網路節點故障下之分散式程式可靠度計算
Computing Distributed Program Reliability with Unreliable Nodes

柯偉震
Wei-Jenn Ke

Chunghwa Telecommunication Laboratories, Taiwan R.O.C.
wjke@ms.tl.gov.tw

王勝德
Sheng-De Wang

Department of Electrical Engineering
National Taiwan University, Taiwan R.O.C.
sdwang@cc.ee.ntu.edu.tw

摘要

本論文提出一計算網路節點故障下之分散式程式可靠度之有效演算法，此方法將不改變原網路分解法之規則與搜尋樹，且額外平方時間複雜度即可求解.

關鍵字: 節點故障，分散式程式可靠度，網路分解

Abstract

In this paper, an efficient reliability evaluation method accounting for unreliable nodes in analyzing distributed program reliability is presented. The proposed method leaves practically unchanged the original network decomposition scheme and hence, the original event tree associated with the decomposition algorithm. It takes additional costs of only square time complexity to account for unreliable nodes.

Key words: Unreliable nodes, Distributed-program reliability, Network Decomposition.

1. Introduction

Acronyms

DCN	distributed computing networks
DPR	distributed program reliability
AGM	Aggarwal, Gupta, Misra Method [12]
NPR/T	node-pair reliability — Torrieri Method [13]
KHR	Kumar, Hariri, and Raghavendra algorithm [5, 6]
ENF	Evaluating Node Failures (presented in this paper)

Definition

DPR the probability that a distributed program can be executed successfully by accessing all the required files from the remote sites in the DCN.

In the DCN with different operational probabilities in computer sites and communication links, different distributions of programs and data files to various sites lead to different system reliabilities. It is important to develop an efficient reliability evaluation algorithm to estimate how reliable the system is. Several traditional reliability measures and their corresponding evaluation algorithms, such as source-to-multiterminal reliability [1], computer network reliability [2], and multi-terminal reliability [3] are not suited to evaluate the reliability of the DCN with programs and data files redundantly distributed, since they don't capture the effect of redundant resources distribution. Therefore Kumar, Hariri, and Raghavendra proposed another measure DPR and developed an algorithm KHR [4, 5] to compute it.

Other methods of computing DPR have been proposed in [6-9]. Unlike KHR, these methods assume that nodes are perfectly reliable to simplify the evaluation complexity. However, this assumption is not realistic since nodes can be failed more frequently than links in opposite situation. Therefore another special method such as AGM or NPR/T is explicitly applied to the resultant expression to compensate for the effect of unreliable nodes. For instance, Kumar and Agrawal propose an algorithm [8] to compute the DPR expression with perfect nodes and then suggest AGM to extend it to include node failures.

In this paper, we develop an efficient procedure for computing the effect of unreliable nodes that is directly integrated into the network decomposition algorithm to compute the DPR expression with unreliable nodes like KHR. Recently, Tsuchiya, Kakuda, and Kikuno [10] apply KHR to compute the DPR for an augmented graph based on the three-mode failure model. However KHR uses a two-pass procedure that generates all minimum file spanning trees comprising all unreliable nodes first, and algorithm SYREL [11] or a similar reliability algorithm is then called for reliability evaluation. Since unreliable nodes are added as inputs to SYREL for reliability evaluation, the time complexity increases exponentially with the number of unreliable nodes.

The proposed method for computing the effect of unreliable nodes called ENF integrates decomposition algorithms in particular to compute DPR with unreliable nodes. The decomposition algorithms are derived from a particular state space decomposition theorem extended from the factoring theorem to decompose the network into several subnetworks recursively rather than only two by the factoring algorithm. Essentially, the network is decomposed and reduced by contracting or deleting a set of links specified in the decomposition event, where the contraction of a link results in its two endnodes collapsed into a single node while the deletion just removes the

edge. The basic idea of the proposed method is to contract or delete a link including an edge and its two end nodes, not just a single edge. This would result in the problem that failures of nodes are not *s*-independent so that it poses difficulty to compute the effect of failure nodes while cutting a link. Nevertheless such problem is naturally avoided by decomposition algorithms owing to the fact that links specified in the decomposition event either failed or work are always incident with a perfect (coalescence) source in every subnetwork and it follows that failures of nodes in the subnetwork always remain *s*-independent.

Using ENF, the same decomposition events and the same set of subnetworks are derived as the original decomposition algorithm. Therefore the event tree remains practically unchanged. The reliability of each unreliable node is first updated in each subnetwork by ENF, which is used to compute the effect of the decomposition event and then, to gain the numerical reliability associated with each subnetwork. The additional cost of ENF accounting for unreliable nodes is in the order of square time complexity.

2. The Decomposition Algorithm

Assumptions

1. Bi-directional communication channels operate between computer sites.
2. The DCN is modeled by a simple undirected graph.
3. The communication channels and computer sites are either working or failed.
4. Failures are *s*-independent.

Notation

DPR(G) distributed program reliability of network G.

P, N_i, F_i program P, node i, file i in the DCN.

PN the set of needed files to execute program P.

n_i, X_i (v_i, e_i, u_i) node i, link X_i consisting of edge e_i and two end nodes v_i and u_i.

$G|\bar{1}\bar{2}\cdots\overline{i-1}i$ The reduced subnetwork indicating that G with X_1, X_2, ..., X_{i-1} failed (cut) but X_i working (contracted).

p_i, q_i success, failure probability of network element i.

x_i, $\bar{x}_i$ Boolean variable indicating that X_i is working, failed.

Definition

For decomposition algorithms, the following terms must be defined:

keystone set: a set of network elements chosen to decompose the network.

positive conservative policy (PCP): Given a set of r Boolean variables $\{x_1, x_2, ..., x_r\}$, the policy yielding a set of $r+1$ disjoint events $\{x_1, \bar{x}_1 x_2, ..., \bar{x}_1 \bar{x}_2 \cdots \bar{x}_{r-1} x_r, \bar{x}_1 \bar{x}_2 \cdots \bar{x}_r\}$.

decomposition events: Given the keystone set and decomposition policy, a set of events is specified to decompose the network.

NE-nodes: In the decomposition algorithm, the NE-nodes are end nodes of the links emanating from the (coalescence) source specified in the decomposition event.

In computing the DPR, mandatory nodes are the source node where the program is located, and the nodes only from which the needed files can be found. All mandatory nodes must be working because the failure of any one implies the failure of the network. Therefore,

$$\mathrm{DPR}(G) = \prod_{n_i \in M} p_{n_i} \cdot \mathrm{DPR}(G'). \qquad (1)$$

Using a set of adjacent links $\{X_1, X_2, ..., X_e\}$ emanating from the source node to be the keystone set, G' can be partitioned into a set of successive smaller subnetworks as follows:

$$\mathrm{DPR}(G') = p_1 \mathrm{DPR}(G'|1) + q_1 p_2 \mathrm{DPR}(G'|\bar{1}2) + \cdots + [\prod_{i=1}^{e-1} q_i] p_e \mathrm{DPR}(G'|\bar{1}\bar{2}\cdots\overline{e-1}e). \qquad (2)$$

The EMD events, $x_1, \bar{x}_1 x_2, ..., \bar{x}_1 \bar{x}_2 \cdots \bar{x}_{e-1} x_e$, generated based on PCP are network decomposition events.

The subnetwork $G|\bar{1}\bar{2}\cdots\overline{i-1}i$ of Eq. (2) indicates that G is reduced by a series cuts of links X_1, X_2, ..., X_{i-1} and a contraction of working link X_i. The contraction of working link X_i yields its two working end nodes collapsed into a single node named a coalescence source. The links incident on the coalescence source generates the new set of network partition events to partition the network recursively. As the recursive partition progresses, the coalescence source forms the collection set of working nodes. This implies that all links specified in the network partition event for every subnetwork are always incident on working nodes. In other words, for each working or failed link X_i in the network partition event, there always exists one end node u_i lying in the coalescence source that can be considered as perfect, and only edge e_i and the other end node v_i need to be considered in computing the reliability expression. Here, the node v_i is called NE-node for link X_i.

Fig.1(a) shows the event tree generated using Eq. (2) for a four-node DCN where P is located at the source N_6 and needs files F_1, F_2 and F_3. First, G is decomposed into $G|1$, $G|\bar{1}3$ and $G|\bar{1}\bar{3}4$ using adjacent links X_1, X_3 and X_4 from N_6. The tree edges annotated with 1, $\bar{1}3$ and $\bar{1}\bar{3}4$ represent three disjoint decomposition events; the event that X_1 is contracted, the event that X_1 is cut but X_3 is contracted and so on. For $G|\bar{1}3$, it is further decomposed into three including one success and two

failed subnetworks using three links X_2, X_4 and X_5 adjacent from the coalescence source (N_6 and N_9) that is generated by the contraction of link X_3. In Fig. 1(b), $G|\bar{1}3$ is depicted, and X_2, X_4 and X_5 specified in the three decomposition events and emanating from the coalescence source result in NE-nodes N_7 (due to X_2) and N_8 (due to X_4, X_5) respectively. Since $G|\bar{1}32$ contains a file spanning tree that program P is successfully executed, the expansion terminates and the probability term $q_1p_3p_2$ is added to DPR(G). On the other hand, $G|\bar{1}3\bar{2}4$ and $G|\bar{1}3\bar{2}\bar{4}5$ are terminated but determined to be failed since no file spanning trees are found and the coalescence sources become isolated.

The best elegant feature that all links specified in the decomposition event are always incident on a perfect (coalescence) source gives an important property as stated in the following theorem:

Theorem 1: In the decomposition algorithms, failures of nodes in every subnetwork are *s*-independent.

proof: For subnetwork $G|\bar{1}\bar{2}\cdots\overline{i-1}i$, when X_j is cut (j=1 to i-1), the edge e_j is removed and failures of end nodes u_j and v_j become *s*-dependent as:

$$p'_{v_j} \equiv \Pr\{v_j \text{success} \mid X_j \text{failed}\} = \frac{p_{v_j}(q_{u_j} + p_{u_j}q_{e_j})}{1 - p_{u_j}p_{e_j}p_{v_j}} \quad (j = 1 \text{ to } i-1).$$

Since all links specified in the decomposition event $\bar{1}\bar{2}\cdots\overline{i-1}i$ are incident on the perfect source, thus assume that NE-nodes are v_j (j=1 to i), all u_j (j=1 to i) are fused into the source and p'_{v_j} are redefined as:

$$p'_{v_j} = \frac{p_{v_j}q_{e_j}}{1 - p_{v_j}p_{e_j}} \quad (j = 1 \text{ to } i-1) \qquad (3)$$

$$p'_{v_j} = 1 \quad (j = i)$$

Eq. (3) means that the failure of node v_j depends only on the failure of edge e_j that has been removed from the subnetwork. Since no other node failures can dependent on the failure of e_j, thus failures of nodes in the subnetwork remain *s*-independent. Q. E. D.

3. The Enf Method

In this section, we will present ENF integrating the preceding stated concepts with the decomposition algorithm to find the numerical reliability considering unreliable nodes. The elegant property stated in Theorem 1 is fully exploited in ENF such that the effect of unreliable nodes in each subnetwork can easily be computed. Especially, the selection of keystone elements and the expansion of the event tree for the original decomposition algorithm remain practically unchanged by ENF.

Notation

n, l	number of nodes and links in DCN.
REL	reliability variables associated with the subnetwork.

In each node of the event tree of the decomposition algorithm, ENF performs the following steps.

1. For each link X_j specified in the decomposition event $\bar{1}\bar{2}\cdots\overline{i-1}i$, its NE-node v_j must be found first. The effect of contracting X_j yields the product $p_{e_j}p_{v_j}$ while cutting link X_j contributes the effect of the product $1 - p_{e_j}p_{v_j}$. When v_j is mandatory, the reliability effect is reduced to p_{e_j} and q_{e_j} respectively. Furthermore, p_{v_j} must be updated according to the description of the proof in Theorem 1.

2. Since failures of nodes remain *s*-independent, the reliability effect of the decomposition event is obtained by directly multiplying the effect of its contracting and cutting links stated in Step 1 without Boolean simplifications involved. The REL for each subnetwork is then updated by multiplying the derived reliability effect.

The additional cost incurred in ENF is O(nl), as finding NE-node of a specified link needs O(n) time and there are O(l) links specified in the decomposition event. Since the decomposition algorithm also spends O(nl) time to find adjacent links from the coalescence source, ENF does not raise the time complexity order for the decomposition algorithm.

For the example in Fig.1, N_6 (the source node) and N_7 (F_2 is only located at N_7) are mandatory so that $\mathrm{DPR}(G) = p_6p_7\mathrm{DPR}(G')$ where G' denotes G with N_6 and N_7 being perfect. The event tree for DPR(G') generated using ENF are shown in Fig. 2. In each nonterminal stage of the event tree, the reliability of each unreliable node must be updated and maintained to compute the effect of the decomposition event. For instance, the event $\bar{1}\bar{3}4$ gives the effect q_1 ($\overline{X_1}$), $1 - p_3p_9$ ($\overline{X_3}$) and p_4p_8 (X_4) respectively and yields the updating of p_8 and p_9 as shown in $G'|\bar{1}\bar{3}4$. Notes 1 to 4 further explain the updating of reliabilities of unreliable nodes. In Note 3, p_9 is consecutively updated twice because N_9 is NE-node for both cut links X_2 and X_3. The REL shown in each node is updated by the reliability of the decomposition event. Eventually, all REL of 'P' nodes are summed to get the DPR(G')=0.96016 and hence, DPR(G)=0.77773.

4. Expermental Results

Notation

G_i^j	benchmark network used in the experiments; subscript i represents the number of nodes in the network while superscript j denotes G with N_1 to N_j are completely connected, $j \le i$

E_s, K_s total number of generated subnetworks for ENF, KHR in computing DPR(G)

E_t, K_t overall computation time in seconds for ENF, KHR in computing DPR(G).

The efficiency of ENF combining with the decomposition algorithm (called ENF simply) is compared with algorithm KHR using a set of benchmark networks G_i^j. Fig. 3 shows an example of G_8^6. There are eight files distributed in the benchmark networks as shown in Table 1. The program is located at N_1 and files F_1, F_3 and F_5 are required to execute it.

Table 2 shows the experimental results running ENF and KHR on a SUN SPARC workstation. The efficiencies of ENF and KHR are determined by (E_s, K_s) & (E_t, K_t). The DPR is also computed assuming that each network element has equal reliability of 0.9. The ratio for generated subnetworks (K_s/E_s) ranges from 2.313 to 8.006. While for the execution time, the speed up of ENR/KW grows dramatically with the network size; as the network enlarges from G_{10}^4, G_{10}^7, G_{10}^8 to G_{10}^9, the time ratio (K_t/E_t) increases from 6.17, 82.07, 351.48 to 1725.92. ENF greatly outperforms KHR.

The major drawback of KHR is its two-pass procedure wherein all subnetworks consisting of unreliable nodes are found first and an extra reliability disjoint algorithm is then used to make them disjoint. Since the cost of disjoint process is exponential with the number of network elements (represented by Boolean variables) involved; therefore the added inputs of unreliable nodes appreciably increase the execution time of the disjoint process and decrease the overall performance of KHR. ENF generates all disjoint subnetworks in one step and directly computes the effect of node failure through a directed graph structure; thus the extra disjoint algorithm is not necessary. This shows why ENF outperforms KHR.

5. Conclusions

From the experimental results and discussions, we conclude that ENF is a feasible efficient approach in terms of less number of generated subnetworks and overall computation time compared with KHR, and thus worth considering to provide an effective reliability expression for computing DPR considering unreliable nodes.

REFERENCES

[1] A. Satyanaraya, J. H. Hagstrom, "A new algorithm for reliability analysis of multi-terminal networks", IEEE Trans. Reliability, vol. R-30, , pp. 325-323, Oct. 1981.

[2] K. K. Aggarwal and S. Rai, "Reliability evaluation in computer-communication networks", IEEE Trans. Reliability, vol. R-30, pp. 32-35, Jan. 1981.

[3] A. Grnarov and M. Gerla, "Multi-terminal reliability analysis of distributed processing systems", Porc. 1981 Int'l Conf. Parallel Processing, pp. 79-86, Aug. 1981.

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

[5] C. S. Raghavendra, V. K. P. Kumar, and S. Hariri, "Reliability analysis in distributed systems", IEEE Trans. Computers, vol. 37, pp. 352-358, March 1988.

[6] A. Kumar, S. Rai, and D. P. Agrawal, "On computer communication network reliability under program execution constraint", IEEE J. Selected Areas in Communications, vol. 6, pp. 1393-1400, Oct. 1988.

[7] D. J. Chen and T. H. Huang, "Reliability analysis of distributed systems based on a fast reliability algorithm", IEEE Trans. Parallel and Distributed Systems, vol. 42, pp. 139-154, march 1992.

[8] A. Kumar and D. P. Agrawal, "A generalized algorithm for evaluating distributed -program reliability", IEEE Trans. Reliability, vol. 42, pp. 416-426, Sep 1993.

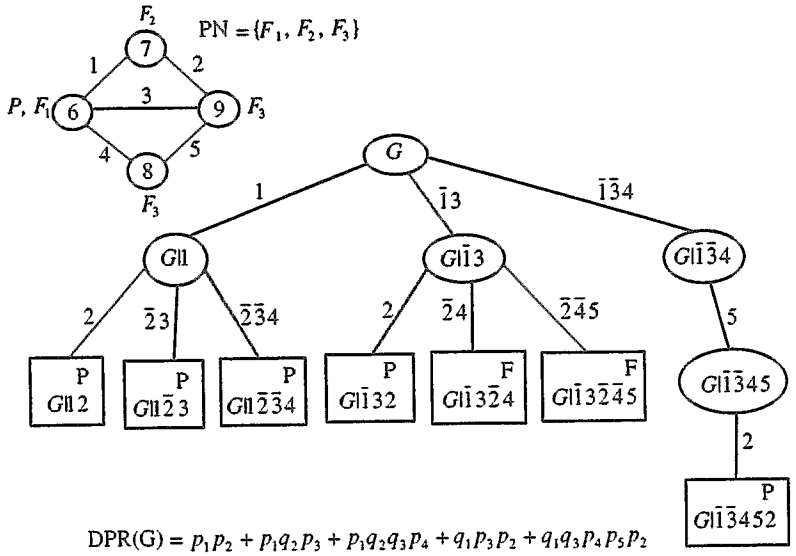
[9] D. J. Chen and M. S. Lin, "On distributed computing systems reliability analysis under program execution constraints," IEEE Trans. Comput., vol. 43, no. 1, pp. 87-97, 1994.

[10] T. Tsuchiya, Y. Kakuda, and T. Kikuno, "Three-mode failure model for reliability analysis of distributed programs," IEICE Trans. Inf. & Syst., vol. E80-D, no. 1, pp. 3-9, Jan. 1997.

[11] S. Hariri, C. S. Raghavendra, "SYREL: A symbolic reliability algorithm based on path and cut set methods", IEEE Trans. Comput., Vol. C-36, 1987 Oct., pp. 1224-1232.

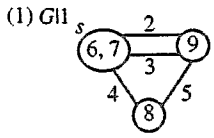
[12] K. K. Aggarwal, J. S. Gupta and K. B. Misra, "A simple method for reliability evaluation of a communication system," IEEE Trans. Communication, vol. 23, pp. 563-566, May 1975.

[13] D. Torrieri, "Calculation of node-pair reliability in large networks with unreliable nodes," IEEE Trans. Reliability, vol. 43, no. 3, , pp. 375-377, Sep. 1994.



$$DPR(G) = p_1p_2 + p_1q_2p_3 + p_1q_2q_3p_4 + q_1p_3p_2 + q_1q_3p_4p_5p_2$$

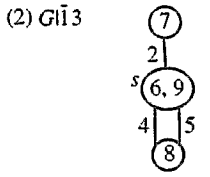
(a) Event Tree Generated Using Eq. (2)



NE-node = N_9 for event 2

NE-node = N_9 for event $\bar{2}$3

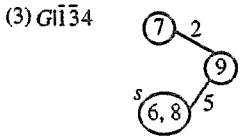
NE-node = N_9, N_8 for event $\bar{2}\bar{3}$4



NE-node = N_7 for event 2

NE-node = N_7, N_8 for event $\bar{2}$4

NE-node = N_7, N_8 for event $\bar{2}\bar{4}$5



NE-node = N_9 for event 5

(b) Some Subnetworks with NE-nodes

Fig.1. A DCN Example

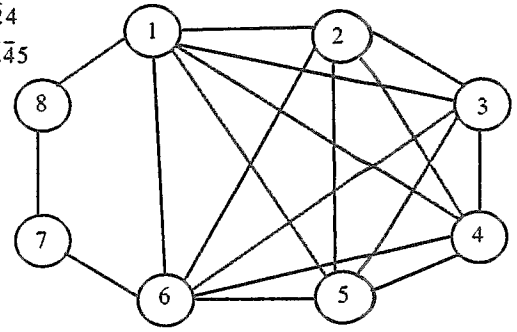


Fig. 3. The Benchmark Network G_8^6

TABLE 1. File Distribution Table

N_1	F_1, F_2, F_3	N_2	F_2, F_3, F_4	N_3	F_3, F_4, F_5	N_4	F_4, F_5, F_6	N_5	F_5, F_6, F_7
N_6	F_6, F_7, F_8	N_7	F_1, F_7, F_8	N_8	F_1, F_2, F_8	N_9	F_3, F_7, F_8	N_{10}	F_1, F_4, F_7

TABLE 2. Experimental Results for Computing the DPR

Network	K_s	E_s	K_t	E_t	DPR
G_8^4	37	16	0.030	0.003	0.891551
G_{10}^4	55	20	0.037	0.006	0.889355
G_8^6	306	72	0.412	0.012	0.898896
G_8^7	1159	289	4.279	0.061	0.899061
G_{10}^7	3443	462	9.848	0.120	0.899057
G_8^8	3225	1196	36.287	0.187	0.899090
G_{10}^8	20464	2556	230.221	0.655	0.899092
G_{10}^9	131899	17832	7999.649	4.635	0.899099

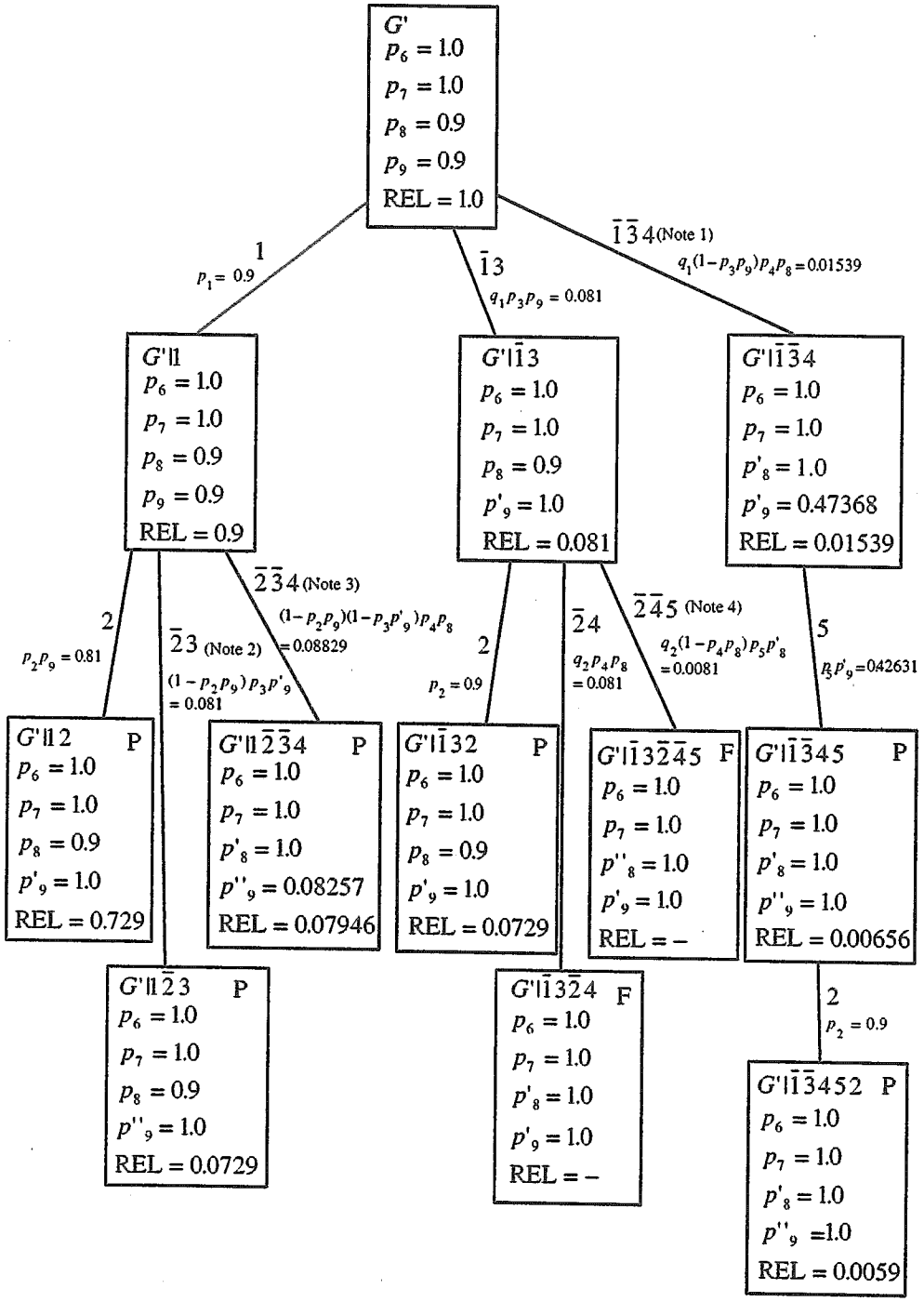


Fig. 2. Event Tree Generated by ENF

Note 1: After X_3 is cut, $p'_9 = \dfrac{p_9 q_3}{1 - p_3 p_9} = 0.47368$. Note 2: After X_2 is cut, $p'_9 = \dfrac{p_9 q_2}{1 - p_2 p_9} = 0.47368$.

Note 3: After X_2 is cut, $p'_9 = 0.47368$, and after X_3 is cut, $p''_9 = \dfrac{p'_9 q_3}{1 - p_3 p'_9} = 0.08257$.

Note 4: After X_4 is cut, $p'_8 = \dfrac{p_8 q_4}{1 - p_4 p_8} = 0.47368$.