An O(n^2)-time algorithm for computing the *K*-terminal reliability of rooted directed path graphs

Chao-Chun Ting*, Min-Sheng Lin**
Department of Electrical Engineering
National Taipei University of Technology, Taipei, Taiwan
*jackey_din2000@yahoo.com.tw, **mslin@ee.ntut.edu.tw

Abstract

Let G denote a graph, and $K \subseteq V(G)$ represent a set of target vertices. Assume that the non-target vertices of G fail independently with given probabilities. The K-terminal reliability of G is defined as the probability that all target vertices in K are connected. Computing K-terminal reliability is #P-complete for general graphs, yet solvable in polynomial time for interval graphs. This work proposes an $O(n^2)$-time algorithm for computing the K-terminal reliability of n-vertex rooted directed path graphs, which are a superclass of interval graphs.

1 Introduction

Commonly found in engineering, reliability analysis problems are characterized by network reliability, i.e. the ability of a network to survive a random or purposeful attack [1-4]. A network can be modeled as a probabilistic graph, in which each edge and vertex has an associated failure probability. The network model presented in this work is a graph *G* with perfect edges and imperfect vertices. Related applications of these graph models include *Radio Broadcast Networks* (*RBN*) [2]. An *RBN* can be modeled using a probabilistic graph in which each site is represented by a vertex with a known probability of failure; an edge exists between the vertices if and only if they can communicate with each other.

A subset of *K* vertices is selected in this work as the target vertices of a network. The *K*-terminal reliability (*KTR*) is the probability that all vertices in *K* are connected to each other by a set of working non-target vertices. Aboeifotoh and Colbourn [5] confirmed that the computation of *KTR*, i.e. the *KTR* problem, is #*P*-complete for general graphs and remains so even for chordal graphs. Valiant [6] defined the class of #*P* problems as those that involve counting access computations for problems in *NP*, while the class of #*P*-complete problems includes the hardest problems in #*P*. As is well known, all algorithms for solving these problems have exponential time complexity, explaining the near impossibility of developing efficient algorithms for solving this class of problems. However, considering only a restricted subclass of #*P*-complete problems can reduce this complexity.

Gavril [7] verified that chordal graphs are the intersection graphs of a family of subtrees in a clique tree. A tree *T* is a clique tree for a graph *G* if each node in *T* corresponds to a maximal clique in *G*. Hereinafter *clique* refers to a *maximal clique*. For $v \in V(G)$, let $C(v)$ denote the set of all cliques of *G* containing vertex *v*. Therefore, *G* is a chordal graph if and only if $C(v)$ is a subtree in a clique tree *T* for every $v \in V(G)$ [7]. This work investigates a restricted subclass of chordal graphs, i.e. rooted directed path graphs. *Rooted directed path graphs* consist of the intersection graphs of a family of directed subpaths in a rooted directed clique tree. A tree is called a rooted directed tree if one node has been designated as the root, and the edges have a natural orientation, away from the root. *Interval graphs* are rooted directed path graphs in which the clique tree is itself a path. Interval graphs are generally defined as the intersection graphs of a family of intervals on a real line. An interval graph *G* is *called proper interval graph* if there is an interval representation of *G* such that no interval contains another one properly. These graph classes are related to each other by the following proper inclusions:

proper interval graph $\subset$ interval graph $\subset$ rooted directed path graph $\subset$ chordal graph [8].

Permutation graphs, trapezoid graphs and *d*-trapezoids graphs are three other important intersection graphs. A *permutation graph* has an intersection model consisting of straight lines (one per vertex) between two parallels. *Trapezoid graphs* are the intersection graphs of a family of trapezoids (one per vertex) between two parallel lines. Trapezoid graphs properly contain both interval and permutation graphs. Flotow [9] generalized the definition of trapezoid graphs as follows. A *d-trapezoid graph* has an intersection model consisting of polygons, called *d*-trapezoids, between *d* parallel lines. The *d*-trapezoid graphs are common generalizations of interval and trapezoid graphs, such that 1-trapezoid graphs coincide with interval ones and 2-trapezoid graphs coincide with trapezoid ones.

Aboeifotoh and Colbourn [5] demonstrated that the *KTR* problem is #*P*-complete even for chordal graphs. That work also developed polynomial time algorithms for computing *KTR* of such interval and permutation graphs. Our earlier works derived a linear-time algorithm for proper interval graphs [10] as well as a polynomial time algorithm for interval, trapezoid and *d*-trapezoid graphs [11]. The status of the *KTR* problem for rooted directed path graphs has remained unclear until now. This work shows that an $O(n^2)$-time algorithm exists for computing the *K*-terminal reliability of a given *n*-vertex rooted directed path graph.

2 O(n^2)-Time Algorithm for Computing *KTR* of Rooted Directed Path Graphs

This section introduces a simple and efficient algorithm for solving the *KTR* problem for rooted directed path graphs. First, assume that the rooted directed clique tree *T* has been constructed for the rooted directed path graph *G*. This construction takes only linear time with an easy modification of the recognition algorithm in [12]. The rooted directed clique tree has the feature that the cliques containing any vertex form a subpath that is directed away from the root. For clarity, some notations and definitions used in this section are described as follows.

Notations

G	rooted directed path graph
K	set of target vertices
T	rooted directed clique tree corresponding to G
r	root clique of T
x	lowest common ancestor of all target cliques in T
$N(T)$	set of nodes (cliques) in T
T_k	subtree of T rooted at clique k ($T_r \equiv T$)
$CHD(k)$	set of children of clique k in T
k^+	parent of clique k in T
$V(k)$	set of vertices in clique k
$C(v)$	set of cliques which contains vertex v (Note: vertex $v \in V(k)$ iff clique $k \in C(v)$)
$S(k)$	set of vertices which start at clique k in T (Note: vertex $v \in S(k)$ iff $v \in V(k)$ and $v \notin V(k^+)$, i.e. $S(k) \equiv V(k) \setminus V(k^+)$)
$S^*(k)$	set of vertices which start at the clique k and all its descendants in T
$S(k,h)$	set of vertices which start at the cliques on the subpath from k to h in T
$EF(k,h)$	event that clique h fail to connect to its parent clique h^+ by using vertices in $S(k,h^+)$, where k is one of the ancestors of h in T
$EC(k,h)$	event that all leaf cliques of T_h can connect to clique h^+ by using vertices in $S^*(k)$, where k is one of the ancestors of h in T
$\Pr[\bullet]$	probability that event $\bullet$ occurs
q_v	failure probability of the vertex v; $q_v=0$ if v is a target vertex
$R(G,K)$	K-terminal reliability of the rooted directed path graph G

Definitions

- A clique in T is called a *target clique* if it contains at least one target vertex in G; otherwise, it is called a *non-target clique*.
- The *lowest common ancestor* of two cliques k and h, denoted by $lca(k,h)$, in T is the clique of greatest depth in T that is an ancestor of both k and h.

Given a rooted directed path graph G, the corresponding rooted directed clique tree T, and a set K of target vertices, $R(G,K)$ can be equivalently defined as the probability that all target cliques in T can connect to each other in T by using vertices of G. Let the clique x denote the lowest common ancestor of all target cliques in T. Obviously, all target cliques can connect to each other if and only if each target clique can connect to the clique x. Additionally, if clique k is a leaf and non-target clique in T, clique k can be deleted without affecting its reliability. Therefore, after deleting all non-target leaf cliques in T, $R(G,K)$ can be equivalently redefined as the probability that all leaf cliques of T connect to the clique x. That is,

$$R(G,K) = \Pr\left[\bigcap_{x' \in CHD(x)} EC(r,x')\right]$$

Since all subtrees $T_{x'}$, for $x' \in CHD(x)$, are disjoint from each other, all events $EC(r,x')$, for $x' \in CHD(x)$, are independent of each other. Thus

$$\mathrm{R}(G,K) = \prod_{x' \in CHD(x)} \Pr\left[EC(r,x')\right]$$

To compute $R(G,K)$, two lemmas, one that computes $\Pr[EF(k,h)]$ and one that computes $\Pr[EC(k,h)]$, for each clique $k \in N(T)$ and clique $h \in N(T_k) \setminus \{k\}$, are given as follows.

Lemma 1. For each clique $k \in N(T) \setminus \{r\}$ and clique $h \in N(T_k)$,

$$\Pr[EF(k^+,h)] = \Pr[EF(k,h)] \times \prod_{v \in S(k^+) \cap V(h)} q_v ,$$

and with boundary condition $\Pr[EF(k,k)] = 1$ for $k \in N(T)$.

Proof. By definition, $\Pr[EF(k,h)] =$

$\prod_{v\in S(k,h^+)\cap V(h^+)\cap V(h)} q_v$. As a result of $h\in N(T_k)$, ($k \to h^+ \to h$) is a directed subpath in T. Therefore, if $v\in S(k,h^+) \cap V(h)$, then $v\in S(k,h^+) \cap V(h^+)$ and thus $\Pr[EF(k,h)] = \prod_{v\in S(k,h^+)\cap V(h)} q_v$. Similarly, $\Pr[EF(k^+,h)] = \prod_{v\in S(k^+,h^+)\cap V(h)} q_v$. Obviously, $S(k,h^+) \subseteq S(k^+,h^+)$ and this implies that $\Pr[EF(k^+,h)] = \Pr[EF(k,h)] \times \prod_{v\in(S(k^+,h^+)\setminus S(k,h^+))\cap V(h)} q_v$. Since ($k^+ \to k \to h$) is also a directed subpath in T, $(S(k^+,h^+)\setminus S(k,h^+)) \cap V(h) = S(k^+) \cap V(h)$ and the lemma follows. □

Lemma 2. For each clique $k\in N(T)$ and clique $h\in N(T_k)\setminus\{k\}$,

$$\Pr[EC(k,h)] = \prod_{h'\in CHD(h)} \Pr\left[EC(k,h')\right] - \Pr[EF(k,h)] \times \prod_{h'\in CHD(h)} \Pr\left[EC(h,h')\right],$$

and with boundary condition $\Pr[EC(k,h)] = 1-\Pr[EF(k,h)]$ if h is a leaf clique in T.

Proof. Two events used to derive the lemma are defined as follows:

$EA \equiv \bigcap_{h'\in CHD(h)}$ {all leaf cliques of $T_{h'}$ are connected to clique h by using vertices in $S^*(k)$} and

$EB \equiv$ {clique h is connected to clique h^+ by using vertices in $S^*(k)$}.

By definition, $EA = \bigcap_{h'\in CHD(h)} EC(k,h')$. Notably, all subtrees $T_{h'}$, for $h' \in CHD(h)$, are disjoint from each other. Hence, all events $EC(k,h')$, for $h' \in CHD(h)$, are independent of each other. Thus $\Pr[EA] = \prod_{h'\in CHD(h)} \Pr\left[EC(k,h')\right]$. Furthermore, event EA can be expressed as union of two disjoint events:

$$EA = (EA\cap EB) \uplus (EA\cap \overline{EB}),$$

where $\uplus$ denotes disjoint union and $\overline{EB}$ denotes the complement of EB.

By definition, $EA\cap EB = EC(k,h)$. Therefore,

$$\prod_{h'\in CHD(h)} \Pr\left[EC(k,h')\right] = \Pr[EC(k,h)] + \Pr[EA\cap \overline{EB}].$$

Next, consider the event $EA \cap \overline{EB} = \bigcap_{h'\in CHD(h)} EC(k,h') \cap \overline{EB}$. Since all leaf cliques of $T_{h'}$ fail to connect to clique h^+, event $EC(k,h') \cap \overline{EB}$ can be refined as two events $E_1\cap E_2$, which are

$E_1 \equiv \bigcap_{h'\in CHD(h)}$ {all leaf cliques of $T_{h'}$ are connected to clique h by using vertices in $S^*(h)$} and

$E_2 \equiv$ {clique h fail to connect to clique h^+ by using the vertices in $S(k,h^+)$}.

Obviously, $\Pr[E_1] = \prod_{h'\in CHD(h)} \Pr\left[EC(h,h')\right]$ and $\Pr[E_2]=\Pr[EF(k,h)]$. Notably, the events E_1 and E_2 involve disjoint sets $S^*(h)$ and $S(k,h^+)$, respectively, and thus are independent of each other. Therefore,

$$\Pr[EA\cap \overline{EB}] = \Pr[EF(k,h)] \times \prod_{h'\in CHD(h)} \Pr\left[EC(h,h')\right]$$

and the lemma follows. □

Based on the above formulation, the algorithm for computing $R(G,K)$ of a rooted directed path graph G is formally described as follows.

Algorithm *Compute_KTR*
Input: A rooted directed path graph G, a set K of target vertices, and the failure probability of each vertex
Output: $R(G,K)$

1. Construct a rooted directed clique tree T of G with the root r;
2. Remove all leaf nodes from T that contain none of target vertex in K;
3. **for each** clique $k\in N(T)\setminus\{r\}$ **do** $S(k)\leftarrow V(k)\setminus V(k^+)$;
4. **call** *Compute_PrEF*;
5. **call** *Compute_PrEC*;
6. $x\leftarrow$**call** *Compute_LCA*;.
7. **return**($\prod_{x'\in CHD(x)} PrEC(r,x')$);

end-algorithm

Procedure *Compute_PrEF*

1. **for each** clique $k \in N(T)$ **do** $PrEF(k,k) \leftarrow 1$;
2. **for each** clique $k \in N(T)\backslash\{r\}$ encountered in the post-order traversal **do**
3. **for each** clique $h \in N(T_k)$ **do** $PrEF(k^+,h) \leftarrow PrEF(k,h)$;
4. **for each** vertex $v \in S(k^+)$ **do**
5. **for each** clique $h \in C(v)$ **do**
6. **if** clique $h \in N(T_k)$ **then** $PrEF(k^+,h) \leftarrow PrEF(k,h) \cdot q_v$;

end-procedure

Procedure *Compute_PrEC*

1. **for each** clique $k \in N(T)$ encountered in the post-order traversal **do**
2. **for each** clique $h \in N(T_k)\backslash\{k\}$ encountered in the post-order traversal **do**
3. **if** (clique h is a leaf clique) **then**
4. $PrEC(k,h) \leftarrow 1 - PrEF(k,h)$;
5. **else**
6. $$PrEC(k,h) \leftarrow \prod_{h' \in CHD(h)} PrEC(k,h') - PrEF(k,h) \times \prod_{h' \in CHD(h)} PrEC(h,h');$$

end-procedure

Procedure *Compute_LCA*

1. $x \leftarrow$ one of target cliques in T;
2. **for each** target clique $k \in N(T)$ **do** $x \leftarrow lca(k,x)$;
3. **return**(x);

end-procedure

Theorem 1. An $O(n^2)$-time algorithm exists for computing the K-terminal reliability of a given n-vertex rooted directed path graph.

Proof. Assume that all sets, $N(T_k)$, $V(k)$, $S(k)$, $C(v)$ used in the algorithm are implemented using bit-vectors. Thus, set-related operations such as *finding a member* and *adding a member* take constant time and operations such as *finding the difference between sets* take linear time. Consider Algorithm *Compute_KTR*. In line 1, given a rooted directed path graph G, the rooted directed clique tree T can be constructed in linear time by easily modifying the recognition algorithm of Dietz *et al*. [13]. As is well known, an n-vertex chordal graph has at most n maximal cliques and rooted directed path graphs are chordal. Therefore, the tree T constructed in line 1 has at most n clique nodes, i.e. $O(|N(T)|)=O(n)$. After constructing T, deriving all of the sets $V(k)$ for $k \in N(T)$, and $C(v)$ for $v \in V(G)$ in $O(n^2)$ time is relatively easy. In line 2, removing all non-target leaf cliques from T can be completed in $O(n^2)$ time. Line 3 takes $O(|N(T)| \times n)=O(n^2)$ time to compute all $S(k)$.

Procedure *Compute_PrEF* needs $O(|N(T)|^2) = O(n^2)$ time to sort all cliques in the post-order and then takes $O(|N(T)| \times |N(T)|) = O(n^2)$ time to set the initial values of all $PrEF(k^+,h)$, for $k \in N(T)\backslash\{r\}$ and $h \in N(T_k)$, in line 3. Obviously, each vertex v belongs to at most one of $S(k^+)$ for $k \in N(T)\backslash\{r\}$ in line 4. Therefore, computing all $PrEF(k^+,h)$ in line 6 takes $O(\sum_{k \in N(T)\backslash\{r\}} (\sum_{v \in S(k^+)} |C(v)|)) = O(\sum_{v \in |V(G)|} |C(v)|) = O(n^2)$ time. Hence, procedure *Compute_PrEF* takes $O(n^2)$ time overall.

Next, the running time of procedure *Compute_PrEC* is analyzed. Clearly, line 6 takes $O(\sum_{k \in N(T)} (\sum_{h \in N(T_k)\backslash\{k\}} |CHD(h)|))$ time to compute all $PrEC(k,h)$. Since $\sum_{h \in N(T_k)\backslash\{k\}} |CHD(h)| = O(\text{the number of branches in } T_k) = O(|N(T_k)|)$, procedure *Compute_PrEC* takes $O(\sum_{k \in N(T)} |N(T_k)|) = O(n^2)$ time overall.

Finally, the running time of procedure *Compute_LCA* is analyzed. Before *Compute_LCA* procedure is executed, T can be preprocessed in $O(|N(T)|) = O(n)$ time. Therefore, for any pair of its nodes k and h, $lca(k,h)$ can be computed in constant time [13,14]. Since $lca(k,x)$ operation in line 2 executes $O(|N(T)|) = O(n)$ times, procedure *Compute_LCA* takes $O(n)$ time.

In summary, the overall computational complexity of the algorithm is $O(n^2)$. □

3 Conclusion

To our knowledge, this work solves the

complexity of the *KTR* problem for rooted directed path graphs for the first time. Notably, the classes of intersection graphs with polynomially solvable *KTR* problem extend from interval graphs [5] to rooted directed path graphs, yet still maintain the same time complexity of $O(n^2)$.

References

[1] Halsall, F. (1996). Data Communications, Computer Networks and Open Systems. 4th ed., Addison-Wesley.

[2] Tanenbaum, A. S. (2010). Computer Networks. 5th ed., Prentice-Hall.

[3] Lin, M. S. (2003). Linear-time algorithms for computing the reliability of bipartite and (#⩽ 2) star distributed computing systems. Computers & Operations Research, 30(11), 1697-1712.

[4] Lin, M. S. (2004). An O(k^2*log(n)) algorithm for computing the reliability of consecutive-k-out-of-n: F systems. IEEE Transactions on Reliability, 53(1), 3-6.

[5] Aboeifotoh, H. M., & Colbourn, C. J. (1990). Efficient algorithms for computing the reliability of permutation and interval graphs. Networks, 20(7), 883-898.

[6] Valiant, L. G. (1979). The complexity of enumeration and reliability problems. SIAM Journal on Computing, 8(3), 410-421.

[7] Gavril, F. (1974). The intersection graphs of subtrees in trees are exactly the chordal graphs. Journal of Combinatorial Theory, Series B, 16(1), 47-56.

[8] Monma, C. L., & Wei, V. K. (1986). Intersection graphs of paths in a tree. Journal of Combinatorial Theory, Series B, 41(2), 141-181.

[9] Flotow, C. (1995). On powers of m-trapezoid graphs. Discrete Applied Mathematics, 63(2), 187-192.

[10] Lin, M. S. (2002). A linear-time algorithm for computing K-terminal reliability on proper interval graphs. IEEE Transactions on Reliability, 51(1), 58-62.

[11] Lin, M. S., & Ting, C. C. (2013). Computing K-terminal reliability of d-trapezoid graphs. Information Processing Letters, 113(19-21), 734-738.

[12] Dietz, P. F., Furst, M., & Hopcroft, J. E. (1979). A linear time algorithm for the generalized consecutive retrieval problem. Cornell University.

[13] Harel, D., & Tarjan, R. E. (1984). Fast algorithms for finding nearest common ancestors. SIAM Journal on Computing, 13(2), 338-355.

[14] Bender, M. A., & Farach-Colton, M. (2000). The LCA problem revisited. In LATIN 2000: Theoretical Informatics (pp. 88-94). Springer Berlin Heidelberg.