10-2013

A collusion-resistant conditional access system for flexible-pay-per-channel pay-TV broadcasting

Zhiguo WAN

June LIU

Rui ZHANG

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1109/TMM.2013.2250493

A Probabilistic Approach to Fault Diagnosis in Linear Lightwave Networks

Robert H. Deng, Aurel A. Lazar, *Fellow, IEEE,* and Weiguo Wang

***Abstract*— The application of probabilistic reasoning to fault diagnosis in Linear Lightwave Networks (LLN's) is investigated. The LLN inference model is represented by a Bayesian network (or causal network). An inference algorithm is proposed that is capable of conducting fault diagnosis (inference) with incomplete evidence and on an interactive basis. Two belief updating algorithms are presented which are used by the inference algorithm for performing fault diagnosis. The first belief updating algorithm is a simplified version of the one proposed by Pearl for singly connected inference models. The second belief updating algorithm applies to multiply connected inference models and is more general than the first. We also introduce a t-fault diagnosis system and an adaptive diagnosis system to further reduce the computational complexity of the fault diagnosis process.**

I. Introduction

THE complexity of communication networks and the volume of information provided by these networks have caused an increase in demand for network management systems and personnel. In particular, the area of network fault management requires a great deal of network expertise (design, operation, management, etc.) which has proved to be difficult to acquire and maintain. The application of expert, or knowledge-based, systems to attack the inherent complexity of network fault management (e.g., NDS [23], YES/MVS [8], ACE [14], Troubleshooter [12], and ISM [7]) is a growing effort. However, most of the network fault management systems were built on an ad-hoc and unstructured basis. The research on network fault management is still in its infancy. There is a pressing need, therefore, for establishing a theoretical foundation of network fault management and for bridging the gap between research and working systems.

Fig. 1 presents a generic network fault diagnosis system architecture. The fault diagnosis system is data-driven and operates in real-time. It consists of four parts: the *alarm acquisition system,* the *event manager,* the *interference engine,* and the *knowledge base.* The alarm acquisition system gathers the network status information (alarm messages) from online monitors, and passes them to the event manager. The event manager filters the alarm messages according to certain criteria. The filtered messages, called evidence, are used as the input to the inference engine. The inference engine conducts fault diagnosis based on the available evidence and the knowledge base according to certain inference algorithms. The knowledge base represents the knowledge about the network under consideration.

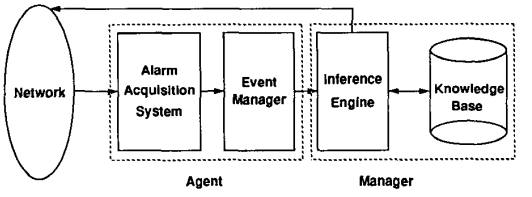


Fig. 1. A generic network fault diagnosis system architecture.

Due to the distributed nature of communication networks, the relationships between fault patterns and network events are often inexact and the conclusions are unavoidably uncertain [22]. Moreover, unavailability, loss, or delay of network messages require the fault diagnosis system to have the ability to conduct inference with incomplete information. To quote from [11], "to deal with the increasing network complexities, most existing approaches have adopted classical logic by designating each network (fault) proposition with a definite truth. However, this proves insufficient in dealing with incompleteness of network information." For reasoning with such imprecise information and relationships, it is highly desirable for the inference engine to have the capability to conduct inference with uncertainty.

Over the last few years, reasoning using probabilities has become very popular within the AI community. For introductions to this area as well as reviews on its recent developments, the reader is directed to [2] and [10]. Most recently, applications of probabilistic reasoning in network fault management have begun to appear in the literature. Network fault recognition using probabilistic data and machine learning was studied by Maxion [13]. Work on incorporating nondeterministic reasoning in managing heterogeneous network faults was investigated by Hong and Sen [11].

The Bayesian network (or causal network) model is a popular probabilistic reasoning model developed recently [18]. It has many advantages over classical rule-based models. It is iterative in nature in that after receiving new evidence, the belief about the causes of the total observed evidence can be recomputed (updated). This feature suites well the network fault diagnosis process in general. Thus, we begin our investigation with applying this model to specific network fault diagnosis problems.

Manuscript received June 1992; revised December 1992.

R. H. Deng and W. Wang are with the National University of Singapore, Singapore 0511.

A. A. Lazar is with the Department of Electrical Engineering, Columbia University, New York, NY 10027.

IEEE Log Number 9212152.

The *Linear Lightwave Network* (LLN) constitutes the physical layer of ACORN, a gigabit research network testbed ([21], [19], and [1]). It is a new type of architecture for lightwave networks based on establishing controllable transparent optical paths among network users. The LLN is based on a single key component: a controllable *linear divider/combiner* (LDC). By appropriately controlling the settings of LDC's placed at each network node, internal connections between input and output ports of an LDC can be set up and, as a result, optical path (also called routes) on which many signals are multiplexed can be created on demand. For a detailed introduction to LLN's the reader is directed to [21].

Fault diagnosis of the LLN's has been studied by Schroff and Schwartz using deterministic approaches [19]. Modeling the same problem using Bayesian networks, a special class of Bayesian networks is obtained on which the belief updating algorithm can be greatly simplified. The resulting fault diagnosis algorithm improves that of Schroff and Schwartz in that it allows multiple simultaneous faults in the network. The work cited allows only for single faults.

The organization of this paper is as follows. In Section II, we illustrate how to derive the inference model (i.e., the Bayesian network) for fault diagnosis from a given LLN. In Section III, we give an inference algorithm that is performed by the inference engine during the fault diagnosis process. Belief updating algorithms are discussed and presented in Section IV. To further simplify the belief function computation, we propose in Section V the concepts of t-fault diagnosis system and adaptive fault diagnosis system. Finally, Section VI contains our concluding remarks.

II. The Inference Model

The inference model is a Bayesian network, i.e., a *directed acyclic graph* (DAG) where the nodes are random variables and certain independence assumptions hold [18], [15], [2]. In this section, we illustrate, by using a simple example, how a Bayesian network is constructed from a given LLN.

A simple LLN with five LDC's is shown in Fig. 2. Each LDC has a number of input and output ports and the LDC's are interconnected by optical links. In Fig. 2, the symbols i_{jk} and o_{jl} denote, respectively, the kth input port and the lth output port of LDC j. The arrows represent unidirectional fiber links that connect output ports to input ports. The label F_{ip} associated with the ith fiber link denotes the optical output power value at the output port that it is connected to. For example, F_{2p} is the power value at port o_{21}. Finally, I_{ip} and O_{kp} denote the input and output power values from/to the LLN periphery, respectively.

We make the following assumptions in our model:

1. The observable messages are the LDC input and output power values. Faults are detected when the observed values deviate from the expected range of power values. We assume to have reliable information about the range of power values under normal operation of each LDC in the LLN. Once a power value falls outside its specified operating range, a fault is said to have occurred.

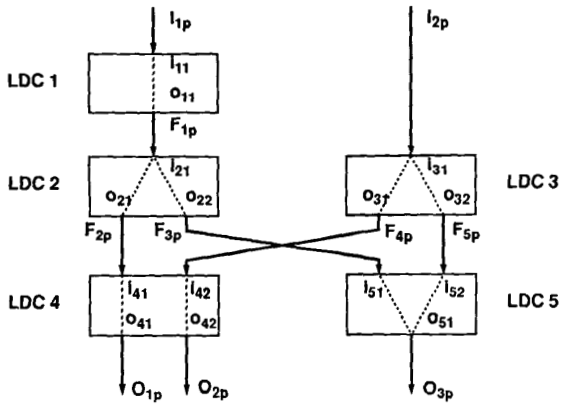


Fig. 2. An example of an LLN.

TABLE I
LLN Routing Table

Route 1	i_{11}	o_{11}	i_{21}	o_{21}	i_{41}	o_{41}
Route 2	i_{11}	o_{11}	i_{21}	o_{22}	i_{51}	o_{51}
Route 3	i_{31}	o_{31}	i_{42}	o_{42}		
Route 4	i_{31}	o_{32}	i_{52}	o_{51}		

2. The basic network components (BNC's) are LDC's. That is, LDC's are treated as black boxes and are the elementary units (atoms) for the network fault manager. If an LDC has failed or some part of it has failed, all its output power values will be incorrect (outside the specified range) but its input power values remain unaffected.
3. We assume that the optical links between LDC's are not subject to failure. This is not really a restriction because we can model a link as a one input/one output "LDC" which is subject to failure. Thus, the set of BNC's also includes the optical links.
4. The network routes are considered to be quasistatic. That is, the network configuration changes very slowly—connections that are established remain up for a long time when compared with the time required to carry out the fault diagnosis.

The LLN routes/connections determine the internal connections between the input and the output ports of every LDC. Thus, the routing information is required by the fault diagnosis system for constructing the inference model. Assuming the routing table as given in Table I, the internal connections of each LDC shown in Fig. 2 by dotted lines. Hence, Fig. 2 summarizes the network configuration, which together with the four assumptions constitutes the knowledge about the LLN. The inference model is constructed based on this knowledge.

Since a power value, say F_{ip}, is assumed to be in one of the two states, within the expected range or outside it, we define a binary-valued random variable (RV) F_i for power value F_{ip}:

$$F_i = \begin{cases} 1 & \text{if the power value } F_{ip} \text{ is within the expected range,} \\ 0 & \text{if the power value } F_{ip} \text{ is not within the expected range.} \end{cases} \quad (1)$$

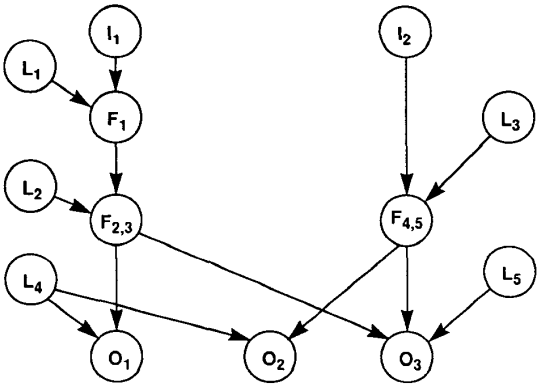


Fig. 3. The Bayesian network for the LLN.

Similarly, for each LDC we define a binary-valued RV

$$L_j = \begin{cases} 1 & \text{if LDC } j \text{ is under normal operation,} \\ 0 & \text{if LDC } j \text{ has failed.} \end{cases} \quad (2)$$

Furthermore, we assume that the L_j's are mutually independent RV's with probability distribution

$$P(L_j = 0) = p_j, P(L_j = 1) = 1 - p_j. \quad (3)$$

With these RV's so defined, based on Fig. 2 and the assumptions 1, 2, and 3, the following dependence relations among the RV's are obvious:

$$\begin{aligned} &F_1 = I_1 \wedge L_1, \quad F_2 = F_3 = F_1 \wedge L_2, \\ &F_4 = F_5 = I_2 \wedge L_3, \quad O_1 = F_2 \wedge L_4, \\ &O_2 = F_4 \wedge F_4, \quad O_3 = F_3 \wedge F_5 \wedge L_5 \end{aligned} \quad (4)$$

where $\wedge$ denotes the logical "AND" operation. The dependence relations in (4) can be graphically represented as a DAG, in which each node represents a RV and the arcs indicate the dependence relations (Fig. 3) such that a node representing an RV on the left-hand side of an equation in (4) is pointed by arcs from the nodes representing the RV's on the right-hand side of the same equation. Note that since RV's F_2 and F_3 always have the same value according to (4), they are drawn in the same node for simplicity. This is similar to F_4 and F_5. In general, for a non-root RV X in the DAG of Fig. 3, X and its parents $U_1, \cdots, U_n$ are related by:

$$X = U_1 \wedge U_2 \wedge \cdots \wedge U_n. \quad (5)$$

The conditional probability $P(x|u_1, \cdots, u_n)$ (for a RV X, we denote its value by the lower case letter x) is then derived as:

$$P(x|u_1, \cdots, u_n) = \begin{cases} 1 & \text{if } x = u_1 \wedge \cdots \wedge u_n \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

which is the probabilistic representation of the dependence relation given in (5).

The DAG in Fig. 3 and the probability distributions of (3) and (6) constitute the inference model for the LLN of Fig. 2. This inference model is a Bayesian network [18], [15]. It captures the knowledge required for LLN fault diagnosis; hence, it is the knowledge base to be consulted by the inference engine in carrying out fault diagnosis.

From this example, it is clear that the construction of the Bayesian network inference model from an arbitrary LLN with routing information is straightforward and can be easily automated.

III. The Inference Algorithm

The RV's in the inference model of an LLN (see Fig. 3) can be classified into two types: *observable* and *unobservable*. The power-related RV's such as F_1, I_1, O_1 belong to the first type, since their values can be determined from measuring the corresponding power values. On the other hand, LDC-related RV's such as L_j are of the second type because we have treated an LDC as a black box. The task of the LLN fault diagnosis system is to infer, based on the values (i.e., instantiations) of the observable RV's, the values of the unobservable RV's.

Referring now to Fig. 1, the fault diagnosis process for LLN's can be briefly summarized as follows: network monitors periodically measure the values of optical powers among other things and generate messages to the management agent. The messages will be filtered by the management agent. Only those messages which indicate possible network faults are passed to the inference engine. The set of messages received at the inference engine is called *evidence* for network faults. Based on this evidence and the inference model, the *inference engine* computes its belief that an LDC has failed using the inference algorithm to be presented in this section.

If the fault diagnosis system were supplied with power measurements at all the LDC's, fault diagnosis would be a trivial process. However, power measurements, or evidence gathering, is usually an expensive process. To minimize the cost in evidence gathering, we assume that power measurements are only performed at the periphery of the LLN under normal network operation; power measurements at other input and output ports are conducted only if additional evidence is needed in order to complete the fault diagnosis process. This suggests that the inference algorithm must have the following two features:

1. It should be able to conduct inference with incomplete evidence and be able to identify the faulty LDC or a plausible set of faulty LDC's.
2. It should perform inference on an interactive basis. Initially, it should identify a plausible subset based on the currently available evidence. Then, with this subset as a guide, it should decide what additional evidence needs to be acquired. With the new evidence, it should narrow down the possible set of faulty LDC's and finally pinpoint the faulty ones.

With respect to the LLN inference model, the evidence is a set of instantiated observable RV's, $W = \{W_1 = w_1, \cdots, W_z = W_z\}$. Relating to our earlier example, the W_j's can be either F, I, or O variables. The instantiation is derived from power measurements. For example, if measurement indicates that the power value W_{ip} is incorrect, then $W = \{\cdots, W_i = 0, \cdots\}$. The belief function of an LDC-related RV

L_j, is defined as:

$$b(L_j = l_j) = P(L_j = l_j|W) \qquad \text{for } l_j = 0, 1. \qquad (7)$$

The belief function tells us the likelihood that the RV L_j takes value 0 or 1. Therefore, $b(L_j = 0)$ is the likelihood that LDC j has failed given evidence W. We illustrate the previously mentioned concepts with a fault diagnosis example for the LLN in Fig. 2.

Example 1: For the LLN in Fig. 2, suppose that LDC 1 and LDC 5 have failed simultaneously. We also assume that the LLN input power I_{1p} and I_{2p} are both correct; otherwise, fault sources will be in the transmitters which are outside of the LLN. Due to the failures of LDC's 1 and 5, the observed LLN output power values O_{1p} and O_{3p} will be both incorrect and O_{2p} will be correct. Therefore, with respect to the reference model in Fig. 3, the evidence is $W = \{I_1 = 1, I_2 = 1, O_1 = 0, O_2 = 1, O_3 = 0\}$. The job of the inference engine is to locate the fault sources, i.e., $L_1 = L_5 = 0$, based on the evidence W. This is achieved in an iterative fashion.

Since $O_2 = 1$, all the ancestors of O_2, i.e., $L_4, F_{4,5}, L_3$, and I_2, must also have value 1. Therefore, O_2 and all its ancestors can be eliminated from the inference model in Fig. 3. Similarly, I_1 can be deleted. For the purpose of fault diagnosis, therefore, we need only to work with the trimmed inference model shown in Fig. 4(a).

Iteration 1: The evidence in Fig. 4(a) is $W = \{O_1 = 0, O_3 = 0\}$. Through direct application of the Bayes formula, we obtain:

$$b(L_1 = 0) = \frac{p_1}{p_1 + p_2 - p_1 p_2},$$
$$b(L_2 = 0) = \frac{p_2}{p_1 + p_2 - p_1 p_2}, \quad \text{and}$$
$$b(L_5 = 0) = p_5.$$

For simplicity, assume that $p_i = p$ for all i. Then, $b(L_1 = 0)$ and $b(L_2 = 0)$ achieve the maximum of all belief functions. In other words, LDC's 1 and 2 are the most probable failed LDC's which have caused the evidence W. We arbitrarily select LDC 2 for further probing. Measuring the input and output power values at LDC 2 will reveal that they are all incorrect. This corresponds to $F_1 = F_{2,3} = 0$ in Fig. 4(a). This measurement does not tell us directly anything about the status of LDC 2.

Iteration 2: Now the evidence available to us is updated to $W = \{F_1 = 0, F_{2,3} = 0, O_1 = 0, O_3 = 0\}$. For this simple model, we immediately recognize that $L_1 = 0$, i.e., LCD 1 has failed. However, to illustrate the diagnosis process, we proceed with the belief function calculation based on the updated evidence. Simple calculations using Bayes formula show that $b(L_1 = 0) = 1, b(L_2 = 0) = p$, and $b(L_5 = 0) = p$. Therefore, we are certain that LDC 1 has failed.

Iteration 3: With LDC 1 being repaired, the updated evidence becomes $W = \{L_1 = 1, O_1 = 1, O_3 = 0\}$.Since $O_1 = 1, O_1$ and all its ancestors can be removed from Fig. 4(a). The resulting inference model is now as shown in Fig. 4(b). The evidence in Fig. 4(b) is $W = \{O_3 = 0\}$. The belief function calculated based on Fig. 4(b) and the evidence W is $b(L_5 = 0) = 1$. Hence, LDC 5 has failed. After LDC 5 is repaired, power measurements at the LLN output will be all correct. Therefore, no more faults are detected. □

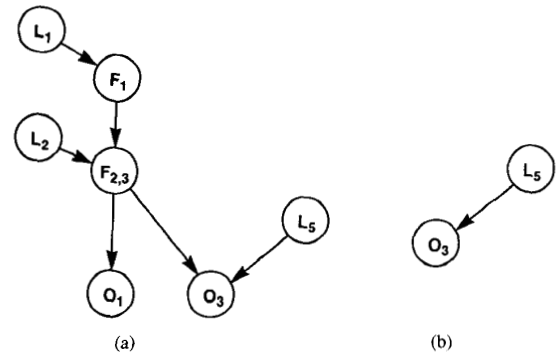


Fig. 4. Trimmed Bayesian network.

The previous example shows one important step in the LLN fault diagnosis process—trimming of the inference model—an operation that can significantly simplify belief function calculations. Given the evidence, $W = \{W_1 = w_1, \cdots, W_z = w_z\}$ if $w_i = 1$; then, the RV W_i and all its ancestors can be deleted from the inference model. The inference engine needs only to conduct its inference on the trimmed inference model. We are now ready to present the inference algorithm for LLN fault diagnosis.

Algorithm 1: (*LLN fault diagnosis algorithm*)

Step 1: Read the input and output power values at the periphery of the LLN:

- If all the input power values are correct but some output power values are incorrect, then go to step 2, otherwise exit.

Step 2: Let W denote the evidence obtained from Step 1. Trim the inference model if possible. Compute $b(L_j = 0)$, for all j, based on some belief updating algorithm (to be described in more detail later). Find the L_k with the largest belief. If $b(L_k = 0) = 1$, then LDC k has failed. Go to step 5; otherwise, go to step 3.

Step 3: Measure the input and output power values of LDC k:

- If some or all of the output power values are correct, LDC k is not faulty; clear "uncertain" marks if any; go to step 4.
- If all the input power values are correct and some output power values are incorrect, then LDC k has failed. Go to step 5.
- If some input and all output power values are incorrect, mark RV L_k "uncertain"; go to step 4.

Step 4: Add to W the evidence obtained from the power measurement of LDC k in step 3. Trim the inference model if possible. Compute $b(L_j = 0)$ for all j. Find the RV, say L_i, with the largest belief and *not* marked "uncertain." Let $k = i$; go to step 3.

Step 5: Repair LDC k. Clear all "uncertain" marks. Go to Step 1. □

Algorithm 1 did not include the belief updating part. This, however, will be the center of discussion for the rest of the

paper. To see that the algorithm always converges, note that the number of "uncertain" marks is strictly increasing until a faulty LDC is repaired or a good LDC is identified. By the definition of step 3, an LDC is marked uncertain if and only if it has an faulty ancestor. Since the network is finite and acyclic, there is always an LDC that does not have any faulty ancestor. Therefore, step 4 will always be successful. A faulty LDC without faulty ancestor will be picked up in step 4 and repaired. This follows the argument of the bound on the number of power tests needed for the algorithm to locate the repair all faults. Assuming there are N LCD's, the number of "uncertain" marks can increase to at most N, and there are at most N resets to the counting. So, the total number of power tests is bounded by $O(N^2)$.

IV. Belief Updating Algorithms

In this section, general belief updating algorithms will be discussed. In Section IV-A, Pearl's belief updating algorithm and its limitations are briefly mentioned. For the benefit of the reader, the algorithm is presented in the Appendix. In Section IV-B, belief updating for multiply connected inference models is described. A new belief updating algorithm for multiply connected LLN inference models is presented in Section IV-C.

A. Belief Updating for Singly Connected Inference Models

Since our inference model for fault diagnosis of LLN's is a Bayesian network, the core operation in the diagnosis algorithm presented in the last section is the belief function computation also known as belief updating. Belief updating for general Bayesian networks has been proven to be NP-hard [4]. For singly connected Bayesian networks (namely, networks with no more than one path between any two RV's), the belief updating can be done in polynomial time [18], [15]. In the Appendix, we summarize Pearl's belief updating algorithm for singly connected Bayesian networks [18], [15] and call it Algorithm 2.

Pearl's algorithm deals directly with the conditional probability $P(X|U_1, \cdots, U_n)$. For the LLN inference model, all RV's are binary-valued and the conditional probability is given simply by (6). For such a special case, the computations in the belief updating algorithm can be simplified considerably. Specifically, it is straightforward to show that (31) and (32) (see the Appendix) can be reduced to:

$$\begin{aligned} \pi(X=1) &= \prod_i \pi_X(U_i=1), \\ \pi(X=0) &= 1-\pi(X=1), \\ &\text{if } X \text{ is not instantiated,} \end{aligned} \tag{8}$$

and

$$\begin{aligned} \lambda_x(U_i=0) &= \lambda(X=0), \\ \lambda_x(U_i=1) &= \lambda(X=1)\prod_{k\neq i}\pi_X(U_k=1) \\ &\quad + \lambda(X=0)\left(1-\prod_{k\neq i}\pi_X(U_k=1)\right) \end{aligned} \tag{9}$$

respectively.

B. Belief Updating for Multiply Connected Inference Models

Pearl's belief updating algorithm described in the Appendix cannot be applied to multiply connected Bayesian networks. A multiply connected Bayesian network is evidenced by the presence of loops. Here, loops are defined as undirected cycles in the underlying network. The inference model in Fig. 3 is multiple connected. There is one loop in this model, formed by the nodes $F_{2,3}, O_1, L_4, O_2, F_{4,5}$, and O_3.

There are three general approaches of belief updating for multiply connected Bayesian networks: *clustering, conditioning,* and *simulation* [18], [15]. Clustering involves forming compound nodes in such a way that the resulting network of clusters is singly connected. For example, the model in Fig. 3 becomes singly connected if we cluster the nodes $F_{2,3}, O_1, L_4, O_2$, and $F_{4,5}$ into a compound node. Every Bayesian network can be structured as singly connected if we do not limit the size of the clusters. Unfortunately, in the LLN case, the sizes of the clusters can be quite large, and the structureless nature of the compound node makes it difficult to compute, much less explain, the belief functions with the clustered nodes. For this reason, clustering does not appear to be a feasible approach for the LLN.

The simulation technique provides an approximate solution to the belief functions. It uses Monte Carlo techniques to estimate probabilities by counting how frequently events occur in a series of simulation runs. The first simulation method in this contest, called logic sampling, was proposed by Henrion [9]. In logic sampling, simulation starts at the root nodes and ends at leaf nodes. A value is assigned to each root node based on its prior probability. Once root nodes have been assigned values, their children are assigned values based on the conditional probabilities relating the root nodes to their children. This process of setting values of the children based on the values of their parents and the related conditional probabilities continues until all the nodes in the network have been assigned values. The assignment of a value to each node in the network constitutes one simulation cycle.

Logic sampling applies to any Bayesian network; however, it does not permit evidence nodes to be clamped to their known values, unless they just happen to be root nodes. Since the simulation proceeds in a top-down fashion, there is no way to account for nonroot node evidence known to have occurred until the nodes corresponding to these evidences are sampled. If the assigned values match the evidence, the simulation cycle is then counted as one positive instance of the simulation; otherwise, it must be discarded. To get a reasonable approximation to the belief function, the process is repeated until enough positive instances are generated to be statistically significant. In those cases where there are a large number of instantiated nodes, many repetitions may be needed to produce just one positive instance [9].

The stochastic simulation method proposed by Pearl [17] accounts for the evidence in the sampling process. It permanently clamps the evidence nodes to the values observed and then conducts the stochastic simulation on the clamped network. Although the stochastic simulation method seems to work well with Bayesian networks having nodes that are not highly

dependent on each other, it is prone to convergence problems when the network contains links that are nearly deterministic. Therefore, it cannot be applied to the LLN inference model where the links are deterministic.

Chin and Cooper [3] have proposed logic sampling with evidential integration, which employs link reversal to convert nonroot evidence nodes to root nodes. However, this does not seem to be a general approach since the link reversal process is liable to combinatorial problems. More recently, other simulation approaches have been investigated, such as the likelihood weighting technique by Fung and Chang [6] and the importance sampling technique by Shachter and Peot [20]. While the convergence properties of these two approaches seem hard to predict, their numerical results suggest that they work well even for networks with near 0 and 1 probabilities values. The applicability of the latter two techniques to the LLN model remains an open problem.

The third method, conditioning, involves breaking the loops by instantiating a selected set of nodes to render the network singly connected so that Pearl's belief updating algorithm can be applied [18], [15]. In Fig. 3, for example, instantiating L_4 will render the rest of the network singly connected. Given the evidence W, we instantiate L_4 for each of its values. We then propagate the evidence W, using Pearl's algorithm, under each of the instantiations of L_4. The belief functions, say $b(L_3)$ of L_3, can be obtained by:

$$b(L_3) = b(L_3|L_4 = 0)P(L_4 = 0|W) + b(L_3|L_4 = 1)P(L_4 = 1|W) \quad (10)$$

where $b(L_3|L_4 = 0)$ and $b(L_3|L_4 = 1)$ are the belief functions obtained by instantiating L_4 to 0 and 1, respectively. Using Bayes rule, we have:

$$P(L_4|W) = \alpha P(W|L_4)P(L_4) \quad (11)$$

where α is a normalizing constant, $P(L_4)$ is the *a priori* probability of L_4, and $P(W|L_4)$ can be obtained by propagating all instantiations of L_4 through a single connected network.

In the LLN inference model, the nodes are all binary-valued. If a model contains K loops, one node in each loop needs to be instantiated. Therefore, the amount of computation is approximately equal to the execution of Pearl's algorithm for 2^K singly connected networks. If the number of loops is not too large, conditioning would be a possible approach for the LLN application.

C. A Belief Updating Algorithm for LLN's

In what follows, we present a belief updating algorithm for multiply connected LLN inference models. The computational complexity of the algorithm is $O(N^Z)$ in the worst case, where N is the number of LDC RV's, i.e., those corresponding to LDC's, in the trimmed inference model, and Z is the number of RV's instantiated to value 0 in the evidence.

Let $W = \{W_1 = 0, \cdots, W_Z = 0\}$ be the evidence in the trimmed inference model. For each LDC RV L_j, we denote the event $L_j = 0$ by e_j. From (3), we have that:

$$P(e_j) = p_j \quad (12)$$

and these events are mutually independent.

For $i = 1, \cdots, Z$, define sets:

$$E_i = \{e_j | L_j \text{ has a path to } W_i \text{ in the inference mode}\}. \quad (13)$$

Note that sets E_i are all nonempty since there must be some cause for the faulty power outputs at W_i. Moreover, any single event in E_i alone can cause the instantiation of W_i to 0. In other words, the disjunction of the events in the set is logically equivalent to $W_i = 0$. Depending on the context, symbol E_i will denote the set as defined in (13) or the disjunction of the events in the set. Therefore, the conjunction of E_i for $i = 1, 2, \cdots, Z$:

$$V = E_1 \wedge E_2 \wedge \cdots \wedge E_Z \quad (14)$$

is logically equivalent to the total observed evidence W.

Using the laws of Boolean algebra, we can rewrite (14) in the irreducible disjunctive normal form (DNF), i.e., in a disjunction of conjunctions of basic events in which each disjunct contains no identical basic events and no disjunct is a subset of another.

The DNF form of V represents the precise set of all joint events, each of which *alone* can cause the observed evidence W. Due to the logical equivalence between W and V, the belief function of event e_j, i.e., $L_j = 0$, given W can be expressed as:

$$b(e_j) = \frac{P(V \wedge e_j)}{P(V)}. \quad (15)$$

The computation of $P(V \wedge e_j)$ and $P(V)$ is straightforward by the laws of probability [16].

Example 2: Consider the trimmed inference model of Fig. 4(a) with evidence $W = \{O_1 = 0, O_3 = 0\}$. The sets E_1 and E_3 are as follows:

$$E_1 = \{e_1, e_2\}, E_3 = \{e_1, e_2, e_5\}. \quad (16)$$

From (14), we have:

$$V = (e_1 \vee e_2) \vee (e_1 \vee e_2 \vee e_5) \quad (17)$$

and the irreducible DNF is:

$$V = e_1 \vee e_2. \quad (18)$$

Using (15), we obtain:

$$b(e_1) = \frac{P(V \wedge e_1)}{P(V)} = \frac{P((e_1 \vee e_2) \wedge e_1)}{P(e_1 \vee e_2)} = \frac{P(e_1)}{P(e_1 \vee e_2)} = \frac{p_1}{p_1 + p_2 - p_1 p_2} \quad (19)$$

and, similarly,

$$b(e_2) = \frac{p_2}{p_1 + p_2 - p_1 p_2}, \qquad b(e_5) = p_5. \quad (20)$$

□

The belief updating algorithm to be presented consists of $Z + 1$ steps. Step 0 finds the sets $E_i, i = 1, \cdots, Z$; steps 1 to $Z - 1$ compute the DNF form of V iteratively; and the last step computes the belief function. The proof of the correctness of steps 1 to $Z - 1$ is straightforward but

tedious; therefore, it is omitted. In what follows, we will use $A \cap B, A \cup B, A - B$ to denote set operations, and $A \wedge B, A \vee B$ to denote Boolean operations. We will treat sets of (possibly joint) events interchangeably with Boolean formulas in the following way: a set is converted to a formula by taking the Boolean OR of its elements, and a formula in DNF form is converted to a set by treating the disjuncts as elements of the set.

Algorithm 3. (*Belief Updating Algorithm for LLN's.*)

Step 0: Given the inference model and the evidence $W = \{W_1 = 0, \cdots, W_z = 0\}$, compute $E_i, i = 1, \cdots, Z$ by finding all ancestors of W_i that are LDC RV's.

Step 1: Find the intersection of E_1 and $E_2, S_1 = E_1 \cap E_2$, let

$$V_2 = S_1 \cup [(E_1 - S_1) \wedge (E_2 - S_1)]. \tag{21}$$

Specifically, the right-hand side of (21) is computed as follows: remove from E_1 and E_2 the events in S_1, and treat the resulting sets as disjunctions. Then, apply the distributive law to obtain a DNF. The resulting disjunction is again treated as a set and unioned with S_1. Note that if either $E_1 - S_1$ or $E_2 - S_1$ is empty, then $V_2 = S_1$ and, if S_1 is empty, then $V_2 = E_1 \wedge E_2$.

Step i(for $1 < i < Z - 1$):

1. Find the *generalized intersection* of V_i and E_{i+1}:

$$S_i = E_{i+1} \otimes V_i = \{v \in V_i : \exists e \in E_{i+1}, v \text{ contains } e \text{ as a basic event}\}. \tag{22}$$

   In other words, S_i contains those (possibly joint) events in V_i that have some events in E_{i+1} as basic event.
2. In a similar fashion to step 1, let

$$V_{i+1} = S_i \cup [(E_{i+1} - S_i) \wedge (V_i - S_i)]. \tag{23}$$

3. As we shall see later, the set V_{i+1} thus obtained may not be in the irreducible DNF form because some of its joint events might contain some other joint events in it. The former should be eliminated. This can be done by arranging the joint events in V_{i+1} in increasing cardinality order and then scan from the smallest upwards checking for proper containment. The resulting set is V_{i+1} in the irreducible DNF form.

Step Z:

Let $V = V_Z$. Compute $P(V), P(V \wedge e_j)$ and $b(e_j)$ for $j = 1, \cdots, N$ by using (15).

□

Note that step $i(3)$ is necessary because step $i(2)$ does not necessarily return V_{i+1} in irreducible DNF form. Here is an example: Let $V_2 = \{e_1 \wedge e_2, e_2 \wedge e_3, e_1 \wedge e_3\}, E_3 = \{e_1\}$. (It is easy to see that this can be the case for some LLN). Now, by (22), $S_2 = \{e_1 \wedge e_2, e_1 \wedge e_3\}$ and, by (23), $V_3 = \{e_1 \wedge e_2, e_1 \wedge e_3, e_1 \wedge e_2 \wedge e_3\}$ and the joint event $e_1 \wedge e_2 \wedge e_3$ is redundant.

If an element in V_i is a joint event of k basic events, we say that the element has multiplicity k. In particular, if $k = 1$, it is called a singleton. Note that if the elements in V_i are all singletons, then the generalized intersection defined in (22) reduces to the ordinary intersection.

Let us estimate the worst-case time complexity of this algorithm. Recall that we have N LDC nodes and Z output ports whose power measurements are incorrect. In step 0, to compute each E_i, it takes at most $O(N^2)$ amount of time to traverse the DAG. Thus, this step takes at most $O(ZN^2)$ amount of time. In step i, all three subsets take time proportional to the size V_i and E_{i+1}. The former is bounded by $O(N^i)$, and the latter by $O(N)$. Thus, in total, steps 1 through $Z - 1$ take at most $O(ZN^Z)$ amount of time. Step Z takes time proportional to the size V_Z which is bounded by $O(ZN^Z)$. Therefore, in the worst case the entire algorithm takes $O(ZN^Z)$ amount of time. This bound is not very good but, as we will see in the next section, this method will allow us to come up with a better average case algorithm.

V. The t-Fault Diagnosis System and Adaptive Fault Diagnosis System

A fault diagnosis system employing Algorithm 1 in conjunction with Algorithm 2 or 3 is capable of identifying any number of faults in the LLN. Hence, we call such systems general fault diagnosis systems. In a practical LLN, simultaneous failure of a large number of LDC's is very unlikely. Consider, for example, an LLN with 100 LDC's. Assuming that the *a priori* probability of LDC failure is $p = 0.001$, the probabilities of one, two, and three LDC failures are $9 \times 10^{-2}, 4.5 \times 10^{-3}$, and 1.45×10^{-4}, respectively. Therefore, for practical purposes, it is worthwhile to consider fault diagnosis systems with more limited fault diagnosis capabilities. In this section, we introduce the concept of t-fault identification capability for an inference system, and show that the computational complexity of a t-fault identification algorithm is on the order of $O(ZN^t)$ in the worst case.

A fault diagnosis system is said to have a t-fault diagnosis capability if it is designed to identify all fault patterns with t or fewer LDC failures. We call such systems t-fault diagnosis systems. We observe that if all the fault patterns have t or fewer faults, then the corresponding element in the irreducible set V will be of multiplicity at most t. The t-fault diagnosis system conducts its inference using Algorithm 1 and a modified version of Algorithm 3.

The basic idea of the modified Algorithm 3 is as follows. Step 0 remains the same as in Algorithm 3. In step $i(2)$, when forming set V_{i+1} we discard all those joint events that have *more than* t basic events. We denote the resulting set of step $i(3)$ by V_{i+1}^t. As a result, the final V^t as well as all intermediate V_i^t's will contain only joint events of multiplicity less than or equal to t. Step Z also remains the same as in Algorithm 3:

$$b(t, e_j) = \frac{P(V^t \wedge e_j)}{P(V^t)}. \tag{24}$$

Following the same complexity analysis of Algorithm 3, we can see that the worst-case time complexity of the t-fault

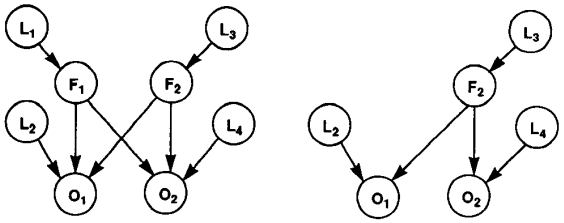


Fig. 5. The LLN inference model for Example 3.

diagnosis algorithm is bounded by $O(ZN^t)$. Note that this is a significant reduction in time complexity since it is now a polynomial in the size of the LLN.

During the steps $i = 1, 2, \cdots, Z-1$, set V_i^t might become empty. This means that no t or less faults can cause the given evidence. In other words, the actual number of faults must be larger than t, and such fault combinations cannot be identified by the t-fault diagnosis algorithm. Therefore, there is a tradeoff between the fault diagnosis capability and the computational complexity. As a special case, when $t = 1$, finding V^1 is particularly simple:

$$V^1 = E_1 \cap E_2 \cap \cdots \cap E_Z. \tag{25}$$

Note that $\cap$ denotes the set intersection operation instead of the Boolean "and." It can be shown that if the *a priori* probabilities p_i are all identical, the 1-fault diagnosis system becomes the deterministic fault diagnosis system presented in [19].

Example 3: Consider an LLN with its inference model as shown in Fig. 5(a). Assume that the LLN input power values are all correct and that LDC's 1 and 3 have failed. Then, the evidence for this inference model will be $W = \{I_1 = 1, I_2 = 1, O_1 = 0, O_2 = 0\}$. The nodes I_1 and I_2 have been deleted from the model since their values are 1.

a) Assuming that the 1-fault diagnosis system is used, we proceed as follows:

Iteration 1: The evidence for the trimmed inference model is $W = \{O_1 = 0, O_2 = 0\}$. We have $E_1 = \{e_1, e_2, e_3\}$ and $E_2 = \{e_1, e_3, e_4\}$. Since $V^1 = \{e_1, e_3\}, V^1 \wedge e_1 = \{e_1\}, V^1 \wedge e_2 = \{e_1 \wedge e_2, e_3 \wedge e_2\}, V^1 \wedge e_3 = \{e_3\}$, and $V^1 \wedge e_4 = \{e_1 \wedge e_4, e_3 \wedge e_4\}$, we have $b(1, e_1) \approx p_1/(p_1 + p_3), b(1, e_2) = p_2, b(1, e_3) \approx p_3/p_1 + p_3$, and $b(1, n_4) = p_4$. Assuming that $p_i = p$ for all i, then $b(1, e_1)$ and $b(1, e_3)$ are the maximum. If LDC 1 is selected for power measurement, we will find that it has failed and will need to be repaired.

Iteration 2: The updated evidence is $W = \{O_1 = 0, O_2 = 0, F_1 = 1\}$. After deleting node F_1 and its ancestor L_1, the trimmed inference mode is shown in Fig. 5(b). The evidence corresponding to Fig. 5(b) is $W = \{O_1 = 0, O_2 = 0\}$. We have $E_1 = \{e_2, e_3\}$ and $E_2 = \{e_3, e_4\}$. Thus, $V^1 = \{e_3\}, V^1 \wedge e_2 = \{e_3 \wedge e_2\}, V^1 \wedge e_3 = \{e_3\}$, and $V^1 \wedge e_4 = \{e_3 \wedge e_4\}$. Then, $b(1, e_2) = p_2, b(1, e_3) = 1$, and $b(1, e_4) = p_4$. After LDC 3 is fixed, the LLN output levels will be all correct and, hence, no further faults are detected.

b) Assuming that the general fault diagnosis system is used, we proceeds as follows:

Iteration 1: E_1 and E_2 will be the same as in iteration 1 of a). Following Algorithm 3, we have $V = \{e_1, e_3, e_2 \wedge e_4\}, V \wedge e_1 = \{e_1\}, V \wedge e_2 = \{e_1 \wedge e_2, e_3 \wedge e_2, e_2 \wedge e_4\}, V \wedge e_3 = \{e_3\}$, and $V \wedge e_4 = \{e_1 \wedge e_4, e_2 \wedge e_4, e_3 \wedge e_4\}$. Then, $b(e_1) = p_1/(p_1 + p_3 + p_2p_4), b(e_2) = (p_1 + p_3 + p_4)p_2/(p_1 + p_3 + p_2p_4), b(e_3) = p_3/(p_1 + p_3 + p_2p_4)$, and $b(e_4) = (p_1 + p_2 + p_3)p_4/(p_1 + p_3 + p_2p_4)$. (This, again, assumes that $p_i = p$ for all i. The $b(e_1)$ and $b(e_3)$ are the maximum. If LDC 1 is selected, we will find that it has failed and have it repaired.

Iteration 2: The same as in a). □

Note that the 1-fault diagnosis system corrects the two faults for this LLN. Also, note that the belief functions $b(1, e_j)$ and $b(e_j)$ are generally different due to the omission of larger joint events in the modified version of Algorithm 3. If the *a priori* probabilities $p_i \ll 1$, the effects due to the difference between the two sets of belief functions on the inference process become negligible.

To take advantage of the low computational complexity for small t's, we propose the adaptive inference algorithm that runs the modified version of Algorithm 3 with increasing t values. More precisely, it starts with $t = 1$. If the set V^t is not empty, it computes $b(t, e_j)$ and conducts the inference process as described in Algorithm 1. Only when the set V^t is found to be empty, it increases t by 1 and restarts the modified Algorithm 3 again. It repeats this until it finds a nonempty V^t.

To analyze the time complexity of this adaptive inference algorithm, we assume that in some instance of execution there are actually T faults. In the worst case, the algorithm will find all $V^t, t = 1, \cdots, T-1$, to be empty. For $t = T, V^t$ will be nonempty. As shown previously, for each $t = 1, \cdots, T-1$, the algorithm takes $O(ZN^t)$ amount of time and the last step takes $O(ZN^T)$. Therefore, it takes at most $O(\Sigma_{t=1}^{T} ZN^t) = O(ZN^{T+1})$ amount of time when there are exactly T faults. Normally, each LDC has limited fanout. Therefore, the number of output ports Z should be bounded by a constant times N, the number of LDC's. Thus, the time complexity of the algorithm is $O(N^{T+2})$ for T faults. Assuming that the *a priori* failure probabilities of the LDC's are all equal to p, then the average time complexity of the algorithm is, at most,

$$O\left(\sum_{T=1}^{N} N^{T+2} \frac{\binom{N}{T} p^T (1-p)^{N-T}}{1-(1-p)^N}\right) \le O\left(\sum_{T=1}^{N} (N^2 p)^T\right). \tag{26}$$

When $p < 0.5N^{-2}$, (26) is bounded by a constant. In other words, whenever some fault occurs, the adaptive inference algorithm would spend (on average) a constant amount of time to compute the belief function, independent of the size of the network as long as $p < 0.5N^{-2}$. As an example, when p can be guaranteed to be smaller than 5×10^{-5} (a reasonable assumption given today's optical technology), the network can be made as large as having 100 nodes.

VI. Simulation Results

We have conducted some simulation experiments to assess the performance of Algorithm 1 with our belief updating

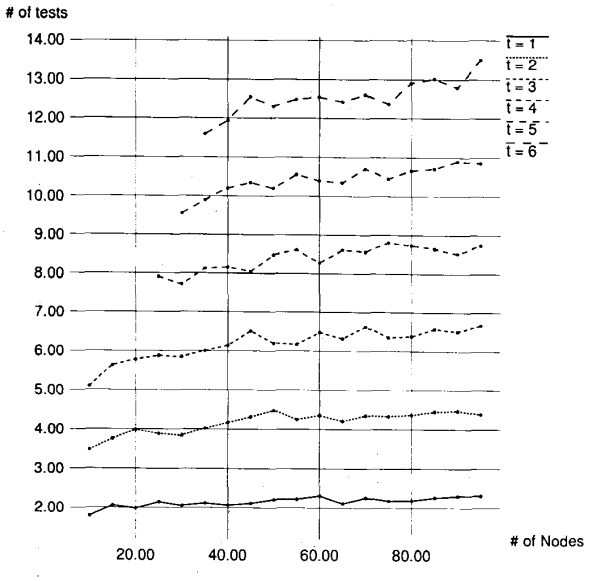


Fig. 6. Average time complexity.

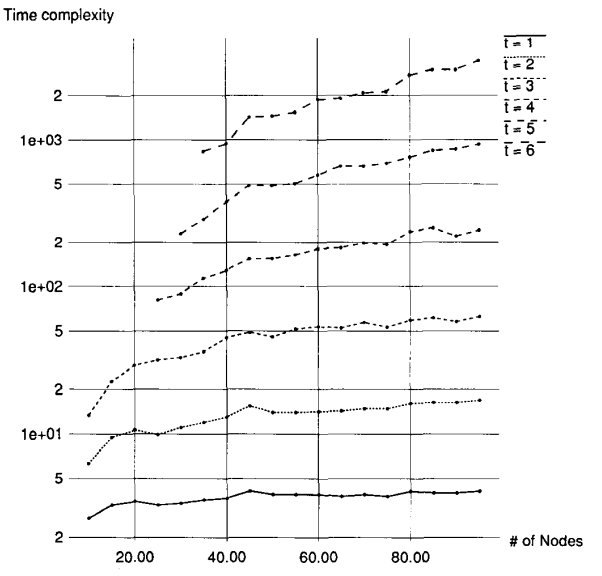


Fig. 7. Average time complexity.

algorithm (Algorithm 3) proposed in Section IV-C. We are mainly concerned with two performance measures: the number of power tests required to locate all faulty LDC's and the computational complexity. The former can be directly counted as the number of iterations of step 3 of Algorithm 1. The latter could be, in principle, found out by measuring the time it takes for the algorithm to find all the faults. But, in a time-shared multiuser system, the time thus obtained does not really reflect the true computational complexity. On the other hand, we have seen in the complexity analysis of Section IV-C that the size of the sets V_i (see Algorithm 3) is proportional to the time complexity of the algorithm. Therefore, we counted the number of terms in sets V_i for the estimation of the computational complexity.

We have tested LLN's with various sizes from 15 to 95 nodes. The LLN's and the routes were generated at random. There is a fixed number of output ports at the periphery. Each output port has at least two routes from some input

The *a priori* failure probability was assumed to be the same for all LDC's and, for the simulation, it was arbitrarily chosen to be 0.001. For each of the sizes, we generated 10 sample networks with routes. Then, for each sample LLN with routes, we generated fault patterns containing one faulty LDC's up to six faulty LDC's. For each of the fixed number of faults $t = 1, \cdots, 6$, we generated ten faulty patterns at random. Therefore, for each size and number of faults combination, there were 100 samples all together. Then, the algorithm was run for the network with these fault patterns. The total number of power tests and the maximum number of terms in the sets V_i were recorded. The simulation was implemented in C with little effort in coding efficiency, and was run on a SUN SPARC 2. It took a few seconds for the smallest generated network, to more than 10 hours for a 95 node with six faults. The results are presented in Figs. 6 and 7, after taking the average over the 100 samples for each size and the number of faults.

From Fig. 6, we find that the average number of power tests grows as the network size and the number of faults increase but very mildly. The average number of power test per fault grows even more slowly. This means that for each fault, the system spends an almost constant amount of extra number of tests to locate it.

On the computational complexity side, from Fig. 7 we can see that with a few exceptions due to random fluctuations, the time complexity increases as the network size and the number of faults increase at different rates. It increases slowly as the network size increases but fast when the number of faults grows. The figures and the real execution time taken seem to suggest an exponential increase in the number of faults. Fortunately, multiple simultaneous faults occur very rarely as long as the single failure probability of the LDC's is small.

VII. Conclusions

In this paper, we proposed a probabilistic approach to fault diagnosis in LLN's. In particular, the application of the Bayesian network inference model to fault diagnosis in LLN was investigated. A general inference algorithm and a belief update algorithm for multiply connect LLN inference models was proposed. We also proposed the t-fault and adaptive diagnosis concept for the tradeoff between computational cost and fault diagnosis capability.

Some simulation of our algorithm was reported. The results suggest that the algorithm performs well with single and multiple faults. The average number of power tests grows slowly as the network size and number of faults increase. Although the computational complexity grows fast as the number of faults and network size increase, it has been shown in the simulation that with today's available midrank workstations, the computational demand of our system for six simultaneous faults in a 100-mode network can be coped with. This limit will certainly be pushed up with better

implementations and more powerful machines. As possible future research topics, it is interesting to consider extension of the model to cover the multibandwidth aspect and to cope with partial LDC failure of the LLN.

APPENDIX

This Appendix presents Pearl's belief updating algorithm. Pearl's algorithm is distributed in nature, with each node/RV of the Bayesian network regarded as an individual processor. Each node performs local computations, and the results are communicated only between neighboring nodes. A typical fragment of the singly connected Bayesian network is shown in Fig. 8, along with the messages to be passed between neighboring nodes. In this figure, node X has n parents $U_1, \cdots, U_n, m$ children, $Y_1, \cdots, Y_m$. The conditional probability $P(x|u_1, \cdots, u_n)$ quantitatively relates the node X to it parents.

Let W^-_{XYj} denote the evidence contained in the subnetwork on the head side of the arc $X- > Y_j$, and W^+_{UiX} denote the evidence in the subnetwork on the tail side of the arc $U_i- > X$. The total available evidence is given by $W = \{W^-_X, W^+_X\}$, where $W^-_X = \{W^-_{XY1}, \cdots, W^-_{XYm}\}$ and $W^+_X = \{W^+_{U1X}, \cdots, W^+_{UmX}\}$. Note that for singly connected networks, all W^-_{XYi} and W^+_{UiX} are disjoint.

In Fig. 8, the π message

$$\pi_X(u_i) = P(u_i|W^+_{UiX}) \tag{27}$$

is the current strength of the causal support contributed by incoming arc $U_i- > X$, and the λ message

$$\lambda_{Yj}(x) = P(W^-_{XYj}|x) \tag{28}$$

is the current strength of the diagnostic support contributed by each outgoing arc $X- > Y_j$.

Algorithm 2 (*Pearl's belief updating algorithm [18], [15]*)

A node X is activated when it receives the π messages from its parents, the λ messages from its children, or the node itself is instantiated for a specific value x. Upon the activation, X performs the following three steps in any order.

Step 1: Belief updating. The node X updates its belief measure to

$$b(x) = \alpha\lambda(x)\pi(x) \tag{29}$$

where

$$\lambda(x) = \begin{cases} \prod_j \lambda_{Yj}(x) & \text{if } X \text{ is not instantiated,} \\ 1 & \text{if } X \text{ is instantiated for } x \\ 0 & \text{if } X \text{ is instantiated but not for } x \end{cases} \tag{30}$$

is the λ value of node X,

$$\pi(x) = \begin{cases} \sum_{u_1,\cdots,u_n} P(x|u_1, \cdots, u_n)\prod_i \pi_x(u_i) & \\ \quad \text{if } X \text{ is not instantiated,} & \\ 1 & \\ \quad \text{if } X \text{ is instantiated for } x & \\ 0 & \\ \quad \text{if } X \text{ is instantiated but not for } x & \end{cases} \tag{31}$$

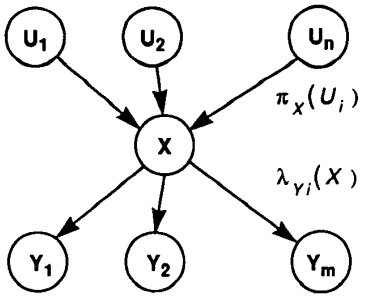


Fig. 8. A typical node X in a Bayesian network.

is the π value of node X, and where α is a normalizing constant rendering $\Sigma_X\, b(x) = 1$.

Note that (29)–(31) implies that $b(x) = 1$ if X is instantiated with value x and 0 if X is instantiated with values other than x.

Step 2: Bottom-up propagation. The node X computes new λ messages and posts them to its parents:

$$\lambda_X(u_i) = \sum_x \lambda(x) \sum_{u_k:k\neq i} \cdot p(x|u_1, \cdots, u_n) \prod_{k\neq i} \pi(u_k). \tag{32}$$

Step 3: Top-down propagation. The node X computes new π messages and posts them to its

$$\pi_{Yj}(x) = \begin{cases} b(x)/\lambda_{Yj}(x) & \text{if } X \text{ is not instantiated,} \\ 1 & \text{if } X \text{ is instantiated for } x \\ 0 & \text{if } X \text{ is instantiated but not for } x. \end{cases} \tag{33}$$

This algorithm needs to be initialized by the following procedures:

1. Set all λ values, λ messages, and π messages to 1.
2. For all roots U, set $\pi(u) = P(u)$.
3. For all roots U and all children X of U, the node U posts new π messages to X:

$$\pi_x(u) = \begin{cases} P(u) & \text{if } U \text{ is not instantiated,} \\ 1 & \text{if } U \text{ is instantiated for } u \\ 0 & \text{if } U \text{ is instantiated but not for } u. \end{cases} \tag{34}$$

Initially, when no evidence is available, the probability distribution embedded in the Bayesian network is in equilibrium. Upon the instantiation of a node (i.e., the arrival of a new piece of evidence), the equilibrium state is broken. In order for the network to enter a new equilibrium state (i.e., the belief functions converge to their true values), the number of belief updates to be performed by each node is proportional to the diameter of the Bayesian network [18].

REFERENCES

[1] K. Bala, "Routing in linear lightwave networks," Ph.D. thesis, Columbia Univ., 1992.

[2] E. Charniak, "Bayesian networks without tears," *AAAI AI Mag.*, pp. 50–63, Win. 1991.

[3] H. L. Chin and G. F. Cooper, "Stochastic simulation of Bayesian belief networks," in *Uncertainty in Artificial Intelligence 2*, L. N. Kanal, J. Kemmer, and T. S. Levitt, Eds. North Holland: Amsterdam, 1989, pp. 129–148.

[4] G. F. Cooper, "Probabilistic inference using belief networks is NP-hard," Tech. Rep. KSL-87-27, Stanford Univ., Stanford, CA, 1988.

[5] N. W. Dawes, J. Altoft, and B. Pagurek, "Network diagnosis by reasoning in uncertain nested evidence spaces," submitted to *IEEE Trans. Commun.*

[6] R. Fung and K. C. Chang, "Weighing and integrating evidence for stochastic simulation in Bayesian networks," in *Uncertainty in Artificial Intelligence 5*, M. Henrion, R. D. Shachter, L. N. Kanal, and J. Kemmer, Eds. North Holland: Amsterdam, 1990, pp. 209–219.

[7] R. M. Goodman, J. Miller, and P. Smyth, "Real time autonomous expert systems in network management," in *Integrated Network Management I*, B. Meandzija and J.Westcoot, Eds. Boston, MA: Elsevier Science, 1989, pp. 599–624.

[8] J. H. Griesmer, "YES/MVS: A continuous real time expert system," in *Proc. AAAI-84*, Austin, TX, 1984.

[9] M. Henrion, "Propagating uncertainty by logic sampling in Bayes' network," in *Uncertainty In Artificial Intelligence 2*, L. N. Kanal and J. F. Lemmer, Eds. North Holland: Amsterdam, 1988, pp. 149–163.

[10] M. Henrion, J. S. Breese, and E. J. Horvitz, "Decision analysis and expert systems," *AAAI AI Mag.*, pp. 64–91, Winter 1991.

[11] P. Hong and P. Sen, "Incorporating nondeterministic reasoning in managing heterogeneous network faults," *Integrated Network Management II*, I. Krishnan and W. Zimmer, Eds. Boston, MA: Elsevier Science, 1991, pp. 481–492.

[12] T. E. Marques, "A symptom driven expert system for isolating and correcting network faults," *IEEE Commun. Mag.*, vol. 26, Mar. 1988.

[13] R. A. Maxion, "Anomaly detection for diagnosis," in *Proc. 20th IEEE Int. Conf. Fault Tolerant Comput.*, 1990, pp. 20–27.

[14] F. M. Miller, G. V. E. Otto, E. M. Siegfried, and P. E. Zeldin, "ACE: A knowledge based maintenance analyzer," in *Proc. IEEE Automated Testing Conf.*, 1985.

[15] R. E. Neapolitan, *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. New York: Wiley, 1990.

[16] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1991.

[17] J. Pearl, "Evidential reasoning using stochastic simulation of casual models," *Art. Intell.*, vol. 32, pp. 245–257, 1987.

[18] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.

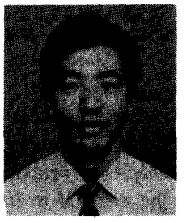
[19] N. Schroff and M. Schwartz, "Fault detection/identification in the linear lightwave networks," Tech. Rep. CU/CTR/TR 243-91-24, Columbia Univ., New York, 1991.

[20] R. D. Shachter and M. A. Peot, "Simulation approaches to general probabilistic inference on belief networks," in *Uncertainty in Artificial Intelligence 5*, M. Henrion, R. D. Shachter, L. N. Kanal, and J. Kemmer, Eds. North Holland: Amsterdam, 1990.

[21] T. E. Stern, "Linear lightwave networks: How far can they go?," in *Proc. IEEE Global Telecommun. Conf.*, San Diego, CA, Dec. 2–5, 1990, pp. 1866–1872.

[22] Z. Wang, "Model of network faults," *Integrated Network Management I*, B. Meandzija and J. Westcott, Eds. Boston, MA: Elsevier Science, 1989, pp. 345–352.

[23] T. L. Williams, P. J. Orgren, and C. L. Smith, "Diagnosis of multiple faults in a nationwide communications network," in *Proc. 8th IJICAI*, Karsruhe, West Germany, Aug. 1983.



Robert H. Deng received the B.E. degree in electrical engineering from Changsha Institute of Technology, Changsha, China, in 1981 and the M.S. and Ph.D. degrees in electrical engineering from the Illinois Institute of Technology, Chicago, in 1983 and 1985, respectively.

From 1986 to 1987, he was a Postdoctorate Research Associate in the Department of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, IN. Between 1987 and 1991, he was a Research Staff Member at the Institute of Systems Science, National University of Singapore (NUS), Singapore. Currently, he is a Senior Lecturer in the Department of Electrical Engineering, NUS. His research interests include error control coding for digital communication and magnetic recording systems, coding and diversity techniques for mobile radio systems, trellis-coded modulations, network interconnection, local area networks, performance evaluation, and network fault management.

Dr. Deng is a member of the IEEE Information Theory and Communications Societies.

Aurel A. Lazar (S'77–M'80–SM'90–F'93), for a photograph and biography, see p. 1335.

Weiguo Wang received the B.E. degree in computer science and technology from the University of Science and Technology of China, Hefei, China in 1983, and the M.A. and Ph.D. degrees in computer science from Boston University, Boston, MA, in 1985 and 1991, respectively.

He has been working at the Institute of Systems Science, National University of Singapore as an Associate Research Staff since November 1990. He has been interested in theory of computation, complexity theory, reliable computation, and cellular automata since he was a graduate student. His current research interest is network fault management.

Dr. Wang is a member of the IEEE and its Computer and Communications Societies, ACM SIGACT, and the European Association for Theoretical Computer Science (EATCS).