Resilience of Partial k-tree Networks with Edge and Node Failures

Erick Mata-Montero

CIS-TR-89-15
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 is a #P-complete problem even for planar networks with fail-safe nodes. We generalize an $\mathcal{O}(n^2)$ time algorithm to compute the resilience of n-node k-tree networks with fail-safe nodes to obtain an $\mathcal{O}(n)$ time algorithm that computes the resilience of n-node partial k-tree networks with edge and node failures (given a fixed k and an embedding of the partial k-tree in a k-tree).

Department of Computer and Information Science
University of Oregon

Resilience of Partial k-tree Networks with Edge and Node Failures

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

October 31, 1989

Abstract

The resilience of a network is the expected number of pairs of nodes that can communicate. Computing the resilience of a network is a #P-complete problem even for planar networks with fail-safe nodes. We generalize an $\mathcal{O}(n^2)$ time algorithm to compute the resilience of n-node k-tree networks with fail-safe nodes to obtain an $\mathcal{O}(n)$ time algorithm that computes the resilience of n-node partial k-tree networks with edge and node failures (given a fixed k and an embedding of the partial k-tree in a k-tree).

1 Introduction

Reliability measures of communication networks are an important parameter in network design. We model a computer communication network as a *probabilistic* graph $G = (V, E)$ in which each node v in V represents a *communication site* and each edge e in E represents a bidirectional *communication line* between two sites. Furthermore, edges and nodes have an associated *probability of operation.* The probability of operation of a component (node or edge) c of G is a fixed precision real number p_c such that $0 \leq p_c \leq 1$. Components of the network are in either *operational* or *failed* state. Component failures are assumed to be statistically independent.

Traditionally, the reliability of a network G is defined as the probability that a given communication task T can be performed in G. For example, if the task T consists of exchanging information between k distinguished nodes of G, the reliability of G (*k-terminal reliability*) is defined as the probability that the graph contains paths between each pair of the k nodes. The n-terminal (*all-terminal*) and the 2-terminal reliability are two of the most widely used measures of the reliability of a network. In the former case we are interested in computing the probability that the network contains a spanning tree, in the latter we are concerned with the probability that there is a path connecting two distinguished nodes in G. Computing the all-terminal reliability of a network is a #P-complete problem, even for networks with fail-safe nodes [15]. Furthermore, computing the 2-terminal reliability of a network is also a #P-complete problem, even when the network is planar, acyclic, with bounded degree nodes, with fail-safe nodes, and with all the edges having identical

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

probability of operation [14]. Colbourn [7] presents an excellent survey of the combinatorics of network reliability.

The *resilience* of a network is the expected number of pairs of distinct nodes that can communicate. This measure provides some additional, fine grain information about the reliability of a network. For example, Figure 1 presents two fail-safe networks that have the same all-terminal reliability but whose resilience is quite different. The all-terminal reliability of both G_1 and G_2 is 0. However, the resilience of G_1 is 5 and the resilience of G_2 is 15. In general, it has not been determined what relationships (if any) exist between all-terminal reliability and resilience [8].

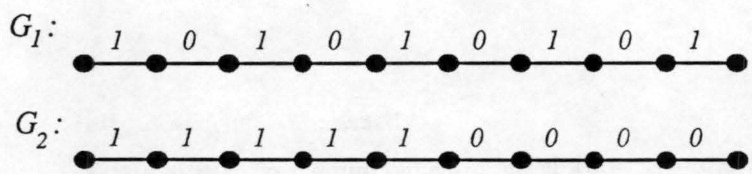


Figure 1: Two graphs with the same 10-terminal reliability but different resilience.

The resilience problem, RES, consists of computing the resilience of a network. This is also a #P-complete problem, even when the network is planar and the nodes are fail-safe [8]. The apparent complexity of RES has lead to the development of efficient algorithms on restricted classes of networks, especially on the class of partial 2-tree networks ([8], [12], and [16]). The class of partial k-trees is an attractive subject of study not only because it contains several important classes of graphs (e.g., series-parallel graphs and outerplanar graphs [5],[11]), but also because many NP-complete graph problems have polynomial, and even linear time solutions when restricted to the class of partial k-trees [4]. Table 1 describes the complexity of the polynomial algorithms so far obtained for the resilience problem on partial 2-tree networks. Table 2 describes the complexity of the polynomial algorithms known for the class of partial k-tree networks (for a fixed $k > 2$). Our main result consists of a linear time algorithm for all the classes of networks described in tables 1 and 2.

This paper is organized as follows. Section 2 introduces some basic terminology. Section 3 presents some background material about k-trees and partial k-trees. Section 4 describes a linear time algorithm to compute the resilience of partial k-tree networks given with a suitable embedding in a k-tree (for a fixed k).

2 Terminology

Except for a few explicitly defined concepts, we use the basic graph theoretic terminology as defined in [10]. Throughout this paper we assume that all graphs are probabilistic. Let $G = (V, E)$ be a graph with n nodes and m edges. 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 if H is a spanning subgraph of G. We use $H \leq G$ to denote that H is a subgraph of G.

		Edge failures		
		$\forall e,\ p_e = 1$	$\forall e, p_e = c_1$	$\forall e,\ 0 \le p_e \le 1$
Node	$\forall v,\ p_v = 1$	-	$\mathcal{O}(n)$ [1]	$\mathcal{O}(n)$ [12]
failures	$\forall v,\ p_v = c_2$	$\mathcal{O}(n)$ [1]	open	open
	$\forall v,\ 0 \le p_v \le 1$	open	open	open

Table 1: Polynomial time algorithms for RES on partial 2-tree networks.

		Edge failures		
		$\forall e,\ p_e = 1$	$\forall e,\ p_e = c_1$	$\forall e,\ 0 \le p_e \le 1$
Node	$\forall v,\ p_v = 1$	-	$\mathcal{O}(n^2)$ [12]	$\mathcal{O}(n^2)$ [12]
failures	$\forall v,\ p_v = c_2$	open	open	open
	$\forall v,\ 0 \le p_v \le 1$	open	open	open

Table 2: Polynomial time algorithms for RES on partial k-tree networks.

The state S of a network G is the set of nodes and edges of G that are operational. Nodes and edges are in one of two states: *up* (operational) or *down* (failed). Let p_v and p_e denote the probability that node v is up and edge e is up respectively. The probability that G is in state S is

$$\prod_{v \in S} p_v \prod_{v \in V \setminus S} (1 - p_v) \prod_{e \in S} p_e \prod_{e \in E \setminus S} (1 - p_e)$$

We use subgraphs of G to represent states of the network. Notice however that, unless G has no edges, there are more states than subgraphs of G. So, each subgraph $H = (V_H, E_H)$ of G represents a class of states of G, namely those states of G in which nodes in V_H are up, nodes in $V \setminus V_H$ are down, edges in E_H are up, and edges in $E'_H \setminus E_H$ are down, where E'_H is the set of edges of the subgraph of G induced by V_H (see Figure 2). The *operational subgraph of* G is the subgraph of G defined by the operational nodes and the operational edges that are incident on two operational nodes. $P_G[H]$ denotes the probability that H is the operational subgraph of G (equivalently, it

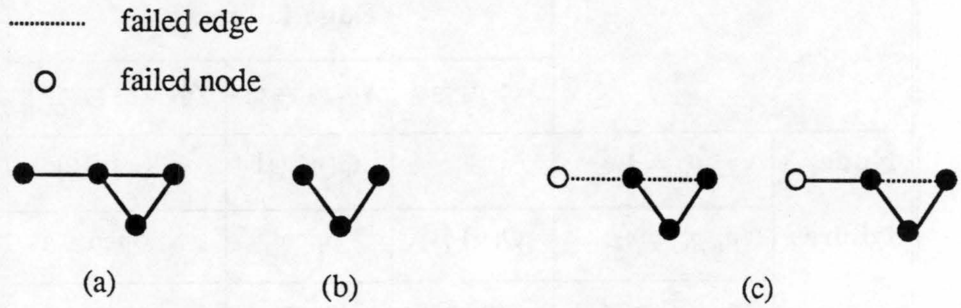


Figure 2: (a) Graph G. (b) Subgraph H. (c) States represented by H.

denotes the probability that the state of G is one of the states represented by H). Thus,

$$P_G[H] = \prod_{v \in V_H} p_v \prod_{v \in V \setminus V_H} (1 - p_v) \prod_{e \in E_H} p_e \prod_{e \in E'_H \setminus E_H} (1 - p_e)$$

We extend the definition of P_G to the domain of sets of subgraphs of G in the natural way. Let A be a set of subgraphs of G, $P_G[A]$ denotes the probability that the operational subgraph of G is in A. Therefore $P_G[A] = \sum_{H \in A} P_G[H]$.

Let H be a subgraph of G and u, v be two nodes of H. We say that node u *is connected to* node v *via* H ($u \overset{H}{\sim} v$) iff there is a path, consisting of zero or more edges of H, that connects node u to node v; when $H = G$ we prefer the notation $u \sim v$ over $u \overset{G}{\sim} v$. A node v is connected to a set of nodes C via a graph H ($v \overset{H}{\sim} C$) if $v \overset{H}{\sim} w$, for all nodes $w \in C$.

The set of all connected components of a graph defines a partition of the set of vertices of the graph. A set π is a *subpartition* of the set of nodes V if π is a partition of a subset of nodes of V. We use V_π to denote the set of nodes of which π is a partition. Given a subpartition π of V, $P_G[\pi]$ denotes the probability that the operational subgraph of G has precisely the connected components defined by π [1].

The resilience 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. We use *Res*(G) to denote the resilience of G. We can formulate *Res*(G) as

$$Res(G) = \sum_{H \leq G} P_G[H] \; Pairs(H)$$

where *Pairs(H)* is the number of pairs $\{u, v\}$ of nodes in V such that $u \overset{H}{\sim} v$ and $u \neq v$. We can also formulate *Res(G)* in terms of certain information about the connected components of the subgraphs

[1]Notice that π may be the empty set.

of G. It is easy to verify that

$$Res(G) = \frac{1}{2}\,(\sum_{H \leq G} P_G[H] \sum_{\substack{CC \text{ connected} \\ \text{component of } H}} |V(CC)|^2 \;-\; \sum_{v \in V} p_v) \qquad (1)$$

In the next ection we use equation 1 to devise an $\mathcal{O}(n)$ time algorithm to compute the resilience of partial k-tree networks given with an embedding in a k-tree.

3 Partial k-tree networks

Important classes of networks can be classified as partial k-trees (graphs with bounded tree-width) [5]. 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.

A graph is a *partial k-tree* if it is a partial graph of a k-tree. We refer the reader to [3] or [2] for an overview of properties of k-trees and to [5, 11] for surveys of classes of graphs related to the class of (partial) k-trees.

3.1 The reduction paradigm

Arnborg and Proskurowski [4] 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-hard 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. To simplify our presentation, 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 [4] 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 removing v we combine the states of these k-cliques 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. 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 $K^{+}(v)$ [2]. 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.

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* $R)$.

[2] 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 devising the functions f and g for specific problems.

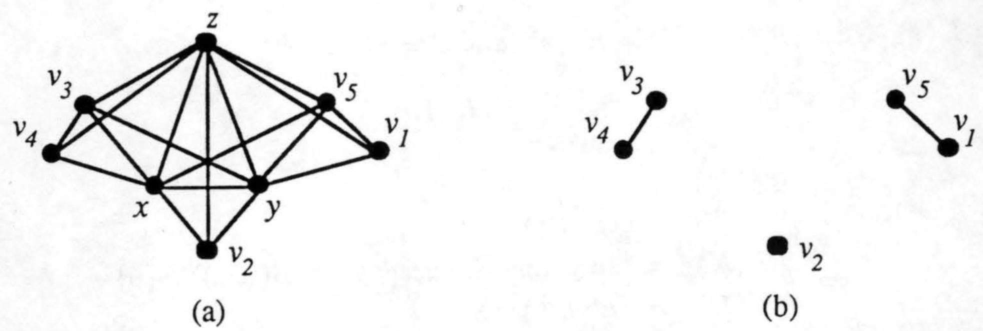


Figure 3: (a) A 3-tree. (b) Branches on K.

The state of 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 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 state of each k-clique. Then we specify how to compute e, f, g, and h in Algorithm 1.

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 member of K or a descendant of K. The connected components of the subgraph induced by all descendants of K are *branches* on K. Figure 3 (a) depicts a 3-tree in which K is the 3-clique induced by the nodes x, y, and z. Figure 3 (b) presents the 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 (completely) removed branches on K. $B'(K)$ denotes the subgraph induced by the nodes in $K \cup B(K)$ without the edges between nodes in K. We call $B'(K)$ the *shell* of K. The state of a k-clique K describes solutions to problems restricted to the shell $B'(K)$. Figure 4 illustrates these concepts; after v_1, v_2, v_3, and v_4 have been removed from the graph in Figure 3, $B(K)$ becomes the graph in Figure 4 (a), and $B'(K)$ becomes the graph in Figure 4 (b).

The following equations describe how $B(K)$ and $B'(K)$ change during the execution of Algorithm 1. Notice that these equations also define $B(K^+)$ and $B'(K^+)$.

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

Initialization step.

$$B(K) = (\emptyset, \emptyset) \quad (2)$$
$$B'(K) = (V(K), \emptyset) \quad (3)$$

Reduction steps.

Let $K = K(v)$, $K^+ = K^+(v)$, $K^u = K^u(v)$, and $S = S(v)$.

$$B(K^+) = \bigcup_{u \in V(K^+)} B(K^u) \quad (4)$$

$$B'(K^+) = \bigcup_{u \in V(K^+)} B'(K^u) \quad (5)$$

$$B(K) = \text{subgraph induced by } V(B(K^+)) \cup \{v\} \quad (6)$$

$$B'(K) = B'(K^+) \cup S \quad (7)$$

Termination step.

$$B(R) = G - R \quad (8)$$

$$B'(R) = G \text{ without the edges in } R \quad (9)$$

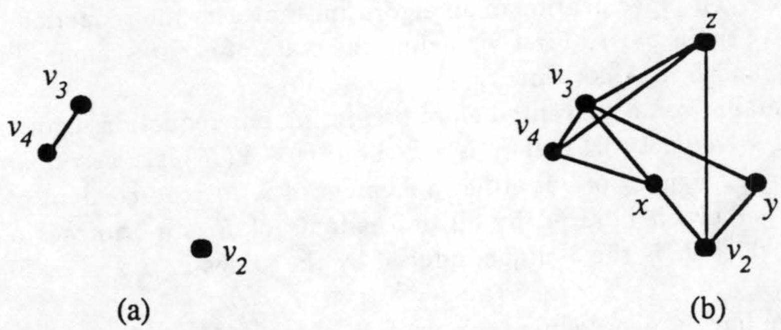
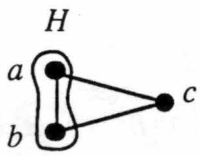


Figure 4: $B(K)$ and $B'(K)$ after removing v_1, v_2, v_3 and v_4 from the graph in Figure 3.

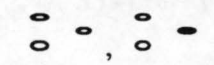
4 Resilience problem on partial k-tree networks

4.1 Resilience of k-tree networks

Before defining the state of each k-clique we need to introduce some additional notation. Let $G = (V, E)$ be a graph and W be a set of nodes. The ***projection of the connected components of G onto W*** (*Proj(G, W)*) is the subpartition of W defined by intersecting each connected component of G with W (if V and W have no common nodes $Proj(G, W) = \emptyset$). Let us now consider $H = (V_H, E_H)$, a subgraph of G. We use $\Pi(H)$ and $\Pi(V_H)$ to denote the set of subpartitions of the nodes in V_H. For each subpartition π in $\Pi(H)$, $SG(G, H, \pi)$ denotes the set of subgraphs G' of G such that $Proj(G', V_H) = \pi$. It is easy to verify that the set of subgraphs of G can be partitioned into equivalence classes, each of which is $SG(G, H, \pi)$, where π is a subpartition of the set of nodes of H. Figure 5 illustrates the partition of the set of subgraphs of a 2-tree into five equivalence classes. Each equivalence class is induced by a subpartition of the set of nodes $\{a, b\}$; white nodes represent failed nodes.



(a)

Subpartition π	Graphs in $SG(H,\pi)$
{ {a,b} }	
{ {a} , {b} }	
{ {a} }	
{ {b} }	
∅	

(b)

Figure 5: (a) A 2-tree G. (b) Equivalence classes induced by subpartitions of $V(H)$.

The idea of partitioning the set of subgraphs 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 subpartitions of the nodes in K [3]. We can partition the set of subgraphs of the shell $B'(K)$ into the following q equivalence classes: $SG(B'(K), K, \pi_1), \ldots, SG(B'(K), K, \pi_q)$. The state of K consists of statistical information about each equivalence class of subgraphs of the shell $B'(K)$. The following values define the state of a k-clique or $(k+1)$-clique K:

- $s(\pi, K)$, for each subpartition π in $\Pi(K)$. We define $s(\pi, K)$ as the probability that a subgraph of the shell $B'(K)$ belongs to the class $SG(B'(K), K, \pi)$, given that $up(V_\pi)$ (the nodes in V_π are up) and $dn(V(K)\backslash V_\pi)$ (the nodes in $V(K)\backslash V_\pi$ are down). If the probability that the nodes in $V(K)\backslash V_\pi$ are down is zero, $s(\pi, K)$ is defined as zero [4]. So

$$\begin{aligned} s(\pi, K) &= P_{B'(K)}[SG(B'(K), K, \pi) \parallel up(V_\pi) \wedge dn(V(K)\backslash V_\pi)] \\ &= \sum_{H \in SG(B'(K),K,\pi)} P_{B'(K)}[H \parallel up(V_\pi) \wedge dn(V(K)\backslash V_\pi)] \qquad (10) \end{aligned}$$

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

[4] For the sake of simplicity we assume that the probability of operation of each node v in G is positive. If some p_v is zero we can either modify the formulas in this section or remove v and apply the algorithm to the resulting partial k-tree.

where $P[A \parallel B]$ denotes $P[A \mid B]$ if $P[B] > 0$, otherwise it is 0.

- $E(\pi, K, C)$, for all non-empty subpartitions π in $\Pi(K)$, and $C \in \pi$. We define $E(\pi, K, C)$ as follows:

$$E(\pi, K, C) \quad = \sum_{H \in SG(B'(K), K, \pi)} P_{B'(K)}[H \parallel up(V_\pi) \wedge dn(V(K) \setminus V_\pi)] \; BN(K, H, C) \quad (11)$$

where $BN(K, H, C)$ is the number of *branch nodes* (nodes in $B(K)$) connected to C via H. It is easy to verify that if $s(\pi, K) > 0$, $E(\pi, K, C)/s(\pi, K)$ is a conditional expected value, namely the expected number of nodes in $B(K)$ that are connected to C, via H, given that H is a member of the class $SG(B'(K), K, \pi)$.

- $EP(\pi, K, C_1, C_2)$, for all non-empty subpartitions π in $\Pi(K)$, and C_1, C_2 blocks of π. We define $EP(\pi, K, C_1, C_2)$ as follows:

$$EP(\pi, K, C_1, C_2) = \sum_{\substack{H \text{ in} \\ SG(B'(K), K, \pi)}} P_{B'(K)}[H \parallel up(V_\pi) \wedge dn(V(K) \setminus V_\pi)]i \; BN(K, H, C_1) \; BN(K, H, C_2) \quad (12)$$

Again, it is easy to verify that if $s(\pi, K) \neq 0$, $E(\pi, K, C_1, C_2)/s(\pi, K)$ is a conditional expected value, namely the expected number of pairs (s, t) of nodes in $B(K)$ such that $s \overset{H}{\sim} C_1$, and $t \overset{H}{\sim} C_2$, given that H, a subgraph of $B'(K)$, is a member of $SG(B'(K), K, \pi)$.

- $EIP(K)$. We use $EIP(K)$ to denote the expected number of pairs of nodes in $B(K)$ that can communicate but are separated (isolated) from K. Formally, we define

$$EIP(K) = \sum_{H \leq B'(K)} P_{B'(K)}[H] \sum_{\substack{CC \text{ connected} \\ \text{component of } H \\ V(CC) \cap V(K) = \emptyset}} |V(CC)|^2 \quad (13)$$

The next four lemmata define the initialization, reduction, and termination steps of our algorithm for the resilience problem. The initialization lemma follows from the definitions of $B(K)$, $B'(K)$, $s(\pi, K)$, $E(\pi, K, C)$, $EP(\pi, K, C_1, C_2)$, and $EIP(K)$ (equations 2, 3, 10, 11, 12, and 13, respectively).

Lemma 4.1 (initialization) *Let G be a k-tree network. Then*

(i) For all K and π such that K is a k-clique of G, and π is a subpartition in $\Pi(K)$

$$s(\pi, K) = \begin{cases} 1 & \text{if } \pi \text{ consists of zero or more singletons and } \prod_{v \in V(K) \setminus V_\pi} (1 - p_v) \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

(ii) *For all K, π, and C such that K is a k-clique of G, π is a non-empty subpartition in $\Pi(K)$, and $C \in \pi$*

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

(iii) *For all K, π, C_1, and C_2 such that K is a k-clique of G, π is a non-empty subpartition in $\Pi(K)$, and C_1 and C_2 are sets of nodes in π*

$$EP(\pi, K, C_1, C_2) = 0$$

(iv) *For all K such that K is a k-clique of G*

$$EIP(K) = 0$$

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. Let K^+ be the graph induced by $V(K) \cup \{v\}$, and $u_1, \ldots, u_{k+1}$ be the nodes in K^+. Also, for all $i = 1, \ldots, k+1$, let K^i denote the k-clique induced by the nodes in $V(K^+) \setminus \{u_i\}$. The reduction step consists of two parts. First we compute the state of K^+ by combining the information in the states of K^i for all $i = 1, \ldots, k+1$ (lemma 4.2). Then we update the state of K by considering the state of K^+ and the effect of the edges that connect v to K (lemma 4.3).

We need some additional notation. Let π be a partition. Following [4], 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. Furthermore, the *join* of two partitions π_1 and π_2, denoted $\pi_1 \vee \pi_2$, is the partition obtained by taking the union of intersecting blocks until a partition of the union remains (e.g., $\{\{a,b\},\{c\},\{d\}\} \vee \{\{a,d\},\{b,c,e\},\{f\}\} = \{\{a,b,c,d,e\}\{f\}\}$).

To compute the state of K^+ we consider all possible ways of obtaining π^+, a subpartition in $\Pi(K^+)$, as the join of $\pi_1, \ldots, \pi_{k+1}$, where π_i is a partition of $V_{\pi^+} \setminus \{u_i\}$ ($i = 1, \ldots, k+1$). We use $T(\pi^+, K^+)$ to denote the set of $(k+1)$-tuples of subpartitions of nodes in K^+ such that their join is π^+ and the i-th subpartition is a partition of $V_{\pi^+} \setminus \{u_i\}$ ($i = 1, \ldots, k+1$). Formally,

$$T(\pi^+, K^+) = \{(\pi_1, \ldots, \pi_{k+1}) \mid \bigvee_{i=1}^{k+1} \pi_i = \pi^+ \wedge \forall i = 1, \ldots, k+1,\ \pi_i \text{ is a partition of } V_{\pi^+} \setminus \{u_i\}\}$$

We use $\vec{\pi}$ to denote a $(k+1)$-tuple in $T(\pi^+, K^+)$ and $\vec{\pi}_i$ to denote the i-th entry of $\vec{\pi}$.

By definition , a subgraph H of the shell $B'(K^+)$ is the (graph) union of $k+1$ graphs $H_1, \ldots, H_{k+1}$ such that each H_i is a subgraph of $B'(K^i)$ and $i = 1, \ldots, k+1$ (cf. equation 5). Furthermore, the subgraph H is in $SG(B(K^+), K^+, \pi^+)$ if and only if each H_i is in $SG(B(K^i), K^i, \pi_i)$ for subpartitions π_i such that $(\pi_1, \ldots, \pi_{k+1})$ is an element of $T(\pi^+, K^+)$. Formally, we make the following observation.

Observation 4.1 *There is a bijection ϕ from $SG(B'(K^+), K^+, \pi^+)$ to*

$$\bigcup_{\substack{(\pi_1, \ldots, \pi_{k+1}) \\ \text{in } T(\pi^+, K^+)}} SG(B'(K^1), K^1, \pi_1) \times \ldots \times SG(B'(K^{k+1}), K^{k+1}, \pi_{k+1})$$

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

The following observation is useful in proving lemma 4.2.

Observation 4.2 *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,*

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

We can now prove the following lemma.

Lemma 4.2 *Let* G *be a* k*-tree network that has been partially reduced using some perfect elimination ordering and the general reduction paradigm. Let* v *be the next* k*-leaf to be removed. Let* K *be the neighborhood of* v*, and* K^+ *be the subgraph of* G *induced by* $V(K) \cup \{v\}$*. Then*

(i) For all π^+ *such that* π^+ *is a subpartition in* $\Pi(K^+)$

$$s(\pi^+, K^+) = \sum_{\vec{\pi} \in T(\pi^+, K^+)} \prod_{i=1}^{k+1} s(\vec{\pi}_i, K^i)$$

(ii) For all π^+ *and* C *such that* π^+ *is a non-empty subpartition in* $\Pi(K^+)$*, and* $C \in \pi^+$

$$E(\pi^+, K^+, C) = \sum_{\vec{\pi} \in T(\pi^+, K^+)} \sum_{i=1}^{k+1} \prod_{\substack{j=1 \\ j \neq i}}^{k+1} s(\vec{\pi}_j, K^j) \sum_{\substack{D \in \vec{\pi}_i \\ D \subseteq C}} E(\vec{\pi}_i, K^i, D)$$

(iii) For all π^+*,* C_1*, and* C_2 *such that* π^+ *is a non-empty subpartition in* $\Pi(K^+)$*, and* C_1*,* $C_2 \in \pi^+$

$$EP(\pi^+, K^+, C_1, C_2) = \sum_{\vec{\pi} \in T(\pi^+, K^+)} \sum_{i=1}^{k+1} \sum_{j=1}^{k+1} \sum_{\substack{D_1 \in \vec{\pi}_i \\ D_1 \subseteq C_1}} \sum_{\substack{D_2 \in \vec{\pi}_j \\ D_2 \subseteq C_2}} F(i, j, D_1, D_2)$$

where

$$F(i,j,D_1,D_2) = \begin{cases} \prod_{\substack{l=1 \\ l \neq i}}^{k+1} s(\vec{\pi}_l, K^l)\ EP(\vec{\pi}_i, K^i, D_1, D_2) & \text{if } i = j \\ \prod_{\substack{l=1 \\ l \neq i \wedge l \neq j}}^{k+1} s(\vec{\pi}_l, K^l)\ E(\vec{\pi}_i, K^i, D_1)\ E(\vec{\pi}_j, K^j, D_2) & \text{otherwise} \end{cases}$$

(iv) $EIP(K^+) = \sum_{i=1}^{k+1} EIP(K^i)$

Proof: Let us use Y to denote the condition $up(V_{\pi^+}) \wedge dn(V(K)\setminus V_{\pi^+})$. Also, let us use Y_i to denote the condition $up(V_{\vec{\pi}_i}) \wedge dn(V(K^i)\setminus V_{\vec{\pi}_i})$.

(i) Using the definition of $s(\pi^+, K^+)$ (equation 10) and observation 4.1 we get

$$
\begin{aligned}
s(\pi^+, K^+) &= \sum_{\substack{H \text{ in} \\ SG(B'(K^+), K^+, \pi^+)}} P_{B'(K^+)}[H \mid\mid Y] \\
&= \sum_{\vec{\pi} \in T(\pi^+, K^+)} \sum_{\substack{(H_1,\ldots,H_{k+1}) \text{ in} \\ X_1 \times \ldots \times X_{k+1}}} P_{B'(K^+)}[H_1 \cup \ldots \cup H_{k+1} \mid\mid Y]
\end{aligned}
$$

where $X_i = SG(B'(K^i), K^i, \vec{\pi}_i)$, $1 \le i \le k+1$.

Notice that the graphs $H_1, \ldots, H_{k+1}$ are edge-disjoint. Besides, component failures are statistically independent. So,

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

The result follows by observation 4.2.

(ii) Analogously, we can use the definition of $E(\pi^+, K^+, C)$ (equation 11), observation 4.1, and the statistical independence of component failures to obtain

$$
E(\pi^+, K^+, C) = \sum_{\vec{\pi} \in T(\pi^+, K^+)} \sum_{\substack{(H_1,\ldots,H_{k+1}) \text{ in} \\ X_1 \times \ldots \times X_{k+1}}} \prod_{i=1}^{k+1} P_{B'(K^i)}[H_i \mid\mid Y_i] \; BN(K^+, H_1 \cup \ldots \cup H_{k+1}, C)
$$

But

$$
BN(K^+, H_1 \cup \ldots \cup H_{k+1}, C) = \sum_{j=1}^{k+1} \sum_{\substack{D \subseteq C \\ D \in \vec{\pi}_j}} BN(K^j, H_j, D) \tag{14}
$$

So, simple algebraic manipulation and observation 4.2 yield the desired result.

(iii) Similarly, we use the definition of $EP(\pi^+, K^+, C_1, C_2)$ (equation 12), and the arguments employed in (ii) above to get that $EP(\pi^+, K^+, C_1, C_2)$ is

$$
\sum_{\substack{\vec{\pi} \text{ in} \\ T(\pi^+, K^+)}} \sum_{\substack{(H_1,\ldots,H_{k+1}) \text{ in} \\ X_1 \times \ldots \times X_{k+1}}} \prod_{i=1}^{k+1} P_{B'(K^i)}[H_i \mid\mid Y_i] \; BN(K^i, H_1 \cup \ldots \cup H_{k+1}, C_1) \; BN(K^i, H_1 \cup \ldots \cup H_{k+1}, C_2)
$$

Equation 14 and additional algebraic manipulation suffice to prove (iii).

(iv) By similar arguments we obtain that

$$
\begin{aligned}
EIP(K^+) &= \sum_{H \leq B'(K^+)} P_{B'(K^+)}[H] \sum_{\substack{CC \text{connected} \\ \text{component of } H \\ V(CC)\cap V(K^+)=\emptyset}} |V(CC)|^2 \\
&= \sum_{\substack{(H_1,\ldots,H_{k+1}) \text{ in} \\ B'(K^1)\times\ldots\times B'(K^{k+1})}} \prod_{j=1}^{k+1} P_{B'(K^j)}[H_j] \sum_{i=1}^{k+1} \sum_{\substack{CC \text{connected} \\ \text{component of } H_i \\ V(CC)\cap V(K^i)=\emptyset}} |V(CC)|^2
\end{aligned}
$$

The proof follows by observation 4.2. ∎

We now show how to update the state of K given the state of K^+. Let S be the star network consisting of the k edges that link v to K. Let $\Pi'(S)$ denote the set of subpartitions of nodes in S that consist of singletons only possibly with exception of 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. So we update the state of K by considering all possible ways of obtaining each subpartiton π in $\Pi(K)$ as the join of pairs (π_1, π_2) of subpartitions in $\Pi(K^+)$ and $\Pi'(S)$, respectively. The following set defines formally the pairs of subpartitions that we want to consider:

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

The following observation is useful in proving lemma 4.3.

Observation 4.3 *There is a bijection ψ such that*

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

and $\psi(H) = (H_1, H_2)$ iff $H = H_1 \cup H_2$ [5] .

We can now establish how to update the state of K from the state of K^+ and the star graph S.

Lemma 4.3 *Let G be a k-tree network that has been partially reduced using some perfect elimination ordering and the general reduction paradigm. Let v be the next k-leaf to be removed. Let K be the neighborhood of v, and K^+ be the subgraph of G induced by $V(K) \cup \{v\}$. In addition, let S be the star graph consisting of the k edges that link v to K. Then*

(i) For all π such that π is a subpartition in $\Pi(K)$,

$$s(\pi, K) = \sum_{\substack{(\pi_1,\pi_2) \\ \text{in } PS(\pi,K)}} s(\pi_1, K^+)\, P_S[\pi_2 \parallel up(V_\pi) \wedge dn(V(K)\backslash V_\pi)]$$

---

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

(ii) *For all* π, C, *such that* π *is a non-empty subpartition in* $\Pi(K)$ *and* $C \in \pi$,

$$E(\pi,K,C) = \sum_{\substack{(\pi_1,\pi_2)\\ in\ PS(\pi,K)}} P_S[\pi_2 \parallel up(V_\pi) \wedge dn(V(K)\setminus V_\pi)] \, (\sum_{\substack{D\in\pi_1\\ D\setminus\{v\}\subseteq C}} E(\pi_1,K^+,D) + r(v,\pi_1))$$

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

(iii) *For all* π, C_1, *and* C_2 *such that* π *is a non-empty subpartition in* $\Pi(K)$, *and* C_1 *and* C_2 *are blocks of the subpartition* π,

$$\begin{aligned} EP(K,\pi,C_1,C_2) &= \sum_{\substack{(\pi_1,\pi_2)\\ in\ PS(\pi,K)}} P_S[\pi_2 \parallel up(V_\pi) \wedge dn(V(K)\setminus V_\pi)] \\ &\quad \sum_{\substack{D_1\in\pi_1\\ D_1\setminus\{v\}\subseteq C_1}} \sum_{\substack{D_2\in\pi_1\\ D_2\setminus\{v\}\subseteq C_2}} (EP(\pi_1,K^+,D_1,D_2) + R(v,D_1,D_2)) \end{aligned}$$

where

$$R(v,D_1,D_2) = \begin{cases} 2\,E(\pi_1,K^+,D_1) + s(\pi_1,K^+) & \text{if } v \in D_1 \wedge v \in D_2 \\ E(\pi_1,K^+,D_1) & \text{if } v \notin D_1 \wedge v \in D_2 \\ E(\pi_1,K^+,D_2) & \text{if } v \in D_1 \wedge v \notin D_2 \\ 0 & \text{otherwise} \end{cases}$$

(iv) *Let* $\Pi(v,K^+,S)$ *be the set of pairs* (π_1,π_2) *of subpartitions in* $\Pi(K^+)$ *such that* $V_{\pi_1} = V_{\pi_2}$, $\pi_2 \in \Pi'(S)$, $\{v\} \in \pi_1$, *and* $\{v\} \in \pi_2$. *Then*

$$EIP(K) = EIP(K^+) +$$

$$\sum_{\substack{(\pi_1,\pi_2)\\ in\ \Pi(v,K^+,S)}} (EP(\pi_1,K^+,\{v\},\{v\}) + 2\,E(\pi_1,K^+,\{v\}) + s(\pi_1,K^+)) \;\; P_S[\pi_2 \parallel up(V_\pi) \wedge dn(V(K)\setminus V_\pi)]$$

Proof: The proof follows by applying observation 4.3 to the definition of $s(\pi,K)$, $E(\pi,K,C)$, $EP(\pi,K,C_1,C_2)$, and $EIP(K)$, and then performing basic algebraic manipulations. Let us use Y to denote the condition $up(V_\pi) \wedge dn(V(K) \setminus V_\pi)$. In addition, let Y_1 denote the condition $up(V_{\pi_1}) \wedge dn(V(K^+) \setminus V_{\pi_1})$.

(i) By definition (equation 10) and observation 4.3

$$\begin{aligned} s(\pi,K) &= \sum_{H \in SG(B'(K),K,\pi)} P_{B'(K)}[H \parallel Y] \\ &= \sum_{\substack{(\pi_1,\pi_2)\\ in\ PS(\pi,K)}} \sum_{\substack{H_1\ in\\ SG(B'(K^+),K^+,\pi_1)}} \sum_{\substack{H_2\ in\\ SG(S,S,\pi_2)}} P_{B'(K)}[H_1 \cup H_2 \parallel Y] \end{aligned}$$

But

$$P_{B'(K)}[H_1 \cup H_2 \parallel Y] = P_{B'(K^+)}[H_1 \parallel Y_1]\; P_S[H_2 \parallel Y] \tag{15}$$

Therefore

$$s(\pi, K) = \sum_{\substack{(\pi_1,\pi_2) \\ in\ PS(\pi,K)}} s(\pi_1, K^+) \sum_{\substack{H_2\ in \\ SG(S,S,\pi_2)}} P_S[H_2 \parallel Y]$$

which is the desired result [6].

(ii) Analogously, we can use the definition of $E(\pi, K, C)$ (equation 11) and observation 4.3 to obtain

$$E(\pi, K, C) = \sum_{\substack{(\pi_1,\pi_2) \\ in\ PS(\pi,K)}} \sum_{H_1 \in X_1} \sum_{H_2 \in X_2} P_{B'(K)}[H_1 \cup H_2 \parallel Y]\; BN(K, H_1 \cup H_2, C) \tag{16}$$

where $X_1 = SG(B'(K^+), K^+, \pi_1)$ and $X_2 = SG(S, S, \pi_2)$.

Notice that a block C in $(\pi_1 \vee \pi_2) \setminus \{v\}$ is obtained by taking $\bigcup_{\substack{D\in\pi_1 \\ D\setminus\{v\}\subseteq C}} D \setminus \{v\}$. Thus,

$$BN(K, H_1 \cup H_2, C) = \sum_{\substack{D\in\pi_1 \\ D\setminus\{v\}\subseteq C}} BN(K^+, H_1, D) + \delta(v, \pi_1, C) \tag{17}$$

where $\delta(v, \pi_1, C) = \begin{cases} 1 & \text{if } \exists D \in \pi_1 \text{ such that } v \in D \text{ and } D \setminus \{v\} \subseteq C \\ 0 & otherwise \end{cases}$

So, combining equations 15, 16, annd 17

$$E(\pi, K, C) =$$
$$\sum_{\substack{(\pi_1,\pi_2) \\ in\ PS(\pi,K)}} \Big(\sum_{H_1 \in X_1} P_{B'(K^+)}[H_1 \parallel Y_1] \big(\sum_{\substack{D\in\pi_1 \\ D\setminus\{v\}\subseteq C}} BN(K^+, H_1, D) + \delta(v, \pi_1, C)\big)\Big) \sum_{H_2 \in X_2} P_S[H_2 \parallel Y]$$

Simple algebraic manipulation completes the proof.

(iii) By definition (equation 12) and observation 4.3

$$EP(\pi, K, C_1, C_2) =$$
$$\sum_{\substack{(\pi_1,\pi_2) \\ in\ PS(\pi,K)}} \sum_{H_1 \in X_1} \sum_{H_2 \in X_2} P_{B'(K)}[H_1 \cup H_2 \parallel Y] BN(K, H_1 \cup H_2, C_1) BN(K, H_1 \cup H_2, C_2) \tag{18}$$

where X_1 and X_2 are defined as in (ii) above.

---

[6] Recall from section 2 that $P_S[\pi_2]$ is the probability that the connected components of of S are those defined by π_2.

Besides, by equation 17, the product $BN(K, H_1 \cup H_2, C_1)\ BN(K, H_1 \cup H_2, C_2)$ is one of the following values:

$$(\sum_{\substack{D_1 \in \pi_1 \\ D_1 \setminus \{v\} \subseteq C_1}} BN(K^+, H_1, D_1) + 1)\ (\sum_{\substack{D_2 \in \pi_1 \\ D_2 \setminus \{v\} \subseteq C_2}} BN(K^+, H_1, D_2) + 1)$$

if $\delta(v, \pi_1, C_1)\delta(v, \pi_1, C_2) = 1$, or

$$(\sum_{\substack{D_1 \in \pi_1 \\ D_1 \setminus \{v\} \subseteq C_1}} BN(K^+, H_1, D_1) + 1) \sum_{\substack{D_2 \in \pi_1 \\ D_2 \setminus \{v\} \subseteq C_2}} BN(K^+, H_1, D_2)$$

if $\delta(v, \pi_1, C_1) = 1$ but $\delta(v, \pi_1, C_2) = 0$, or

$$\sum_{\substack{D_1 \in \pi_1 \\ D_1 \setminus \{v\} \subseteq C_1}} BN(K^+, H_1, D_1)(\sum_{\substack{D_2 \in \pi_1 \\ D_2 \setminus \{v\} \subseteq C_2}} BN(K^+, H_1, D_2) + 1)$$

if $\delta(v, \pi_1, C_1) = 0$ but $\delta(v, \pi_1, C_2) = 1$, or

$$\sum_{\substack{D_1 \in \pi_1 \\ D_1 \setminus \{v\} \subseteq C_1}} BN(K^+, H_1, D_1) \sum_{\substack{D_2 \in \pi_1 \\ D_2 \setminus \{v\} \subseteq C_2}} BN(K^+, H_1, D_2)$$

otherwise.

The result follows by considering each of the four cases above and simplifying equation 18 accordingly.

(iv) By definition (equation 13) and observation 4.3

$$EIP(K) = \sum H_1 \subseteq B'(K^+) \sum_{H_2 \subseteq S} P_{B'(K)}[H_1 \cup H_2 \parallel Y] \sum_{\substack{CC \text{ connected} \\ \text{component of } H_1 \cup H_2 \\ V(CC) \cap V(K) = \emptyset}} |V(CC)|^2$$

Notice that

$$\sum_{\substack{CC \text{ connected} \\ \text{component of } H_1 \cup H_2 \\ V(CC) \cap V(K) = \emptyset}} |V(CC)|^2 = \sum_{\substack{C_1 \text{ connected} \\ \text{component of } H_1 \\ V(C_1) \cap V(K) = \emptyset}} |V(C_1)|^2 + |V(C_v)|^2$$

where C_v is the connected component of H_1 that contains the removed node v if H_2 has no edges, otherwise C_v is the empty graph.

Notice also that $|V(C_v)| = |BN(K^+, H_1, \{v\})| + 1$. Thus,

$$EIP(K) = \sum_{H_1 \subseteq B'(K^+)} \sum_{H_2 \subseteq S} P_{B'(K)}[H_1 \parallel Y_1] \sum_{\substack{C_1 \text{ connected} \\ \text{component of } H_1 \\ V(C_1) \cap V(K^+) = \emptyset}} P_S[H_2 \parallel Y]\ |V(C_1)|^2 +$$

$$\sum_{\substack{(\pi_1, \pi_2) \\ in\ \Pi(v, K^+, S)}} \sum_{H_1 \in X_1} \sum_{H_2 \in X_2} P_{B'(K)}[H_1 \cup H_2 \parallel Y]\ (|BN(K^+, H_1, \{v\})| + 1)^2$$

Simple algebraic manipulation of the expression above concludes the proof. ∎

We can use lemmata 4.1, 4.2, and 4.3 to reduce any k-tree G to a k-clique R. We compute *Res(G)* by combining the information in the state of the k-clique R with the effect of the edges between nodes in R. Before computing *Res(G)* we extend the values in the state of R (statistics about $B'(R)$) to values about the graph G itself. Some additional notation is in order. Let $Res'(G)$ denote the expected number of *ordered* pairs of nodes in G that can communicate (including pairs of the form (u, u)). So

$$Res'(G) = \sum_{H \leq G} P_G[H] \sum_{\substack{CC \text{ connected} \\ \text{component of } H}} |V(CC)|^2$$

Notice that by equation 1 in section 2

$$Res(G) = \frac{1}{2}\,(Res'(G) - \sum_{v \in V} p_v)$$

Therefore we only need to prove that $Res'(G)$ can be computed from the state of the root R and the effect of the edges between nodes in R.

To account for the effect of the edges between nodes in R we define the following functions. Let π be any non-empty subpartition of the nodes in the root R, and $C \in \pi$, define

$$EP'(\pi, R, C) = \sum_{H \in SG(G,R,\pi)} P_G[H]\; N(H, C)^2$$

where $N(H, C) = |\{y \in G \mid y \overset{H}{\sim} C\}|$. Finally, let $EIP'(R)$ denote the expected number of ordered pairs (u, v) of nodes in G that can communicate such that $u \not\sim R$ and $v \not\sim R$. So

$$EIP'(R) = \sum_{H \leq G} P_G[H] \sum_{\substack{CC \text{ connected} \\ \text{component of } H \\ V(CC) \cap V(R) = \emptyset}} |V(CC)|^2$$

The following lemma states how to compute $Res'(G)$ from the state of the root R.

Lemma 4.4 (termination) *Let $G = (V, E)$ be a k-tree network and R be a root of G obtained by applying the reduction paradigm and lemmata 4.1-4.3 to G. Then*

(i) For all π, C, such that π is a non-empty subpartition in $\Pi(V)$, and C is a block of π

$$EP'(\pi, R, C) = \sum_{\substack{(\pi_1, \pi_2) \\ \pi_1 \wedge \pi_2 \text{ part. of } V_\pi \\ \pi_1 \vee \pi_2 = \pi}} P_R[\pi_2]\,(s(\pi_1, R)\,|C|^2 + \sum_{\substack{D \in \pi_1 \\ D \subseteq C}} (2\,|C|\,E(\pi_1, R, D) + EP(\pi_1, R, D, D)))$$

(ii) $EIP'(R) = EIP(R)$

(iii) $Res'(G) = EIP'(R) + \sum_{\pi \in \Pi(R)} \sum_{C \in \pi} EP'(\pi, R, C)$

Proof: The proofs follow easily by algebraic manipulation of the definitions of $EP'(\pi, R, C)$, $EIP'(R)$, and $Res'(G)$. We present some details of the proof for (i) only. Let Y denote the condition $up(V_{\pi_1}) \wedge dn(V(R)\backslash V_{\pi_1})$. Clearly,

$$\begin{aligned} EP'(\pi, R, C) &= \sum_{H \in SG(G,R,\pi)} P_G[H]\, N(H, C) \\ &= \sum_{\substack{(\pi_1,\pi_2) \\ \pi_1 \wedge \pi_2 \text{ part. of } V_\pi \\ \pi_1 \vee \pi_2 = \pi}} \sum_{\substack{H_1 \text{ in} \\ SG(B'(R),R,\pi_1)}} \sum_{\substack{H_2 \text{in} \\ SG(R,R,\pi_2)}} P_{B'(R)}[H_1 \,||\, Y]\, P_R[H_2]\, N(H_1 \cup H_2, C)^2 \end{aligned}$$

and the result follows because

$$N(H_1 \cup H_2, C)^2 = |C|^2 + 2\,|C| \sum_{\substack{D \in \pi_1 \\ D \subseteq C}} BN(H_1, D) + (\sum_{\substack{D \in \pi_1 \\ D \subseteq C}} BN(H_1, D))^2$$

∎

Therefore, lemmata 4.1-4.4 and the general reduction paradigm (Algorithm 1 in section 2) give us the following theorem.

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

Proof: Correctness follows from lemmata 4.1-4.4. Timing follows from lemmata 4.1-4.4 and an implementation of Algorithm 1 in section 2 that keeps a stack of k-leaves and uses an adjacency list representation of the the graph (see [12] for an example of such an implementation). ∎

Although our algorithm runs in $\mathcal{O}(n)$ time, the constants involved are exponential in k. This seems unavoidable as any graph on n nodes is a partial n-tree and the resilience problem is NP-hard in general. Thus our algorithm is of practical interest for small values of k only.

Even though we are interested in the asymptotic time complexity of the resilience algorithm, we include a table that gives some idea of the magnitude of the constants involved (see Table 3). The second column of Table 3 presents the number of subpartitions of a set of k elements, i.e., $\sum_{i=1}^{k+1} \left\{ {k+1 \atop i} \right\}$, where $\left\{ {n \atop m} \right\}$ is the number of ways to partition a set of n elements into m non-empty disjoint subsets (a Stirling number of the second kind). The third column of Table 3 shows the number of values that constitute the state of a k-clique (the *size* of $state(K)$). A naive implementation of our algorithm makes $k = 4$ already impractical (consider the number of join operations performed in the reduction step). A careful implementation of the reduction step may make our algorithm practical for $k = 4$.

4.2 Complexity of the resilience problem on partial k-tree networks

We can compute the resilience of a partial k-tree network G by finding an *embedding* in a k-tree G', assigning probability zero to the added edges, and then applying the resilience algorithm for k-trees to G'. In [2], Arnborg, Corneil and Proskurowski give an $\mathcal{O}(n^{k+2})$ time algorithm to find an embedding of a partial k-tree, for a fixed k. However, for $k = 2$ and $k = 3$ the embedding of a

k	$\|\Pi(K)\|$	size of $state(K)$
1	2	5
2	5	17
3	15	69
4	49	293

Table 3: Number of subpartitions of a k-clique and number of values in $state(K)$.

partial k-tree in a k-tree can be found in $\mathcal{O}(n)$ time ([17], [13]). When $k = 1$ we simply find the resilience of each 1-tree in the forrest G. Therefore, we can state the following corollary of theorem 4.1

Corollary 4.1 *Let G be a partial k-tree network that has n nodes. The resilience of G can be computed in $\mathcal{O}(n^{k+2})$ time. If an embedding of G in a k-tree is given, or $k \leq 3$, Res(G) can be computed in $\mathcal{O}(n)$ time.*

We can also use theorem 4.1 to devise an NC algorithm that computes the resilience of a partial k-tree network. Consider A, a sequential algorithm obtained using the reduction paradigm. Let us assume that A runs in linear time on partial k-trees given with an embedding in a k-tree. Bodlaender [6] has proved that if the initialization step, each reduction step, and the termination step of A can each be solved in NC, then there is an NC algorithm to solve the same problem (e.g., the resilience problem) on *partial* k-trees (assuming only that k is fixed). From the identities used in lemmata 4.1, 4.2, 4.3, and 4.4 we clearly see that Bodlaender's result is applicable. So, we obtain the following corollary.

Corollary 4.2 *Let G be a partial k-tree network given with an embedding in a k-tree. There is an NC algorithm that computes the resilience of G.*

Corollary 4.2 is mainly of theoretical interest as the number of processors, although polynomial in the number of nodes of the graph, is very large [6].

5 Conclusions

The reduction paradigm introduced in [4] is a powerful tool to solve reliability problems on partial k-tree networks. We have developed an $\mathcal{O}(n)$ time algorithm to compute the resilience of partial k-tree networks given with an embedding in a k-tree (for a fixed value of k). This algorithm was obtained by generalizing and speeding up an $\mathcal{O}(n^2)$ time algorithm for the same problem on fail-safe k-tree networks [12]. The speed up was achieved by keeping more information in the state of each k-clique, namely, the values $EP(\pi, K, C_1, C_2)$ and $EIP(K)$. The generalization was attained by modeling the state of a network using subgraphs instead of partial graphs and by computing conditional probabilities (e.g., $s(\pi, K)$ is now a conditional probability). We can use this same generalization technique to define an $\mathcal{O}(n)$ time algorithm to compute the l-terminal reliability (for a fixed value l) of partial k-tree networks given with an embedding in a k-tree (we generalize the algorithm given in [4], which assumes that nodes are fail-safe). An NC algorithm can also be derived from our sequential algorithm and results in [6].

It is easy (but tedious) to verify that a previously known linear time algorithm to compute the resilience of partial 2-tree networks with fail-safe nodes [12] is a special case of the algorithm presented in section 4. We need only substitute $k = 2$, $p_v = 1$ for all nodes v in the network, and make a few ad-hoc simplifications (e.g., eliminate redundant information and change slightly the definition of $B'(K)$).

Because of the large constants involved in our algorithm, the result is of practical interest for small values of k only ($k \leq 4$). However, as partial 4-trees include several important classes of graphs (e.g., series-parallel, outerplanar, Halin, Δ-Y reducible, and Cube-free graphs [9]), the domain of application of the algorithm is still considerable.

References

[1] A. T. Amin, K. T. Siegrist, and P. J. Slater. On the expected number of pairs of connected vertices: pair connected reliability. In *Proceddings of the Second New Mexico Conf. on Applications of Graph Theory to Computer Networks*, 1986.

[2] 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.

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

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

[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] H. L. Bodlaender. NC-algorithms for graphs with small treewidth. Technical report, Department of Computer Science, University of Utrecht, 1988.

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

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

[9] E. L. El-Mallah and C. J. Colbourn. Partial k-tree algorithms. In *Proceedings of the 250th Anniversary Conference on Graph Theory*, March 13-15 1986.

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

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

[12] E. Mata-Montero. Resilience of partial k-tree networks. Technical Report CIS-TR-89-09, Department of Computer and Information Science, University of Oregon, 1989.

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

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

[15] 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.

[16] P. J. Slater. A summary of results on pair-connected reliability. Technical Report 556, Department of Mathematical Sciences, Clemson University, 1987.

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