General Reduction Methods for the Reliability Analysis of Distributed Computing Systems

MIN-SHENG LIN AND DENG-JYI CHEN[1]

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

The reliability of a distributed computing system is the probability that a distributed program which runs on multiple processing elements and needs to communicate with other processing elements for remote data files will be executed successfully. This reliability varies according to (1) the topology of the distributed computing system, (2) the reliability of the communication links, (3) the data files and program distribution among processing elements, and (4) the data files required to execute a program. Thus, the problem of analyzing the reliability of a distributed computing system is more complicated than the *K*-terminal reliability problem, and many of the reliability-preserving reductions for speeding up the computation of the *K*-terminal reliability cannot be applied to this problem. In this paper, we shall propose several reduction methods for computing the reliability of distributed computing systems. These reduction methods can dramatically reduce the size of a distributed computing system, and therefore speed up the reliability computation.

Received December 1992, revised June 1993

1. INTRODUCTION

Recently, distributed computing systems (DCS) have become increasingly popular because they offer high fault tolerance, the potential for parallel processing and better reliability in comparison with other processing systems [3, 4]. A typical DCS consists of processing elements (PE), memory units, data files and programs. These resources are interconnected through a communication network that dictates how information flows between PEs. Programs residing on some PEs can run using the data files stored in other PEs. For successful execution of a program, it is essential that communication links between the PE containing the program and other PEs that have the required data files are operational. Distributed program reliability (DPR) is defined as the probability that a distributed program which runs on multiple processing elements and needs to communicate with other processing elements for remote files will be executed successfully. To illustrate the definition of DPR, consider the specific DCS shown in Figure 1, which consists of four processing elements (x_1, x_2, x_3, x_4) and five communication links ($x_{1,2}, x_{1,3}, x_{2,3}, x_{2,4}, x_{3,4}$). Program P1 requires data files, f1, f2 and f3 to complete execution, and it is running at node x_1, which holds data files f1 and f2. Hence, it must access data file f3, which is resident at both node x_2 and node x_4. Therefore, the reliability of the distributed program P1 can be formulated as follows:

$$\text{DPR(program P1)} = Prob\ (x_1 \text{ and } x_2 \text{ are connected})$$
$$\text{or } (x_1 \text{ and } x_4 \text{ are connected}))$$

In Kumar [11], a Minimum File Spanning Tree (MFST) is proposed to represent the multiterminal connections required to execute a distributed program, and

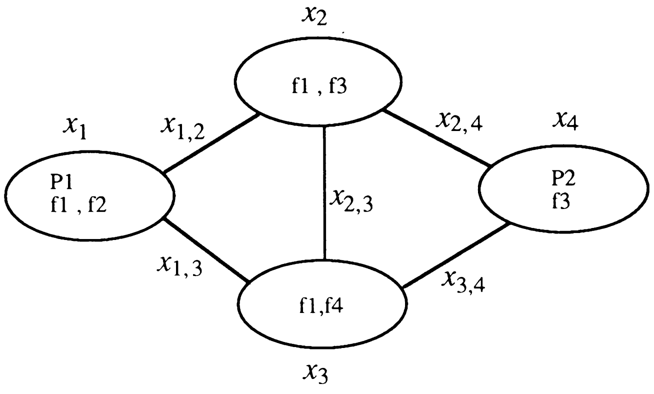

FIGURE 1. A simple DCS. Program P1 needs data files f1, f2 and f3 to complete execution.

a two-pass method for the reliability analysis of a DCS is developed. In this method, all MSFTs are obtained by using the breadth-first search method. Since the MSFTs it finds are not disjointed, the algorithm requires other reliability evaluation algorithms, such as SYREL [5], to generate the reliability expression. Although Kumar's method is elegant, it generates many replicated subgraphs during the procedure for finding all MSFTs and needs extra time to convert these MSFTs into the reliability expression. Thus, it is not an efficient reliability analysis algorithm. Another algorithm, called FARE [6, 7], has been proposed to compute the DPR directly by using a connection matrix. FARE does not require additional reliability evaluation algorithms to convert MSFTs into the reliability expression. The shortcoming of this algorithm is that it is not applicable to distributed programs running from more than one node. Therefore, more efficient algorithms are needed, such as FST-SPR [2] and FREA [9], which uses a graph cutting and expanding approach to generate disjoint File Spanning Trees (FSTs). Both FST-SPR and FREA were proven to be the factoring equivalent problem in [2, 9]. The principal difference between FST-SPR and FREA is that in FREA the edges incident to the nodes containing the executing programs are substituted to factor for the edges of a spanning tree found in FST-SPR. Hence, FREA does not incur the computational cost of searching for a spanning tree in each induced graph, and it is in general more efficient that FST-SPR.

To reduce the state space of the associated reliability problem, most algorithms apply reliability-preserving reductions. For example, the K-terminal reliability, i.e. the probability that a given set K of nodes in a network are connected to each other, can be computed in linear time for a series-parallel graph by repeated application of reliability-preserving reductions developed for the K-terminal reliability problem. Unlike the K-terminal reliability problem [8], in which K-terminal nodes are fixed and given, the distributed program reliability problem does not have fixed K-terminal nodes and the effect of redundant distributions of data files and programs must be taken into consideration. Since we do not specify the set of K-target nodes in DCS, we cannot directly apply the reductions [8] used in computing K-terminal reliability to the DPR problem. Obviously, if there are no replicated files, i.e. if there is only one copy of each file in the DCS, then the DPR problem can be transformed into the K-terminal reliability equivalent problem in which the K set is just 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, so the reduction methods for the K-terminal problem cannot be applied to the DPR problem. The reliability algorithms presented in Kumar [11] and Kumar [6] do not perform any reliability-preserving reductions, while FST-SPR and FREA use some special reliability-preserving reductions developed for DPR problem analysis.

In this paper, we shall introduce several general reliability-preserving reductions for DPR problem analysis and propose an algorithm, called MFREA (Modified FREA), which incorporates these general reliability-preserving reductions to speed up reliability computation.

2. NOTATION AND DEFINITIONS

In this paper we will make use of the following notation:

x_i	a node representing a processing element i
e_k	an edge representing a communication link k
$p_{e_k}(q_{e_k})$	probability that the link e_k works (fails)
$x_{i,j}$	an edge between processing elements i and j
$p_{i,j}(q_{i,j})$	probability that the link $x_{i,j}$ works (fails)
f_i	the data file i
P_i	the distributed program i
FA_i	the set of data files available at node x_i
$D=(V, E, FA)$	an undirected DCS graph with vertex (node) set V, edge set E and file distribution $FA=\{FA_1, FA_2, FA_3, \dots\}$.
D_F	an undirected DCS graph with a needed file set F
$R(D_F)$	the DPR of the D_F
$D_F - x_{i,j}$	the graph D with edge $x_{i,j}$ deleted
$D_F + x_{i,j}$	the graph D with edge $x_{i,j}$ contracted such that nodes x_i and x_j are merged into a single node. This new merged node contains all data files and programs that were in nodes x_i and x_j.

Without loss of generality, we identify a program with a special type of file. In the above notation, the DPR of P1 in the example in Figure 1 can be represented by $R(D_F)$, where $D=(V, E, FA)$ and

$$V=\{x_1, x_2, x_3, x_4\}$$

$$E=(x_{1,2}, x_{1,3}, x_{2,3}, x_{2,4}, x_{3,4}\}$$

$$FA=\{FA_1, FA_2, FA_3, FA_4\} \quad \text{and} \quad FA_1\{\text{P1}, \text{f1}, \text{f2}\},$$

$$FA_2=\{\text{f1}, \text{f3}\}, FA_3=\{\text{f1}, \text{f4}\}, FA_4=\{\text{P2}, \text{f3}\}$$

$$F=\{\text{P1}, \text{f1}, \text{f2}, \text{f3}\}$$

Definition. A node x_i is called a *reducible node* for a distributed program P_j in graph G if and only if (1) the degree of node x_i is two in graph G, and (2) x_i is not a leaf node of any MFST of program P_j.

Definition. A *File Spanning Tree* (FST) [11] is a tree whose nodes hold all needed files, i.e. $\cup_{x_i \in \text{FST}} FA_i \supseteq F$.

Definition. A *Minimal File Spanning Tree* (MFST) [11] is an FST such that there exists no other FST which is a subset of it. By the definition of MFST, the DPR can be written as

$$R(D_F) = \text{Prob(at least one MFST is operational), or}$$

$$R(D_F) = \text{Prob}\left(\bigcup_{j=1}^{\#\text{mfst}} \text{MFST}_j\right)$$

where *#mfst* is the number of MFSTs for a given needed file set F.

Definition. Let $D=(V, E, FA)$ be a DCS graph. We say node x_i *is useless for node* x_j under the set F of needed files being considered, represented by $x_i \Rightarrow x_j$ in our notation, if and only if

1. edge $x_{i,j}$ is in D
2. for the given set of needed files F, there is no such MFST in D in which x_i is a leaf node and x_j is the incident node of x_i.

Definition. In a DCS graph, a *chain* is an alternating sequence of distinct nodes and edges. Without loss of generality, a chain may be labeled $(x_0, x_{0,1}, x_1, x_{1,2}, \dots,$

$x_{n-1,n}, x_n$) such that the internal nodes, $x_1, x_2, \ldots, x_{n-1}$, are all of degree 2 and the end nodes, x_0 and x_n are of a degree greater than 2.

3. PRELIMINARIES

3.1. The generalized factoring theorem used in FREA

The factoring theorem [10] consists of picking an edge e_i in the graph and decomposing the original problem with respect to the two possible states of the edge e_i:

$$R(D_F) = p_{e_i} R(D_F \mid e_i \text{ works}) + q_{e_i} R(D_F \mid e_i \text{ fails}) \quad (3.1)$$

which, for a DCS with perfect nodes, can be formulated as

$$R(D_F) = p_{e_i} R(D_F + e_i) + q_{e_i} R(D_F - e_i) \quad (3.2)$$

Equation (3.2) can be generalized and represented as follows:

$$\begin{aligned} R(D_F) = {} & p_{e_1} R(D_F + e_1) + q_{e_1} q_{e_1} p_{e_2} R(D_F - e_1 + e_2) \\ & + \ldots + q_{e_1} q_{e_2} \cdots q_{e_{d-1}} p_{e_d} \\ & \times R(D_F - e_1 - e_2 - \ldots - e_{d-1} + e_d) \\ & + q_{e_1} q_{e_2} \cdots q_{e_d} R(D_F - e_1 - e_2 - \ldots - e_d) \quad (3.3) \end{aligned}$$

where $\{e_d, e_2, \ldots, e_d\}$ is the set of edges incident to the nodes containing the programs being executed. Since the subgraph $D_F - e_1 - e_2 - \ldots - e_d$ indicates that the nodes that include the programs being executed are disconnected from the other nodes, the reliability of subgraph $D_F - e_1 - e_2 - \ldots - e_d$ is equal to 0. Thus, we do not need to generate the subgraph $D_F - e_1 - e_2 - \ldots - e_d$ to compute its reliability. Therefore, (3.3) can be rewritten as

$$\begin{aligned} R(D_F) = {} & p_{e_1} R(D_F + e_1) + q_{e_1} p_{e_2} R(D_F - e_1 + e_2) \\ & + \ldots + q_{e_1} q_{e_2} \cdots q_{e_{d-1}} p_{e_d} \\ & \times R(D_F - e_1 - e_2 - \ldots - e_{d-1} + e_d) \quad (3.4) \end{aligned}$$

Equation (3.4) can be recursively applied to the induced graph until either (1) the further induced graph with a node contains all data files needed for the programs to be executed or (2) the further induced graph contains no FSTs. The induced graph of the former case represents success (reliability = 1); the latter case represents failure (reliability = 0). Since the subgraphs generated using (3.4) will be completely disjoint, no duplicated subgraphs will be generated during the expansion of the computation tree.

3.2. Special reduction methods for DCS reliability evaluation

Several reduction methods for the DCS reliability evaluation are proposed in [2, 9]. These special reduction methods are reviewed briefly below.

*Degree-*1 *reduction.* Degree-1 reduction removes (1) degree-1 nodes which contain none of the needed data files and programs under consideration, and (2) their incident edges.

Irrelevant component deletion. Let $D^0 = (V^0, E^0)$ be a connected component of D that is not connected to the rest of the components of D. If there are no FSTs in D^0 then the component D^0 is irrelevant and is deleted.

Parallel reduction. Let $x_{i,j}$ and $x'_{i,j}$ be two parallel edges in D. D' is obtained by replacing $x_{i,j}$ and $x'_{i,j}$ with a single edge $x''_{i,j}$ such that $p''_{i,j} = 1 - q_{i,j} * q'_{i,j}$ (or $p''_{i,j} = p_{i,j} + p'_{i,j} - p_{i,j} * p'_{i,j}$). The parallel reduction for the DPR problem is the same as the parallel reduction for the K-terminal reliability problem.

Series reduction. There are some differences in series reduction between the DCS reliability problem and the K-terminal network reliability problem. The series reduction for the K-terminal network reliability problem is recalled here:

Let $x_{i,j}$ and $x_{i,k}$ be two series edges in G such that degree(x_i) = 2 and $x_i \notin K$. Then G′ is obtained by replacing $x_{i,j}$ and $x_{i,k}$ with a single edge $x_{j,k}$ such that $p_{j,k} = p_{i,k} * p_{i,j}$.

The series reduction for the DCS reliability problem is the same as the above description except that the condition $x_i \notin K$ is replaced by $FA_i \cap FN = \emptyset$. In other words, if degree(x_i) = 2 and node x_i contains no required data files and programs to be executed, then we can apply series reduction on D.

*Degree-*2 *reduction.* Suppose node x_i is a reducible node; then one can apply series reduction on node x_i and move data files and programs within node x_i to one of its adjacent nodes x_j or x_k.

These reduction methods are not general enough for every case encountered during subgraph generation. Therefore, some of these reductions cannot be applied. For example, degree-1 reduction cannot be applied to degree-1 nodes containing needed data files.

4. GENERAL REDUCTION METHODS FOR THE DPR PROBLEM

The reduction methods proposed in [2, 9] and Section 3.2 are not general, and some of the reductions may not be applicable during subgraph generation. In this section we first discuss some properties of DCS graphs with respect to topology, file distribution, and reliability, and then propose more general reduction methods for DPR problem analysis.

4.1. Properties of DCS graphs

THEOREM 1. Consider a DCS graph D with a set of needed files F and an edge $x_{i,j}$. If $(FA_i \cap F) \subseteq (FA_j \cap F)$ then $x_i \Rightarrow x_j$.

Proof. To show that $x_i \Rightarrow x_j$, we must prove that for the given set F of needed files there exists no MFST that contains a leaf node x_i with an incident node x_j. Suppose there exists one MFST T that contains a leaf node x_i with an incident node x_j. Then $(FA_i \cap F) \subseteq (FA_j \cap F)$, implying that the subset T' of T with deleted x_i and $x_{i,j}$ is also an FST. By the definition of MFST, T is not an MFST. This is a contradiction. Therefore, there exists no MFST in D that contains a leaf node x_i with an incident node x_j. So $x_i \Rightarrow x_j$. QED

COROLLARY 1. Let $(x_0, x_{0,1}, x_1, x_{1,2}, \ldots, x_{n-1,n}, x_n)$ be a chain in a DCS graph $D = (V, E, FA)$ with a set of needed files F. If there exist x_i and x_j between x_0 and x_n such that $(FA_i \cap F) \subseteq (FA_j \cap F)$, then

$$x_i \Rightarrow x_{i+1} \quad \text{for } 0 \leqslant i < j \leqslant n, \quad \text{or}$$

$$x_i \Rightarrow x_{i-1} \quad \text{for } 0 \leqslant j < i \leqslant n$$

THEOREM 2. For a DCS graph $D = (V, E, FA)$ with a set of needed files F, if $x_i \Rightarrow x_j$ then $R(D_F) = R(D'_F)$, where $D' = (V, E, FA')$ and FA' is FA with $FA_i \cup FA_j$ substituted for FA_j (i.e. copying the files in node x_i into node x_j).

Proof. Figure 2 illustrates the basic concept behind the proof of this theorem. By (3.2), we get

$$\begin{aligned} R(D_F) &= p_{i,j}R(D_F + x_{i,j}) + q_{i,j}R(D_F - x_{i,j}) \\ &= p_{i,j}R(D1_F) + q_{i,j}R(D2_F), \quad \text{and} \\ R(D'_F) &= p_{i,j}R(D'_F + x_{i,j}) + q_{i,j}R(D'_F - x_{i,j}) \\ &= p_{i,j}R(D1_F) + q_{i,j}R(D3_F) \end{aligned}$$

Hence, if we can show $R(D2_F) = R(D3_F)$, then $R(D_F) = R(D'_F)$. To prove $R(D2_F) = R(D3_F)$, we create a new DCS graph $D4$ that is $D2$ with a new node x_k, new edge $x_{j,k}$, and $FA_k = FA_i$. Since $x_i \Rightarrow x_j$ in D, by the definition of $\Rightarrow$, D has no MFST that contains a leaf node x_i with an incident node x_j. Therefore, $D4$ has no MFST in which x_k is a leaf node with an incident node x_j. Thus, $D4$ and $D2$ must have the same MFSTs. So

$$R(D4_F) = R(D2_F)$$

and

$$R(D4_F) = p_{j,k}R(D3_F) + q_{j,k}R(D2_F)$$

by (3.2). Then we get $R(D2_F) = R(D4_F) = R(D3_F)$. Hence Theorem 2 follows. QED

THEOREM 3. For a DCS graph $D = (V, E, FA)$ with a set of needed files F, if $x_i \Rightarrow x_j$, $x_j \Rightarrow x_k$, and degree $(x_j) = 2$, then $R(D_F) = R(D'_F)$, where $D' = (V, E, FA')$ and FA' is FA with $FA_k = FA_i \cup FA_j \cup FA_k$.

Proof. This can be proved in a manner similar to the proof of Theorem 2.

COROLLARY 2. Let $(x_0, x_{0,1}, x_1, x_{1,2}, \ldots, x_{n-1,n}, x_n)$ be a chain in a DCS graph $D = (V, E, FA)$ with a set of needed files F. If there exist x_i and x_j $(i < j)$ between x_0 and x_n such that $x_i \Rightarrow x_{i+1} \Rightarrow x_{i+2} \Rightarrow \ldots \Rightarrow x_{j-1} \Rightarrow x_j$, then $R(D_F) = R(D'_F)$, where $D' = (V, E, FA')$ and FA' is FA with setting $FA_j = \cup_{k=i}^{j} FA_k$.

4.2. General reliability preserving reductions for the DPR problem

For a DCS graph $D = (V, E, FA)$ with a set of needed files F, we shall now introduce the following reduction methods to reduce the size of D for the computation of DPR.

4.2.1. R1 (general degree-1) reduction

Let x_i be a degree-1 node in D such that $x_{i,j}$ is its incident edge and $x_i \Rightarrow x_j$. Then, a reduced DCS graph

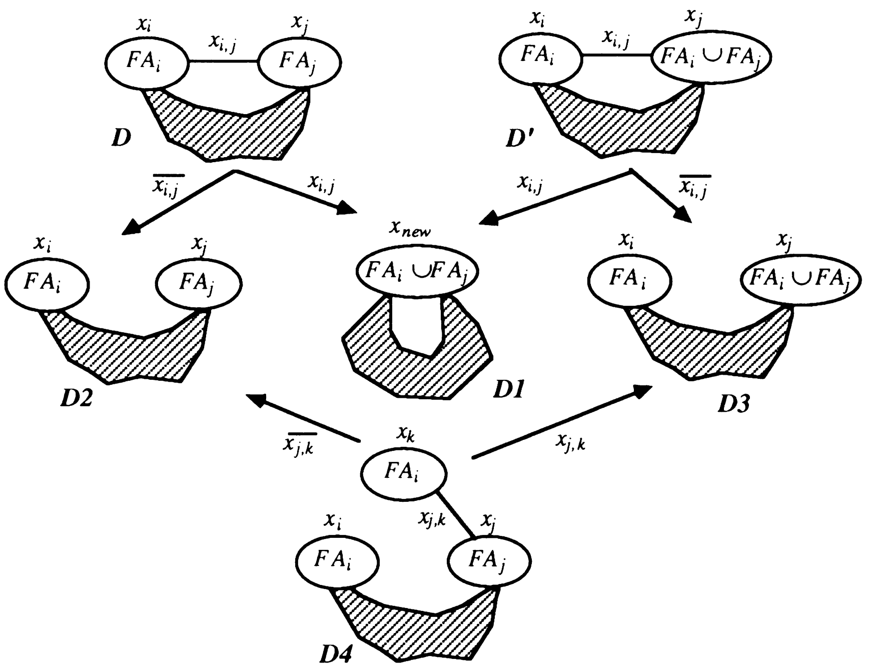

FIGURE 2. A snapshot of the derivation of the graph D's.

D' is obtained by (1) deleting x_i and $x_{i,j}$ and (2) replacing FA_j with $FA_i \cup FA_j$.

Proof. Consider the DCS graph in Figure 3. Since $x_i \Rightarrow x_j$, by Theorem 2, $R(D_F) = R(D1_F)$. Moreover, by (3.2),

$$\begin{aligned} R(D1_F) &= p_{i,j}R(D1_F + x_{i,j}) + q_{i,j}R(D1_F - x_{i,j}) \\ &= p_{i,j}R(D'_F) + q_{i,j}R(D'_F) \\ &= R(D'_F) \end{aligned}$$

Hence $R(D_F) = R(D'_F)$ and R1 reduction is reliability-preserving.

It is clear that the degree-1 reduction proposed in [8, 9] is just a special case of R1. Since $FA_i \cap FN = \emptyset$, the condition in degree-1 reduction just meets the condition $(FA_i \cap F) \subseteq (FA_j \cap F)$ in Theorem 1. Then $x_i \Rightarrow x_j$ can be obtained by Theorem 1 and the R1 reduction can be performed.

4.2.2. R2 (general degree-2) reduction

Let $x_{i,k}$ and $x_{i,k}$ be two series edges in D such that *degree* $(x_i) = 2$, $x_i \Rightarrow x_j$, and $x_i \Rightarrow x_k$. Then a reduced DCS graph D' is obtained by (1) replacing $x_{i,k}$ and $x_{i,k}$ with a single edge $x'_{j,k}$, (2) setting $p'_{j,k} = p_{i,j} * p_{i,k}$, and (3) replacing FA_j and FA_k with $FA_j \cup FA_i$ and $FA_k \cup FA_i$, respectively.

Proof. Consider the DCS graph in Figure 4. Since $x_i \Rightarrow x_j$ and $x_i \Rightarrow x_k$, by Theorem 2, $R(D_F) = R(D1_F)$. Moreover, by (3.2),

$$\begin{aligned} R(D1_F) &= q_{i,j}q_{i,k}R(D\beta_F) + q_{i,j}p_{i,k}R(D\beta_F) + p_{i,j}q_{i,k}R(D\beta_F) \\ &\quad + p_{i,j}p_{i,k}(D\alpha_F) \\ &= (1 - p_{i,j}p_{i,k})R(D\beta_F) + p_{i,j}p_{i,k}R(D\alpha_F) \\ R(D'_F) &= (1 - p'_{j,k})R(D\beta_F) + p'_{j,k}R(D\alpha_F) \end{aligned}$$

Letting $R(D1_F) = \Omega R(D'_F)$, we get the equations

$$1 - p_{i,j}p_{i,k} = \Omega(1 - p'_{j,k})$$
$$p_{i,j}p_{i,k} = \Omega p'_{j,k}$$

Solving these equations, we obtain

$$\Omega = 1 \quad \text{and} \quad p'_{j,k} = p_{i,j} * p_{i,k}$$

Hence, the R2 reduction is reliability-preserving.

COROLLARY 3. Let $(x_0, x_{0,1}, x_1, x_{1,2}, \ldots, x_{n-1,n}, x_n)$ be a chain in D. If there exist x_i, x_j, and x_k, $j < i < k$, between x_0 and x_n such that $(FA_i \cap F) \subseteq (FA_j \cap F)$ and $(FA_i \cap F) \subseteq (FA_k \cap F)$, then we can supply the R2 reduction on D by (1) replacing $x_{i,i-1}$ and $x_{i,i+1}$ with a single edge $x'_{i-1,i+1}$, (2) setting $p'_{i-1,i+1} = p_{i,i-1} * p_{i,i+1}$, and (3) replacing FA_{i-1} and FA_{i+1} with $FA_{i-1} \cup FA_i$ and $FA_{i+1} \cup FA_i$, respectively.

It is clear that the series reduction and degree-2 reduction proposed in [2, 9] are just special cases of the R2 reduction.

4.2.3. R3 reduction

Let $x_{i,j}$ and $x_{i,k}$ be two series edges in D such that *degree* $(x_i) = 2$, $x_i \Rightarrow x_j$ and there exists an edge $x_{j,k}$ in D. Then a reduced DCS graph D' is obtained by (1) deleting $x_{j,k}$ and (2) replacing FA_j with $FA_i \cup FA_j$. Then the new

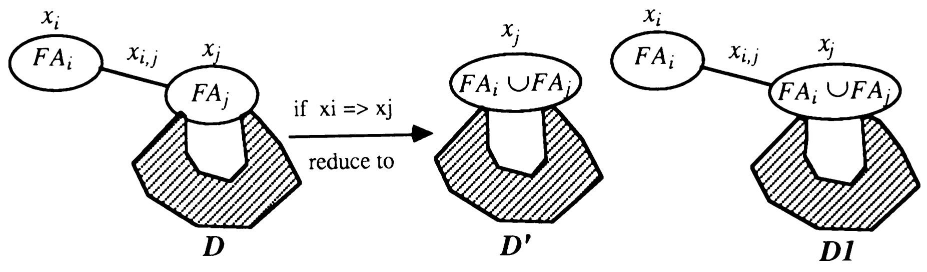

FIGURE 3. A snapshot of general degree-1 reduction.

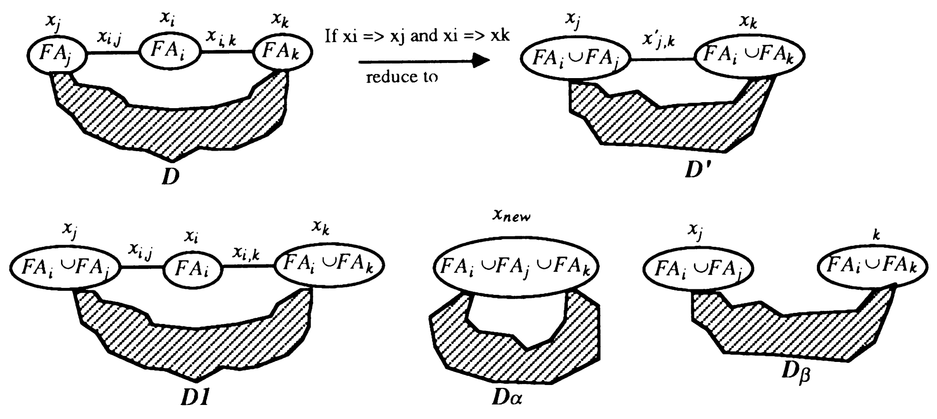

FIGURE 4. A snapshot of general degree-2 reduction.

reliability of links $x'_{i,k}$ and $x'_{i,j}$ will be

$$p'_{i,k} = 1 - q_{j,k} * q_{i,k}$$

and

$$p'_{i,j} = (p_{j,k} + q_{j,k} * p_{i,j} * p_{i,k})/(p_{j,k} + q_{j,k} * p_{i,k})$$

Proof. Consider the DCS graph in Figure 5. Since $x_i \Rightarrow x_j$ in D, by Theorem 2, $R(D_F) = R(D1_F)$. For D_1 and D', there are three classes of subgraphs, namely $D_\alpha D_\beta$, and D_γ. We must now enumerate the success and failure combinations of edges $x_{i,j}$, $x_{i,k}$, and $x_{j,k}$ in D_1 (see Table 1), and edges $x'_{i,j}$ and $x'_{i,k}$ in D' (see Table 2). Let $R(D1_F) = \Omega R(D'_F)$. We then obtain the equations

$$\alpha: \quad q_{i,j}p_{i,k} + p_{i,j}q_{i,k}p_{j,k} + p_{i,j}p_{i,k} = \Omega p'_{i,j}p'_{i,k}$$

$$\beta: \quad q_{i,k}q_{j,k} = \Omega q'_{i,k}$$

$$\gamma: \quad q_{i,j}p_{i,k}q_{j,k} = \Omega q'_{i,j}p'_{i,k}$$

Solving these equations, we obtain

$$\Omega = 1$$

$$p'_{i,k} = 1 - q_{j,k} * q_{i,k}$$

$$p'_{i,j} = (p_{j,k} + q_{j,k} * p_{i,j} * p_{i,k})/(p_{j,k} + q_{j,k} * p_{i,k})$$

Hence the R3 reduction is reliability-preserving.

TABLE 2. Success and failure modes of D'

$p'_{i,j}$	$p'_{i,k}$	Category
0	0	$D\beta$
0	1	$D\gamma$
1	0	$D\beta$
1	1	$D\alpha$

The R3 reduction can be generalized as stated in the following theorem.

THEOREM 4. If a DCS graph $D = (V, E, FA)$ contains the topology of Figure 6(a) such that there exists an edge $x_{0,n}$ and a chain $(x_0, x_{0,1}, x_{1,2}, x_2, \ldots, x_r, x_{r,r+1}, \ldots, x_{n-1,n}, x_n)$ between x_0 and x_n, and

$$x_{i+1} \Rightarrow x_i, \quad \text{for } 0 \leqslant i \leqslant r-1$$
$$x_i \rightarrow x_{i+1}, \quad \text{for } r+1 \leqslant i \leqslant n-1 \qquad (4.1)$$

then, a reduced DSC graph D' is obtained by (1) deleting edge $x_{0,n}$ and (2) replacing FA_0 and FA_n with $\cup_{i=0}^{r} FA_i$ and $\cup_{i=r+1}^{n} FA_i$, respectively. The new reliability of links $x_{0,1}, x_{1,2}, \ldots, x_{r,r+1}, \ldots$ and $x_{n-1,n}$ will be

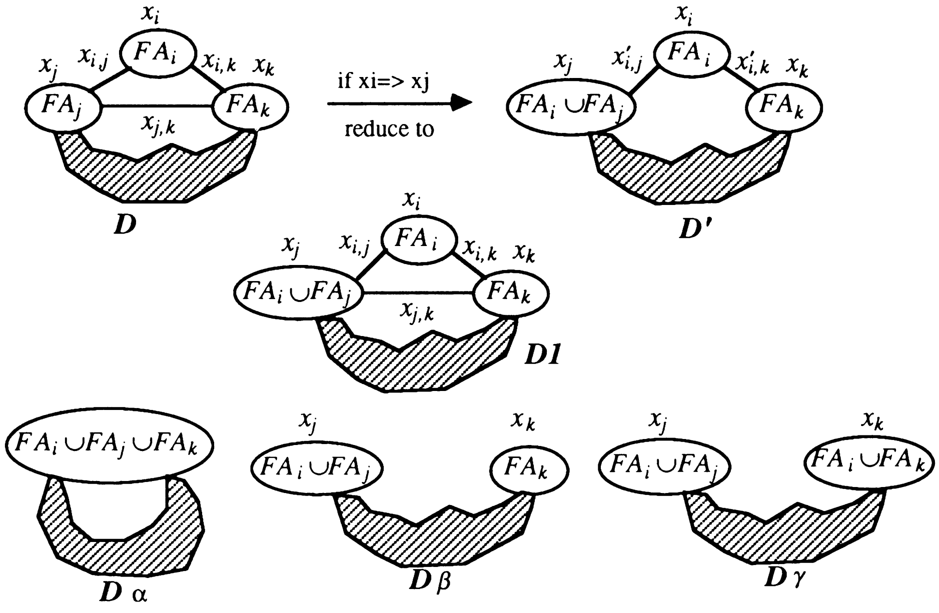

FIGURE 5. A snapshot of the R3 reduction.

TABLE 1. Success and failure modes of graph $D1$

$p_{i,j}$	$p_{i,k}$	$p_{j,k}$	Category
0	0	0	$D\beta$
0	0	1	$D\alpha$
0	1	0	$D\gamma$
0	1	1	$D\alpha$
1	0	0	$D\beta$
1	0	1	$D\alpha$
1	1	0	$D\alpha$
1	1	1	$D\alpha$

$$p'_{i,i+1} = \begin{cases} \dfrac{p_{0,n} + q_{0,n} * \prod_{j=i}^{n-1} p_{j,j+1}}{p_{0,n} + q_{0,n} * \prod_{j=i+1}^{n-1} p_{j,j+1}} & \text{for } 0 \leqslant i \leqslant r-1; \\ 1 - q_{0,n} * q_{i,i+1} & \text{for } i = r; \\ \dfrac{p_{0,n} + q_{0,n} * \prod_{j=0}^{i} p_{j,j+1}}{p_{0,n} + q_{0,n} * \prod_{j=0}^{i-1} p_{j,j+1}} & \text{for } r+1 \leqslant i \leqslant n-1 \end{cases} \qquad (4.2)$$

Proof. Consider the DCS graph in Figure 6. Since condition (4.1) holds, we get $R(D_F)=R(D1_F)$ by Corollary 2. For $D1$ and D', there are $n+1$ classes of subgraphs, i.e. α_0, α_1, ..., α_r, ..., α_{n-1}, and β. We now enumerate the success and failure combinations of edges $x_{0,1}, x_{1,2}, \ldots, x_{r,r+1}, \ldots, x_{n-1,n}$ and $x_{0,n}$ (see Table 3) in D, and edges $x'_{0,1}, x'_{1,2}, \ldots, x'_{r,r+1}, \ldots$ and $x'_{n-1,n}$ in D' (see Table 4). Let $R(D1_F)=\Omega R(D'_F)$. We then obtain the equations

α_0: $\quad q_{0,n}q_{0,1}p_{1,2}\cdots p_{n,n+1}=\Omega(q'_{0,1}p'_{1,2}\cdots p'_{n,n+1})$

α_1: $\quad q_{0,n}q_{1,2}p_{2,3}\cdots p_{n,n+1}=\Omega(q'_{1,2}p'_{2,3}\cdots p'_{n,n+1})$

... ...

... ...

α_{r-1}: $\quad q_{0,n}q_{r-1,r}p_{r,r+1}\cdots p_{n,n+1}$

$$=\Omega(q'_{r-1,r}p'_{r,r+1}\cdots p'_{n,n+1})$$

α_r: $\quad q_{0,n}q_{r,r+1}=\Omega(q'_{r,r+1})$

α_{r+1}: $\quad q_{0,n}p_{0,1}p_{1,2}\cdots p_{r,r+1}q_{r+1,r+2}$

$$=\Omega(p'_{0,1}p'_{1,2}\cdots p'_{r,r+1}q'_{r+1,r+2})$$

α_{r+2}: $\quad q_{0,n}p_{0,1}p_{1,2}\cdots p_{r,r+1}p_{r+1,r+2}q_{r+2,r+3}$

$$=\Omega(p'_{0,1}p'_{1,2}\cdots p'_{r,r+1}p'_{r+1,r+2}q'_{r+2,r+3})$$

... ...

... ...

α_n: $\quad q_{0,n}p_{0,1}p_{1,2}\cdots p_{n-2,n-1}q_{n-1,n}$

$$=\Omega(p'_{0,1}p'_{1,2}\cdots p'_{n-2,n-1}q'_{n-1,n})$$

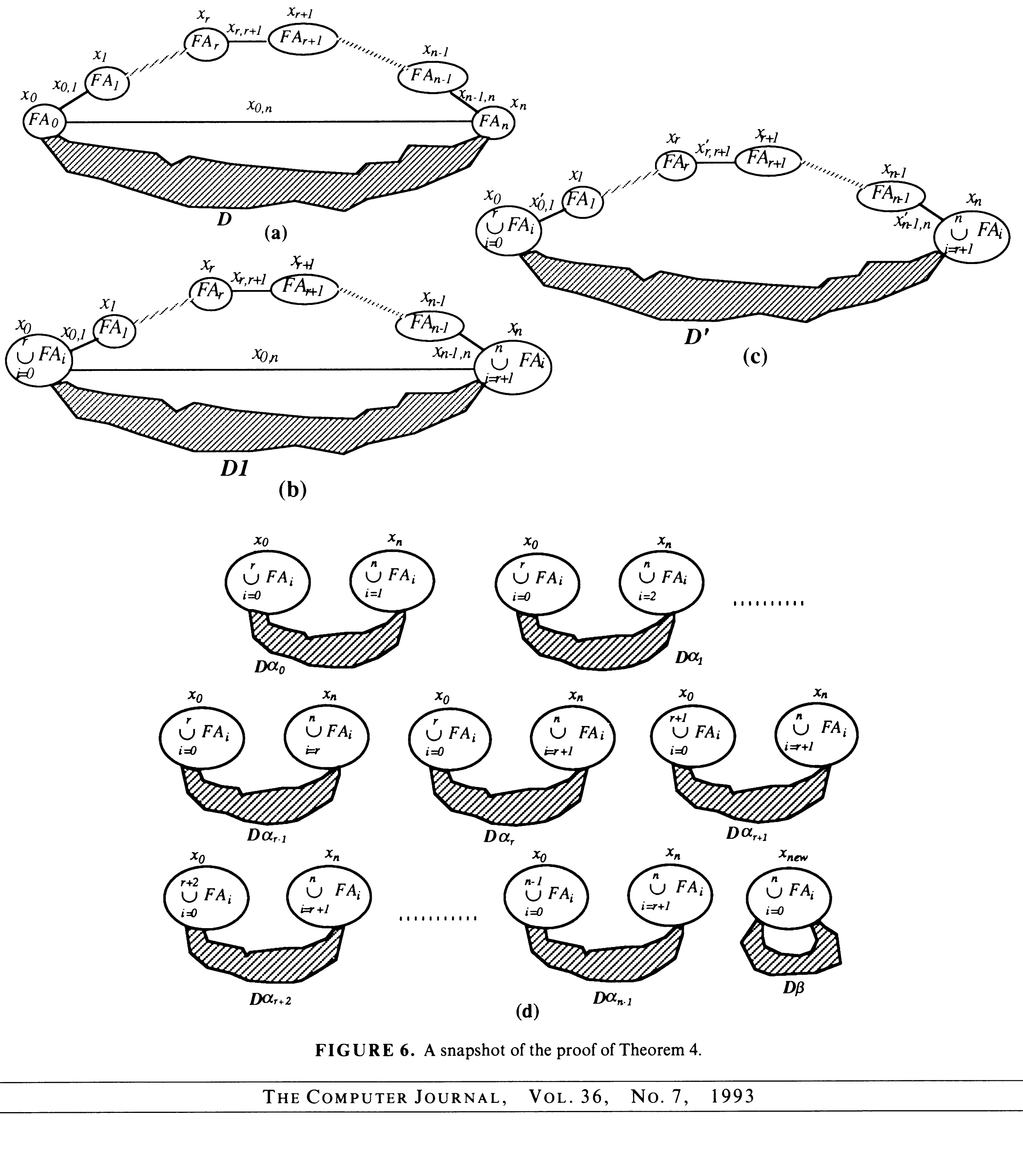

FIGURE 6. A snapshot of the proof of Theorem 4.

TABLE 3. Success and failure modes of graph $D1$ (where an asterisk indicates 'DON'T CARE')

$p_{0,n}$	$p_{0,1}$	$p_{1,2}$	$\cdots$	$p_{r-1,r}$	$p_{r,r+1}$	$p_{r+1,r+2}$	$\cdots$	$p_{n-2,n-1}$	$p_{n-1,n}$	*Category*
0	0	1	1 … 1	1	1	1	1 … 1	1	1	α_0
0	*	0	1 … 1	1	1	1	1 … 1	1	1	α_1
0	…	…	…	…	…	…	…	…	…	…
0	…	…	…	…	…	…	…	…	…	…
0	*	*	* … *	0	1	1	1 … 1	1	1	α_{r-1}
0	*	*	* … *	*	0	*	* … *	*	*	α_r
0	1	1	1 … 1	1	1	0	* … *	*	*	α_{r+1}
0	1	1	1 … 1	1	1	1	0 * … *	*	*	α_{r+2}
0	…	…	…	…	…	…	…	…	…	…
0	…	…	…	…	…	…	…	…	…	…
0	1	1	1	1	1	1	1	1	0	α_{n-1}
0	1	1	1	1	1	1	1	1	1	β
1	*	*	*	*	*	*	*	*	*	β

TABLE 4. Success and failure modes of graph D' (where an asterisk indicates 'DON'T CARE')

$p'_{0,1}$	$p'_{1,2}$	$\cdots$	$p'_{r-1,r}$	$p'_{r,r+1}$	$p'_{r+1,r+2}$	$\cdots$	$p'_{n-2,n-1}$	$p'_{n-1,n}$	*Category*
0	1	1 … 1	1	1	1	1 … 1	1	1	α_0
*	0	1 … 1	1	1	1	1 … 1	1	1	α_1
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
*	*	* … *	0	1	1	1 … 1	1	1	α_{r-1}
*	*	* … *	*	0	*	* … *	*	*	α_r
1	1	1 Γ 1	1	1	0	* … *	*	*	α_{r+1}
1	1	1 … 1	1	1	1	1 * … *	*	*	α_{r+2}
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
1	1	1	1	1	1	1	1	0	α_{n-1}
1	1	1	1	1	1	1	1	1	β

$$\beta: \quad q_{0,n}p_{0,1}p_{1,2}\cdots p_{n-1,n} + p_{0,n} = \Omega(p'_{0,1}p'_{1,2}\cdots p'_{n-1,n})$$

Solving these equations, we obtain

$$p'_{i,i+1} = \begin{cases} \dfrac{p_{0,n} + q_{0,n} * \prod_{j=i}^{n-1} p_{j,j+1}}{p_{0,n} + q_{0,n} * \prod_{j=i+1}^{n-1} p_{j,j+1}} & \text{for } 0 \leqslant i \leqslant r-1; \\ 1 - q_{0,n} * q_{i,i+1} & \text{for } i = r; \\ \dfrac{p_{0,n} + q_{0,n} * \prod_{j=0}^{i} p_{j,j+1}}{p_{0,n} + q_{0,n} * \prod_{j=0}^{i-1} p_{j,j+1}} & \text{for } r+1 \leqslant i \leqslant n-1 \end{cases}$$

Therefore, the generalization of the R3 reduction is also reliability-preserving. QED

COROLLARY 4. If a DCS graph $D = (V, E, FA)$ has the topology in Figure 6(a) and satisfies the conditions

$$\left(\bigcup_{i=1}^{r} FA_i \cap F\right) \subseteq (FA_0 \cap F)$$

and

$$\left(\bigcup_{i=r+1}^{n-1} FA_i \cap F\right) \subseteq (FA_n \cap F) \qquad (4.3)$$

then we can apply the reduction in Theorem 4 on D.

Proof. From condition (4.3), we get $(FA_i \cap F) \subseteq (FA_0 \cap F)$ for $1 \leqslant i \leqslant r$ and $(FA_i \cap F) \subseteq (FA_n \cap F)$ for $r+1 \leqslant i \leqslant n-1$. Then, by Corollary 1, we get $x_{i+1} \Rightarrow x_i$ for $0 \leqslant i \leqslant r-1$ and $x_i \Rightarrow x_{i+1}$ for $r+1 \leqslant i \leqslant n-1$. This meets condition (4.1) in Theorem 4. Hence we can apply the reduction in Theorem 4. QED

The R3 reduction follows immediately from Theorem 4 by letting $n = 2$ and $r = 1$. Note that parallel reduction is also a special case of Theorem 4 where $n = 1$ and $r = 0$. Moreover, by applying Theorem 4 we have the following reductions.

4.2.4. R4 reduction

Suppose D has the topology in Figure 7(a), with $x_i \Rightarrow x_j$ and $x_k \Rightarrow x_l$. Then a reduced DCS graph D' (see Figure 7(b)) is obtained by (1) deleting edges $x_{j,l}$ and (2) replacing FA_j and FA_l with $FA_j \cup FA_i$ and $FA_l \cup FA_k$, respectively. Then the new reliability of links $x_{i,j}$, $x_{i,k}$,

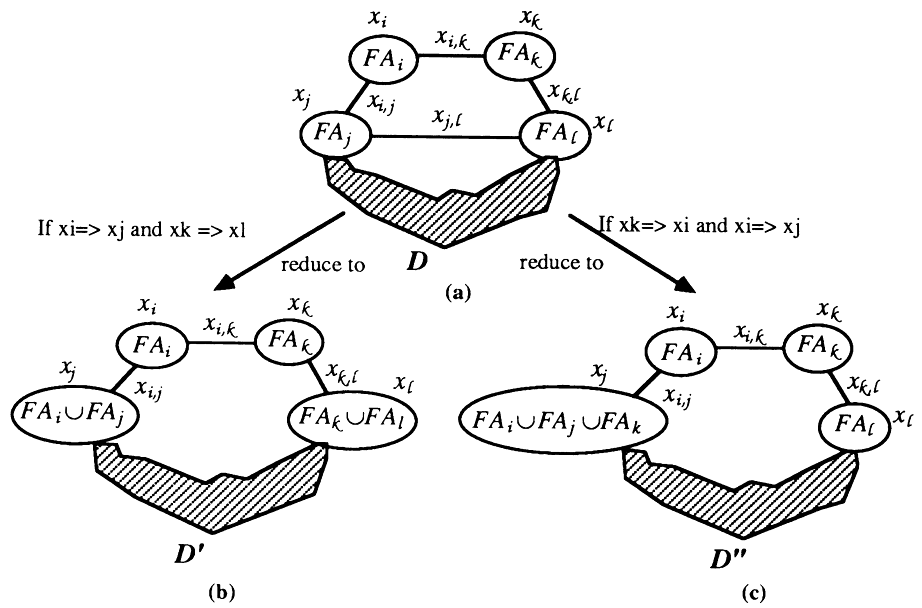

FIGURE 7. A snapshot of the R4 and R5 reductions.

and $x_{k,l}$ will be

$$p'_{i,j} = (p_{j,l} + q_{j,l} * p_{i,j} * p_{i,k} * p_{k,l})/(p_{j,l} + q_{j,l} * p_{i,k} * p_{k,l})$$

$$p'_{i,k} = 1 - q_{j,l} * q_{i,k}$$

$$p'_{k,l} = (p_{j,l} + q_{j,l} * p_{i,j} * p_{i,k} * p_{k,l})/(p_{j,l} + q_{j,l} * p_{i,j} p_{i,k})$$

Proof. This follows immediately from Theorem 4 by letting $n = 3$ and $r = 1$.

4.2.5. R5 reduction

Suppose D has the topology in Figure 7(a) with $x_i \Rightarrow x_j$ and $x_k \Rightarrow x_i$. Then a reduced DCS graph D'' (see Figure 7(c)) is obtained by (1) deleting edge $x_{j,l}$ and (2) replacing FA_j with $FA_j \cup FA_i \cup FA_k$. Then the new reliability links $p'_{i,j}$, $p'_{i,k}$, and $p'_{k,l}$ are as follows:

$$p'_{i,j} = (p_{j,l} + q_{j,l} * p_{i,j} * p_{i,k} * p_{k,l})/(p_{j,l} + q_{j,l} * p_{i,k} * p_{k,l})$$

$$p'_{i,k} = (p_{j,l} + q_{j,l} * p_{i,k} * p_{k,l})/(p_{j,l} + q_{j,l} * p_{k,l})$$

$$p'_{k,l} = 1 - q_{j,l} * q_{k,l}$$

Proof. This reduction follows immediately from Theorem 4 by letting $n = 3$ and $r = 2$.

4.3. The identification of $x_i \Rightarrow x_j$

Reductions R1 through R5 proposed above are based on checking the relation of $\Rightarrow$ for any pair (x_i, x_j). Using Theorem 1 or Corollary 1 to check whether $x_i \Rightarrow x_j$ is very efficient, but the condition in Theorem 1 is only a sufficient condition, and not a necessary condition. Hence there may exist some $x_i \Rightarrow x_j$ that do not satisfy the condition in Theorem 1, i.e. FA_i is not a subset of FA_j. Therefore, if we rely on Theorem 1 alone we may not find all x_i and x_j for which $x_i \Rightarrow x_j$. We now present an algorithm called CHECK_USELESS for checking if $x_i \Rightarrow x_j$. By using this algorithm to check all (x_i, x_j) pairs, we can find all $x_i \Rightarrow x_j$.

```
Input:
        node $x_i$ and node $x_j$
Output:
        true    if and only if $x_i \Rightarrow x_j$
        false   otherwise
/* assume the original DCS graph $D$ has performed
the parallel reduction, i.e. there is no parallel edge in
$D$ */
CHECK_USELESS($x_i$, $x_j$)
begin
  if there is no edge $x_{i,j}$ in $D$ then
    return (false);
  $D1$ = deleting all incident edges of $x_i$ except edge
  $x_{i,j}$ from $D$;
    for each file $f_i$ in $FA_i \cap F$ do
    $D2$ = deleting all nodes except $x_i$ such that contain
    file $f_i$ from $D1$;
      if there exists one or more FSTs in $D2$ then
        return (false);
  od
  return (true);
end
```

4.4. Example

Consider the DSC graphs in Figure 8. If program P4 needs data files f1, f2, f4 and f6 to complete execution, then the original DCS graph $D1$ with eight edges can be reduced to $D8$, which has only one edge, by the following steps:

Step 1: Since $D1$ has $x_1 \Rightarrow x_2$ and $x_1 \Rightarrow x_3$, it can be reduced to $D2$ by applying the R2 reduction.

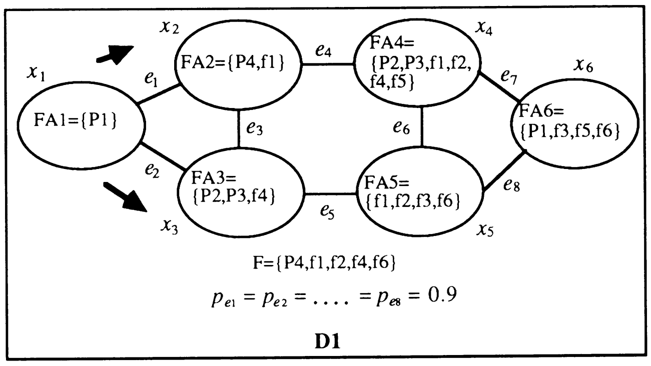

FIGURE 8. An example of applying general reliability-preserving reductions for the DPR problem.

Step 2: $D2$ can be reduced to $D3$ by applying parallel reduction.
Step 3: Since $D3$ has $x_3 \Rightarrow x_2$ and $x_3 \Rightarrow x_5$, it can be reduced to $D4$ by applying the R2 reduction.
Step 4: Since $D4$ has $x_6 \Rightarrow x_5$, it can be reduced to $D5$ by applying the R3 reduction.
Step 5: Since $D5$ has $x_4 \Rightarrow x_2$ and $x_4 \Rightarrow x_6$, it can be reduced to $D6$ by applying the R2 reduction.
Step 6: Since $D6$ has $x_5 \Rightarrow x_6$, it can be reduced to $D7$ by applying the R3 reduction.
Step 7: Since $D7$ has $x_6 \Rightarrow x_5$, it can be reduced to $D8$ by applying the R1 reduction.

5. THE MFREA ALGORITHM

5.1. The algorithm

MFREA uses (3.4) and the general reliability-preserving reductions discussed in Section 4.2 to compute the reliability of DCS. The complete MFREA algorithm is stated below.

```
Input
D = (V, E, FA): the DCS graph D with node set E,
edge set V and file distribution FA
F: the set of needed files to be connected
Output:
  R(D_F): the distributed program reliability
MFREA(D, F)
begin
Step 1: The checking step
  if there exists one node $x_i$ such that $FA_i \supseteq F$ then
    return (1);
  if there are no FSTs in D then        /* using DFS */
    return (0);
Step 2: The reduction step for D
  repeat
      Perform parallel reduction
      Perform R1, R2, R3, R4, and R5 reduction
  Until no reductions can be made
Step 3: The formulating step for (3.4)
  $D' =$ the new graph after the above reduction
  $D''' = D'' = D'$ /* $D'''$ and $D''$ are temporary variables
                for graph $D'$ */
  $R = 0$        /* set reliability to 0 */
  $C = 1$        /* the constant terms in (3.4) */
  for all $e_i \in$ the set of edges incident on the nodes
          containing the executing programs
          $P_j \in PN$ do
    $C = C^* p_{e_i}$
    $R = R + C^*$MFREA$(D''' + e_i, F)$
    $C = C^* q_{e_i}$
    $D'' = D''' - e_i$
    $D''' =$ the new graph after deleting irrelevant
    components from $D''$
    if there are no FSTs in $D'''$ then
      return(R)
    od
    return(R)
  end /* MFREA */
```

5.2. Example

Consider the DCS $D1$ in Figure 8 and substitute $FA_5 = \{f1, f2, f3\}$ for $FA_5 = \{f1, f2, f3, f6\}$. Using MFREA to evaluate the reliability of program P4, which needs data files f1, f2, f4 and f6 for its execution, we generate the subgraphs shown in Figure 9.

The DPR of program P4 can be compared as follows:

$$
\begin{aligned}
\mathrm{DPR}_4 & \\
&= p_{e_4}p_{e_{11}} + q_{e_4}p_{e_9}p_{e_{12}} \\
&= p_{e_4}[1 - q_{e_7}(1 - p_{e_8}p_{e_{10}})] \\
&\quad + q_{e_4}[(1-(1-p_{e_1}p_{e_2})q_{e_3})p_{e_5}]p_{e_{12}} \quad /\text{* } 10 = 6|9 \text{ */} \\
&= p_{e_4}(1 - q_{e_7}(1 - p_{e_8}[1 - q_{e_6}(1 - p_{e_9}])) \\
&\quad + q_{e_4}((1-(1-p_{e_1}p_{e_2})q_{e_3})p_{e_5})[1 - q_{e_8}(1 - p_{e_6}p_{e_7})] \\
&= p_{e_4}(1 - q_{e_7}(1 - p_{e_8}(1 - q_{e_6} \\
&\quad \times (1 - [(1-(1-p_{e_1}p_{e_2})q_{e_3})p_{e_5}])))) \\
&\quad + q_{e_4}((1-(1-p_{e_1}p_{e_2})q_{e_3})p_{e_5})(1 - q_{e_8}(1 - p_{e_6}p_{e_7})) \\
&= p_{e_4} - p_{e_4}q_{e_7} + p_{e_4}q_{e_7}p_{e_8} - p_{e_4}q_{e_6}q_{e_7}p_{e_8} \\
&\quad + p_{e_4}p_{e_5}q_{e_6}q_{e_7}p_{e_8} - q_{e_3}p_{e_4}p_{e_5}q_{e_6}q_{e_7}p_{e_8} \\
&\quad + p_{e_1}p_{e_2}q_{e_3}p_{e_4}p_{e_5}q_{e_6}q_{e_7}p_{e_8} + q_{e_4}p_{e_5} - q_{e_3}q_{e_4}p_{e_5}
\end{aligned}
$$

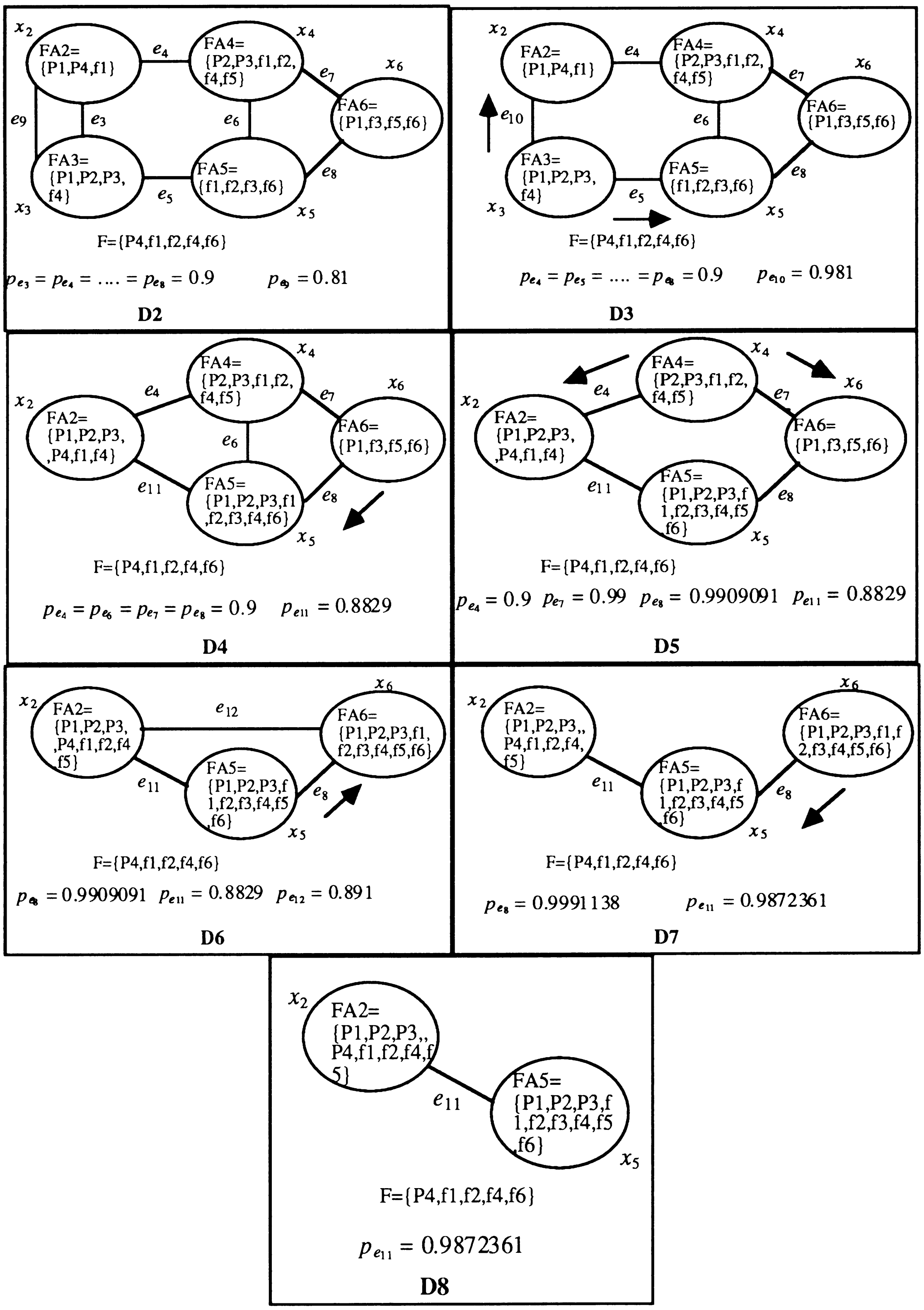
FA2= {P1,P4,f1}
FA4= {P2,P3,f1,f2,f4,f5}
FA3= {P1,P2,P3,f4}
FA5= {f1,f2,f3,f6}
FA6= {P1,f3,f5,f6}
x_2 x_3 x_4 x_5 x_6
e_3 e_4 e_5 e_6 e_7 e_8 e_9
F={P4,f1,f2,f4,f6}
$p_{e_3} = p_{e_4} = ... = p_{e_8} = 0.9$ $p_{e_9} = 0.81$
D2
FA2= {P1,P4,f1}
FA4= {P2,P3,f1,f2,f4,f5}
FA3= {P1,P2,P3,f4}
FA5= {f1,f2,f3,f6}
FA6= {P1,f3,f5,f6}
e_4 e_5 e_6 e_7 e_8 e_{10}
F={P4,f1,f2,f4,f6}
$p_{e_4} = p_{e_5} = ... = p_{e_8} = 0.9$ $p_{e_{10}} = 0.981$
D3
FA4= {P2,P3,f1,f2,f4,f5}
FA2= {P1,P2,P3,P4,f1,f4}
FA5= {P1,P2,P3,f1,f2,f3,f4,f6}
FA6= {P1,f3,f5,f6}
e_4 e_6 e_7 e_8 e_{11}
F={P4,f1,f2,f4,f6}
$p_{e_4} = p_{e_6} = p_{e_7} = p_{e_8} = 0.9$ $p_{e_{11}} = 0.8829$
D4
FA4= {P2,P3,f1,f2,f4,f5}
FA2= {P1,P2,P3,P4,f1,f4}
FA5= {P1,P2,P3,f1,f2,f3,f4,f5,f6}
FA6= {P1,f3,f5,f6}
F={P4,f1,f2,f4,f6}
$p_{e_4} = 0.9$ $p_{e_7} = 0.99$ $p_{e_8} = 0.9909091$ $p_{e_{11}} = 0.8829$
D5
FA2= {P1,P2,P3,P4,f1,f2,f4,f5}
FA6= {P1,P2,P3,f1,f2,f3,f4,f5,f6}
FA5= {P1,P2,P3,f1,f2,f3,f4,f5,f6}
e_8 e_{11} e_{12}
F={P4,f1,f2,f4,f6}
$p_{e_8} = 0.9909091$ $p_{e_{11}} = 0.8829$ $p_{e_{12}} = 0.891$
D6
FA2= {P1,P2,P3,,P4,f1,f2,f4,f5}
FA6= {P1,P2,P3,f1,f2,f3,f4,f5,f6}
FA5= {P1,P2,P3,f1,f2,f3,f4,f5,f6}
F={P4,f1,f2,f4,f6}
$p_{e_8} = 0.9991138$ $p_{e_{11}} = 0.9872361$
D7
FA2= {P1,P2,P3,,P4,f1,f2,f4,f5}
FA5= {P1,P2,P3,f1,f2,f3,f4,f5,f6}
F={P4,f1,f2,f4,f6}
$p_{e_{11}} = 0.9872361$
D8

FIGURE 8. continued.

$$+ p_{e_1}p_{e_2}q_{e_3}q_{e_4}p_{e_5} - q_{e_4}p_{e_5}q_{e_8} + q_{e_3}q_{e_4}p_{e_5}q_{e_8}$$
$$- p_{e_1}p_{e_2}q_{e_3}q_{e_4}p_{e_5}q_{e_8} + q_{e_4}p_{e_5}p_{e_6}p_{e_7}q_{e_8}$$
$$- q_{e_3}q_{e_4}p_{e_5}p_{e_6}p_{e_7}q_{e_8} + p_{e_1}p_{e_2}q_{e_3}q_{e_4}p_{e_5}p_{e_6}p_{e_7}q_{e_8}$$

where p_{e_i} is the probability of link i working and $q_{e_i} = 1 - p_{e_i}$.

Let the probability of any link being operational be 0.9. Then the DPR_4 is computed as 0.9766640.

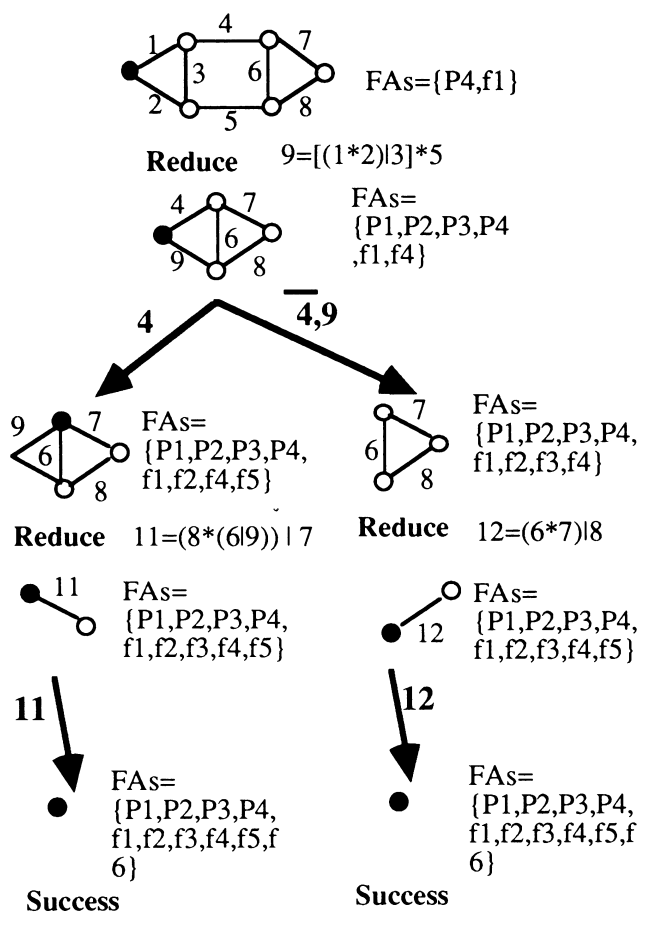
FAs={P4,f1}
Reduce 9=[(1*2)|3]*5
FAs={P1,P2,P3,P4,f1,f4}
4
4,9
FAs={P1,P2,P3,P4,f1,f2,f4,f5}
FAs={P1,P2,P3,P4,f1,f2,f3,f4}
Reduce 11=(8*(6|9)) | 7
Reduce 12=(6*7)|8
FAs={P1,P2,P3,P4,f1,f2,f3,f4,f5}
FAs={P1,P2,P3,P4,f1,f2,f3,f4,f5}
11
12
FAs={P1,P2,P3,P4,f1,f2,f3,f4,f5,f6}
FAs={P1,P2,P3,P4,f1,f2,f3,f4,f5,f6}
Success
Success

FIGURE 9. The subgraphs generated by MFREA supplied to Figure 8 with $FA_5 = \{f1, f2, f3\}$. Key: $*$, R2 reduction; |, parallel reduction; ●, the node including the executing programs; ○, nodes; $\bar{e}$, edge e is deleted; e, edge e is merged; s, the index of dark nodes.

TABLE 5. File distributions

Node	*File distribution* FA_i
1	P1, f2, f6
2	P3, f3, f11
3	f7
4	P10, f10
5	f3, f6
6	P8, f5
7	f11
8	P9, f12
9	P5, f4, f8
10	f9, f10
11	f1, f7
12	f5
13	f10
14	P2, f1, f2
15	P4, f4, f7
16	f8
17	f3
18	f6, f9
19	P7, f1
20	f5
21	P6, f2

TABLE 6. Data files required to execute the program P_i

Program	*Required files*
P3	f9, f10, f11
P4	f10, f11, f12
P7	f1, f8, f12
P9	f1, f11
P10	f4, f8, f12

5.3. Complexity analysis

It is well known that computing K-terminal reliability in general is NP-hard, or $\#P$-complete [11]. We have stated that the K-terminal problem is a special case of the DPR problem, so the DPR problem is also NP-hard. Thus, there exists no polynomial time algorithm for computing the reliability of distributed programs for general distributed computing systems. However, for the K-terminal reliability problem, some classes of networks, e.g. tree and series–parallel networks, can be computed in polynomial time by applying well-known reductions like series, degree-2, parallel, and polygon-to-chain reductions [8]. However, the DPR problem is much more complicated than the K-terminal problem, since its computational complexity depends not only on the topology of the network but also on the file distributions. Hence tree and series–parallel networks cannot yet be computed in polynomial time for the DPR problem. Actually, we have proven that the DPR problem for tree and series–parallel networks is still NP-hard [10]. However, we have proposed a number of reduction

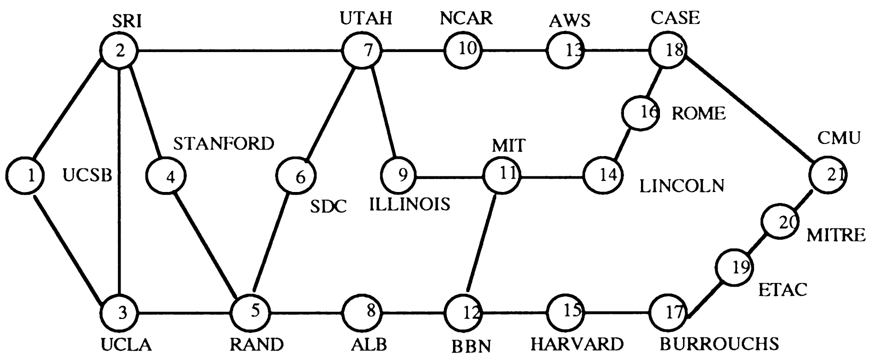

FIGURE 10. ARPA computer network.

TABLE 7. Comparison of algorithms for ARPA network

Comparison factor	*Algorithm*	*Program*				
		P3	P4	P7	P9	P10
No. of subgraphs generated	Kumar [11]	172907	197541	82759	72005	257333
	Kumar [6]	35515	38120	25135	22436	66752
	FST-SPR	2755	57	57	285	1339
	MFREA	50	145	23	16	267
Execution time (s)	Kumar [11]	1462.69	>1800	474.28	246.17	>1800
	Kumar [6]	7.29	7.75	5.11	4.56	13.40
	FST-SPR	0.71	7.34	0.31	0.71	3.40
	MFREA	0.39	1.05	0.14	0.16	1.33
DPR		0.97668	0.93457	0.91438	0.97039	0.96955

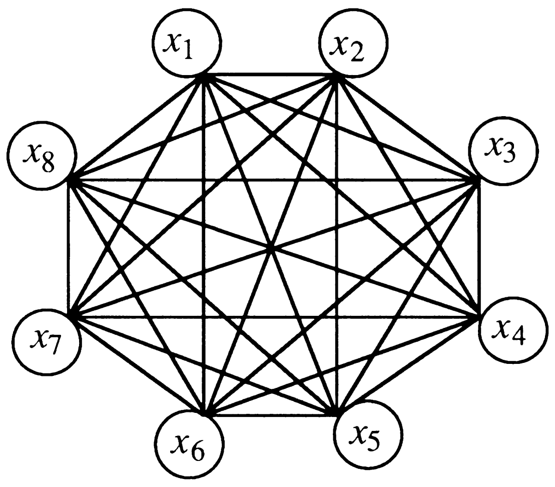

FIGURE 11. An eight-node completely connected DCS graph.

TABLE 8. File distributions

Node	*File distribution FA_i*
1	P1, P8, f1, f2
2	P2, P7, f2, f3
3	P3, P6, f3, f4
4	P4, P5, f4, f5
5	P4, P5, f1
6	P3, P6, f2
7	P2, P7, f3
8	P1, P8, f4

methods developed for the DPR problem to speed up its computation.

6. COMPARISONS

We implemented our algorithm and reduction methods in approximately 1100 lines of C. We then ran our program on an IBM RS/6000 workstation and tested it against FST-SPR [2] and the algorithm in Kumar [11] and Kumar [6]. We used the number of subgraphs generated during algorithm execution and the execution time for our comparison indexes.

TABLE 9. Data files required to execute the program P_i

Program	*Required files*
P1	f1, f2, f3
P4	f1, f2, f4, f5
P5	f1, f3, f5
P8	f1, f4, f5

6.1. Example 1

Figure 10 depicts a well-known example of a computer communications network, the ARPA computer network, in which there are 21 nodes and 26 links with a reliability of 0.9. Suppose that there are 12 data files and 10 programs distributed in the ARPA computer network and that the file distribution is as shown in Table 5. We ran five distributed programs, P3, P4, P7, P9 and P10; the data files required for these programs are shown in Table 6. A comparison of the number of subgraphs generated by and the execution time of the different algorithms is presented in Table 7.

6.2. Example 2

Consider the DCS graph in Figure 11, which has eight completely connected nodes. Assume that all links have a reliability of 0.5. The file distributions are shown in Table 8. We ran four programs, P1, P4, P5 and P8; the data files they required are shown in Table 9. Table 10 gives the comparison results for the four algorithms for example 2.

7. CONCLUSION

The DPR problem is more complicated than the K-terminal problem, since for the DPR problem we need to consider the file distributions in the DCS and we cannot specify the set of K target nodes. Therefore, most reliability-preserving reductions for the analysis of K-terminal reliability cannot be applied to the DPR problem. In this paper, we have investigated some properties of DCS and developed many general reduction methods for the DPR problem. We have also

TABLE 10. Comparison of algorithms for the eight-node completely connected graph

Comparison factor	*Algorithm*	*Program*			
		P1	P4	P5	P8
No. of subgraphs generated	Kumar [11	34321	>300000	>300000	>300000
	Kumar [6]	16173	65468	69701	123274
	FST-SPR	8343	11673	14018	9663
	MFREA	956	840	890	3986
Execution time (s)	Kumar [11]	357.26	>1800	>1800	>1800
	Kumar [6]	3.26	12.22	13.07	27.23
	FST-SPR	12.02	17.87	20.36	15.31
MFREA	MFREA	2.17	1.98	2.09	9.15
DPR		0.99929	0.99081	0.99065	0.99050

designed and implemented the MFREA algorithm, a modified version of FREA that incorporates new general reductions. Our comparisons of computational time and number of subgraphs generated show that MFREA is much faster than several previously proposed algorithms, such as MFST [11], FARE [6] and FST-SPR [2].

ACKNOWLEDGEMENTS

This research work was supported in part by the National Science Council of the ROC under contract no. NSC-80-0408-E009-16 and in part by the Chung San Institute of Technology under contract no. 7S79-0210-D009-03.

REFERENCES

[1] M. O. Ball, Computational complexity of network reliability analysis: an overview, *IEEE Transactions on Reliability*, **R-35**, pp. 230–239 (1986).
[2] D. J. Chen and T. H. Huang, Reliability analysis of distributed systems based on a fast reliability algorithm, *IEEE Transactions on Parallel and Distributed Systems*, **3**, pp. 139–153 (1992).
[3] P. Enslow, What is a distributed data processing system, *IEEE Computer*, **11**, pp. 00–00 (1978).
[4] J. Garcia-Molina, Reliability issues for fully replicated distributed database, *IEEE Computer*, **16**, pp. 34–42 (1982).
[5] S. Hariri and C. S. Raghavendra, *SYREL: A Symbolic Reliability Algorithm based on Path and Cutset Methods.* Technical Report, University of South Carolina (1984).
[6] A. Kumar, S. Rai and D. P. Agrawal, Reliability evaluation algorithms for distributed systems, in *Proceedings of IEEE INFOCOM 88*, pp. 851–860 (1988).
[7] A. Kumar, S. Rai and D. P. Agrawal, On computer communications network reliability under program execution constraints, *IEEE Journal on Selected Areas in Communication*, **6**, pp. 1393–1399 (1988).
[8] R. Kevin Wood, Factoring algorithms for computing *K*-terminal network reliability, *IEEE Transactions on Reliability*, **R-35**, pp. 269–278 (1986).
[9] M. S. Lin and D. J. Chen, Distributed program reliability analysis, *Proceedings of the IEEE 3rd Workshop on Future Trends of Distributed Computing Systems*, pp. 395–401 (1992).
[10] M. S. Lin and D. J. Chen, *Computational Complexity of the Reliability Problem in Distributed Computing Systems.* Technical Report, National Chiao-Tung University, Hsin Chu, Taiwan, ROC (1993).
[11] V. K. Prasnna Kumar, S. Hariri and C. S. Raghavendra, Distributed program reliability analysis, *IEEE Transactions on Software Engineering*, **SE-12**, pp. 42–50 (1986).
[12] A. Satyanarayana and M. K. Chang, Network reliability and the factoring theorem, *Networks*, **13**, pp. 107–120 (1983).