Workshop on Algorithms and Computational Molecular Biology

Efficient Algorithms For Distributed Program Reliability Analysis

Ming-Sang Chang[1]
Department of Information management
National Central Police University, Taoyuan, Taiwan, R.O.C.
Email: mschang@sun4.cpu.edu.tw
Mobil: 0932015668

Min-Sheng Lin
Department of Electrical Engineering
National Taipei University of Technology, Taipei, Taiwan, R.O.C.
Email: mslin@ee.ntut.edu.tw

Key Words — Distributed computing systems, distributed program reliability, computational complexity

Abstract — In this paper, we investigated the problem of distributed program reliability in various classes of distributed computing systems. We showed that this problem is computationally intractable for arbitrary distributed computing systems, even when it is restricted to the class of star distributed computing systems. One particular solvable case for star distributed computing systems is identified, in which data files are distributed with respective to a consecutive property, and a polynomial-time algorithm is developed for this case. We also proposed a linear-time algorithm to test whether or not an arbitrary star distributed computing system has this consecutive file distribution property.

[1] All correspondence should be sent to Prof. Ming-Sang Chang.

1. INTRODUCTION

A distributed computing system (DCS), in general, is considered to be one in which the computing functions are distributed among several physically distinct computing elements [4]. These elements or resources (e.g. processing elements, data files, and programs) may be geographically separated or co-located. Thus, each program can run on one or more computers and may frequently access files stored in other sites. Banking systems, travel agency systems, and power control systems are just a few examples of such a distributed computing environment [15]. There are many measures to evaluate the performance of DCS's. Reliability is an important issue [5]. For traditional networks, many reliability indices have been proposed. They include two-terminal reliability, all-terminal reliability, and *K*-terminal reliability [1, 7, 12, 13, 18]. However, these measures are not applicable to practical DCS's since the reliability measure for DCS's should capture the effects of redundant distribution of data files.

Kumar et al. [8, 9] introduce a new reliability measure, namely, distributed program reliability (DPR) to accurately model the reliability of DCS's. The DPR is defined as the probability that a program with distributed files can run successfully in spite of some faults occurring in the communication edges. A model used to represent such situations is a probabilistic graph. A probabilistic graph has a collection of nodes representing the processing elements (sites) which contain some data files and programs, together with a collection of edges representing communication links. Each edge fails independently with known failure probability. As an example, consider a possible DCS of a banking system [8, 15] shown in figure 1. Each local disk stores some of the following information:

- consumer accounts file (CAF),
- automated teller machine accounts file (TAF),
- administrative aids file (ADF), and
- interest and exchange rates file (IXF).

Management report generation (MRG) in computer A indicates a query (program) to be executed for report generation. Figure 2 shows the graph model for this system. A node represents any computer

location and the links show the communication network. We assume that the query (program) MRG requires data files CAF, TAF, ADF and IXF to complete its execution, and it is running at node v_1, which holds data files CAF, ADF and IXF. Hence, it must access data file TAF, which is stored in both nodes v_2 and v_3. Therefore, the DPR of MRG shown in figure 2 can be formulated as:

DPR = Prob[(v_1 and v_2 are connected) or (v_1 and v_3 are connected)].

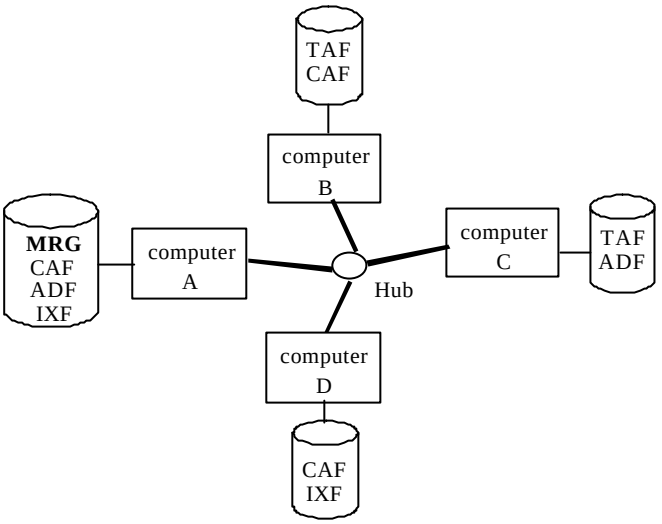

Figure 1: A distributed banking system.

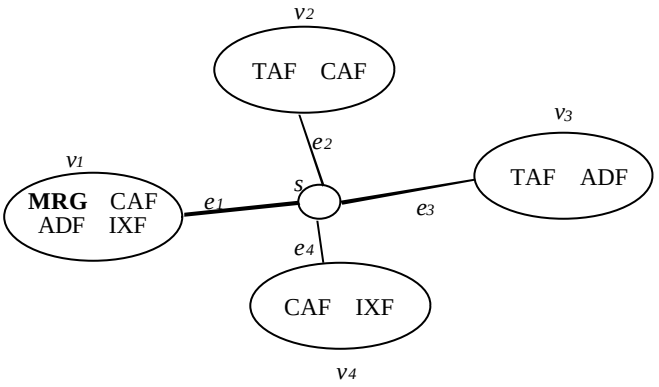

Figure 2: The graph model for the distributed banking system in figure 1.

Most network reliability problems (e.g., *K*-terminal reliability) are *#P*-complete. The class of *#P*-complete problems was introduced by Valiant [16]. The class *#P* contains those problems that involve counting the accepting computations for problems in *NP*; the class of *#P*-complete problems contains the hardest problems in *#P*. As widely recognized, all known exact algorithms for these problems have exponential time complexity, thereby making it unlikely that efficient (polynomial time) algorithms can be

developed for this class of problems. Clearly, computing the DPR for general DCS's is also *#P*-complete. This complexity can be averted by considering only a restricted class of DCS's. Our overall objective will be to examine the boundary of problem classes separating the polynomially solvable cases from the *#P*-complete cases. However, polynomial-time algorithms have been developed for computing the DPR over the DCS's with linear and ring topologies [10].

Classes of interest here include star, 2-tree, serial-parallel, and planar topologies. The results of section 2 show that most of them continue to be *#P*-complete. In section 3, we propose a polynomially solvable case of the DPR problem for star topologies in which data files are restricted to a certain type of distribution. Section 4 shows a linear time algorithm to verify whether or not a star DCS has this restricted class of file distribution.

Assumptions

1. The nodes are perfect.
2. The edges are *s*-independent and either function or fail with known probabilities.

Acronyms & Abbreviations

DCS	distributed computing system
DPR	distributed program reliability
KTR	K-terminal reliability
#EC	number of edge covers
FST	file spanning tree

Notation (General)

G	a general graph (of a network).
D	a DCS graph
E	set of edges
V	set of nodes
e_i	a component of E

v_i a component of V

K subset of V

A_i set of files available at node v_i

p_i probability that edge e_i functions

q_i probability that edge e_i fails; $\equiv 1 - p_i$

f_i data file i

Notation (Star DCS)

D *a star DCS with n + 1 nodes {s, v1, v2, …, vn} and n edges {(s,v1), (s,v2), …,(s, vn)}*

n *number of edges in D*

ei $\circ$ *edge (s, vi); 1* $\leq i \leq n$

m *number of distinct files in D*

t *total number of files in D*

$A\text{-}1j$ *set of indexes of nodes which contain the file fj*

$\boldsymbol{P}$ $\circ$ *[* $\pi(1), \pi(2), \ldots, \pi(n)$ *] a permutation of numbers {1, 2, …, n} such that if file fd* $\in A_{\pi(i)}$ *and fd* $\in A_{\pi(j)}$*, then fd* $\in A_{\pi(k)}$ *for all k, i < k < j*

$\boldsymbol{F}$ *ordered set of all minimal file cutsets according to their minimal components*

r *number of minimal file cutsets in* $\boldsymbol{F}$

Ci *the ith minimal file cutset in* $\boldsymbol{F}$*;* $1 \leq i \leq r$

αi $\circ$ $\min\{k \mid e_{\pi(k)} \in Ci\}$ *, i.e. , the index of the minimal component in Ci;* $1 \leq i \leq r$

βi $\circ$ $\max\{k \mid e_{\pi(k)} \in Ci\}$ *, i.e. , the index of the maximal component in Ci;* $1 \leq i \leq r$

$H(i, j)$ $\circ\{e_{\pi(i)}, e_{\pi(i+1)}, \ldots, e_{\pi(j)}\}$; $1 \leq i \leq j \leq n$ *(note that Ci* $\circ H(\alpha i, \beta i)$ *)*

$X(i, j)$ *event: all edges in H(i, j) fail*

Wi $\circ \prod_{j=1}^{i} X(\alpha_j, \beta_j)$ *(note that the DPR of D can be expressed as 1 – Pr(Wr))*

Fi *event: the star DCS D' fails in which it consists of i + 1 nodes s,* $v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(i)}$ *and i edges* $e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(i)}$

$\overline{W}$ *complement of event W*

Definitions (Star DCS)

- ***Consecutive file distribution:*** A star DCS D has the consecutive file distribution property iff its nodes can be linearly ordered such that, for each distinct file f_d, the nodes containing file f_d occur consecutively. More formally, a star DCS D has the consecutive file distribution property iff there exists a permutation $\Pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ of numbers $\{1, 2, \ldots, n\}$ such that if file $f_d \in A_{p(i)}$ and $f_d \in A_{p(j)}$, then $f_d \in A_{p(k)}$ for all k, $\pi(i) < \pi(k) < \pi(j)$.
- ***File cutest:*** A set C_d of edges of D is referred to as a file cutset for file f_d if it consists of all edges (s, v_i) such that node v_i contains file f_d, i.e., $C_d = \{(s, v_i) \mid f_d \in A_i\}$.
- ***Minimal file cutest:*** *A file cutset C is referred to as minimal if there is no other file cutset C' such that $C' \subset C$. Without loss of generality, we reorder the minimal file cutsets, if necessary, by their minimal component, i.e., for two distinct minimal file cutsets Ci and Cj, $i < j$ iff $\min\{k \mid (s, v_{p(k)}) \in Ci\} < \min\{k \mid (s, v_{p(k)}) \in Cj\}$.*

2. THE COMPUTATIONAL COMPLEXITY OF THE DPR PROBLEM

In this section we assume that the reader is familiar with the basic notions of *NP*-completeness. We refer the reader to [6] for an excellent exposition of the theory of *NP*-completeness. First we state some known *#P-complete* problems.

- K-Terminal Reliability (KTR) [12]

  Input: an undirected graph $G = (V, E)$ where V is the set of nodes and E is the set of edges that fail s-independently of each other with known probabilities. A set $K \subseteq V$ is distinguished with $|K| \geq 2$.

  Output: $R(G_K)$, the probability that the set K of nodes of G is connected in G.

- Number of Edge Covers (#EC) [2]

  Input: an undirected graph $G = (V, E)$.

  Output: the number of edge covers for G

  $\equiv |\{EC \subseteq E$: each node of G is an end of some edge in $EC\}|$.

Theorem 1. The KTR problem is polynomially reducible to the DPR problem.

Proof. Let $G = (V, E)$ be a network graph with a subset of nodes $K \subseteq V$. Construct a DCS graph $D = (V, E)$ from G such that node v_i of D contains file f_i iff node $v_i \in K$ in G. Clearly, all distinct files in D are interconnected iff all nodes of K are connected in G. In addition, let each edge of D have the same operational probability as the corresponding edge of G. Then, the DPR of D is equal to the KTR of G. In this case the DCS D can be obtained from G in polynomial time. ***Q.E.D.***

By Theorem 1, if we have a polynomial-time algorithm for computing the DPR of D, then we can obtain a polynomial-time algorithm for computing the KTR of G using this construction. However, Rosenthal [12] showed that the problem of computing the KTR in general is *#P*-complete, so computing the DPR in general is also *#P*-complete. Therefore, we have the following corollary.

Corollary 1. Computing the DPR for a general DCS is *#P*-complete.

Corollary 2. Computing the DPR for a planar DCS is *#P*-complete.

*Proof***.** From the proof of Theorem 1, it is clear that the KTR problem is just a special case of the DPR problem. It has been shown that computing the KTR over a planar network is *#P*-complete [11]. This also immediately implies that computing the DPR over a planar DCS is still *#P*-complete. ***Q.E.D.***

For the KTR problem, polynomial-time (or linear-time) algorithms have been developed for other restricted networks, such as a star network, a 2-tree network, and a series-parallel network [14]. If there are no replicated files in DCS's, i.e., if there is only one copy of each file in DCS's, the DPR problem can be transformed into the equivalent KTR problem in which the K set is the set of nodes that contain the data files needed for the program under consideration. However, data files are usually replicated and distributed in DCS's, so these two problems are different. In the remainder of this section, we will see that computing the DPR over a star DCS, a tree DCS, or a series-parallel DCS in general is still *#P*-complete.

Theorem 2. The #EC problem is polynomially reducible to the DPR problem over a star DCS.

Proof. Given a graph G with n edges $e_1, e_2, \ldots, e_n$ and nodes $v_1, v_2, \ldots$, we shall construct a star DCS D such that the number of edge covers in G can be expressed as a function of the DPR of D. Construct a star

DCS $D = (V', E')$ where $V' = \{s, v'_1, v'_2, \ldots, v'_n\}$, $E' = \{(s, v'_1), (s, v'_2), \ldots, (s, v'_n)\}$ and node v'_i contains files f_g and f_h iff $e_i = (v_g, v_h)$ in G. We now consider a *file spanning tree* (FST) T, which is a subgraph of D and its nodes hold all the needed data files, i.e.,

$$\bigcup_{v'_i \in T} \{f_j | v'_i \text{ contains file } f_j\} = \bigcup_{v'_i \in V'} \{f_j | v'_i \text{ contains file } f_j\}.$$

It is easy to see that there is a one-to-one correspondence between one of the sets of edge covers in G and one FST in D. The DPR of D can be expressed as

$$\text{DPR} = \sum_{\substack{\text{for all} \\ \text{FST } T \text{ in } D}} \left(\prod_{(v_0, v'_i) \in E' - T'} (1 - p_i) \prod_{(v_0, v'_i) \in T} p_i \right)$$

where p_i is reliability of edge (s, v'_i) of D, $1 \le i \le n$. If we set each $p_i = \frac{1}{2}$ for all $1 \le i \le n$, then we have

$$\text{DPR} = \sum_{\substack{\text{for all FST} \\ T \text{ in } D}} \left(\frac{1}{2}\right)^n, \text{ or}$$

$$\text{DPR} \cdot 2^n = \sum_{\substack{\text{for all FST} \\ T \text{ in } D}} 1$$

$$= \#\text{ of FST's in } D$$

$$= \#\text{ of edge covers in } G$$

Since D can be constructed from G in polynomial time, the number of edge covers in G can be solved in polynomial time if we have a polynomial-time algorithm for computing the DPR of D.

Q.E.D.

Corollary 3. Computing the DPR for a star DCS is *#P*-complete.

Proof. Follows from Theorem 2 and the fact that #EC have been shown to be *#P*-complete [2].

Q.E.D.

Now we shall show that computing the DPR remains difficult for a 2-tree topology. A *2-tree* is defined recursively as follows:

- The complete graph K2 (a single edge) is a 2-tree.
- Given any 2-tree G on $n \geq 2$ nodes, let (v_i, v_j) be an edge of G. Adding a new node v_k and two edges (v_k, v_i) and (v_k, v_j) produces a 2-tree on $n + 1$ nodes.

Corollary 4. Computing the DPR for a 2-tree DCS in general is *#P*-complete.

Proof. Let D be a star DCS with $n + 1$ nodes $s, v_1, v_2, ..., v_n$ and n edges $(s, v_1), (s, v_2), \ldots, (s, v_n)$. We shall construct from D a 2-tree DCS D' such that D and D' have the same DPR. Embed the 2-tree DCS D' into the star DCS D by adding some virtual edges (v_i, v_{i+1}), $1 \leq i \leq n$-1. Now, it is easy to see that D' is a 2-tree DCS on $n + 1$ nodes. If we stipulate that each virtual edge has operational probability 0, the DPR of D is reduced to the DPR of D'. By corollary 3, since computing the DPR over a star topology in general is *#P*-complete, computing the DPR over a 2-tree topology is also *#P*-complete. ***Q.E.D.***

Corollary 5. Computing the DPR over a series-parallel DCS is *#P*-complete.

Proof. From [17], a 2-tree graph is a maximal series-parallel graph. A maximal series-parallel graph is a series-parallel graph with neither loops nor parallel edges. Since computing the DPR over a 2-tree topology is *#P*-complete, computing the DPR over a series-parallel DCS is also *#P*-complete. ***Q.E.D.***

3. A POLYNOMIAL-TIME ALGORITHM FOR COMPUTING THE DPR OF STAR DCS's

The results of the previous section indicate that computing the DPR over a star DCS is *#P*-complete. These results imply that polynomial algorithms unlikely exist for solving them. However, an efficient algorithm possibly exists for computing the DPR over a star DCS with a certain restricted class of file distribution. In this section we present a polynomial-time algorithm for computing the DPR of a star DCS with a consecutive file distribution.

Let D be a star DCS with the consecutive file distribution property. Then, the minimal file cutsets can be ordered by their minimal component, i.e. for two distinct minimal file cutsets C_i and C_j, $i < j$ iff $\min\{k \mid (s, v_{p(k)}) \in C_i\} < \min\{k \mid (s, v_{p(k)}) \in C_j\}$. By definition, D fails iff at least one event $X(\boldsymbol{a}_i, \boldsymbol{b}_i)$, $1 \leq i \leq r$,

occurs, where α_i and β_i are the indexes of the minimal and maximal components in C_i, respectively. Clearly, if $r = 1$, the unreliability of D can be obtained as $\Pr[W_1] = \Pr[X(\alpha_1, \beta_1)]$. Next consider the case with $r \geq 2$. The unreliability of D with the first i's file cutsets is

$$\Pr[W_i] = \Pr[W_{i-1} \cup X(\alpha_i, \beta_i)]$$

This expression can be decomposed using conditional probability as

$$\Pr[W_i] = \Pr[W_{i-1}] + \Pr[\overline{W_{i-1}} \cap X(a_i, \beta_i)]. \quad (1)$$

Consider the event $\overline{W_{i-1}} \cap X(\alpha_i, \beta_i)$, which implies

- E_1: For each k, $1 \leq k \leq i-1$, at least one edge $e \in H(\alpha_k, \beta_k)$ ° C_k functions and
- E_2: All edges $\in H(\alpha_i, \beta_i)$ ° C_i fail.

By event E_2, event E_1 can be rewritten as

- E_1': For each k, $1 \leq k \leq i-1$, at least one edge $e \in \{H(\alpha_k, \beta_k) - H(\alpha_i, \beta_i)\}$ functions.

A fundamental difficulty in calculating $\Pr(E_1')$ is that events in E_1' are not, in general, disjoint. However, we can define events S_j's that are disjoint by

S_j = { E_1' occurs and edge $e_{p(j)}$ is the last good one}, for $\alpha_{i-1} \leq j \leq \alpha_i - 1$.

Thus, $E_1' \cap E_2 = \bigcup_{j=\alpha_{i-1}}^{\alpha_i - 1} (S_j \cap E_2)$. and

$$\Pr[\overline{W_{i-1}} \cap X(a_i, \beta_i)] = \Pr[\bigcup_{j=\alpha_{i-1}}^{\alpha_i - 1} (S_j \cap E_2)] \quad (2)$$

Since S_j's are disjoint events, we have

$$\Pr[\bigcup_{j=\alpha_{i-1}}^{\alpha_i - 1} (S_j \cap E_2)] = \sum_{j=\alpha_{i-1}}^{\alpha_i - 1} \Pr(S_j \cap E_2) \quad (3)$$

The event S_j 3 E_2, $\alpha_{i-1} \le j \le \alpha_i - 1$, can be decomposed into three independent events:{no file cutset fails between edges $e_{p(1)}$ and $e_{p(j-1)}$ }, {edge $e_{p(j)}$ functions}, and {all edges between $e_{p(j+1)}$ and $e_{p(\beta_i)}$ fail}. So

$$\Pr(S_j \text{ I } E_2) = [1 - \Pr(F_{j-1})] \cdot p_{p(j)} \cdot \Pr[X(j+1, \beta_i)]. \tag{4}$$

Therefore, according to Eqs. (1), (2), (3), and (4) , we have

$$\Pr(W_i) = \Pr(W_{i-1}) + \sum_{j=\alpha_{i-1}}^{\alpha_i - 1} \left\{ [1 - \Pr(F_{j-1})] \cdot p_{p(j)} \cdot \Pr[X(j+1, \beta_i)] \right\}$$

The following theorem can now be established.

Theorem 3. For $2 \le i \le r$:

$$\Pr(W_i) = \Pr(W_{i-1}) + \sum_{j=\alpha_{i-1}}^{\alpha_i - 1} \left\{ [1 - \Pr(F_{j-1})] \cdot p_{p(j)} \cdot \Pr[X(j+1, \beta_i)] \right\}, \tag{5}$$

with the boundary conditions: $\Pr(W_1) = \Pr[X(\alpha_1, \beta_1)]$, and $\Pr(F_k) = 0$ for $0 \le k < \beta_1$.

Before applying Theorem 3, initially compute the values of $\Pr[X(j+1, \beta_i)]$ and $\Pr(F_{j-1})$ for $2 \le i \le r$ and $\alpha_{i-1} \le j \le \alpha_i - 1$. By noting that $\alpha_g < \alpha_h$ whenever $g < h$, the recursive formula can be obtained as follows.

$$\Pr[X(j+1, \beta_i)] = \begin{cases} \dfrac{1}{q_{p(\alpha_{i-1})}} \cdot \Pr[X(\alpha_{i-1}, \beta_{i-1})] \cdot \displaystyle\prod_{k=\beta_{i-1}+1}^{\beta_i} q_{p(k)} & \text{for } j = \alpha_{i-1} \\ \dfrac{1}{q_{p(j)}} \cdot \Pr[X(j, \beta_i)] & \text{for } \alpha_{i-1} < j \le \alpha_i - 1 \end{cases} \tag{6}$$

By starting with $\Pr[X(\alpha_1, \beta_1)] = \displaystyle\prod_{k=\alpha_1}^{\beta_1} q_{p(k)}$, we successively determine that

$$\Pr[X(\alpha_1+1,\beta_2)],\ \Pr[X(\alpha_1+2,\beta_2)],\ \ldots,\ \Pr[X(\alpha_2,\beta_2)],$$

$$\Pr[X(\alpha_2+1,\beta_3)],\ \Pr[X(\alpha_2+3,\beta_3)],\ \ldots,\ \Pr[X(\alpha_3,\beta_3)],$$

$$\ldots$$

$$\Pr[X(\alpha_{r-1}+1,\beta_r)],\ \Pr[X(\alpha_{r-1}+2,\beta_r)],\ \ldots,\ \text{and}\ \Pr[X(\alpha_r,\beta_r)].$$

To obtain the values of $\Pr(F_{j-1})$ in Theorem 3, by definition, we have that

$$\Pr(F_k)=\begin{cases}\Pr(W_{i-1}) & \text{for } \beta_{i-1}\le k\le \beta_i-1\\ 0 & \text{for } k\le \beta_1-1\end{cases} \qquad (7)$$

Hence, while computing $\Pr(W_i)$ by Theorem 3, we can also obtain $\Pr(F_k)$, for $\beta_{i-1} \le k \le \beta_i-1$.

3.1 A Polynomial-Time Algorithm

The major algorithm-related strategies to compute the DPR of star DCS's are outlined. Assume a given star DCS D and the file distributions A_i for each node. By assuming that D has the property of consecutive file distribution, let Π be a permutation of numbers $\{1, 2, \ldots, n\}$ such that if file $f_d \in A_{p(i)}$ and $f_d \in A_{p(j)}$, then $f_d \in A_{p(k)}$ for all k, $i < k < j$. All file cutsets can be enumerated from A_i in the following manner: if node v_i contains file f_d, then file cutset C_d contains edge e_i. Subsequently, α_i and β_i values of C_i can be determined from the permutation Π such that $\alpha_i = \min\{k|\ e_{p(k)} \in C_i\ \}$ and $\beta_i = \max\{k|\ e_{p(k)} \in C_i\ \}$. The next step removes the file cutsets which are not minimal and rearranges the remaining minimal file cutsets according to their α_i and β_i values. Finally Theorem 3, Eqs. (6) and (7) are used to compute the DPR (= $1 - \Pr[W_r]$). The algorithm is formally described below.

Algorithm *REL*

Input: A star DCS D with n +1 nodes $\{ s, v_1, v_2, \ldots, v_n\}$ and n edges $\{(s,v_1), (s,v_2), \ldots,(s,v_n)\}$.

A permutation $\Pi = [\pi(1),\ \pi(2), \ldots,\ \pi(n)]$ of numbers $\{1, 2, \ldots, n\}$ such that if file $f_d \in A_{p(i)}$, $f_d \in A_{p(j)}$, then $f_d \in A_{p(k)}$ for all k, $i < k < j$, where A_i represents the set of files available at node v_i.

Output : the DPR of D

begin

Step 1: // find all file cutsets //

for $i \leftarrow 1$ **to** m **do** $C_i \leftarrow \varnothing$; // initialization step; m is the number of distinct files //

for $i \leftarrow 1$ **to** n **do**

for each $f_d \in A_i$ **do** $C_d \leftarrow C_d \cup \{e_i\}$; // For convenience, let e_i denote edge (s, v_i) //

Step 2: // set the values of α_i and β_i for $1 \le i \le m$ //

for $i \leftarrow 1$ **to** m **do**

begin

$\alpha_i \leftarrow \min\{k \mid e_{p(k)} \in C_i\}$;

$\beta_i \leftarrow \max\{k \mid e_{p(k)} \in C_i\}$;

end

Step 3: // find all minimal file cutset //

$\Phi \leftarrow \varnothing$;

for $i \leftarrow 1$ **to** m do $\Phi \cup \{C_i\}$;

for $1 \le i, j \le m$ **do**

if ($\alpha_i \ge \alpha_j$ and $\beta_i \le \beta_j$) **then** remove C_j from Φ; // which implies $C_i \subseteq C_j$ //

Step 4: reorder the minimal file cutsets in Φ for two distinct minimal file cutsets C_i and C_j, $i < j$ iff $\alpha_i < \alpha_j$;

Step 5: // compute $\Pr[X(j+1, \beta_i)]$, for $2 \le i \le r$ and $\alpha_{i-1} \le j \le \alpha_i - 1$, by Eq. (6) //

$$\Pr[X(\alpha_1, \beta_1)] \leftarrow \prod_{k=\alpha_1}^{\beta_1} q_{p(k)};$$

for $i \leftarrow 2$ **to** r **do** // r is the number of minimal file cutsets in Φ//

begin

$$\Pr[X(\alpha_{i-1}+1, \beta_i)] \leftarrow \frac{1}{q_{p(\alpha_{i-1})}} \cdot \Pr[X(\alpha_{i-1}, \beta_{i-1})] \cdot \prod_{k=\beta_{i-1}+1}^{\beta_i} q_{p(k)};$$

for $j \leftarrow \alpha_{i-1}+2$ **to** $\alpha_i - 1$ **do** $\Pr[X(j+1, \beta_i)] \leftarrow \frac{1}{q_{\pi(j)}} \cdot \Pr[X(j, \beta_i)]$;

end

Step 6: // Apply Theorem 3 and Eq. (7) to compute $\Pr(W_i)$ and $\Pr(F_j)$ //

$\Pr(W_1) \leftarrow \Pr[X(\alpha_1, \beta_1)]$; // boundary condition //

for $k \leftarrow 0$ **to** $\beta_1 - 1$ **do** $\Pr(F_k) \leftarrow 0$; // boundary condition //

for $i \leftarrow 2$ **to** r **do**

begin

for $k \leftarrow \beta_{i-1}$ **to** $\beta_i - 1$ **do** $\Pr(F_k) \leftarrow \Pr(W_{i-1})$;

$$\Pr(W_i) \leftarrow \Pr(W_{i-1}) + \sum_{j=\alpha_{i-1}}^{\alpha_i - 1} \left\{ [1 - \Pr(F_{j-1})] \cdot p_{\pi(j)} \cdot \Pr[X(j+1, \beta_i)] \right\};$$

end

Step 7: $DPR \leftarrow 1 - \Pr(W_r)$; **Output**(*DPR*);

end ***REL***

3.2 Complexity Analysis

The time complexity of Algorithm ***REL*** is analyzed as follows. Step 1 performs $O(m + \sum_{i=1}^{n} |A_{\pi(i)}|) = O(m+t) = O(t)$ time (since $m < t$) to identify all file cutsets, where t denotes the total number of files in D. Step 2 requires $O(2 \cdot \sum_{i=1}^{m} |C_i|) \approx O(t)$ time to set α_i and β_i, $1 \le i \le m$ and step 3 takes $O(m^2)$ time to obtain all minimal file cutsets. Step 4 requires the reordering of all minimal file cutsets in a nondecreasing order of their index of the minimal component. This ordering can be executed in $O(r \cdot \log r)$ using an efficient sorting algorithm, where r denotes the number of minimal file cutsets. In step 5, evaluating $\Pr[X(j+1, \beta_i)]$ by making use of Eq. (6) requires that

$$\begin{cases} O\{\sum_{i=2}^{r}[(\beta_i - \beta_{i-1}) + 2]\} = O(\beta_r - \beta_1 + r) \approx O(n+r), & \text{for } j = \alpha_{i-1} \\ O\{\sum_{i=2}^{r}(1)\} = O(r-1) = O(r), & \text{for } \alpha_{i-1} \le j \le \alpha_i - 1 \end{cases}$$

Hence, the total time to evaluate all Pr[$X(j+1, \beta_i)$] is therefore $O(n + r)$. In step 6, computing all Pr(F_k) takes $O[\sum_{i=2}^{r}(\beta_i - \beta_{i-1})] = O(\beta_r - \beta_1) \approx O(n)$ time and computing all Pr(W_i) takes $O\{\sum_{i=2}^{r}[1 + (\alpha_i - \alpha_{i-1}) \cdot 3] = O[1 + 3 \cdot (\alpha_r - \alpha_1)] \approx O(n)$ time. Therefore, the total time in step 6 is $O(n)$. Clearly, step 7 performs in constant time. Finally, the entire algorithm has time complexity $O[t + t + m^2 + r \times log\ r + (n + r) + n]$. Since $t \le m \times n$, and $r \le n$, the complexity of Algorithm ***REL*** can be obtained as $O(m^2 + m \times n)$.

3.3 An Example of Application of Algorithm REL

To illustrate Algorithm ***REL*** as stated above, consider the star DCS in figure 3 in which there is a consecutive file distribution property and the associative permutation Π = [3, 6, 4, 2, 5, 1, 7]. (In Section 4, we will show how to identify the associative permutation when the star DCS has the consecutive file distribution property .) The overall procedure is as follows:

Step 1: The file cutsets are found to be

$$C_1=\{e_2, e_5\},\ C_2=\{e_1, e_5, e_7\},\ C_3=\{e_1, e_2, e_5\},\ C_4=\{e_3, e_6\},\ C_5=\{e_2, e_4, e_5\}.$$

Step 2: According to the permutation

$$\pi(1) = 3,\ \pi(2) = 6,\ \pi(3) = 4,\ \pi(4) = 2,\ \pi(5) = 5,\ \pi(6) = 1,\ \pi(7) = 7$$

and the results of Step 1, we have

$$\alpha_1 = 4,\ \beta_1 = 5,\ \alpha_2 = 5,\ \beta_2 = 7,\ \alpha_3 = 4,\ \beta_3 = 6,\ \alpha_4 = 1,\ \beta_4 = 2,\ \alpha_5 = 3,\ \beta_5 = 5.$$

Step 3: Since $C_1 \subset C_3$ and $C_1 \subset C_5$, remove C_3 and C_5. Thus, the set of minimal file cutsets is

$$\Phi = \{ C_1, C_2, C_4\}.$$

Step 4: Reorder the minimal file cutsets in such a manner that for C_i and C_j, $i < j$ iff $\boldsymbol{a}_i < \boldsymbol{a}_j$, and we obtain

$$C_1=\{e_3, e_6\},\ \boldsymbol{a}_1 = 1,\ \boldsymbol{b}_1 = 2,$$

$$C_2=\{e_2, e_5\},\ \boldsymbol{a}_2 = 4,\ \boldsymbol{b}_2 = 5,$$

$$C_3=\{e_1, e_5, e_7\},\ \boldsymbol{a}_3 = 5,\ \boldsymbol{b}_3 = 7.$$

Step 5: By using Eq. (6), we have

$$\Pr[X(1,2)] = q_3q_6,\ \Pr[X(2,5)] = q_6q_4q_2q_5,\ \Pr[X(3,5)] = q_4q_2q_5,$$

$$\Pr[X(4,5)] = q_2q_5, \text{ and } \Pr[X(5,7)] = q_5q_1q_7.$$

Step 6: We use Theorem 3 and Eq. (7) to compute $\Pr(W_i)$ and $\Pr(F_k)$ for $2 \le i \le 3$ and $\boldsymbol{b}_{i-1} \le k \le \boldsymbol{b}_i - 1$, and obtain

$$\Pr(W_1) = q_3q_6; \quad \Pr(F_0) = \Pr(F_1) = 0 \qquad \text{(boundary condition)}$$

i =2: $\Pr(F_2) = \Pr(F_3) = \Pr(F_4) = \Pr(W_1) = q_3q_6,$

$$\Pr(W_2) = \Pr(W_1) + [1-\Pr(F_0)] \cdot p_3 \cdot \Pr[X(2,5)] \qquad (j = 2)$$

$$+ [1-\Pr(F_1)] \cdot p_6 \cdot \Pr[X(3,5)] \qquad (j = 3)$$

$$+ [1-\Pr(F_2)] \cdot p_4 \cdot \Pr[X(4,5)] \qquad (j = 4)$$

$$= q_3q_6 + p_3q_6q_4q_2q_5 + p_6q_4q_2q_5 + (1-q_3q_6) \cdot p_4q_2q_5$$

i =3: $\Pr(F_5) = \Pr(W_2)$

$$\Pr(W_3) = \Pr(W_2) + [1-\Pr(F_3)] \cdot p_2 \cdot \Pr[X(5,7)] \qquad (j = 5)$$

$$= q_3q_6 + p_3q_6q_4q_2q_5 + p_6q_4q_2q_5 + (1-q_3q_6) \cdot p_4q_2q_5 + (1-q_3q_6) \cdot p_2q_5q_1q_7$$

Step 7: Therefore, DPR is

$$\text{DPR} = 1 - \Pr(W_3)$$

$$= 1 - \{q_3q_6 + p_3q_6q_4q_2q_5 + p_6q_4q_2q_5 + (1-q_3q_6) \cdot p_4q_2q_5 + (1-q_3q_6) \cdot p_2q_5q_1q_7\}$$

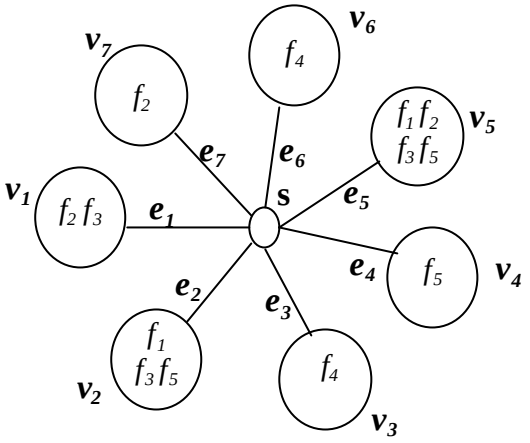

Figure 3 : A star DCS with the consecutive file distribution property

4. A LINEAR TIME ALGORITHM OF TESTING FOR THE CONSECUTIVE FILE DISTRIBUTION PROPERTY IN A STAR DCS

The previous section has presented a polynomial-time algorithm for computing the DPR of a star DCS when it has the consecutive file distribution property. In this section, we are interested in testing whether or not a star DCS has the consecutive file distribution property. The problem statement would be:

Input: A star DCS D with $n + 1$ nodes s , v_1, v_2, …, v_n and file distributions A_i, $1 \le i \le n$.

Output: A permutation $\Pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ of numbers $\{1, 2, \ldots, n\}$ such that if file $f_d \in A_{p(i)}$ and $f_d \in A_{p(j)}$, then $f_d \in A_{p(k)}$ for all k, $\pi(i) < \pi(k) < \pi(j)$.

Note that a solution does not always exist. To facilitate our search for the correct ordering of Π, we use a data structure of a *PQ*-tree proposed by Booth and Leuker [3]. A *PQ*-tree is a rooted tree that has nodes of two varieties: *P*-nodes and *Q*-nodes. A *P*-node is a node whose children can be arbitrarily permuted. A *Q*-node is a node whose children are ordered or reverse ordered. The frontier of a *PQ*-tree is the permutation of leaves from left to right. Two *PQ*-trees are equivalent iff one can be transformed into the other by applying a sequence of the following transformation rules.

- arbitrarily permute the children of a *P*-node
- reverse the children of a *Q*-node

Using a *PQ*-tree data structure, we have the following algorithm.

Algorithm *Check_Consecutive_File_Distribution*

Input : A star DCS D with $n + 1$ nodes $s, v_1, v_2, \ldots, v_n$, n edges $e_1, e_2, \ldots, e_n$, where $e_i = (s, v_i)$ for $1 \le i \le n$, and file-available set $A_i = \{f_j \mid \text{for each } f_j \text{ stored in node } v_i\}$ for $1 \le i \le n$.

Output : A permutation $\Pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ of numbers$\{1, 2, \ldots, n\}$such that if file $f_d \in A_{p(i)}$ and $f_d \in A_{p(j)}$, then $f_d \in A_{p(k)}$ for all k, $i < k < j$.

begin

$T \leftarrow$ universal tree; // a single *P*-node connected to all the leaf nodes of $\{1, 2, \ldots, n\}$ //

for $j \leftarrow 1$ to m **do** $A^{-1}_j \leftarrow \varnothing$; // m denotes the number of distinct files in D //

// A^{-1}_j is the set of indexes of nodes that contain the file f_j //

for $i \leftarrow 1$ to n **do**

for each $f_j \in A_i$ **do** $A^{-1}_j \leftarrow \{i\}$;

for $j \leftarrow 1$ **to** m **do** $T \leftarrow REDUCE(T, A^{-1}_j)$;

if T is a null tree

then

print out "*D* has no consecutive file distribution property" ;

else

print out the frontier of T ;

end *Check_Consecutive_File_Distribution*

The routine *REDUCE* attempts to apply a set of eleven templates. Each template consists of a pattern to be matched against the current *PQ*-tree and the set A^{-1}_j and a replacement to be substituted for the pattern. The templates are applied from the bottom to the top of the tree. The null tree may be returned when no template applies. For details of the algorithm, the reader is directed to Booth and Leuker [3].

Complexity Analysis:

For A^{-1}_j, $1 \le j \le m$, it can be obtained in $O(m+\sum_{i=1}^{n}|A_i|)$ steps. According to [3], the loop of the *REDUCE* routine can be computed in $O(m+n+\sum_{j=1}^{m}|A^{-1}{}_j|)$ steps. Further, $\sum_{i=1}^{n}|A_i| = \sum_{j=1}^{m}|A^{-1}{}_j| = t$ (the total number of files in D). Therefore, the time complexity for the above algorithm is $O(m + t) + O(m + n + t) = O(m + n + t)$.

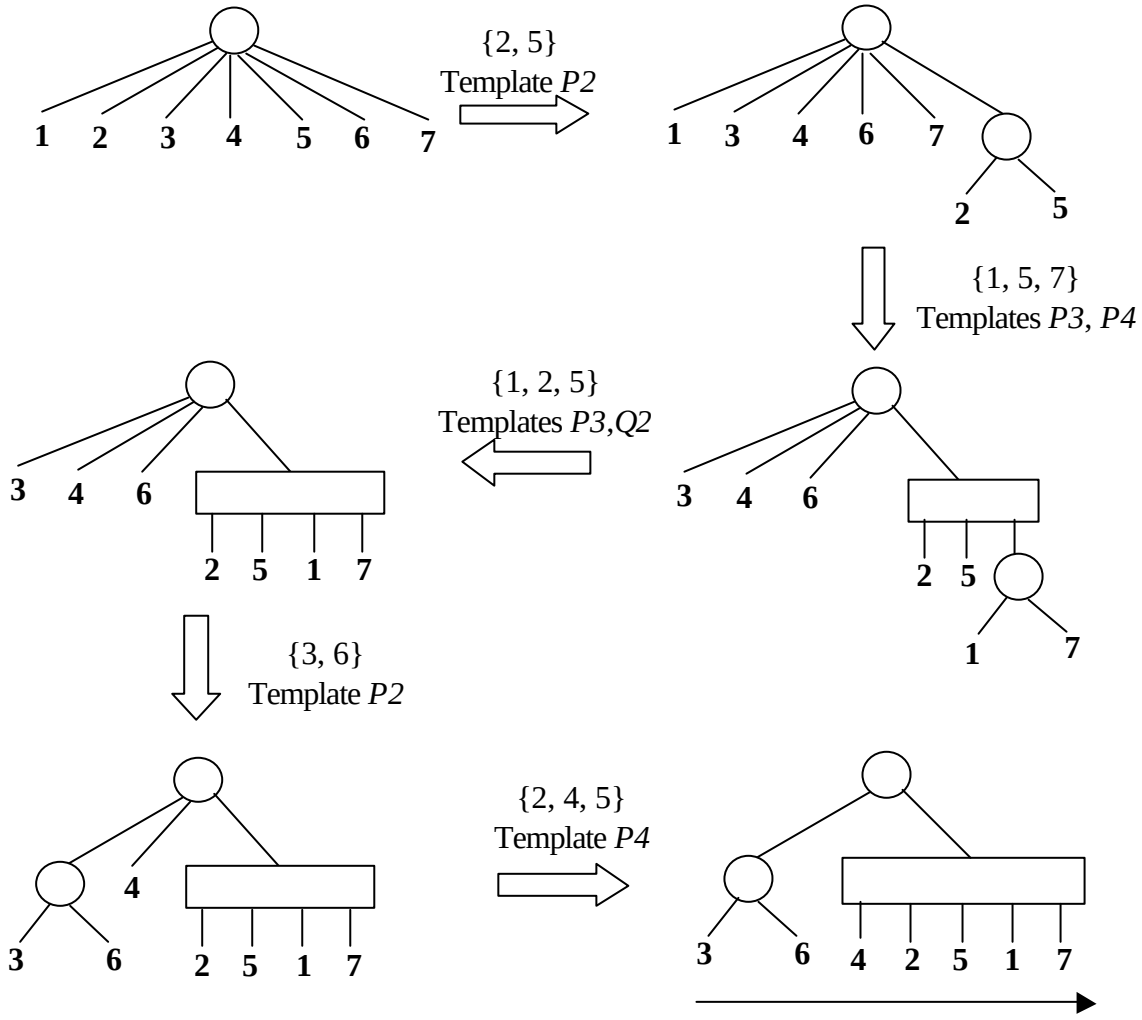

Figure 4. The reduction steps by using a PQ-tree

An Illustrative Example:

Consider the star DCS D shown in figure 3. Applying the above algorithm leads to

$$A^{-1}_1 = \{2, 5\}, A^{-1}_2 = \{1, 5, 7\}, A^{-1}_3 = \{1, 2, 5\}, A^{-1}_4 = \{3, 6\}, A^{-1}_5 = \{2, 4, 5\}$$

Figure 4 displays the reduction steps. In an illustration of a *PQ*-tree, a *P*-node is drawn as a circle and a *Q*-node as a rectangle. From this figure, we can conclude that the star DCS *D* of figure 3 has the consecutive file distribution property and one of the associative permutations is:

$$\Pi = [3, 6, 4, 2, 5, 1, 7]$$

REFERENCES

[1] K.K. Agrawal, S. Rai, "Reliability evaluation in computer-communication networks", *IEEE Trans. Reliability*, vol R-30, 1981 Apr, pp 32-35.

[2] M.O. Ball, J.S. Provan, D.R. Shier, "Reliability covering problems", *Networks*, vol 21, 1991, pp 345-357.

[3] K.S. Booth, G.S. Leuker, "Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms", *J. Comput. Syst. Sci*., vol 13, 1976, pp 335-379.

[4] P. Enslow, "What is a distributed data processing system", *Computer*, vol 11, 1978 Jan.

[5] J. Garcia-Molina, "Reliability issues for fully replicated distributed database", *IEEE Trans. Computer*, vol 16, 1982 Sep, pp 34-42.

[6] M.R. Garey, D.S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[7] A.P. Grnarov, M. Gerla, "Multi-terminal reliability analysis of distributed processing system", *Proc. 1981 Int. Conf. Parallel Processing*, 1981 Aug, pp 79-86.

[8] A. Kumar, S. Rai, D.P. Agrawal, "On computer communication network reliability under program execution constraints", *IEEE JSAC*, vol 6, 1988 Oct, pp 1393-1399.

[9] V.K.P. Kumar, S. Hariri, C.S. Raghavendra, "Distributed program reliability analysis", *IEEE Trans. Software Eng.*, vol SE-12, 1986 Jan, pp 42-50.

[10] M.S. Lin, D.J. Chen, "Two polynomial-time algorithms for computing reliability in a linear and a circular distributed system", PDPTA'97, 1997 Jun, Las Vegas, Nevada, USA.

[11] J.S. Provan, "The complexity of reliability computations in planar and acyclic graphs", *SIAM Journal on Computing*, vol 15, 1986, pp 694-702.

[12] A. Rosenthal, "A computer scientist looks at reliability computations in: reliability and fault tree analysis", *SIAM*, 1975, pp 133-153.

[13] A. Satyanarayana, J.N. Hagstrom, "A new algorithm for the reliability analysis of multi-terminal networks", *IEEE Trans. Reliability*, vol R-30, 1981 Oct, pp 325-334.

[14] A. Satyanarayana, R.K Wood, "A linear-time algorithm for computing *k*-terminal reliability in series-parallel networks", *SIAM Journal of Computing*, vol 14, 1985 Nov, pp 818-832.

[15] D.A. Sheppard, "Standard for banking communication system", *IEEE Trans. Computer*, 1987 Nov, pp 92-95.

[16] L.G. Valiant, "The complexity of enumeration and reliability problems", *SIAM J. Computing*, vol 8, 1979, pp 410-421.

[17] P. Winter, "Steiner problem in networks: a survey", *Networks*, vol 17, 1987, pp 129-167.

[18] R.K. Wood, "Factoring algorithms for computing k-terminal network reliability", *IEEE Trans. Reliability*, vol R-35, 1986 Aug, pp 269-278.