P_L[z \sim x] \times P_R[z \sim y]$

5. $S_x(B) = S_x(L) + 2 \times E^z_x(L) \times E^z(R) + S_z(R) + S^z(R) \times P_L[z \not\sim x] -$
$2 \times (E^z_x(L) + E^z(R) \times P_L[z \not\sim x]) + P_L[z \not\sim x]$

6. $S_y(B) = S_y(R) + 2 \times E^z_y(R) \times E^z(L) + S_z(L) + S^z(L) \times P_R[z \not\sim y] -$
$2 \times (E^z(L) \times P_R[z \not\sim y] + E^z_y(R)) + P_R[z \not\sim y]$

7. $S^x(B) = S^x(L) + 2 \times E^z(R) \times E^{xz}(L) + S^z(R) \times P_L[z \sim x] -$
$2 \times (E^{xz}(L) + E^z(R) \times P_L[z \sim x]) + P_L[z \sim x]$

8. $S^y(B) = S^y(R) + 2 \times E^z(L) \times E^{yz}(R) + S^z(L) \times P_R[z \sim y] -$
$2 \times (E^{yz}(R) + E^z(L) \times P_R[z \sim y]) + P_R[z \sim y]$

9. $S_{xy}(B) = S_{xz}(L) + S^z_x(L) \times P_R[z \not\sim y] + 2 \times E^z_x(L) \times E^z_y(R) + S_{yz}(R) +$
$S^z_y(R) \times P_L[z \not\sim x] - 2 \times (E^z_x(L) \times P_R[z \not\sim y] + E^z_y(R) \times P_L[z \not\sim x]) +$
$P_L[z \not\sim x] \times P_R[z \not\sim y]$

10. $\overline{S}_B = \overline{S}_L \times P_R[z \sim y] + E^x(L) \times E^y_z(R) + E^x_z(L) \times E^{yz}(R) + \overline{S}_R \times P_L[x \sim z] -$
$(E^y_z(R) \times P_L[x \sim z] + E^x_z(L) \times P_R[z \sim y]))$

Proof:

1. We know that node z is an x-y separator in B; besides, edge failures are statistically independent and irrelevant edges may be ignored. Thus

$$\begin{aligned} P_B[x \sim y] &= P_B[x \sim z \wedge z \sim y] \\ &= P_B[x \sim z] \times P_B[z \sim y] \\ &= P_L[x \sim z] \times P_R[z \sim y] \end{aligned}$$

2. We prove 2, 3, and 4 using the same method. Notice first that we can enumerate the set $V(B)$ (without repetitions) by enumerating the sets $V(L)$, $V(R)$, and then eliminating the repeated elements (node z). So

$$\begin{aligned} E^x_y(B) &= \sum_{t \in B} P_B[t \sim x \wedge t \not\sim y] \\ &= \sum_{t \in L} P_B[t \sim x \wedge t \not\sim y] + \sum_{t \in R} P_B[t \sim x \wedge t \not\sim y] - P_B[z \sim x \wedge z \not\sim y] \end{aligned}$$

Let us call the first, second, and third terms in the expression above T_L, T_R, and $T_{\{z\}}$ respectively. We prove the result by first expressing T_L, T_R, and $T_{\{z\}}$ in terms of statistics for L and R, and then, adding up the results. The following identities follow from the observation that L and R are edge-disjoint, failures are statistically independent, and irrelevant edges

can be ignored.

$$\begin{aligned} T_L &= \sum_{t \in L} P_B[t \sim x \wedge t \not\sim y] \\ &= \sum_{t \in L} (P_L[t \sim x \wedge t \not\sim z] + P_L[t \sim x \wedge t \sim z] \times P_R[z \not\sim y]) \\ &= E_z^x(L) + E^{xz}(L) \times P_R[z \not\sim y] \end{aligned}$$

$$\begin{aligned} T_R &= \sum_{t \in R} P_B[t \sim x \wedge t \not\sim y] \\ &= \sum_{t \in R} P_L[x \sim z] \times P_R[z \sim t \wedge t \not\sim y] \\ &= P_L[x \sim z] \times E_y^z(R) \end{aligned}$$

$$T_{\{z\}} = P_L[z \sim x] \times P_R[z \not\sim y]$$

3. This case is symmetric to the previous one, here we obtain

$$E_x^y(B) = \overbrace{\sum_{t \in L} P_B[t \sim y \wedge t \not\sim x]}^{T_L} + \overbrace{\sum_{t \in R} P_B[t \sim y \wedge t \not\sim x]}^{T_R} - \overbrace{P_B[z \sim y \wedge z \not\sim x]}^{T_{\{z\}}}$$

where

$$\begin{aligned} T_L &= E_x^z(L) \times P_R[y \sim z] \\ T_R &= E_z^y(R) + E^{yz}(R) \times P_L[z \not\sim x] \\ T_{\{z\}} &= P_R[z \sim y] \times P_L[z \not\sim x] \end{aligned}$$

4. In this case

$$E^{xy}(B) = \sum_{t \in B} [t \sim x \wedge t \sim y] = T_L + T_R - T_{\{z\}}$$

where

$$\begin{aligned} T_L &= \sum_{t \in L} P_B[t \sim x \wedge t \sim y] \\ &= \sum_{t \in L} P_B[t \overset{L}{\sim} x \wedge t \overset{L}{\sim} z \wedge z \overset{R}{\sim} y] \\ &= \sum_{t \in L} P_L[t \sim x \wedge t \sim z] \times P_R[z \sim y] \\ &= E^{xz}(L) \times P_R[z \sim y] \end{aligned}$$

$$
\begin{aligned}
T_R &= \sum_{t \in R} P_B[t \sim x \wedge t \sim y] \\
&= \sum_{t \in R} P_B[t \overset{R}{\sim} z \wedge z \overset{L}{\sim} x \wedge t \overset{R}{\sim} y] \\
&= \sum_{t \in R} P_R[t \sim z \wedge t \sim y] \times P_L[z \sim x] \\
&= E^{yz}(R) \times P_L[z \sim x] \\
T_{\{z\}} &= P_L[z \sim x] \times P_R[z \sim y]
\end{aligned}
$$

5. To prove 5-10 we observe that we can sum over all pairs of nodes in $V(B) \times V(B)$ by summing over all pairs in $V(L) \times V(L)$, $V(L) \times V(R)$, $V(R) \times V(L)$, and $V(R) \times V(R)$, and then subtracting the repeated elements (notice that (z, z) occurs four times). It is easy to verify that the repeated pairs are precisely the elements in $\{z\} \times V(B) \cup V(B) \times \{z\}$. So we can write

$$
S_x(B) = T_{LL} + T_{LR} + T_{RL} + T_{RR} - (T_{\{z\}L} + T_{\{z\}R} + T_{L\{z\}} + T_{R\{z\}}) + T_{\{z\}\{z\}}
$$

where T_{IJ} denotes $\sum_{s \in I} \sum_{t \in J} P_B[s \sim t \wedge t \not\sim x]$, for any I, J subgraphs or sets of nodes of B. Exhaustive case analysis and simple algebraic manipulation gives

$$
\begin{aligned}
T_{LL} &= \sum_{s \in L} \sum_{t \in L} P_L[s \sim t \wedge t \not\sim x] \\
&= S_x(L) \\
T_{LR} &= \sum_{s \in L} \sum_{t \in R} P_L[s \not\sim x \wedge s \sim z] \times P_R[t \sim z] \\
&= \sum_{s \in L} P_L[s \not\sim x \wedge s \sim z] \times E^z(R) \\
&= E^z_x(L) \times E^z(R) \\
&= T_{RL} \\
T_{RR} &= \sum_{s \in R} \sum_{t \in R} P_B[s \sim t \wedge t \not\sim x] \\
&= \sum_{s \in R} \sum_{t \in R} P_B[s \sim t \wedge t \not\sim z] + \sum_{s \in R} \sum_{t \in R} P_B[s \sim t \wedge t \sim z] \times P_L[z \not\sim x] \\
&= S_z(R) + S^z(R) \times P_L[z \not\sim x] \\
T_{\{z\}L} &= \sum_{t \in L} P_B[z \overset{L}{\sim} t \wedge t \overset{L}{\not\sim} x] \\
&= E^z_x(L) \\
&= T_{L\{z\}}
\end{aligned}
$$

$$
\begin{aligned}
T_{\{z\}R} &= \sum_{t \in R} P_B[z \overset{R}{\sim} t \wedge z \overset{L}{\not\sim} x] \\
&= E^z(R) \times P_L[z \not\sim x] \\
&= T_{R\{z\}} \\
T_{\{z\}\{z\}} &= P_L[z \not\sim x]
\end{aligned}
$$

6. This is clearly a case symmetric to 5.

7. We know that $S^x(B) = \sum_{s \in B} \sum_{t \in B} P_B[s \sim t \wedge t \sim x]$. So, breaking down the sum as in 5, and doing an exhaustive analysis of cases, we get

$$
\begin{aligned}
T_{LL} &= \sum_{s \in L} \sum_{t \in L} P_L[s \sim t \wedge t \sim x] \\
&= S^x(L) \\
\\
T_{LR} &= \sum_{s \in L} \sum_{t \in R} P_R[t \sim z] \times P_L[z \sim s \wedge s \sim x] \\
&= E^z(R) \times E^{xz}(L) \\
&= T_{RL} \\
\\
T_{RR} &= \sum_{s \in R} \sum_{t \in R} P_B[s \sim t \wedge t \sim x] \\
&= \sum_{s \in R} \sum_{t \in R} P_R[s \sim t \wedge s \sim z] \times P_L[z \sim x] \\
&= S^z(R) \times P_L[z \sim x] \\
\\
T_{\{z\}L} &= \sum_{t \in L} P_L[z \sim t \wedge t \sim x] \\
&= E^{xz}(L) \\
&= T_{L\{z\}} \\
\\
T_{\{z\}R} &= \sum_{t \in R} P_B[z \overset{R}{\sim} t \wedge z \overset{L}{\sim} x] \\
&= \sum_{t \in R} P_R[t \sim z] \times P_L[[z \sim x] \\
&= E^z(R) \times P_L[[z \sim x] \\
&= T_{R\{z\}} \\
\\
T_{\{z\}\{z\}} &= P_B[z \sim x] \\
&= P_L[z \sim x]
\end{aligned}
$$

8. This case is symmetric to 7.

9. Proceeding in the same fashion as in 5 we obtain

$$\begin{aligned}
T_{LL} &= S_{xz}(L) + S_x^z(L) \times P_R[z \not\sim y] \\
T_{LR} &= E_x^z(L) \times E_y^z(R) \\
T_{RR} &= S_{yz}(R) + S_y^z(R) \times P_L[z \not\sim x] \\
T_{\{z\}L} &= \sum_{t \in L} P_B[x \overset{L}{\not\sim} t \wedge t \overset{L}{\sim} z \wedge z \overset{R}{\not\sim} y] \\
&= E_x^z(L) \times P_R[z \not\sim y] \\
T_{\{z\}R} &= \sum_{t \in R} P_B[y \overset{R}{\not\sim} t \wedge t \overset{R}{\sim} z \wedge z \overset{L}{\not\sim} x] \\
&= E_y^z(R) \times P_L[x \not\sim z] \\
T_{\{z\}\{z\}} &= P_L[z \not\sim x] \times P_R[z \not\sim y]
\end{aligned}$$

10. We know that

$$\overline{S}_B = \sum_{s \in B} \sum_{t \in B} P_B[s \not\sim t \wedge s \sim x \wedge t \sim y]$$

So by exhaustive case analysis we obtain

$$\overline{S}_B = T_{LL} + T_{LR} + 0 + T_{RR} - (T_{\{z\}L} + T_{L\{z\}} + T_{\{z\}R} + T_{R\{z\}}) + T_{\{z\}\{z\}}$$

where

$$\begin{aligned}
T_{LL} &= \sum_{s \in L} \sum_{t \in L} P_B[x \overset{L}{\sim} t \wedge s \overset{L}{\not\sim} t \wedge s \overset{L}{\sim} z \wedge z \overset{R}{\not\sim} \\
&= \overline{S}_L \times P_R[z \sim y] \\
T_{LR} &= \sum_{s \in L} \sum_{t \in R} (P_B[x \overset{L}{\sim} s \wedge z \overset{R}{\not\sim} t \wedge t \overset{R}{\sim} y] + P_B[x \overset{L}{\sim} s \wedge s \overset{L}{\not\sim} z \wedge z \overset{R}{\sim} t \wedge t \overset{R}{\sim} y]) \\
&= E^x(L) \times E_z^y(R) + E_z^x(L) \times E^{yz}(R) \\
T_{RR} &= \sum_{s \in R} \sum_{t \in R} P_B[x \overset{L}{\sim} t \wedge z \overset{R}{\sim} t \wedge z \overset{R}{\not\sim} s \wedge s \overset{R}{\sim} y] \\
&= \overline{S}_R \times P_L[x \sim z]
\end{aligned}$$

$$
\begin{aligned}
T_{\{z\}L} &= 0 \\
T_{\{z\}R} &= \sum_{t \in R} P_B[x \overset{L}{\sim} z \wedge t \overset{R}{\not\sim} z \wedge t \overset{R}{\sim} y] \\
&= P_L[x \sim z] \times E_z^y(R) \\
T_{R\{z\}} &= 0 \\
T_{L\{z\}} &= \sum_{s \in L} P_B[s \overset{L}{\sim} x \wedge s \overset{L}{\not\sim} z \wedge z \overset{R}{\sim} y] \\
&= E_z^x(L) \times P_L[x \sim z] \\
T_{\{z\}\{z\}} &= 0 \ \blacksquare
\end{aligned}
$$

Lemma 4.2 guarantees that the statistics for B can be computed from the information in the states of $\{x, z\}$ and $\{z, y\}$. We now turn our attention to the computation of the statistics for $C = M \cup B$.

Lemma 4.3 *Let G be a (partially) reduced 2-tree. Let z be a 2-leaf of G with neighbors x and y. Let L, R, and M be the shells of $\{x, z\}$, $\{y, z\}$, and $\{x, y\}$, respectively. Let $B = L \cup R$, with identified nodes x and y, and $C = M \cup B$, with identified nodes x and y. Then*

1. $P_C[x \sim y] = P_M[x \sim y] + P_M[x \not\sim y] \times P_B[x \sim y]$
2. $E_y^x(C) = E_y^x(M) \times P_B[x \not\sim y] + E_y^x(B) \times P_M[x \not\sim y] - P_C[x \not\sim y]$
3. $E_x^y(C) = E_x^y(M) \times P_B[x \not\sim y] + E_x^y(B) \times P_M[x \not\sim y] - P_C[x \not\sim y]$
4. $E^{xy}(C) = E^{xy}(M) + E_y^x(M) \times P_B[x \sim y] + E_x^y(M) \times P_B[x \sim y] + E^{xy}(B) +$
   $E_y^x(B) \times P_M[x \sim y] + E_x^y(B) \times P_M[x \sim y] - 2 \times P_C[x \sim y]$
5. $S_x(C) = S_{xy}(M) + S_x^y(M) \times P_B[x \not\sim y] + S_{xy}(B) + S_x^y(B) \times P_M[x \not\sim y] +$
   $2 \times (E_x^y(M) \times E_x^y(B) - E_x^y(C)) - P_C[x \not\sim y]$
6. $S_y(C) = S_{xy}(M) + S_y^x(M) \times P_B[x \not\sim y] + S_{xy}(B) + S_y^x(B) \times P_M[x \not\sim y] +$
   $2 \times (E_y^x(M) \times E_y^x(B) - E_y^x(C)) - P_C[x \not\sim y]$
7. $S^x(C) = S^x(M) + S_x^y(M) \times P_B[y \sim x] + \overline{S}_M \times P_B[x \sim y] + S^x(B) +$
   $S_x^y(B) \times P_M[y \sim x] + \overline{S}_B \times P_M[x \sim y] +$
   $2 \times (E^x(M) \times E^x(B) + E^{xy}(M) \times E_x^y(B) + E^{xy}(B) \times E_x^y(B)) -$
   $2 \times (E^x(C) + E^{xy}(C)) - 3 \times P_C[x \sim y] + 1$

8. $S^y(C) = S^y(M) + S^x_y(M) \times P_B[x \sim y] + \overline{S}_M \times P_B[x \sim y] + S^y(B) +$
$S^x_y(B) \times P_M[x \sim y] + \overline{S}_B \times P_M[x \sim y] +$
$2 \times (E^y(M) \times E^y(B) + E^{xy}(M) \times E^x_y(B) + E^{xy}(B) \times E^x_y(B)) -$
$2 \times (E^y(C) + E^{xy}(C)) - 3 \times P_C[x \sim y] + 1$

9. $S_{xy}(C) = S_{xy}(M) + S_{xy}(B)$

10. $\overline{S}_C = \overline{S}_M \times P_B[x \not\sim y] + \overline{S}_B \times P_M[x \not\sim y] + E^x_y(M) \times E^y_x(B) + E^x_y(B)E^y_x(M) -$
$(E^y_x(C) + E^x_y(C)) - P_C[x \not\sim y]$

Proof: We follow the approach used in the proof of lemma 4.2. To prove 1, observe that M and B are edge-disjoint, failures are statistically independent, and irrelevant edges can be ignored. The proofs of 2, 3, and 4 exploit the fact that $V_C = V(M) \oplus V(B) \cup \{x, z\}$[5] to break down sums over nodes in C into sums over $V(M)$ and $V(B)$. Similarly, using the same notation as in the proof of lemma 4.2, it is easy to verify that

$$T_{CC} = T_{MM} + T_{MB} + T_{BM} + T_{BB} - (T_{\{x,y\}M} + T_{\{x,y\}B} + T_{M\{x,y\}} + T_{B\{x,y\}}) + T_{\{x,y\}\{x,y\}}$$

So we can prove 5-10 using the identity above and doing an exhaustive analysis of cases. ∎

The following lemma defines how to compute $Res(G)$ from the one-edge graph obtained after applying lemmata 4.1-4.3 to G.

Lemma 4.4 *(termination) Let $G = (V, E)$ be a 2-tree on n nodes, and H a one-edge 2-tree obtained by applying the reduction paradigm and lemmata 4.1-4.3 to G. Let x be a node in H. Then*

$$Res(G) = \frac{1}{2} \times (S^x(G) + S_x(G) - n)$$

Proof:

$$\begin{aligned} Res(G) &= \frac{1}{2} \times \sum_{t \in G} (E^t(G) - 1) \quad \text{(by equation 1 in section 2)} \\ &= \frac{1}{2} \times (\sum_{t \in G} (E^{xt}(G) + E^t_x(G)) - n) \quad \text{(by Observation 4.1(i))} \\ &= \frac{1}{2} \times (S^x(G) + S_x(G) - n) \quad \text{(by definition)} \ \blacksquare \end{aligned}$$

The reduction paradigm together with lemmata 4.1-4.4 imply the following theorem.

Theorem 4.1 *The resilience $Res(G)$ of a partial 2-tree network on n nodes can be computed in $\mathcal{O}(n)$ time.*

Proof: Correctness follows from lemmata 4.1-4.4. The running time is linear because we can find an embedding of a partial 2-tree in a 2-tree in linear time [18] and because we can reduce the

[5] $\oplus$ denotes the disjoint set union operation

embedding 2-tree in linear time using Algorithm 3 and lemmata 4.1-4.4. (see proof of theorem 3.1) ∎

5 Conclusions

We presented an $\mathcal{O}(n^2)$ time algorithm for a general class of networks, namely the class of partial k-tree networks (given a fixed k and an embedding of the input network in a k-tree network). We also presented a linear time algorithm for partial 2-tree networks. Although we conjecture that the resilience problem may be solvable in linear time for k-trees (or partial k-trees given with an embedding in a k-tree), we do not think that our linear time algorithm for partial 2-tree networks can be easily generalized to deal with k-tree networks. The extensive analysis of cases, even for $k = 3$, suggests that a different approach would be more productive. We believe that further analysis of the $\mathcal{O}(n^2)$ time algorithm for k-tree networks would be a more fruitful line of attack.

This research was limited to networks with fail-safe nodes. In a forth comming technical report we consider the resilience problem on partial 2-tree and k-tree networks with edge and node failures. Preliminary results indicate that only minor modifications to the algorithms presented here are needed to cope with the presence of node failures.

References

[1] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[2] S. Arnborg and A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Alg. Discr. Meth.*, 7:305–314, 1986.

[3] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-Hard problems restricted to partial k-trees. *Discrete Appl. Math.*, 23:11–24, 1988.

[4] L. W. Beineke and R. E. Pippert. Properties and characterizations of k-trees. *Mathematika*, 18:141–151, 1971.

[5] H. L. Bodlaender. Classes of graphs with bounded tree-width. Technical Report RUU-CS-86-22, Dept. of Computer Science, University of Utrecht, Utrecht, 1986.

[6] C. J. Colbourn. *The Combinatorics of Network Reliability.* Oxford University Press, New York, 1987.

[7] C. J. Colbourn. Network resilience. *Networks*, 8:404–409, 1987.

[8] R. J. Duffin. Topology of series-parallel networks. *J. Math. Anal. Appl.*, 10:303–318, 1965.

[9] E. S. El-Mallah and C. J. Colbourn. Optimum communication spanning trees in series-parallel graphs. *SIAM J. Computing*, 14:915–925, 1985.

[10] A. M. Farley and A. Proskurowski. Extremal graphs with no disconnecting independent set of matchings. Technical Report CIS-TR-80-21, Dept. of Computer and Information Science, University of Oregon, 1980.

[11] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.

[12] F. Harary. *Graph Theory.* Addison-Wesley, Reading, Mass., 1969.

[13] D. S. Johnson. The NP-completeness column: An ongoing guide. *J. of Algorithms*, 6:434–451, 1985.

[14] J. Matousek and R. Thomas. Algorithms finding tree-decompositions of graphs. Submitted for publication, 1988.

[15] J. S. Provan. The complexity of reliability computations in planar and acyclical graphs. *SIAM Journal on Computing*, 15:694–702, 1986.

[16] J. S. Provan and M. O. Bell. The complexity of counting cuts and of computing that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.

[17] D. Rose. Triangulated graphs and the elimination process. *J. Math Anal. Appl.*, pages 597–609, 1970.

[18] J. A. Wald and C. J. Colbourn. Steiner trees, partial 2-trees, and minimal IFI networks. *Networks*, 1983:159–167, 1983.