The st-Connectedness Problem for a Fibonacci Graph

MARK KORENBLIT and VADIM E. LEVIT
Department of Computer Science
Holon Academic Institute of Technology
52 Golomb Str., P.O. Box 305, Holon 58102
ISRAEL

Abstract: - The paper presents a method for the solution of the st-connectedness problem for a Fibonacci graph. It is shown that this problem has a polynomial time complexity. The number of mincuts of a Fibonacci graph is computed.

Key-Words: - Fibonacci graph, mincut, operating path, probabilistic graph, reliability, st-connectedness, st-dag

1 Introduction

We consider the well-known problem of computing the probability that there exists an operating path from a source to a target in a stochastic network (probabilistic graph). The problem and its generalizations concerning directed and undirected graphs belong to the class of network reliability problems. Network reliability has been considered in a large number of papers. The problem is NP-complete in network size in the general case (see [1], [2], [4], [6], [8]). In this paper, we investigate the problem in relation to a special graph called a Fibonacci graph.

The input to network reliability problems is a *probabilistic graph* $G=(V,E)$, where V is a set of *vertices* and E is a set of *edges*, representing pairs of vertices. If the pairs are ordered (i.e., the pair (v,w) is different from the pair (w,v)) then we call the graph *directed* (*digraph*). All edges of a probabilistic graph can fail randomly and independently of one another, according to certain known probabilities. Hence, each edge $e \in E$ is characterized by a known failure probability p_e and by an operation probability $q_e = 1 - p_e$.

We say that a graph $G'=(V',E')$ is a *subgraph* of $G=(V,E)$ if $V' \subset V$ and $E' \subset E$. A two-terminal directed acyclic graph (*st-dag*) has only one source s and only one target t. In an st-dag, every vertex lies on some path from s to t.

For a probabilistic graph G and specified vertices s and t of G, we define the *two-terminal reliability* to be the probability that there exists an *operating path* (a path of operating edges) between s and t. We call such a state a *system operation* and corresponding event is $EP(s,t)$. A state when no operating path exists between s and t is said to be a *system failure*. In the directed case, the problem of computing the probability $\Pr[EP(s,t)]$ is usually called *st-connectedness*.

We define a *cutset* or simply a *cut* to be a set of edges whose failure implies system failure. A *size of a cut* is a number of edges in the cut. A *mincut* is a minimal cut. A set of all mincuts of an st-dag is denoted $\mathbf{C}(s,t)$.

2 An st-Connectedness for a Fibonacci Graph

The notion of a *Fibonacci graph* (*FG*) was introduced in [3]. In such an st-dag, two edges leave each of its n vertices except the two final vertices (n–1 and n). Two edges leaving the i vertex ($1 \le i \le n-2$) enter the i+1 and the i+2 vertices. The single edge leaving the n–1 vertex enters the n vertex. No edge leaves the n vertex. This graph is illustrated in Fig. 1.

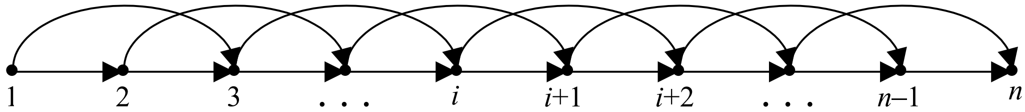

Fig.1. A Fibonacci graph

We explore the algorithm of Provan and Ball [5] for computing $\Pr[EP(s,t)]$ of a probabilistic *FG*. The algorithm determines two-terminal reliability in time that is polynomial in the number of mincuts. Some preliminary definitions are in order.

For any mincut C in the st-dag $G=(V,E)$, we identify the two sets: $SN(C)=\{u\in V$: there exists a path from s to u containing no edges of $C\}$ and $TN(C)=\{v\in V$: there exists a path from v to t containing no edges of $C\}$. The mincut C consists exactly of those edges with one endpoint in $SN(C)$ and one endpoint in $TN(C)$. The set of exit vertices associated with C is defined to be $SE(C)=\{u\in SN(C)$: there exists an edge (u,v) with $v\in TN(C)\}$. For any $C\in \mathbf{C}(s,t)$ of G define the event $EC(C)$=[there is an operating path from s to all vertices of $SE(C)$, but not to vertex of $TN(C)$].

The method for computing $\Pr[EP(s,t)]$ adduced in [5] involves computing the probability $\Pr[EC(C)]$ for all $C\in \mathbf{C}(s,t)$ and is based on the two following results:

$$\Pr[EP(s,t)] = 1 - \sum_{C\in \mathbf{C}(s,t)} \Pr[EC(C)] \qquad (1)$$

and:

For any $C\in \mathbf{C}(s,t)$,

$$\Pr[EC(C)] = \prod_{e\in C} p_e \{1 - \sum_{C'\in \mathbf{C}(s,t) \text{ with } SN(C')\subset SN(C)} \Pr[EC(C')] \Big/ \prod_{e\in C'\cap C} p_e\} \qquad (2)$$

where $\prod_{e\in C'\cap C} p_e$ is defined to be 1 if $C'\cap C = 0$.

Hence, all mincuts of G should be revealed and enumerated for computing $\Pr[EP(s,t)]$ by equations (1) and (2). With that end in view, the algorithm for enumerating mincuts of a graph that proposed in [7] can be used. The time complexity of the algorithm is $O((m+n)\mu)$, where m is a number of edges in G and $\mu = |\mathbf{C}(s,t)|$. As shown in [5], the total time complexity of the algorithm for computing $\Pr[EP(s,t)]$, based on equations (1) and (2), is $O((m+n)\mu^2)$.

In order to estimate μ for *FG*, we derive some recursive relations for the set of mincuts in *FG*.

Suppose that all vertices of the certain *FG* are numerated successively by increased order from the source to the target. We identify vertices by their ordinal numbers. We denote *FG* enclosed between a source numbered i and a target numbered j ($i<j$) as $FG(i,j)$. Therefore, $FG(i,j–1)$ is a subgraph of $FG(i,j)$, $FG(i,j–2)$ is a subgraph of $FG(i,j)$ and $FG(i,j–1)$, etc. We define a mincut of $FG(i,j)$ that causes also the system failure of its subgraph $FG(i,j–1)$ as a *strong mincut* of $FG(i,j)$. We define a mincut of $FG(i,j)$ that does not cause the system failure of its subgraph $FG(i,j–1)$ as a *weak mincut* of $FG(i,j)$. We denote a set of all mincuts of $FG(i,j)$ as $\boldsymbol{CF}(i,j)$, a set of all strong mincuts of $FG(i,j)$ as $\boldsymbol{CF}(i,j–1,j)$, and a set of all weak mincuts of $FG(i,j)$ as $\boldsymbol{CF}(i,\overline{j-1},j)$.

The n-vertex *FG* depicted in Fig. 1 is $FG(1,n)$. The source of the initial *FG* is supposed to be numbered 1. We reveal the subgraphs from the *FG* in such a way that all the subgraphs, including the initial *FG*, have the same source. For this reason, the source number may be omitted when denoting sets of mincuts, strong mincuts, and weak mincuts. In such a case, $\boldsymbol{CF}(n)$, $\boldsymbol{CF}(n–1,n)$, and $\boldsymbol{CF}(\overline{n-1},n)$ denote a set of all mincuts, a set of all strong mincuts, and a set of all weak mincuts, respectively, in an n-vertex *FG*.

We continue our denotation in the following way. Let $\boldsymbol{S}$ be a set of sets of edges. In such a case, the set composed by adding an edge (x,y) to each set of edges of $\boldsymbol{S}$ will be denoted $\boldsymbol{S}\times(x,y)$.

It is clear that a set of all mincuts in an n-vertex *FG* can be presented as

$$\boldsymbol{CF}(n) = \boldsymbol{CF}(n–1,n) \cup \boldsymbol{CF}(\overline{n-1},n). \qquad (3)$$

Consider the general case, when $n > 3$. All strong mincuts of $\boldsymbol{CF}(n–2,n–1)$ (and only them!) block the access to vertices $n–2$ and $n–1$, and, thus, block the access to vertices $n–1$ and n. For this reason, they are strong mincuts of $\boldsymbol{CF}(n–1,n)$ also. Weak mincuts of $\boldsymbol{CF}(\overline{n-2},n–1)$ leave the vertex $n–2$ reachable. In such a case, failure of the edge $(n–2,n)$ only can block the access to the vertex n. Therefore, $\boldsymbol{CF}(n–1,n)$ is defined recursively as follows:

$$\boldsymbol{CF}(n–1,n) = \boldsymbol{CF}(n–2,n–1) \cup \boldsymbol{CF}(\overline{n-2},n–1)\times(n–2,n). \qquad (4)$$

Weak mincuts of $\boldsymbol{CF}(\overline{n-1},n)$ block the access to the vertex n but should leave the vertex $n–1$ reachable. For this reason, any weak mincut of $\boldsymbol{CF}(\overline{n-1},n)$ includes the edge $(n–1,n)$. Now, if the vertex $n–2$ is reachable then the failure of the edge $(n–2,n)$ is sufficient to block the access to the vertex n; otherwise, the vertex $n–3$ should be reachable in order to support the access to the vertex $n–1$.

Therefore, $\boldsymbol{CF}(\overline{n-1},n)$ is defined recursively as follows:

$$\boldsymbol{CF}(\overline{n-1},n) = \{(n-2,n),(n-1,n)\} \cup \boldsymbol{CF}(\overline{n-3},n-2)\times(n-1,n). \quad (5)$$

In the special case, for a 3-vertex *FG*

$$\boldsymbol{CF}(2,3) = \{(1,2),(1,3)\} \quad (6)$$

and

$$\boldsymbol{CF}(\overline{2},3) = \{(1,3),(2,3)\}. \quad (7)$$

A 2-vertex *FG* including the single edge (1,2) has no strong mincut. Its single weak mincut is this edge itself:

$$\boldsymbol{CF}(\overline{1},2) = \{(1,2)\}. \quad (8)$$

Hence, (3)-(8) describe relations between mincuts in *FG*.

Lemma 1. $|\boldsymbol{CF}(n-1,n)| = |\boldsymbol{CF}(n-1)|$, $n \geq 3$.

Proof. For $n = 3$, it is clear. If $n > 3$, then

$$|\boldsymbol{CF}(\overline{n-2},n-1)\times(n-2,n)| = |\boldsymbol{CF}(\overline{n-2},n-1)|,$$

and, according to (4) and (3),

$$|\boldsymbol{CF}(n-1,n)| = |\boldsymbol{CF}(n-2,n-1) \cup \boldsymbol{CF}(\overline{n-2},n-1)|$$

$$= |\boldsymbol{CF}(n-1)|. \blacksquare$$

Lemma 2. $\boldsymbol{CF}(\overline{n-1},n) = \left\lfloor \frac{n}{2} \right\rfloor$, $n \geq 2$.

Proof. For $n = 2, 3$, it is clear. If $n > 3$, then

$$|\boldsymbol{CF}(\overline{n-3},n-2)\times(n-1,n)| = |\boldsymbol{CF}(\overline{n-3},n-2)|.$$

Hence, according to (5),

$$|\boldsymbol{CF}(\overline{n-1},n)|$$

$$= |\{(n-2,n),(n-1,n)\} \cup \boldsymbol{CF}(\overline{n-3},n-2)|$$

$$= |\boldsymbol{CF}(\overline{n-3},n-2)| + 1.$$

For odd n, using (7), we have

$$|\boldsymbol{CF}(\overline{n-1},n)| = |\boldsymbol{CF}(\overline{n-3},n-2)| + 1$$

$$= |\boldsymbol{CF}(\overline{n-5},n-4)| + 2$$

$$= |\boldsymbol{CF}(\overline{n-7},n-6)| + 3$$

$$= \dots$$

$$= |\boldsymbol{CF}(\overline{n-(n-2)},n-(n-3))| + \frac{n-3}{2}$$

$$= |\boldsymbol{CF}(\overline{2},3)| + \frac{n-3}{2}$$

$$= 1 + \frac{n-3}{2} = \frac{n-1}{2} = \left\lfloor \frac{n}{2} \right\rfloor.$$

For even n, using (8) we have

$$|\boldsymbol{CF}(\overline{n-1},n)| = |\boldsymbol{CF}(\overline{n-3},n-2)| + 1$$

$$= |\boldsymbol{CF}(\overline{n-5},n-4)| + 2$$

$$= |\boldsymbol{CF}(\overline{n-7},n-6)| + 3$$

$$= \dots$$

$$= |\boldsymbol{CF}(\overline{n-(n-1)},n-(n-2))| + \frac{n-2}{2}$$

$$= |\boldsymbol{CF}(\overline{1},2)| + \frac{n-2}{2}$$

$$= 1 + \frac{n-2}{2} = \frac{n}{2} = \left\lfloor \frac{n}{2} \right\rfloor.$$

The proof is complete. ■

Theorem 3. For $n \geq 2$, the number of mincuts in an n-vertex *FG* is $\left\lfloor \frac{n^2}{4} \right\rfloor$.

Proof. It is clear for n = 2, 3. If $n > 3$, then, as follows from (3) and Lemmas 1 and 2,

$$|\boldsymbol{CF}(n)| = |\boldsymbol{CF}(n-1,n)| + |\boldsymbol{CF}(\overline{n-1},n)|$$

$$= |\boldsymbol{CF}(n-1)| + \left\lfloor \frac{n}{2} \right\rfloor$$

$$= |\boldsymbol{CF}(n-1-1)| + \left\lfloor \frac{n-1}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor$$

$$= |\boldsymbol{CF}(n-2)| + n - 1.$$

For even n

$$|\boldsymbol{CF}(n)| = |\boldsymbol{CF}(n-2)| + n - 1$$

$$= |\boldsymbol{CF}(n-2-2)| + n - 2 - 1 + n - 1$$

$$= |\boldsymbol{CF}(n-4)| + 2n - 4$$

$$= |\boldsymbol{CF}(n-6)| + 3n - 9$$

$$= |\boldsymbol{CF}(n-8)| + 4n - 16$$

$$= \ldots$$

$$= |\boldsymbol{CF}(n-(n-2))| + \frac{n-2}{2}n - \left(\frac{n-2}{2}\right)^2$$

$$= |\boldsymbol{CF}(2)| + \frac{n^2-4}{4}$$

$$= 1 + \frac{n^2}{4} - 1 = \frac{n^2}{4} = \left\lfloor \frac{n^2}{4} \right\rfloor.$$

Using this result, we have for odd n

$$|\boldsymbol{CF}(n)| = |\boldsymbol{CF}(n-1)| + \left\lfloor \frac{n}{2} \right\rfloor$$

$$= \frac{(n-1)^2}{4} + \frac{n-1}{2} = \frac{n^2-1}{4} = \left\lfloor \frac{n^2}{4} \right\rfloor.$$

Therefore, the proof of the theorem is complete. ■

Corollary 4. $|\boldsymbol{CF}(n-1,n)| = \left\lfloor \frac{(n-1)^2}{4} \right\rfloor$, $n \geq 2$.

Proof. For n = 2, it is clear. If $n > 2$, then, according to Lemma 1 and Theorem 3,

$$|\boldsymbol{CF}(n-1,n)| = |\boldsymbol{CF}(n-1)| = \left\lfloor \frac{(n-1)^2}{4} \right\rfloor. \ ■$$

Therefore, the number of mincuts μ in an n-vertex *FG* is estimated as $O(n^2)$.

It can be easily shown that the number of edges in an n-vertex *FG* is $m = 2n - 3$. Hence, the number of edges in *FG* depends linearly on the number of vertices in the graph. For this reason, the time expended enumerating mincuts of *FG* using the algorithm in [7] is $O((m + n)\mu) = O(n^3)$. The total time complexity of the algorithm computing $\Pr[EP(s,t)]$ for an n-vertex *FG*, based on equations (1) and (2), is $O((m + n)\mu^2) = O(n^5)$.

3 Conclusion

The paper presents a method for the solution of the st-connectedness problem in relation to a Fibonacci graph. The method is based on revealing mincuts in this graph and using one algorithm of Provan and Ball [5]. It is proved that the number of mincuts in an n-vertex Fibonacci graph is equal to $\left\lfloor \frac{n^2}{4} \right\rfloor$. It is also shown that the st-connectedness problem for a Fibonacci graph can be solved in $O(n^5)$ time.

References:

[1] M. O. Ball, C. J. Colbourn, and J. S. Provan, Network Reliability, *Network Models*, Handbooks in OR & MS **7**, North-Holland, Amsterdam, 1995, pp. 673-762.

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

[3] M. Ch. Golumbic and Y. Perl, Generalized Fibonacci Maximum Path Graphs, *Discr. Math.* **28**, 1979, pp. 237-245.

[4] J. S. Provan and M. O. Ball, The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected, *SIAM J. Comput.* **12**, 1983, pp. 777-788.

[5] J. S. Provan, and M. O. Ball, Computing Network Reliability in Time Polynomial in the Number of Cuts, *Oper. Res.* **32**, 1984, pp. 516-526.

[6] D. R. Shier, *Network Reliability and Algebraic Structures*, Oxford University Press, Oxford, New York, 1991.
[7] S. Tsukiyama, I. Shirakawa, H. Ozaki, and H. Ariyoshi, An Algorithm to Enumerate All Cutsets of a Graph in Linear Time per Cutset, *J. ACM* **27**, 1980, pp. 619-632.
[8] L. G. Valiant, The Complexity of Enumeration and Reliability Problems, *SIAM J. Comput.* **8**, 1979, pp. 410-421.