Information Sciences 117 (1999) 89–106

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Efficient algorithms for reliability analysis of distributed computing systems

Min-Sheng Lin [a,*], Ming-Sang Chang [b], Deng-Jyi Chen [b]

[a] *Department of Information Management, Tamsui Oxford University College, 32, Chen-Li Rd., Tamsui, Taipei, 25103, Taiwan, ROC*

[b] *Institute of Computer Science and Information Engineering, National Chiao-Tung University, Hsin Chu, 30050, Taiwan, ROC*

Received 12 March 1998; received in revised form 23 October 1998; accepted 1 January 1999

Abstract

A distributed computing system is modeled as a collection of resources (e.g. processing elements, data files and programs) interconnected via an arbitrary communication network and controlled by a distributed operating system. The distributed program reliability in a distributed computing system is the probability of successful execution of a program running on multiple processing elements and needs to retrieve data files from other processing elements. This reliability varies according to (1) the topology of the distributed computing system, (2) the reliability of the communication edges, (3) the data files and programs distribution among processing elements and (4) the data files required to execute a program. In addition, computing the reliability of distributed computing systems is #*P*-complete even when the distributed computing system is restricted to a series-parallel, a 2-tree, a tree, or a star structure. This paper presents efficient algorithms for computing the reliability of a distributed program running on other restricted classes of networks. © 1999 Elsevier Science Inc. All rights reserved.

Keywords: Distributed computing systems; Distributed program reliability; Computational complexity; Algorithms

* Corresponding author. E-mail: mlin@jupiter.touc.edu.tw

PII: S 0 0 2 0 - 0 2 5 5 (9 9) 0 0 0 0 3 - 1

1. Introduction

A typical distributed computing system (DCS) consists of processing elements (nodes), communication links (links), memory units, data files, and programs [1,2]. These resources are interconnected via a communication network that dictates how information flows between nodes. Programs residing on some nodes can run using data files at other nodes.

A previous investigation [3], introduced distributed program reliability (DPR) to evaluate the reliability of DCSs. Consider DCS in which the nodes are perfectly reliable but the links can fail, *s*-independently of each other, with known probabilities. Successfully executing a distributed program depends on the node containing the program, other nodes that have required data files, and the links between them being operational. DPR is thus defined as the probability that a program with distributed files can run successfully despite some faults in the links. For example, consider the DCS in Fig. 1 which consists of four nodes (processing elements) and five edges (communication links). This figure also includes the available files at each processing element. Assume that program f_1 requires data files f_2, f_3, and f_4 to complete its execution, and it is running at node v_1, which holds data files f_2 and f_3. Hence, it must access data file f_4, which is stored in both nodes v_2 and v_4. Therefore, the DPR of the DCS in Fig. 1 can be formulated as: DPR = Prob[(v_1 and v_2 are connected) or (v_1 and v_4 are connected)].

Although several algorithms have been proposed for evaluation DPR [4,5], none satisfy our desire for more efficient algorithms. We hypothesize that either the approaches examined are ineffective, or that no efficient algorithms exist for our reliability problems. Lin and Chen [6] demonstrated, for the first time, that computing DPR is #*P*-hard even when the distributed computing system is restricted to a series-parallel, a 2-tree, a tree, or a star structure. The class of #*P*-complete problems was introduced by Valiant [7]. The class #*P*

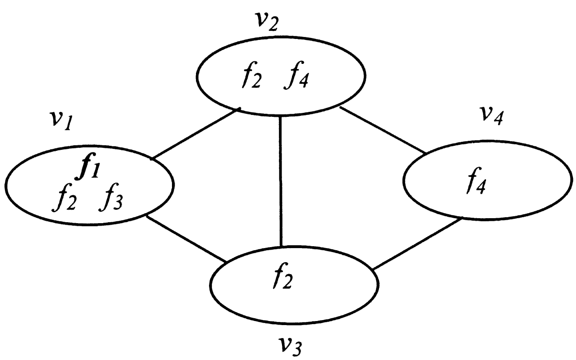

Fig. 1. A simple DCS.

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. This complexity can be averted by considering only a restricted class of DCS's. In light of above discussion, this paper presents a polynomially-solvable case of DPR problem for star topologies in which data files are restricted to a certain type of distribution. A linear time algorithm is also proposed to verify whether or not a star DCS has this restricted class of file distribution. Also proposed herein are two polynomial-time algorithms for computing the DPR of a DCS with a linear and a circular structure, respectively.

2. Assumptions, definitions and notation

Assumptions

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

Definitions

- A star DCS D_s has the *consecutive file distribution property* if and only if its nodes can be linearly ordered such that, for each distinct file f_i, the nodes containing file f_d occur consecutively. More formally, a star DCS D_s has the *consecutive file distribution property* if and only if there exists a permutation $\Pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ of numbers $\{1, 2, \ldots, n\}$ such that if file $f_d \in A\pi(i)$ and $f_d \in A\pi(i)$, then $f_d \in A_\pi(k)$ for all k, $i < k < j$.
- A set C of edges of D_s is referred to as a *file cut set* if and only if all edges in C fail which implies system failure.
- A file cut set C is referred to as *minimal* if there is no other file cut set C' such that $C' \subseteq C$.
- A set I of edges for a linear DCS D_l is referred to as a *file path set* if and only if all edges in I function which implies system functions.
- A file path set I is referred to as *minimal* if there is no other file path set I' such that $I' \subseteq I$.

Notation

(*general*)

D	a Distributed Computing System (DCS)
n	number of edges in D

e_i	edge i in D
v_i	node i in D
f_i	data file i
m	number of distinct files in D
t	total number of files in D
A_i	the set of files available at node v_i
p_i	probability that edge e_i functions
q_i	probability that edge e_i fails; $\equiv 1-p_i$
$\overline{E}$	complement of event E

for star topology

D_s	a star DCS with $n+1$ nodes $\{s, v_1, v_2, \ldots, v_n\}$ and n edges $\{e_1=(s,v_1), e_2=(s,v_2), \ldots, e_n=(s, v_n)\}$
Π	$\equiv [\pi(1), \pi(2), \ldots, \pi(n)]$ a permutation of numbers $\{1, 2, \ldots, n\}$ such that if file $f_d \in A_{\pi(i)}$ and $f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ for all k, $i<k<j$
C_d	the minimal file cut set for file f_d if it consists of all edges (s, v_i) such that node v_i contains file f_d, i.e. $C_d=\{(s, v_i) \mid f_d \in A_i\}$. (Without loss of generality, we reorder the minimal file cut sets, if necessary, by their minimal component, i.e. for two distinct minimal file cut sets C_i and C_j, $i<j$ if and only if $\min\{k \mid (s, v_{\pi(k)}) \in C_i\} < \min\{k \mid (s, v_{\pi(k)}) \in C_j\}$.)
Φ	ordered set of all minimal file cut sets according to their minimal components
r	number of minimal file cut sets in Φ
α_i	$\equiv \min\{k \mid e_{\pi(k)} \in C_i\}$, i.e. the index of the minimal component in C_i
β_i	$\equiv \max\{k \mid e_{\pi(k)} \in C_i\}$, i.e. the index of the maximal component in C_i
$H(i, j)$	$\equiv\{e_{\pi(i)}, e_{\pi(i+1)}, \ldots, e_{\pi(j)}\}$; $1 \leqslant i \leqslant j \leqslant n$ (note that $C_i \equiv H(\alpha_i, \beta_i)$)
$X(i, j)$	event: all edges in $H(i, j)$ fail
W_i	$\equiv \bigcup_{j=1}^{i} X(\alpha_j, \beta_j)$ (note that the *DPR* of D_s can be expressed as $1-\Pr(W_r)$)
F_i	event: the star DCS D_s' fails in which it consists of $i+1$ nodes s, $v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(i)}$ and i edges $e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(i)}$

for linear topology

D_l	a linear DCS with $n+1$ nodes $\{v_0, v_1, v_2, \ldots, v_n\}$ and n edges $\{e_1=(v_0,v_1), e_2=(v_1,v_2), \ldots, e_n=(v_{n-1},v_n)\}$
I_i	the minimal file path set which starts at edge e_i
β_i	$\equiv \max\{k \mid e_k \in I_i\}$, i.e. , the index of the maximal component in I_i
Y_i	event: all edges in I_i function

U_i — $\bigcup_{j=1}^{i} Y_j$ (Notably, the DPR of D_l can be expressed as $1 - \Pr(U_n)$)

R_j — event: there exists an operating event Y_i between edges e_1 and e_j

for ring topology

D_r — a ring DCS with n nodes $\{v_1, v_2, \ldots, v_n\}$ and n edges $\{e_1 = (v_1, v_2), e_2 = (v_2, v_3), \ldots, e_{n-1} = (v_{n-1}, v_n), e_n = (v_n, v_1)\}$

$D_r^* e_i$ — the DCS D_r with edge $e_i = (v_i, v_{i+1})$ contracted so that nodes v_i and v_{i+1} are merged into a single node. This newly merged node contains all data files that were previously in nodes v_i and v_{i+1}, and

$D_r - e_i$ — the DCS D_r with edge e_i deleted.

3. Efficient algorithms for computing DPR of DCS's

According to a previous investigation [6], computing DPR over a star DCS is #P-complete, implying that polynomial algorithms unlikely exist for solving them. However, efficient algorithms possibly exist for computing DPR over some restricted classes.

3.1. Star DCS's with a consecutive 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_s be a star DCS and it have the consecutive file distribution property. Then, the minimal file cut sets can be ordered by their minimal component, i.e. for two distinct minimal file cut sets C_i and C_j, $i<j$ if and only if $\min\{k \mid (s, v_{\pi(k)}) \in C_i\} < \min\{k \mid (s, v_{\pi(k)}) \in C_j\}$. By definition, D_s fails if and only if at least one event $X(\alpha_i, \beta_i)$, $1 \leqslant i \leqslant r$, 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_s can be easily obtained as $\Pr[W_1] = \Pr[X(\alpha_1, \beta_1)]$. Next consider the case with $r \geqslant 2$. The unreliability of D_s with the first i's file cut sets is

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

This expression can be decomposed using conditional probability as

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

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

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

By event E_2, event E_1 can be rewritten as

- E_1': For each k, $1 \leqslant k \leqslant i-1$, 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

$$Sj = \{E_1' \text{occurs and edge } e_{\pi(j)} \text{is the last good one}\}, \text{ for } \alpha_{i-1} \leqslant j \leqslant \alpha_i - 1.$$

Thus,

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

and

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

Since S_j's are disjoint events, we have

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

The event $S_j \cap E_2, \alpha_{i-1} \leqslant j \leqslant \alpha_i - 1$, can be decomposed into three independent events: {no file cut set fail between edges $e_{\pi(1)}$ and $e_{\pi(j-1)}$}, {edge $e_{\pi(j)}$ functions}, and {all edges between $e_{\pi(j+1)}$ and $e_{\pi(\beta_i)}$ fail}. So

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

Therefore, according to Eqs. (1)–(4) , we have

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

The following theorem can now be easily established.

Theorem 1. *For* $2 \leqslant i \leqslant r$:

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

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

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

$$\Pr[X(j+1,\beta_i)] = \begin{cases} \frac{1}{q_{\pi(\alpha_{i-1})}} \cdot \Pr[X(\alpha_{i-1},\beta_{i-1})] \cdot \prod_{k=\beta_{i-1}+1}^{\beta_i} q_{\pi(k)} & \text{for } j=\alpha_{i-1}, \\ \frac{1}{q_{\pi(j)}} \cdot \Pr[X(j,\beta_i)] & \text{for } \alpha_{i-1} < j \leqslant \alpha_i - 1. \end{cases} \quad (6)$$

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

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

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

...

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

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

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

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

Next, the major algorithm-related strategies to compute the DPR of star DCS's are outlined. Given a star DCS D_s and the file distribution A_i's for each node. By assuming that D_s has the property of consecutive file distribution, let Π be a permutation of numbers $\{1,2,\ldots,n\}$ such that if file $f_d \in A_{\pi(i)}$ and $f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ for all k, $i<k<j$. All file cut sets can be easily enumerated from A_i's in the following manner: if node v_i contains file f_d, then file cut set 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 $\alpha_i = \min\{k|\ e_{\pi(k)} \in C_i\}$ and $\beta_i = \max\{k|\ e_{\pi(k)} \in C_i\}$. Then, remove the file cut sets which are not minimal and rearrange the remaining minimal file cut sets according to their α_i's values. Finally, use Theorem 1, Eqs. (6) and (7) to compute the DPR ($= 1 - \Pr[W_r]$). The algorithm is formally described as belows.

Algorithm Reliability_Star_DCS

Input: A star DCS D_s 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 $f_d \in A_{\pi(i)}, f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ for all k, $i<k<j$, where A_i represents the set of files available at node v_i.

Output : the DPR of D_s

begin

Step 1: // find all file cut sets //

for $i \leftarrow 1$ **to** m **do** $C_i \leftarrow \emptyset$; // 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 $1 \leqslant i \leqslant m$ //

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

begin

$\alpha_i \leftarrow \min\{k |\ e_{\pi(k)} \in C_i\ \}$;

$\beta_i \leftarrow \max\{k |\ e_{\pi(k)} \in C_i\ \}$;

end

Step 3: // find all minimal file cut set //

$\Phi \leftarrow \emptyset$;

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

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

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

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

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

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

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

begin

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

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

end

Step 6: // Apply Theorem 1 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** $\Pr(F_k) \leftarrow \Pr(W_{i-1})$;

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

end

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

end Reliability_Star_DCS

Complexity analysis

The time complexity of Algorithm **Reliability_Star_DCS** 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 cut sets, where t denotes the total number of files in Ds. Step 2 requires $O(2 \cdot \sum_{i=1}^{m} |C_i|) \approx O(t)$ time to set α_i and β_i, $1 \leqslant i \leqslant m$ and step 3 takes $O(m^2)$ time to obtain all minimal file cut sets. Step 4 requires the reordering of all minimal file cut sets 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 cut sets. In step 5, evaluating $\Pr[X(j+1, \beta_i)]$ by making use of Eq. (6) requires that

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

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 $O[t + t + m^2 + r \cdot \log r + (n+r) + n]$. Since $t \leqslant m \cdot n$, and $r \leqslant m$, the complexity of Algorithm Reliability_Star_DCS can be obtained as $O(m^2 + m \cdot n)$.

An illustrative example

To illustrate Algorithm **Reliability_Star_DCS** as stated above, consider the star DCS in Fig. 2 in which there is a consecutive file distribution property and the associative permutation $\Pi = [3, 6, 4, 2, 5, 1, 7]$. (In Section 3.2, we will show

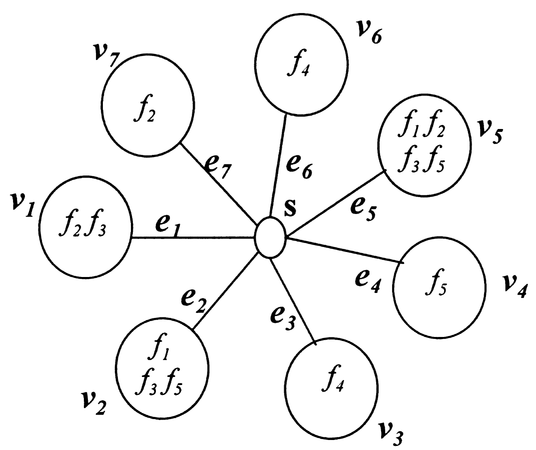

Fig. 2. A star DCS with the consecutive file distribution property.

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 cut sets 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 cut sets is

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

Step 4: Reorder the minimal file cut sets in such a manner that for C_i and C_j, $i < j$ if and only if $\alpha_i < \alpha_j$, and we obtain

$$C_1 = e_3, e_6, \alpha_1 = 1, \beta_1 = 2,$$

$$C_2 = e_2, e_5, \alpha_2 = 4, \beta_2 = 5,$$

$$C_3 = e_1, e_5, e_7, \alpha_3 = 5, \beta_3 = 7.$$

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 1 and Eq. (7) to compute $\Pr(W_i)$ and $\Pr(F_k)$ for $2 \leqslant i \leqslant 3$ and $\beta_i - 1 \leqslant k \leqslant \beta_i - 1$, and obtain

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

$$
\begin{aligned}
i=2: \quad & \Pr(F_2) = \Pr(F_3) = \Pr(F_4) = \Pr(W_1) = q_3q_6, \\
& \Pr(W_2) = \Pr(W_1) + [1 - \Pr(F_0)] \cdot p_3 \cdot \Pr[X(2,5)] && (j=2) \\
& \qquad + [1 - \Pr(F_1)] \cdot p_6 \cdot \Pr[X(3,5)] && (j=3) \\
& \qquad + [1 - \Pr(F_2)] \cdot p_4 \cdot \Pr[X(4,5)] && (j=4) \\
& \quad = q_3q_6 + p_3q_6q_4q_2q_5 + p_6q_4q_2q_5 + (1 - q_3q_6) \cdot \\
& \qquad p_4q_2q_5 \\
i=3: \quad & \Pr(F_5) = \Pr(W_2) \\
& \Pr(W_3) = \Pr(W_2) + [1 - \Pr(F_3)] \cdot p_2 \cdot \Pr[X(5,7)] && (j=5) \\
& \quad = q_3q_6 + p_3q_6q_4q_2q_5 + p_6q_4q_2q_5 \\
& \qquad + (1 - q_3q_6) \cdot p_4q_2q_5 + (1 - q_3q_6) \cdot p_2q_5q_1q_7
\end{aligned}
$$

Step 7: Therefore, DPR is

$$\begin{aligned}\mathrm{DPR} &= 1 - \Pr(W_3)\\ &= 1 - \{q_3q_6 + p_3q_6q_4q_2q_5 + p_6q_4q_2q_5 + (1 - q_3q_6)\cdot p_4q_2q_5\\ &\quad + (1 - q_3q_6)\cdot p_2q_5q_1q_7\}.\end{aligned}$$

3.2. 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 confirm whether or not a star DCS has the consecutive file distribution property. The problem statement would be:

Input: A star DCS *Ds* with $n+1$ nodes s, v_1, v_2, ..., v_n and file distributions A_i, $1 \leqslant i \leqslant n$.

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

Notably a solution does not always exist. To facilitate our search for the finding the correct ordering of Π, we use a data structure of a *PQ*-tree proposed by Booth and Leuker [8]. 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 if and only if 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.

By using *PQ*-tree data structure, we have the following algorithm.

Algorithm *Check_Consecutive_File_Distribution*

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

Output : A permutation $\Pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ of numbers$\{1, 2, \ldots, n\}$such that if file $f_d \in A_{\pi(i)}$ and $f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ 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** $A_j^{-1} \leftarrow \varnothing$; // m denotes the number of distinct files in D_s //

// A_j^{-1} is the set of indexes of nodes which contain the file f_j //

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

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

```
for j ← 1 to m do T ← REDUCE(T, A_j^{-1});
if T is a null tree
  then
    print out "D_s 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_j^{-1} and a replacement to be substituted for the pattern. The templates are applied from the bottom to the top of the tree. Notably, the null tree may be returned when no template is applied. For brevity, the details are omitted herein. Details of the algorithm can be found in Booth and Leuker [8].

Complexity analysis

For A_j^{-1}, $1 \leqslant j \leqslant m$, it can be obtained in $O(m + \sum_{i=1}^{n} |A_i|)$ steps. According to [8], the loop of *REDUCE* routine can be computed in $O(m + n + \sum_{j=1}^{m} |A_j^{-1}|)$ steps. Furthermore, it is very easy to verify that $\sum_{i=1}^{n} |A_i| = \sum_{j=1}^{m} |A_j^{-1}| = t$ (the total number of files in *Ds*). Therefore, the time complexity for the above algorithm is $O(m+t) + O(m+n+t) = O(m+n+t)$.

An illustrative example

Consider the star DCS D_s shown in Fig. 2. Applying the above algorithm lead to

$$A_1^{-1} = \{2,5\}, A_2^{-1} = \{1,5,7\}, A_3^{-1} = \{1,2,5\}, A_4^{-1} = \{3,6\}, A_5^{-1} = \{2,4,5\}.$$

Fig. 3 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_s of Fig. 2 has the consecutive file distribution property and one of the associative permutations is

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

3.3. Linear DCS's

In this section, we extend the results in Section 3.1 for computing the DPR of linear DCS's. Consider a linear DCS D_l with $n+1$ nodes $\{v_0, v_1, v_2, \ldots, v_n\}$ and n edges $\{e_1 = (v_0, v_1), e_2 = (v_1, v_2), \ldots, e_n = (v_{n-1}, v_n)\}$. Let I_i be the minimal file path set which starts at edge e_i. Notably, a linear DCS has the consecutive file distribution property resembling that of a star DCS such that for each minimal file path set I if $e_i \in I$ and $e_j \in I$ then $e_k \in I$ for all k, $i<k<j$. Furthermore, by definition, the reliability of a linear DCS can be expressed as

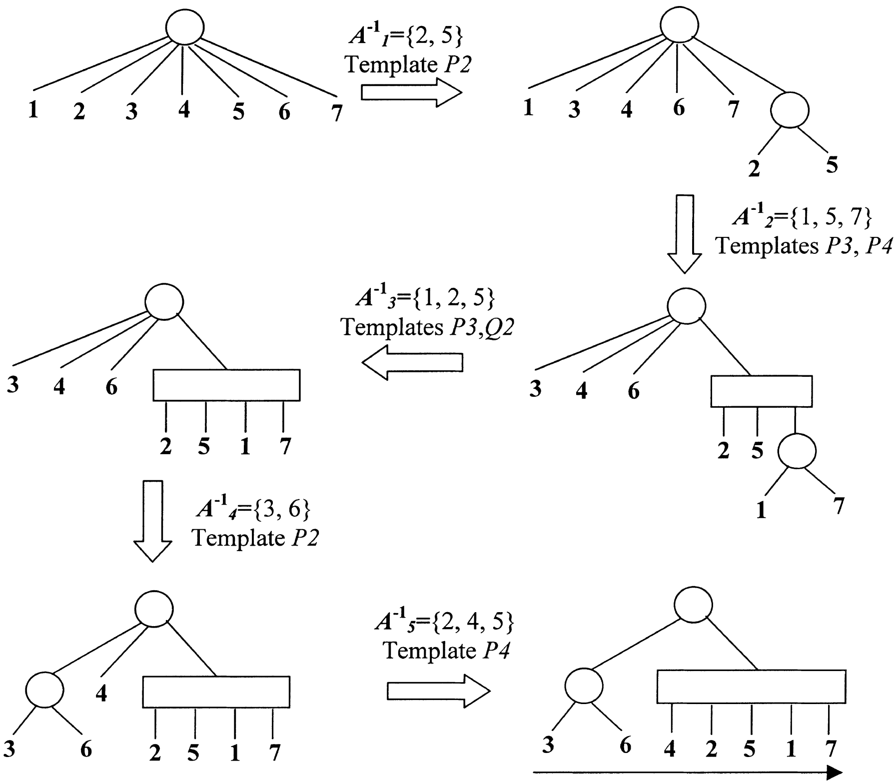

Fig. 3. The reduction steps by using a PQ-tree.

Prob{at least one minimal file path set I whose all edges function} and the unreliability of a star DCS with the consecutive file distribution property can be expressed as Prob{at least one minimal file cut set C whose edges all fail}.Owing to this duality, a simple relationship exists between a linear DCS and a star DCS with the consecutive file distribution property. The relationship is stated as follows.

According to the mirror image described in Table 1, if let $W_i = U_i$, $\alpha_i = \pi(i) = i$, $p_i = q_i$, $\Pr(F_i) = \Pr(R_i)$, and $X(i, \beta_i) = Y_i$, in Theorem 1, then the following theorem can be readily obtained to compute the reliability of a linear DCS D_l.

Theorem 2. *For* $2 \leqslant i \leqslant n$:

$$\Pr(U_i) = \Pr(U_{i-1}) + [(1 - \Pr(R_{i-2})] \cdot q_{i-1} \cdot \Pr(Y_i)$$

with the boundary conditions $\Pr(U_1) = \Pr(Y_1)$ *and* $\Pr(R_j) = 0$ *for* $j \leqslant \beta_1$. □

Table 1
The relationship between a linear DCS and a star DCS with the consecutive file distribution

Star DCS D_s with the consecutive file distribution	↔	Linear DCS D_l
minimal file cut set C	↔	minimal file path set I
$q_i \equiv$ probability that edge e_i fails	↔	$p_i \equiv$ probability that edge e_i functions
$[\pi(1), \pi(2), \ldots, \pi(n)]$ a permutation such that if file $f_d \in A_{\pi(i)}$ and $f_d \in A_{\pi(j)}$, then $f_d \in A_{\pi(k)}$ for all k, $i<k<j$	↔	$[\pi(1), \pi(2), \ldots, \pi(n)] = (1,2, \ldots, n)$
the unreliabilty of D_s	↔	the reliability of D_l

In addition, $\Pr(Y_i)$ and $\Pr(R_j)$ can be easily obtained from Eq. (6) as follows.

$$\Pr(Y_i) = \begin{cases} \frac{1}{p_{i-1}} \cdot \Pr(Y_{i-1}) \cdot \prod_{j=\beta_{i-1}+1}^{\beta_i} p_j & \text{for } \beta_i \leqslant n, \\ 0 & \text{for } \beta_i = \infty, \end{cases} \tag{8}$$

with the boundary condition $\Pr(Y_1) = \prod_{j=1}^{\beta_1} p_j$, and

$$\Pr(R_j) = \begin{cases} \Pr(U_i) & \text{for } \beta_i \leqslant j \leqslant \beta_{i+1} - 1, \\ 0 & \text{for } 0 \leqslant j \leqslant \beta_1 - 1. \end{cases} \tag{9}$$

Next, the complete algorithm for computing the reliability of a linear DCS is presented as follows.

Algorithm Reliability_Linear_DCS

Input: A linear DCS D_l with $n+1$ nodes $\{v_0, v_1, v_2, \ldots, v_n\}$ and n edges $\{e_1 = (v_0,v_1), e_2 = (v_1,v_2), \ldots, e_n = (v_{n-1}, v_n)\}$
A_i : the set of files available at node v_i.

Output: the DPR of D_l

begin
Step 1: // find all β_i's //
  for $i \leftarrow 1$ **to** m **do** $\mathrm{NF}_i \leftarrow 0$ // NF_i is the number of file f_i between v_h and v_t //
  for each $f_i \in A_0$ **do** $\mathrm{NF}_i \leftarrow 1$;
  $h \leftarrow 0$; // h and k are two indexes moving among nodes //
  for $k \leftarrow 1$ **to** n **do**
   begin
    for each file $f_i \in A_k$ **do** $\mathrm{NF}_i \leftarrow \mathrm{NF}_i + 1$; // update the total number of file i for node v_k //
    MFPS ← *true*; // if there is a minimal file path set between v_h and v_t, then MFPS = *true* //
    while MFPS **do**
     begin
      for $i \leftarrow 1$ to m **do if** $\mathrm{NF}_i = 0$ **then** MFPS ← *false*;

```
// check if there exists a minimal file path set
if MFPS then
begin
for each file f_i ∈ A_h do NF_i ← NF_i − 1;
h ← h + 1;
β_h ← k;
end
end
end
for i ← h to n do β_i ← ∞;
Step 2: // compute Pr(Y_i) by Eq. (8) //
Pr(Y_1) ← ∏_{j=1}^{β_1} p_j // boundary condition //
for i ← 1 to n do
begin
if β_i ⩽ n then Pr(Y_i) ← 1/(p_{i−1}) · Pr(Y_{i−1}) · ∏_{j=β_{i−1}+1}^{β_i} p_j
else Pr(Y_i) ← 0
end
Step 3: // Apply Theorem 2 and Eq. (9) to compute Pr(U_i) and Pr(R_j) //
for i ← 0 to β_1 − 1 do Pr(R_i) ← 0; // boundary condition //
Pr(U_1) ← Pr(Y_1) ; // boundary condition //
for i ← β_1 to β_2 − 1 do Pr(R_i) ← Pr(U_1);
for i ← 2 to n do
begin
Pr(U_i) ← Pr(U_{i−1}) + [(1 − Pr(R_{i−2})] · q_{i−1} · Pr(Y_i);
for j ← β_i to β_i + 1 − 1 do Pr(R_j) ← Pr(U_i);
end
Step 4: DPR ← Pr(U_n); Output(DPR);
end Reliability_Linear_DCS
```

Complexity analysis

For step 1, the computational complexity of the procedure β_i is $O(n{\cdot}m)$ since the value of h in the inner while_loop monotonously increases and does not exceed the value of k, i.e. the index of the outer for_loop. Computing $\Pr(Y_i)$ in step 2 is the similar operation as computing $\Pr[X(j, \beta_i)]$ in step 5 of Algorithm Reliability_Star_DCS. Thus, the complexity for step 2 is $O(n+n) = O(n)$. Step 3, which is the same as step 6 of Algorithm Reliability_Star_DCS, can be computed in $O(n)$. Therefore, the algorithm Reliability_Linear_DCS takes $O(n{\cdot}m) + O(n) + O(n) = O(n{\cdot}m)$ time.

An illustrative example

Consider the linear DCS D_l in Fig. 4. Applying the algorithm Reliability_Linear_DCS yields

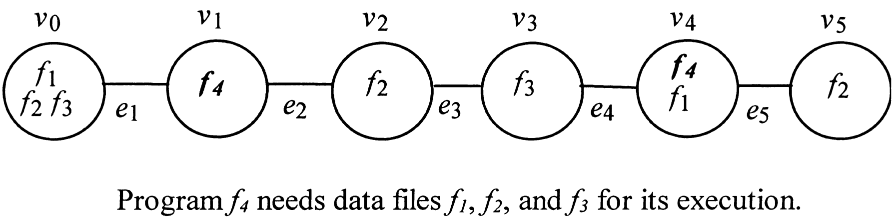

Fig. 4. A DCS with a linear structure.

Step 1:

$$\beta_1 = 1,\ \beta_2 = 2,\ \beta_3 = 4,\ \beta_5 = \infty;$$

Step 2:

$$\Pr(Y_1) = p_1,\ \Pr(Y_2) = p_2 \cdot p_3 \cdot p_4,$$

$$\Pr(Y_3) = p_3 \cdot p_4,\ \{\Pr(\mathrm{Y}_4) = \mathrm{p}_4 \cdot \mathrm{p}_5,\ \Pr(\mathrm{Y}_5) = 0;$$

Step 3:

$$\Pr(R_0) = 0;$$

$$\Pr(U_1) = p_1, \qquad \Pr(R_1) = \Pr(R_2) = \Pr(R_3) = \Pr(U_1) = p_1;$$

$$\begin{array}{lll}
i=2: & \Pr(U_3) & \Pr(U_2) = \Pr(U_1)+[1-\Pr(R_0)]\cdot q_1\cdot \Pr(Y_2) \\
 & & = p_1 + q_1\cdot p_2\cdot p_3\cdot p_4 \\
i=3: & \Pr(U_3) & = \Pr(U_2) + [1-\Pr(R_1)]\cdot q_2\cdot \Pr(Y_3) \\
 & & = p_1 + q_1\cdot p_2\cdot p_3\cdot p_4 + q_1\cdot q_2\cdot p_3\cdot p_4 \\
 & \Pr(R_4) & = \Pr(U_3) = p_1 + q_1\cdot p_2\cdot p_3\cdot p_4 + q_1\cdot q_2\cdot p_3\cdot p_4 \\
i=4: & \Pr(U_4) & = \Pr(U_3)+[1-\Pr(R_2)]\cdot q_3\cdot \Pr(Y_4) \\
 & & = p_1 + q_1\cdot p_2\cdot p_3\cdot p_4 + q_1\cdot q_2\cdot p_3\cdot p_4 + q_1\cdot q_3\cdot p_4\cdot p_5 \\
 & \Pr(R_5) & = \Pr(U_4) = p_1 + q_1\cdot p_2\cdot p_3\cdot p_4 + q_1\cdot q_2\cdot p_3\cdot p_4 + q_1\cdot q_3\cdot p_4\cdot p_5 \\
i=5: & \Pr(U_5) & = \Pr(U_4) + [1-\Pr(R_3)]\cdot q_4\cdot \Pr(Y_5) \\
 & & = \Pr(U_4) \qquad // \text{ since } \Pr(Y_5) = 0 \text{ //} \\
 & & = p_1 + q_1\cdot p_2\cdot p_3\cdot p_4 + q_1\cdot q_2\cdot p_3\cdot p_4 + q_1\cdot q_3\cdot p_4\cdot p_5
\end{array}$$

Step 4: Therefore, DPR is $\Pr(U_5) = p_1 + q_1\cdot p_2\cdot p_3\cdot p_4 + q_1\cdot q_2\cdot p_3\cdot p_4 + q_1\cdot q_3\cdot p_4\cdot p_5$.

3.4. Ring DCS's

A ring DCS is a DCS with a circular communication link. Each node connects two conjoining edges with two neighboring nodes. Assume that D_r is a DCS with a ring structure. According to the well known factoring theorem [7], the DPR of D_r is obtained as follows:

$$\mathrm{DPR}(D_r) = p_i \cdot \mathrm{DPR}(D_r^{*}e_i) + q_i \cdot \mathrm{DPR}(\mathrm{D}_r - e_i), \tag{10}$$

where e_i is an arbitrary edge of D_r. Since $D_r - e_i$ is a DCS with a linear structure with $n-1$ edges, its reliability can be computed by the algorithm Reliability_Linear_DCS in $O(n{\cdot}m)$ time. Notably, $D_r^* e_i$ remains a DCS with a ring structure with $n-1$ edges. The same analysis is then applied to $D_r^* e_i$. By recursively applying Eq. (10), we decompose the ring DCS D_r with n edges into, in the worst case, n linear DCSs. Therefore, we have an $O(n^2{\cdot}m)$ time algorithm for computing the reliability of a DCS with a ring structure.

Algorithm Reliability_Ring_DCS(D_r)
Step 1: if there exists one node that holds all distinct data files then *Return* ($DPR \leftarrow 1$);
Step 2: Select an arbitrary edge e_i of D_r;
Step 3: *Rell* $\leftarrow$ Reliability_Linear_DCS($D_r - e_i$);
Step 4: *Relr* $\leftarrow$ Reliability_Ring_DCS($D_r^* e_i$);
Step 5: *Return*($DRP \leftarrow p_i{\cdot}\mathrm{Rel}_r + q_i{\cdot}\mathrm{Rel}_l$);
end Reliability_Ring_DCS

An illustrative example

Consider the DCS with a ring topology in Fig. 5. This is a simplification of the DCS in Fig. 4 with one edge e_6 added between nodes v_5 and v_0. Applying algorithm Reliability_Ring_DCS yields

$$\begin{aligned}\mathrm{DPR}(D_r) &= q_6 \cdot \mathrm{DPR}(D_r - e_6) + p_6 \cdot \mathrm{DPR}(D_r^* e_6) \\ &= q_6 \cdot \mathrm{DPR}(D_r - e_6) + p_6 \cdot [q_5 \mathrm{DPR}(D_r^* e_6 - e_5) \\ &\quad + p_5 \cdot \mathrm{DPR}(D_r^* e_6^* e_5)].\end{aligned}$$

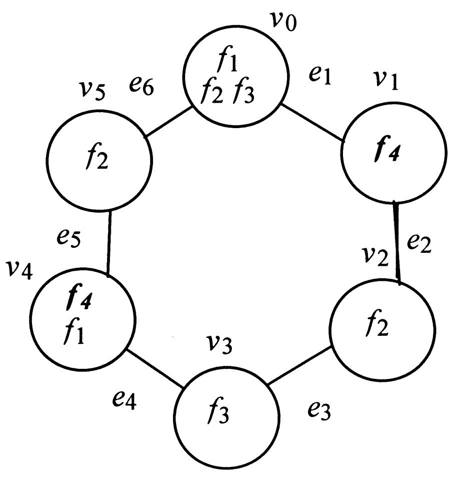

Fig. 5. A DCS with a ring structure.

The fact that there exists one node in $D_r^* e_6^* e_5$ that holds all distinct data files $\{f_1, f_2, f_3, f_4\}$, so we have $\mathrm{DPR}(D_r^* e_6^* e_5) = 1$. The example in Section 3.3 obviously reveals that $\mathrm{DPR}(D_r - e_6) = \Pr(U_5)$ and $\mathrm{DPR}(D_r^* e_6 - e_5) = \Pr(U_4)$. Therefore, we have

$$\begin{aligned}\mathrm{DPR}(D_\mathrm{r}) = {} & q_6 \cdot (p_1 + q_1 \cdot p_2 \cdot p_3 \cdot p_4 + q_1 \cdot q_2 \cdot p_3 \cdot p_4 + q_1 \cdot q_3 \cdot p_4 \cdot p_5) \\ & + p_6 \cdot [q_5 \cdot (p_1 + q_1 \cdot p_2 \cdot p_3 \cdot p_4 + q_1 \cdot q_2 \cdot p_3 \cdot p_4) + p_5].\end{aligned}$$

4. Conclusions

This paper elucidates the distributed program reliability in various classes of distributed computing systems. This reliability is computationally intractable for arbitrarily distributed computing systems, even when it is restricted to the class of star distributed computing systems. A particular solvable case for star distributed computing systems is identified, in which data files are distributed with respect to a consecutive property. In addition, a polynomial-time algorithm is developed for this case as well. Also proposed herein is a linear-time algorithm to verify whether or not an arbitrary star distributed computing system has this consecutive file distribution property. Furthermore, these results are applied towards star DCS's to obtain the reliability of linear and ring DCS's in polynomial time. A future work should attempt to construct efficient algorithms for computing lower and upper bounds on the distributed program reliability for arbitrarily distributed computing systems.

References

[1] P. Enslow, What is a distributed data processing system, Computer, vol. 11, Jan. 1978.
[2] J. Garcia-Molina, Reliability issues for fully replicated distributed database, IEEE Trans. Computer 16 (1982) 34–42.
[3] A. Satyanarayana, J.N. Hagstrom, A new algorithm for the reliability analysis of multi-terminal networks, IEEE Trans. on Reliability 30 (1981) 325–334.
[4] A. Kumar, S. Rai, D.P. Agrawal, On computer communication network reliability under program execution constraints, IEEE JSAC 6 (1988) 1393–1399.
[5] V.K.P. Kumar, S. Hariri, C.S. Raghavendra, Distributed program reliability analysis, IEEE Trans. Software Eng. 12 (1986) 42–50.
[6] M.S. Lin, D.J. Chen, The computational complexity of the reliability problem on distributed systemsInformation Processing Letters 64 (1997) 143–147.
[7] L.G. Valiant, The complexity of enumeration and reliability problems, SIAM J. Computing 8 (1979) 410–421.
[8] K.S. Booth, G.S. Leuker, Testing for the consecutive ones property interval graphs and graph planarity using PQ-tree algorithms, Journal of Computer System and Science 13 (1976) 335–379.