American Journal of Computational Mathematics, **2012, 2, 316-320**
doi:10.4236/ajcm.2012.24043 Published Online December 2012 (http://www.SciRP.org/journal/ajcm)

An Inclusion-Exclusion Algorithm for Network Reliability with Minimal Cutsets

Yan-Rui Sun, Wei-Yi Zhou
[1]School of Sciences, Northeastern University, Shenyang, China
[2]School of Electronic Information, Wuhan University, Wuhan, China
Email: yanruisun@126.com, weiyi0564@126.com

Received August 10, 2012; revised October 9, 2012; accepted October 23, 2012

ABSTRACT

The inclusion-exclusion formula (IEF) is a fundamental tool for evaluating network reliability with known minimal paths or minimal cuts. However, the formula contains many pairs of terms which cancel. Using the notion of comparable node partitions some properties of canceling terms in IEF are given. With these properties and the thought of "dynamic programming" method, a simple and efficient inclusion-exclusion algorithm for evaluating the source-to-terminal reliability of a network starting with cutsets is presented. The algorithm generates all the non-canceling terms in the unreliability expression. The computational complexity of the algorithm is $O\left(n+m^3+M\right)$, where n and m are the numbers of nodes and minimal cuts of the given network respectively, M is the number of terms in the final symbolic unreliability expression that generated using the presented algorithm. Examples are shown to illustrate the effectiveness of the algorithm.

Keywords: Inclusion-Exclusion Formula; Network Reliability; Minimal Cutset; Dynamic Programming

1. Introduction

The reliability of a network is an important parameter in design and operation of networks. There are many methods to compute the reliability of networks [1,2]. Several algorithms exist in the literature for evaluating the reliability of a directed graph by inclusion-exclusion formula (IEF) based on either path (k-tree) enumeration or cutset enumeration [3-8]. In finding the k-terminal reliability by IEF there are two approaches, one based on enumerating all k-trees and the other based on enumerating all k-terminal cuts. If there are m minimal paths (or cuts) in a graph, there are 2^m-1 possible intersection terms in IEF. However, the number of non-cancelling terms in IEF is considerably less.

Starting with the set of paths (or k-trees) of a directed graph, Satyanarayana and coworkers [5,6] developed methods of identifying non-cancelling terms in IEF. They showed that the non-canceling terms of the source-to-terminal reliability correspond one-to-one with the p-acyclic subgraphs of the given graph. An algorithm was given for generating all the p-acyclic subgraphs of a directed graph [5]. Buzacott [3] gave a corresponding result for the non-cancelling terms in IEF starting with the set of cuts of a graph. Since each term in the resulting formula is associated with a partition of the set of nodes of the graph, it was called the node partition formula. Find all the node partitions of a graph is a very tedious work.

Using a lemma (the Lemma 3.4 of [3]) of incomparable node partitions of [3] and characteristics of canceling terms in IEF, by the thought of "dynamic programming" method a simple and efficient inclusion-exclusion algorithm is given in this paper for evaluating the source-to-terminal reliability of a graph based on minimal cuts. The algorithm generates only the non-canceling terms of the reliability expression of the graph.

2. Nomenclature, Notation and Assumption

A network is modeled as a directed graph $G\left(N,E\right)$ (abbreviated to G) which consists of a set of nodes N and a set of edges (links) E. A node $s\in N$ is the source of G and $t\in N-\left\{s\right\}$ is the terminal of G.

2.1. Nomenclature

Source to terminal (s – t) reliability: the probability that the source s is connected to the terminal node t by paths working edges.

$s-t$ cut: a subset of edges whose removal divides the node set of the network into two parts N_i and $\bar{N}_i$ such that $N_i\bigcup\bar{N}_i=N$, $s\in N_i$ and $t\in\bar{N}_i$, *i.e.* the edge set from N_i to $\bar{N}_i$.

Minimal $s-t$ cut: $s-t$ cut which no longer remains a $s-t$ cut if its any edge is removed.

Candidate child set of N_i: an ordered set $\left\{N_{i_1}, N_{i_2}, \cdots, N_{i_r}\right\}$ consisting of all the node sets such that $N_i \subset N_{i_j}$ $\left(j=1,2,\cdots,r\right)$, $\left|N_{i_1}\right| \le \left|N_{i_2}\right| \le \cdots \le \left|N_{i_r}\right|$.

Incomparable node sets: a pair of node sets N_1 and N_2 such that $N_1 \not\subset N_2$ and $N_2 \not\subset N_1$.

2.2. Notation

N_i: subset of N such that $s \in N_i$
$\bar{N}_i$: complement of N_i, and $t \in \bar{N}_i$
$\left(N_i, \bar{N}_i\right)$: cut, *i.e.* the edge set from N_i to $\bar{N}_i$
C_i: 1) cut$\left(N_i, \bar{N}_i\right)$
2) event that all the edges of cut C_i fail
C_iC_j: 1) union of all the edges of cuts C_i and C_j
2) intersection of events C_i and C_j
U_i: union of *i* cuts
$\sigma(N_i)$: candidate child set of N_i
$\Pr(\bullet)$: the probability of event $\bullet$
$Q_G(s,t)$: source-to-terminal ($s-t$) unreliability of *G*

2.3. Assumption

1) *G* has perfectly reliable nodes and s-independent 2-state (good and failed) edges, and the reliability of each edge has been given.

2) Let $C_i=\left(N_i, \bar{N}_i\right)$ and $C_j=\left(N_j, \bar{N}_j\right)$ be mincuts of *G*, then $C_{ij}=\left(N_i \cap N_j, \overline{N_i \cap N_j}\right)$ is also a mincut of *G*.

3. Preliminaries

Let $\left\{C_1, C_2, \cdots, C_m\right\}$ be the set of mincuts of a given network *G* where C_i corresponds one-to-one with the node set N_i, *i.e.* $C_i=\left(N_i, \bar{N}_i\right)$ ($i=1,2,\cdots,m$). The $s-t$ unreliability of *G*, by IEF, can be expressed as

$$\begin{aligned} &Q_G(s,t) \\ &=\Pr\left(C_1 \cup C_2 \cup \cdots \cup C_m\right) \\ &=\sum_{1\le i\le m}\Pr\left(C_i\right)-\sum_{1\le i<j\le m}\Pr\left(C_iC_j\right) \\ &\quad+\sum_{1\le i<j<k\le m}\Pr\left(C_iC_jC_k\right)+\cdots+\left(-1\right)^{m-1}\Pr\left(C_1C_2\cdots C_m\right) \end{aligned} \tag{1}$$

the summations are over all mincuts and mincut combinations.

In formula (1), there exist 2^m-1 possible terms. But it is possible that $U_i \equiv U_j$ for some i, j, $i \ne j$. Indeed the most vexing problem in reliability analysis using (1) is the appearance of large numbers of pairs of identical terms with opposite sign, which cancel. Find the characteristic of canceling terms in (1) is the keystone of an efficient algorithm. Buzacott gave a simple and very useful lemma (Lemma 3.4 of Ref. [3]) to identify some canceling terms in (1).

Lemma 1. Given any two mincuts $C_i=\left(N_i, \bar{N}_i\right)$ and $C_j=\left(N_j, \bar{N}_j\right)$ of *G* such that N_i and N_j are incomparable, all terms in IEF containing both C_i and C_j cancel if $C_{ij}=\left(N_i \cap N_j, \overline{N_i \cap N_j}\right)$ is also a mincut [3].

In formula (1), assume that $\left|N_1\right| \le \left|N_2\right| \le \cdots \le \left|N_m\right|$. According to Lemma 1, (1) can be changed into:

$$\begin{aligned} Q_G(s,t) &= \Pr\left(C_1 \cup C_2 \cup \cdots \cup C_m\right) \\ &= \sum_{1\le i\le m}\Pr\left(C_i\right)-\sum_{\substack{N_i\subset N_j \\ 1\le i<j\le m}}\Pr\left(C_iC_j\right) \\ &\quad+\sum_{\substack{N_i\subset N_j\subset N_k \\ 1\le i<j<k\le m}}\Pr\left(C_iC_jC_k\right) \\ &\quad-\cdots+\left(-1\right)^{k-1}\sum_{\substack{N_{i_1}\subset N_{i_2}\subset\cdots\subset N_{i_k} \\ 1\le i_1<i_2<\cdots<i_k\le m}}\Pr\left(C_{i_1}C_{i_2}\cdots C_{i_k}\right) \end{aligned} \tag{2}$$

the summations are over all mincuts and mincut combinations that satisfy the given conditions.

The terms in (2) can correspond one-to-one the vertex of the *m* rooted trees with the following properties.

1) The root vertex of each rooted tree is the vertex N_i corresponding to cut set C_i, its weight is C_i, sign is +1.

2) Sons of each vertex N_i in every rooted tree are all elements in $\sigma\left(N_i\right)$, each son's weight is the union of its father's weight and the cut set corresponding to this son vertex, sign is its father's sign times –1.

For example, let $N_1 \subset N_2 \subset N_3 \subset N_4$, $C_i=\left(N_i, \bar{N}_i\right)$ $\left(i=1,2,3,4\right)$. **Figure 1** are four rooted trees.

In **Figure 1(a)**, tree (N_4) has a only vertex, the root vertex N_4. Its weight is C_4, sign is 1. In **Figure 1(c)**, tree (N_2)'s root vertex is N_2, its weight is C_2, sign is 1. N_2 has two sons: N_3 and N_4. The son vertex N_3's weight equals to C_2C_3, sign is –1; N_4's weight equals to C_3C_4, sign is –1. N_3's son is N_4, here N_4's weight equals to $C_2C_3C_4$, sign is $1\left(=-1\times\left(-1\right)\right)$.

The weight with its sign of each node in the rooted trees one-to-one corresponds with the term in the expression of formula (2). So we discuss *m* rooted trees' generating. The rooted tree whose root is N_i is denoted as tree $\left(N_i\right)$.

In fact, if we generate the rooted trees in non-increasing order of root's modulus and generate the sons of each vertex in non-decreasing order of the son's modulus, we can use the trees which have already been generated to generate the following trees. For example, **Figure 1** are four rooted trees, where tree $\left(N_3\right)$ is a branch of tree $\left(N_2\right)$, tree $\left(N_2\right)$ and tree $\left(N_3\right)$ are the branches of tree $\left(N_1\right)$. If tree $\left(N_3\right)$ is generated firstly, we can use the result when we generate tree $\left(N_2\right)$. And when we

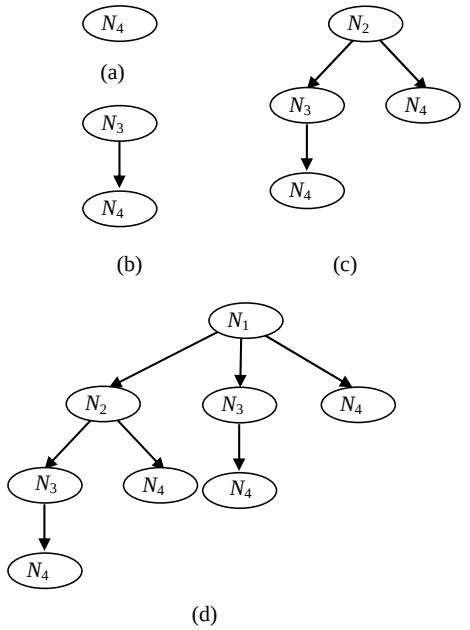

Figure 1. Rooted sub-trees. (a) Sub-tree(N_4); (b) Sub-tree(N_3); (c) Sub-tree(N_2); (d) Sub-tree(N_1).

generate tree (N_1), we can use tree (N_3) and tree (N_2) directly. This is the thought of "dynamic programming". By this thought the process of generating trees is greatly simplified.

In formula (2) there are still many terms that can cancel each other. The properties of canceling terms are discussed as follow.

Theorem 1. Let $C_i=(N_i,\bar{N}_i)$ $(i=1,2,3)$ be three mincuts of a directed graph G, $N_1\subset N_2\subset N_3$, and $C_1C_2C_3=C_1C_3$. Then for any mincuts $C_0=(N_0,\bar{N}_0)$ that $N_0\subset N_1$ and $C_4=(N_4,\bar{N}_4)$ that $N_4\supset N_3$ of G, $C_0C_1C_2C_3=C_0C_1C_3$, $C_1C_2C_3C_4=C_1C_3C_4$.

Theorem 2. Let $C_i=(N_i,\bar{N}_i)$ $(i=1,2,3)$ be three mincuts of a directed graph G with $N_1\subset N_2\subset N_3$. If $C_1C_2C_3\neq C_1C_3$ then doesn't exist mincut $C_0=(N_0,\bar{N}_0)$ of G, $N_0\subset N_1$ such that $C_0C_1C_2C_3=C_0C_1C_3$; and doesn't exist $C_4=(N_4,\bar{N}_4)$ of G, $N_4\supset N_3$ such that $C_1C_2C_3C_4=C_1C_3C_4$.

Theorems 1 and 2 imply that if we find out the all pairs of canceling terms that union of two cuts and three cuts, *i.e.* find out all U_2 and U_3 with $U_2=U_3$, the all canceling terms in (2) can be determined.

The following lemma gives a condition that $U_2=U_3$ in (2).

Lemma 2. Let $C_i=(N_i,\bar{N}_i)$ $(i=1,2,3)$ be three minimal cuts of a directed graph G and $N_1\subset N_2\subset N_3$. Then $C_1C_3=C_1C_2C_3$ if and only if $(N_2-N_1,N_3-N_2)=\varnothing$.

According to Theorems 1 and 2 and Lemma 2 we give the generating processes of the trees with above properties 1) and 2). Such that trees' vertices correspond with the non-canceling terms of (2).

4. Algorithm

This section presents an algorithm for efficiently generating all the non-canceling terms in (2). The algorithm has four parts, The main part is to generate all trees whose vertices correspond with the non-canceling terms of (2).

4.1. Algorithm

1). Find all the mincuts of the given directed network $G(N,E)$ which satisfies assumptions. Let $C_1,C_2,\cdots,C_m$ be the mincuts corresponding to the node partitions $N_1,N_2,\cdots,N_m$, respectively. Order the node partitions as $N_1,N_2,\cdots,N_m$ such that $|N_1|\le|N_2|\le\cdots\le|N_m|$.

2). Find $\sigma(N_i)$ $(i=1,2,\cdots,m)$.

3). Generate *m* rooted trees by the following Algorithm-Tree, *i.e.* generate all the non-canceling terms of $Q_G(s,t)$.

4). Sum up the weights with sign of vertices of all the trees to obtain the symbolic expression of $Q_G(s,t)$. Finally, we get the symbolic reliability expression of $R_G(s,t)=1-Q_G(s,t)$.

4.2. Algorithm Tree

By theorems 1 and 2, all the pairs of canceling terms in IEF can be known if we find the canceling terms which union of two and three cut sets. Using this property an algorithm "Algorithm Tree" is given. It has two parts: "Trees Generation" and "Weighted Trees". It generates rooted trees in the non-increase order of the root vertex's modulus, *i.e.* generates tree (N_m), tree (N_{m-1}), $\cdots$, tree (N_1) successively.

4.2.1. Trees Generation

We shall give an algorithm to generate all trees as follow.

Algorithm Trees Generation

Input: 1) the node partitions of G: $N_1,N_2,\cdots,N_m$ such that $|N_1|\le|N_2|\le\cdots\le|N_m|$ and the corresponding mincuts $C_1,C_2,\cdots,C_m$.

2) the candidate child sets:

$$\sigma(N_1)=\{N_{11},N_{12},\cdots,N_{1t_1}\},\cdots,$$

$$\sigma(N_k)=\{N_{k1},N_{k2},\cdots,N_{kt_k}\},$$

$$\cdots,\sigma(N_{m-1})=\{N_m\},\sigma(N_m)=\varnothing.$$

Output: all the trees.

Begin

Step 1. Generate the first tree (N_m) with the only vertex, *i.e.* root vertex N_m.

Step 2. Generate the second tree (N_{m-1}) with a root

vertex N_{m-1} and its only son vertex N_m.

Step 3. Suppose that trees (N_{m-i+1}), $i=1,2,\cdots,k\le m$ have been generated. Generate tree (N_{m-k}).

1) The root of tree (N_{m-k}) is N_{m-k}.

2) Generate sons (we call them the first-generation offspring) of N_{m-k}: $N_{m-k,1},N_{m-k,2},\cdots,N_{m-k,t_{m-k}}$ (where $\{N_{m-k,1},N_{m-k,2},\cdots,N_{m-k,t_{m-k}}\}=\sigma(N_{m-k})$, $N_{m-k,t_{m-k}}=N_m$);

$G_1(N_{m-k})\leftarrow\sigma(N_{m-k})=\{N_{m-k,1},N_{m-k,2},\cdots,N_{m-k,t_{m-k}}\}$.

3) While $G_1(N_{m-k})\neq\varnothing$ do

Begin

Denote the element with the minimal modulus in $G_1(N_{m-k})$ as $N_{m-k,i}$, $1\le i\le t_{m-k}$.

$G_1(N_{m-k})\leftarrow G_1(N_{m-k})-\{N_{m-k,i}\}$.

Substitute N_{m-k}'s son $N_{m-k,i}$ with tree $(N_{m-k,i})$. Denote the son set of this vertex $N_{m-k,i}$ as $G_1(N_{m-k,i})$. Suppose

$G_1(N_{m-k,i})=\{N_{m-k,i_1},N_{m-k,i_2},\cdots,N_{m-k,i_{k_i}}\}$ (in the non-decreasing order of their modulus).

$G_0(N_{m-k,i})\leftarrow G_1(N_{m-k,i})$.

For j = 1 to k_i

Begin

If $N_{m-k,i_j}\in G_1(N_{m-k})$ and $(N_{m-k,i}-N_{m-k},N_{m-k,i_j}-N_{m-k,i})=\varnothing$, then

$G_1(N_{m-k,i})\leftarrow G_1(N_{m-k,i})-\{N_{m-k,i_j}\}$;

$G_1(N_{m-k})\leftarrow G_1(N_{m-k})-\{N_{m-k,i_j}\}$;

Cut N_{m-k}'s son N_{m-k,i_j} and $N_{m-k,i}$'s son N_{m-k,i_j} with its offspring from current tree $(N_{m-k,i})$;

Else next j

End

If $G_0(N_{m-k,i})=G_1(N_{m-k,i})$, then mark $N_{m-k,i}$, called it still node, denoted as still node $(N_{m-k,i})$.

End.

Step 4. Repeat step 3 until all the trees (N_i), $i=m,m-1,\cdots,2,1$ have been generated.

End.

4.2.2. Weighted Trees

Tree's weight is defined the sum of root's and all vertices' weights with their sign.

In the order of generating trees, starting from each tree's root vertex gives each vertex a weight in depth-first-search. The root's weight is the all edges of the cut set corresponding to the root vertex, sign is +1. Each non-still vertex's weight equals to the union of its father's weight and all the edges of the cut set corresponding this vertex, its sign is its father's sign times –1. Each still node's weight equals to the union of its father's weight and the tree's weight whose root is this vertex and its sign is its father's sign times –1.

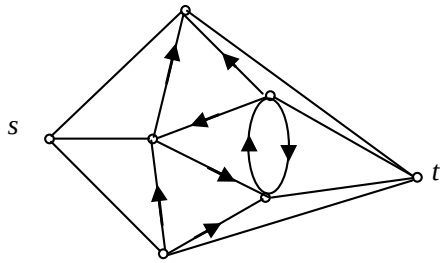

Figure 2. Network.

5. Computational Complexity

The main part of the presented algorithm is "Algorithm Sub-tree". It has two parts. One is Sub-trees Generation, the other is Weighted Sub-trees. The main work of "algorithm Sub-trees Generation" is to determine whether there exist edges between two sets. It at most needs $(m-1)+(m-2)+\cdots+1=m(m-1)/2$ comparison operations for each sub-tree. "Weighted sub-trees" runs in $O(M)$, where M is the number of terms in the last symbolic un-reliability expression. In fact, M is more smaller than 2^m. For example, the network in **Figure 2**, m = 18, $2^m=2^{18}=262144$, but M = 151. For each of sub-trees, other operations at most take $O(m)$ time. It takes $O(n)$ time to find all mincuts, where n is the number of nodes of a given network. It takes $O(m)$ time to find each candidate child set. So the computational complexity of the presented algorithm is $O(n+m^3+M)$.

6. Conclusion

This paper presents an efficient algorithm for evaluating the reliability of network based mincuts. The algorithm generates all the non-canceling terms in the unreliability expression. By the thought of "dynamic programming" method each vertex at most generates two generations children in every sub-tree. The number of vertices of the generated sub-trees are more smaller than the number of non-canceling terms in $Q_G(s,t)$'s expression. The algorithm has smaller time complexity.

REFERENCES

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

[2] M. O. Ball, C. J. Colbourn and J. S. Provan, "Network Reliability," Handbook of Operations Research: Network Models, Elsevier North-Holland, Amsterdam, Vol. 7, 1995, pp. 673-762.

[3] J. A. Buzacott, "Node Partition Formula for Directed Graph Reliability," *Networks*, Vol. 17, No. 2, 1987, pp. 227-240. doi:10.1002/net.3230170207

[4] J. A. Buzacott and S. K. Chang, "Cut Set Intersections and Node Partition," *IEEE Transactions on Reliability*, Vol. 31, No. 4, 1982, pp. 385-389.

[5] A. Satyanarayana and A. Prabhakar, "New Topological Formula and Rapid Algorithm for Reliability Analysis of Complex Networks," *IEEE Transactions on Reliability*, Vol. 27, No. 1, 1978, pp. 82-100. doi:10.1109/TR.1978.5220266

[6] A. Satyanarayana and J. N. Hagstrom, "A New Algorithm for Reliability Analysis of Multi-Terminal Networks," *IEEE Transactions on Reliability*, Vol. 30, No. 4, 1981, pp. 325-334. doi:10.1109/TR.1981.5221103

[7] L. C. Zhao and F. J. Kong, "A New Formula and an Algorithm for Reliability Analysis of Network," *Microelectron Reliability*, Vol. 37, No. 4, 1997, pp. 511-518.

[8] W. C. Yeh, "A Greedy Branch-and-Bound Inclusion-Exclusion Algorithm for Calculating the Exact Multi-State Network Reliability," *IEEE Transactions on Reliability*, Vol. 57, No. 1, 2008, pp. 88-93. doi:10.1109/TR.2008.916871

