Deriving Prognostic Continuous Time Bayesian Networks from D-matrices

Logan Perreault, Monica Thornton, Shane Strasser, and John W. Sheppard
Computer Science Department
Montana State University
Bozeman, MT 59717
{logan.perreault, monica.thornton, shane.strasser, john.sheppard}@cs.montana.edu

***Abstract*—Probabilistic graphical models are widely used in the context of fault diagnostics and prognostics, providing a framework to model the relationships between faults and tests in complex systems. Bayesian networks have sufficient representational power as a model for system-level diagnosis but are inadequate for domains involving prognosis. In order to perform fault prognostics, a model must have the capability to perform probabilistic reasoning over time. One model well suited to this problem is the continuous time Bayesian network (CTBN). In this paper, we propose a method of constructing a continuous time Bayesian network from a D-matrix, a common matrix representation of a diagnostic model. Additionally, we provide procedures for parameterizing the CTBN using reliability information such as the mean time between failures, as well as the false alarm and non-detect probabilities. Through experiments on two different datasets, we demonstrate the correctness of our parameterization process. We also explore the ways in which applying evidence impacts the query results over the network. Finally, we demonstrate the real–world applicability of this approach by performing incremental tests for the purpose of diagnosing and prognosing a fault in the system.**

I. Introduction

As systems increase in complexity and we become more dependent on those systems, accurate testing procedures and fault diagnosis become a significant concern. Within an aerospace context, an unobserved faulty component may result in loss of mission, or more catastrophically, loss of life. The ability to predict faults before they occur is not only a worthwhile problem, but it is also an extremely difficult one.

A diagnostic model requires information about the system as provided by tests, where the results of those tests indicate the likely presence or absence of faults in the system [1]. In addition to the difficulty of designing tests to correctly detect and isolate faults, one must also consider the impact of imperfect testing in the form of false alarm and non-detect events. False alarms and non-detects introduce false indication and false assurance into the testing process, further complicating the process of diagnosing faults.

Probabilistic graphical models provide a convenient means for performing diagnosis in complex systems, as reasoning about the model reflects reasoning about uncertainties in the actual system. This allows for the implementation of multiple diagnostic strategies to evaluate the one best suited for the system [2]. These models are very useful but difficult to build, often requiring the application of learning algorithms or domain knowledge.

If fault diagnosis is a difficult problem, harder still is forecasting what problems are likely to occur in the future. This is the problem of prognostics. Bayesian networks (BNs) have difficulty with this task, as they are incapable of dealing with continuous time. The continuous time Bayesian network (CTBN) is a model that is well suited for applications requiring reasoning in continuous time, and the focus of our work is on using them in prognostic contexts. This paper presents a method of building a CTBN using a D-matrix, an adjacency matrix that encodes the relationships between faults and tests in the system. Unfortunately, the D-matrix, while useful for constructing networks, does not contain the necessary information to parameterize the network. To remedy this, we also propose a method for parameterizing the CTBN using information readily available from historical data or domain knowledge.

II. Background

Prior to describing our proposed method for constructing prognostic CTBNs, we provide a brief overview of the related concepts required to describe the contributions of this work.

A. Notation

Before going into any technical detail, we first describe the notation we will use throughout the paper. A capital letter with a value in parentheses, such as $X(t)$, is used to describe a state instantiation for variable X at time t. A specific state for variable X is often denoted with a corresponding lowercase x^i, where i is the discrete index of the state. Capital letters F_i and T_j are used to denote faults and tests respectively, where the subscript represents the index of the component in the system. A hollowed $\mathbb{F}_i$ is the vector of faults that are detected by a test T_i, therefore $\mathbb{F}_i[j]$ signifies the j^{th} fault detected by T_i. λ_i and μ_i are used to describe the failure and repair rates respectively for fault F_i. Similarly, ND_i and FA_i are the non-detect and false alarm probabilities respectively associated with test T_i, which describe the probabilities of the test failing to do its job. $ND_{i,j}$ and $FA_{i,j}$ also describe non-detect and false alarm probabilities, but do so for the specific test T_i and fault $\mathbb{F}_i[j]$ pair. Rather than describing the probability of complete test failure, $ND_{i,j}$ and $FA_{i,j}$ capture the notion that a test may fail to detect a single fault, while remaining operational. These symbols and their meanings are summarized in Table I.

B. D-matrix

To perform system diagnosis and prognosis, we require certain information about the current state of the unit under

TABLE I. TABLE OF NOTATION

F_i	$\triangleq$	The i^{th} fault in the network
T_i	$\triangleq$	The i^{th} test in the network
$\mathbb{F}_i$	$\triangleq$	The vector of faults monitored by T_i (the parent set of T_i)
λ_i	$\triangleq$	Failure rate for fault F_i
μ_i	$\triangleq$	Repair rate for fault F_i
ND_i	$\triangleq$	The non-detect probability for test T_i
FA_i	$\triangleq$	The false alarm probability for test T_i
$ND_{i,j}$	$\triangleq$	The non-detect probability for the test-fault relationship between T_i and $\mathbb{F}_i[j]$
$FA_{i,j}$	$\triangleq$	The false alarm probability for the test-fault relationship between T_i and $\mathbb{F}_i[j]$

test (UUT). This information is collected via tests performed on the UUT, and the outcome of these tests indicate the presence or absence of faults in the system [1]. In a complex system, this interplay of tests and faults can make diagnosis and maintenance difficult - as a single fault may be monitored by multiple tests, and a single test may monitor multiple faults. To manage this complexity, the relationships between the faults and tests can be represented explicitly and compactly in an adjacency matrix that we call a D-matrix. Within this matrix, the columns correspond to tests and the rows correspond to the potential failures observed by the tests. The D-matrix, has been adopted by a number of modeling tools used to perform fault diagnosis, and the aerospace community is one of the major communities using these tools [3].

Constructing a D-matrix involves assembling the diagnostic signatures for every potential fault within the UUT. Given a set of faults $\mathbb{F}$ and a set of tests $\mathbb{T}$, we can define the function $eval(F_i, T_j)$ as follows:

$$eval(F_i, T_j) = \begin{cases} 1, & \text{if } T_j \text{ detects } F_i \\ 0, & \text{otherwise.} \end{cases}$$

For some fault F_i, when we apply this function to each $T_j \in \mathbb{T}$, the result is the diagnostic signature $\mathbf{F_i} = [eval(F_i, T_1), \ldots, eval(F_i, T_{|\mathbb{T}|})]$ for that fault [3], [4]. This diagnostic signature is constructed for each $F_i \in \mathbb{F}$. Each signature $\mathbf{F_i}$ forms the i^{th} row of the D-matrix.

The relationships between tests and faults, described by a D-matrix, do provide information about how to diagnose the UUT, but they do not offer the complete picture. The logical relationships between the tests and faults are codified in the D-matrix, but probabilistic information is not captured by this format [4]. Similarly, the D-matrix does not provide information on fault-to-fault or test-to-test relationships. For these reasons, although the D-matrix is used to assist in producing a network structure as detailed in Section IV of this work, additional information is required to parameterize the model used to perform diagnosis and prognosis on the UUT.

C. Continuous Time Bayesian Networks

A continuous time Markov process (CTMP) is a model describing a set of discrete state variables X that evolves in continuous time. This model consists of an initial distribution over the state space $P(X(0))$, and an intensity matrix Q_X that describes state transition behavior. Each entry $q_{i,j}$ in row i, column j of Q_X provides the rate parameter for an exponential distribution that indicates how quickly X will transition from state x^i to state x^j. A diagonal entry $q_{i,i}$ is constrained to be the negative sum of the remaining entries in the row, forcing each row to sum to zero. The absolute value of the diagonal entry in row i is the rate for an exponential distribution indicating when the system will transition away from state i and into any other state.

Unfortunately, the size of the state space of X is exponential in the number of variables in the system, which in turn implies that the size of the initial distribution $P(X(0))$ and the intensity matrix Q_X are exponential as well. CTBNs factor CTMPs by taking advantage of conditional independencies between variables in the system [5]. These independencies are encoded using a directed graphical network G, where each node represents a variable in the system. An arc from node A to node B indicates that the behavior of variable B depends on the state of A. Note that the graph G in a CTBN factors a Markov process in much the same manner as a directed acyclic graph in a Bayesian network factors a joint probability distribution. One key difference, however, is that the graphical structure of the CTBN need not be acyclic.

This factored representation provides significant modeling advantages in that, rather than requiring parameterization of a model with a single exponentially sized probability distribution and intensity matrix, define separate local probability distributions and intensity matrices over the domain of each variable individually. The parents of a node in the graph influence the behavior of the node, so a set of intensity matrices are defined for each state instantiation to a node's parents only. These matrices are conditioned on the states of the parents, and are therefore referred to as conditional intensity matrices (CIMs).

Performing exact inference in CTBNs requires the use of the amalgamation operation, which combines two nodes in the network into a single node over an expanded state space. If all nodes in a CTBN are amalgamated, the result is the original unfactored CTMP, which allows for exact inference using the matrix exponential operation. Unfortunately this expansion process results in an initial distribution and intensity matrix that is exponential in size, which nullifies the benefits gained by factorization. To bypass this issue, several approximate inference algorithms have been developed for CTBNs, consisting of both variational and sample-based methods. These algorithms provide a means to infer approximately correct information from the network within a shorter period of time.

Diagnostics and prognostics can be achieved by inferring the state of faults within a model. To represent existing knowledge about a system, evidence may be set to indicate known states. Traditionally, reliability models consist of a network with fault and test nodes. In this case, evidence is always set on the test nodes that describes the test outcomes, and the state of faults is inferred. In some cases there are effects that depend on faults, which can also be inferred from the test results via the fault nodes. For CTBNs, evidence and queries can be applied over the course of time. This means that evidence may describe the state of a test at a particular point in time, or may indicate the state of a test of an entire interval of time. Similarly, queries about faults, effects, or even unknown tests can be made that request probabilities over specified periods of time.

D. Noisy-OR

Although the size of the probability distributions and intensity matrices are reduced to be linear in the size of the variable, the total number of CIMs are still exponential in the number of parents. This representational complexity can be ameliorated in part through the use of a Noisy-OR gate, which is used to model disjunctive interaction. Disjunctive interaction occurs when at least one condition is sufficient to cause an event and the likelihood of that event does not decrease when more than one of the conditions transpire concurrently [6]. This model provides a reduction in the number of necessary parameters, which has several practical benefits. The disjunctive interaction captured by the Noisy-OR model provides savings in storage space, simplified knowledge acquisition, and the potential for accelerated evidence propagation [7].

The Noisy-OR model is a generalized version of the logical OR gate, with the additional capability to model non-determinism through the addition of probabilistic information. To describe the Noisy-OR model in greater detail, we start with the familiar OR gate model which has inputs $U_1, U_2, \ldots, U_N$ and output X. If each U_i is treated as an input line that leads to the Noisy-OR gate, we can associate a non-deterministic line failure function $\mathcal{N}_i$ with each input line. $\mathcal{N}_i$ takes as input the U_i values, and outputs a U_i' value [8]. This line failure function provides the opportunity for events not modeled in the system to have an impact on the output X. In this way, the Noisy-OR gate provides a model with greater representational power, capable of accounting for non-deterministic disjunctive interactions.

The Noisy-OR model has demonstrated utility in the context of BNs, where it has been used to simplify the modeling task by reducing the amount of information required to specify the relationships between causes and effects [8], [9]. Reducing required information has the benefits not only of reducing complexity, but also of being a more accurate reflection of the limited amount of information often available in real–world scenarios. Although work to date has primarily been in the realm of BNs, a recent adaptation to the Noisy-OR gate model allows it to be used in the contexts of CTBNs as well [10]. In this case, while normally the number of CIMs for a node is exponential in the number of parents, Noisy-OR reduces this set to be linear.

III. Related Work

In certain diagnostic environments, the use of probabilistic graphical models is a well-established method of modeling systems with complex interactions. The probabilistic component of these models allows for assessing the likelihood of various scenarios, providing a context-sensitive method for evaluating the interactions in these systems. Within the realm of fault diagnostics, the BN has been adopted widely for numerous practical applications. The advantage gained by using BNs is that they can model a joint probability distribution while exploiting conditional independence properties. This effectively reduces the state space of the problem, and therefore the computational costs associated with inference [6].

In order to reap these benefits, however, we are faced with the problem of constructing the model. Model construction is often accomplished either via learning from historical (or simulated) data or by building the model by hand using domain knowledge [4]. While both of these methods are effective, they can be prohibitively expensive in terms of the time and resources required. An attractive alternative is to derive these models from the kinds of diagnostic models already used in the UUT, and work has been done to this end with BNs [4], [11], [12].

For all of their advantages in combating representational complexity, BNs are not without their limitations. By design, BNs reason about static processes. While this is often sufficient for performing diagnostics, the task of prognostics requires that a system is modeled with respect to time. To avoid this limitation, dynamic Bayesian networks (DBNs) have been developed as an extension to static Bayesian networks that are capable of modeling system change in discrete time. By using multiple copies of a static BN, DBNs provide a method for modeling time indirectly, without changing the semantics of the underlying model.

DBNs have been successfully applied to many prognostic tasks. McNaught and Zagorecki used DBNs to model equipment that deteriorates over time and is monitored by imperfect sensors [13]. They consider multiple repair policies, and investigate how this effects the ability to inform maintenance decisions. DBNs have also been applied to the task of online health estimation [14]. In this case, a DBN is learned during an off-line data collection phase, and the resulting model is used in conjunction with routinely polled sensors to predict the remaining useful life of the system.

In addition to monitoring system health, DBNs have also demonstrated utility for making predictions about complex medical domains. Work has been done to use DBNs to construct prognostic models for clinical patient management [15]. In their work with carcinoid patients, vanGerven *et al.* were able to construct models that took into account temporal dynamics, as well as statistical data, domain literature, expert knowledge and causal factors in order to make predictions about the health state of the patient through time.

The IEEE Std 1232-2010 Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) was introduced to provide a shared framework for testing and diagnostic applications that make use of artificial intelligence (AI) [16]. Recent work has been done to incorporate DBNs into the AI-ESTATE framework, extending the scope of the AI-ESTATE standard beyond data driven diagnosis, to include the ability to perform data driven prognosis as well [17], [18].

Although DBNs do have the ability to model dynamic processes, they are still limited to working in discrete time. In order to perform prognostic tasks, DBNs requires the imposition of a uniform time granularity on the system. For systems with naturally occurring time slices, this is not a problem. However, in many real–world systems, observations can be erratic, making the selection of a uniform time granularity infeasible. For this reason, DBNs are ineffective for structured stochastic processes that evolve over continuous time. Problems that stress the importance of state changes in continuous time occur frequently in the real–world, and are exactly the sort of problems that are well suited to a CTBN model.

While work has been done on deriving BNs from D-matrices, no work to date has explored the process of deriving CTBNs from D-matrices. Given the advantages of the CTBN framework, as well as the prevalence of D-matrices in communities concerned with system-level diagnostics, we chose to extend the algorithm for constructing BNs using D-matrices to CTBNs. Our methods for constructing and parameterizing CTBNs comprise Sections IV and V of this work.

IV. Deriving CTBN Network Structure

In order to take advantage of the prognostic abilities of a CTBN framework, we must produce the correct network structure and parameterize it accordingly. This section describes the process of constructing the CTBN using the information contained in the D-matrix. The parameterization of the CTBN cannot be derived from the D-matrix, but can be derived from reliability and measurement data. The process of parameterizing the CTBN is the subject of Section V.

A. Processing a D-matrix

The relationships between faults and tests that are encoded in a D-matrix can be used to construct a bipartite network structure for a CTBN. Let G be the network structure for the CTBN that is being constructed, and let D be a D-matrix with m rows and n columns. First, fault nodes are added to the network G that correspond to the rows of D. Next, test nodes are added to G corresponding to the columns of D. Finally, arcs are inserted between nodes based on the values in the matrix D. For each entry in the D-matrix where $D_{ij} = 1$, an arc is inserted from fault F_i to test T_j.

The resulting network G has $m + n$ nodes, and $\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}$ arcs, equal to the number of ones in the matrix. Directed arcs only exist from a fault node to a test node, so G is bipartite with one layer consisting of the m faults and the other layer consisting of the n tests. With only two states in the domain of each node, the intensity matrices and initial probability distributions are small. The factor that drives complexity in the network is the total number of CIMs, which is dependent on the number of parents for each test node. In the worst case scenario, all faults are a member of a test's parent set, and so the number of CIMs for the test is $O(2^m)$. The true number of CIMs for a test node T_j is equal to 2^{p_j}, where $p_j = \sum_{i=1}^{m} d_{ij}$.

B. Network Construction Example

Consider the following D-matrix.

$$D = \begin{array}{c} \\ F_1 \\ F_2 \end{array} \begin{array}{c} \begin{array}{cc} T_1 & T_2 \end{array} \\ \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \end{array}$$

This matrix can be used to construct the graph G as shown in Figure 1. The columns T_1 and T_2 are inserted as test nodes into G, while the rows F_1 and F_2 are inserted as fault nodes. Each 1 in the matrix indicates that an arc should be inserted from a fault to a test node, which means that three dependencies are inserted into G. The parent sets for T_1 and T_2 are $\mathbb{F}_1 = \{F_1, F_2\}$ and $\mathbb{F}_2 = \{F_2\}$, resulting in a complete bipartite graph except for the missing arc between F_1 and T_2.

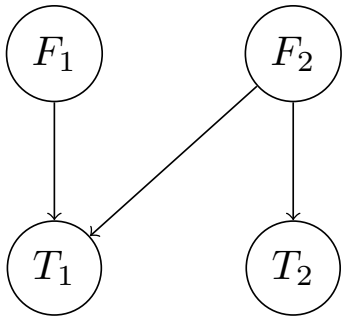

Fig. 1. Example Network

V. Parameterization of Constructed CTBNs

After producing a network structure for a CTBN, the next task becomes parameterizing the CIMs for each of the nodes. These parameters can be derived from information that is already commonly available for diagnostic models. In particular, failure and repair rates can be used to parameterize the fault nodes, while non-detect and false alarm values can be derived from instrumentation data and are sufficient to derive parameters for the test nodes [19]. In this section, we describe the parameterization process for both fault and test nodes. In this work, we assume binary states for both fault and test nodes.

A. Parameterizing Fault Nodes

By definition, fault nodes in the network have no parents. This is advantageous as it associates each fault node with a single unconditional intensity matrix. This matrix describes the transition behavior of the fault between the states in its domain. In the binary case, this means that the CIM defines the probability distributions associated with transitioning to a failing state, and transitioning back to a non-failing state. Domain knowledge and historical data are able to provide failure and repair rates for each fault. Here, a failure rate λ indicates the rate at which a failure will occur given that no failure currently exists. Note that the failure rate λ is related to the mean time between failures (MTBF) in reliability literature as $MTBF = 1/\lambda$.

A repair rate μ indicates the rate at which a fault will transition back to having no failure, and depending on expected repair policy is often set to zero. Note that if no repair policy is implemented, then MTBF is often referred to simply as the mean time to failure (MTTF), implying that the failure state is absorbing. Intensity matrices are simply a collection of transition rates, so parameterization of each matrix is simple. The failure rate λ is assigned to the entry that describes the transition from non-failure to failure, and the repair rate μ is assigned to the reverse direction. As required by the CIM definition, diagonal values are set to be the negative sum of the remaining entries in the row.

The intensity matrix for fault F_i is shown below, where λ_i and μ_i are the failure and repair rates respectively for fault F_i.

$$Q_{F_i} = \begin{array}{c} \\ f_i^0 \\ f_i^1 \end{array} \begin{array}{c} \begin{array}{cc} f_i^0 & f_i^1 \end{array} \\ \begin{pmatrix} -\lambda_i & \lambda_i \\ \mu_i & -\mu_i \end{pmatrix} \end{array} \tag{1}$$

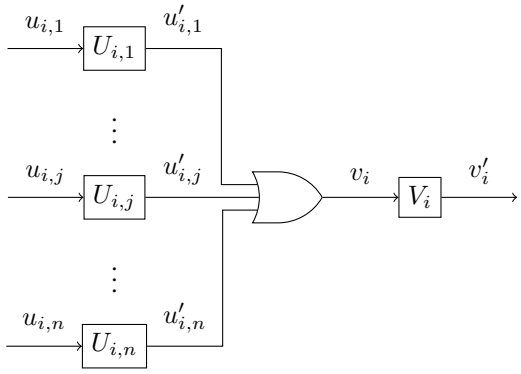

Fig. 2. Line Failure Model

B. Line Failure Model

Prior to discussing the method used to parameterize the test nodes, it is necessary to examine the relative probability of erroneous results in the unit under test. These erroneous results, known collectively as false indications, occur when the tests do not accurately reflect the faults present in the unit under test. False indications can be of two types, false alarms or non-detects, and we adhere to the definitions provided by Sheppard and Kaufman to describe them [19]. A false alarm is defined as an indicated fault where no fault exists, and conversely a non-detect is defined as an indication of no fault where a fault exists. For each of these types of false indications, there can be two potential sources of failure that contribute to the occurrence of either a false alarm or a non-detect event. One failure source is specific to each fault and affects the input received by test T_i from the failure $\mathbb{F}_i[j]$. We denote these failures as $ND_{i,j}$ and $FA_{i,j}$. The second type of test failure occurs due to a malfunction with the test itself. This type of failure will occur regardless of the inputs the test receives, and for test i is denoted ND_i and FA_i.

To parameterize the CIMs for the test nodes in terms of non-detect and false alarm values, we conceptualize the relationship between faults and tests as a line failure model shown in Figure 2. Here, each value $u_{i,j}$ represents the true value for fault $\mathbb{F}_i[j]$, and each $u'_{i,j}$ is the value received by the test node T_i. The $U_{i,j}$ modules are functions that probabilistically alter the value of the line based on non-detect and false alarm values. Specifically, if the input $u_{i,j}$ to $U_{i,j}$ is 1, the output $u'_{i,j}$ will be 0 with probability $ND_{i,j}$. Similarly, if the input is 0, the line will be switched to 1 with a probability of $FA_{i,j}$. A failing output in the test requires at least one fault, so each $u'_{i,j}$ value is fed into an OR gate, whose output v_i represents the input received by the test node. This input is fed through another function V_i, which emulates the failure of the test itself. Similar to each $U_{i,j}$, the V_i function switches the line value according to ND_i and FA_i and produces a final value of v'_i for test T_i.

C. Parameterizing Test Nodes

We seek to define a probability distribution for each test node given the faults that they monitor. In terms of the line failure model, this corresponds to the probability distribution over the value v'_i given input values $u_{i,j}$. The probability that the line failure model outputs a value of 0 is shown by Equation 2. Note that $P(v'_i = 0|v_i = 1)$ is the likelihood that the test i is unable to detect any fault it monitors (ND_i), and $P(v'_i = 0|v_i = 1)$ is the likelihood of a false alarm not occurring for test i ($1 - FA_i$). After replacing these terms in Equation 2, we are left with a formula that accounts for the V_i failure function, as shown in Equation 3.

$$P(v'_i = 0|\mathbf{u_i}) = P(v'_i = 0|v_i = 1)P(v_i = 1|\mathbf{u_i}) + P(v'_i = 0|v_i = 0)P(v_i = 0|\mathbf{u_i}) \quad (2)$$

$$P(v'_i = 0|\mathbf{u_i}) = (ND_i)P(v_i = 1|\mathbf{u_i}) + (1 - FA_i)P(v_i = 0|\mathbf{u_i}) \quad (3)$$

Next, we focus our efforts on calculating the remaining $P(v_i = 0|\mathbf{u_i})$ term, which will also provide a value for $P(v_i = 1|\mathbf{u_i})$. As shown by the line failure model, v_i is the output of a logical OR gate, which takes as input the values $\mathbf{u'_i}$. This means that $P(v_i = 0|\mathbf{u_i})$ can equivalently be interpreted as $\prod_j P(u'_{i,j}|u_{i,j})$. Note that each input to the OR gate is independent, which is why the conditionals for each probability can be reduced only to the value corresponding to the current line. The probability of each value $u'_{i,j}$ can be written in terms of the non-detect and false alarm values associated with line failure function $U_{i,j}$. This is similar to how the probability for v'_i was calculated, except that each $u_{i,j}$ is assumed to be a particular value based on the conditional intensity matrix we are currently parameterizing. The likelihood that v_i is 0 given a fixed set of values $\mathbf{u_i}$ is as follows:

$$\begin{aligned} P(v_i = 0|\mathbf{u_i}) &= \prod_{\{u_{i,j} \in \mathbf{u_i}\}} P(u'_{i,j}|u_{i,j}) \\ &= \prod_{\{u_{i,j} \in \mathbf{u_i} | u_{i,j} = 0\}} P(u'_{i,j} = 0|u_{i,j} = 0) \cdot \\ &\quad \prod_{\{u_{i,j} \in \mathbf{u_i} | u_{i,j} = 1\}} P(u'_{i,j} = 0|u_{i,j} = 1) \\ &= \prod_{\{u_{i,j} \in \mathbf{u_i} | u_{i,j} = 0\}} (1 - FA_{i,j}) \cdot \\ &\quad \prod_{\{u_{i,j} \in \mathbf{u_i} | u_{i,j} = 1\}} (ND_{i,j}). \end{aligned} \quad (4)$$

For each CIM, $P(v_i = 0|\mathbf{u_i})$ can be calculated using the parent states associated with the CIM. The inverse of this value is the probability that v_i takes on a value of one $(1 - P(v_i = 0|\mathbf{u_i})) = P(v_i = 1|\mathbf{u_i})$. These two values can be inserted into the corresponding terms in Equation 3, producing a formula for $P(v'_i = 0|\mathbf{u_i})$ written as a function of the non-detect and false alarm values for test i, and all fault relationships associated with that test. Since the non-detect and false alarm values for the fault-test relationships depend on the state of the faults, the value for $P(v'_i = 0|\mathbf{u_i})$ is potentially unique for each CIM. The line failure model describes how faults ultimately affect the tests. Each $u_{i,j}$ is the value for each of the faults, while v'_i is the value for the test; therefore, $P(v'_i|\mathbf{u_i}) \equiv P(T_i|\mathbb{F}_i)$ by construction.

We are now able to calculate the likelihood that each test is in state zero for each of its CIMs. Again, we can obtain the likelihood of being in state one by simply taking the inverse $(P(T_i = 1|\mathbb{F}_i) = (1 - P(T_i = 0|\mathbb{F}_i)))$. These values can be

used to parameterize the CIMs that they are associated with. The expected value obtained from an exponential distribution with a rate of λ is $1/\lambda$, which corresponds to the MTBF. To produce the expected transition behavior, the entries in row x of the CIM are assigned a value of $P(T_i = x|Fi)^{-1}$. This equates to spending $P(T_i = 0|\mathbb{F}_i)$ portion of the time in state zero, and $P(T_i = 1|\mathbb{F}_i)$ amount of time in state one. Since these are the likelihoods for being in each of these states, and the expected transition behavior conforms to this, inference in a CTBN constructed in this fashion will result in probabilities consistent with the conceptual model. A CIM for a test node with entries defined in terms of the state likelihoods is shown in the following:

$$Q_{T_i|\mathbb{F}_i} = \begin{array}{c} \\ t_i^0 \\ t_i^1 \end{array} \begin{array}{c} \begin{array}{cc} t_i^0 & t_i^1 \end{array} \\ \begin{pmatrix} -P(T_i = 0|\mathbb{F}_i)^{-1} & P(T_i = 0|\mathbb{F}_i)^{-1} \\ P(T_i = 1|\mathbb{F}_i)^{-1} & -P(T_i = 1|\mathbb{F}_i)^{-1} \end{pmatrix} \end{array}. \tag{5}$$

D. Special Case of Test Node Parameterization

Although it is useful to have a general method for parameterizing CIMs given the non-detect and false alarm values for tests and fault-test relationships, it may be unreasonable to expect that these values are available for the system in question. The number of non-detect and false alarm values necessary is equal to the number of edges in the CTBN network, plus the number of total tests: $O(|E| + |V|)$. Although this number is quite manageable from a computational standpoint, it may be that no data exists to describe the non-detect and false alarm values for specific fault-test relationships.

One method that may be used to reduce the number of necessary parameters is to assume that the non-detect and false alarm values for fault-test relationships are all zero. In terms of the line failure model, the failure functions $U_{i,j}$ never fail, and the input is always passed directly to the output ($u_{i,j} = u'_{i,j}$). Any non-detect or false alarm events that may occur are summarized by the ND_i and FA_i values for test i.

By assuming that no line failure exists for each $U_{i,j}$ function, the number of parameters that must be specified is drastically reduced. The only non-detect and false alarm values that need to be specified are for the test nodes: $O(|V|)$. Obtaining this subset of the parameters is a more manageable task, and may be a necessary alternative if the task of collecting non-detect and false alarm values for each fault-test relationship is too difficult. An additional benefit to this assumption is that Equation 3 is sufficient to describe the probability distribution for a test node without further computation. By setting all non-detect and false alarm values for the fault-test relationships to the same value (zero), $P(v_i|\mathbf{u_i})$ is deterministic based on the value produced by the OR gate.

The OR gate forces $P(v_i|\mathbf{u_i})$ to take on a value of either 1.0 or 0.0, depending on whether or not a fault has occurred. In the event that no faults have occurred, $P(v_i = 0|\mathbf{u_i}) = 1.0$, and the Equation 3 reduces to the term $(1 - FA_i)$. For cases where at least one fault is on, the probability is forced to zero, and the equation reduces to (ND_i). The CIM constructed for the former case is shown in Equation 6, while the latter case is given by Equation 7.

$$Q_{\{T_i|(\wedge_{F\in\mathbb{F}_i}F=0\}} = \begin{array}{c} \\ t_i^0 \\ t_i^1 \end{array} \begin{array}{c} \begin{array}{cc} t_i^0 & t_i^1 \end{array} \\ \begin{pmatrix} -(1-FA_i)^{-1} & (1-FA_i)^{-1} \\ (FA_i)^{-1} & -(FA_i)^{-1} \end{pmatrix} \end{array} \tag{6}$$

$$Q_{\{T_i|(\vee_{F\in\mathbb{F}_i}F=1)\}} = \begin{array}{c} \\ t_i^0 \\ t_i^1 \end{array} \begin{array}{c} \begin{array}{cc} t_i^0 & t_i^1 \end{array} \\ \begin{pmatrix} -(ND_i)^{-1} & (ND_i)^{-1} \\ (1-ND_i)^{-1} & -(1-ND_i)^{-1} \end{pmatrix} \end{array} \tag{7}$$

E. Parameterization Example

Continuing the example from Section IV-B, we now parameterize the CIMs for the nodes in Figure 1 using the method that was described in this section. The values we selected to parameterize this example are shown below.

Failure rates for F_1 and F_2	$\lambda_1 = 0.05$, $\lambda_2 = 0.02$
Repair rates for F_1 and F_2	$\mu_1 = 0.1$, $\mu_2 = 0.01$.
ND values for T_1	$ND_1 = 0.05$, $ND_{1,1} = 0.03$, $ND_{1,2} = 0.01$
ND values for T_2	$ND_2 = 0.04$, $ND_{2,1} = 0.04$
FA values for T_1	$FA_1 = 0.01$, $FA_{1,1} = 0.02$, $FA_{1,2} = 0.03$
FA values for T_2	$FA_2 = 0.08$, $FA_{2,1} = 0.04$

As discussed in Section V-A, the failure and repair rates for the fault nodes can be inserted directly into the corresponding intensity matrices. The intensity matrices Q_{F_1} and Q_{F_2} for faults F_1 and F_2 are given below, based on the parameterization shown in Equation 1.

$$Q_{F_1} = \begin{array}{c} \\ f_1^0 \\ f_1^1 \end{array} \begin{array}{c} \begin{array}{cc} f_1^0 & f_1^1 \end{array} \\ \begin{pmatrix} -0.05 & 0.05 \\ 0.1 & -0.1 \end{pmatrix} \end{array}$$

$$Q_{F_2} = \begin{array}{c} \\ f_2^0 \\ f_2^1 \end{array} \begin{array}{c} \begin{array}{cc} f_2^0 & f_2^1 \end{array} \\ \begin{pmatrix} -0.02 & 0.02 \\ 0.01 & -0.01 \end{pmatrix} \end{array}$$

Parameterizing the test nodes is somewhat more complicated. To demonstrate the process, we work out the parameters for the CIM $Q_{T_1|F_{1,1}=0,F_{1,2}=1}$. We start by using Equation 4 to compute $P(v_1 = 0|u_{1,1} = 0, u_{1,2} = 1) = (1 - FA_{1,1}) \cdot (ND_{1,2}) = (1 - 0.02) \cdot (0.01) = 0.0098$. This value can then be used in Equation 3, along with non-detect and false alarm values for test T_1, to obtain a final value $P(v'_1 = 0|u_{1,1} = 0, u_{1,2} = 1) = 0.05 \cdot (1-0.0098) + (1-0.01) \cdot 0.0098 = 0.059212$. Finally, the value is used to produce the matrix below, according to Equation 5. A similar procedure is used to produce the remaining CIMs for the test nodes.

$$Q_{\{T_1|F_{1,1}=0,F_{1,2}=1\}} = \begin{array}{c} \\ t_1^0 \\ t_1^1 \end{array} \begin{array}{c} \begin{array}{cc} t_1^0 & t_1^1 \end{array} \\ \begin{pmatrix} -16.88847 & 16.88847 \\ 1.06294 & -1.06294 \end{pmatrix} \end{array}.$$

VI. Experiments

We conducted three separate experiments meant to illustrate various properties of the constructed networks. The first experiment is meant to verify the correctness of the network construction and parameterization process. This is

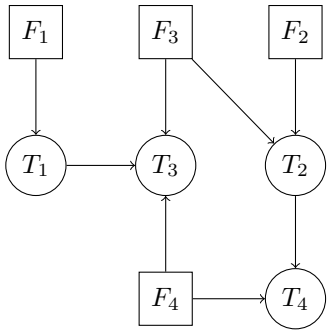

Fig. 3. Logic Diagram for Synthetic

accomplished by comparing the query results over the constructed network to the expected results based on mean time between failure (MTBF), non-detect and false alarm values. The details of this experiment along with the results can be found in Section VI-B. The second experiment was designed to investigate how the behavior of the network changes when evidence is applied at various time intervals. This evidence application experiment is detailed in Section VI-C. Finally, the third experiment follows the testing procedure that a technician might perform to diagnose a fault. Here, the model is queried and a new test is performed repeatedly until enough information is obtained to correctly identify the fault. This experiment is reported in Section VI-D.

Note that all experiments use importance sampling as the underlying inference algorithm. The first two experiments use a synthetic dataset that we constructed for the purpose of this study, while the final experiment makes use of another model that is used in related literature.

A. Datasets

For our experiments, we worked with two different datasets. The first is a manually constructed dataset consisting of four faults and four tests. The MTBF for each fault F_1, F_2, F_3, and F_4 is 1, 2, 4, and 8 respectively, measured in thousands of hours. For these experiments, we assume that there is no repair rate policy associated with the failures, therefore we set $\mu = 0$. Furthermore, we assume that all tests and faults start in a working state at time 0, and therefore set the initial probability distribution to $(1.0, 0.0)$. The non-detect probabilities for the tests are set to 0.02, 0.02, 0.01, and 0.01, while the false alarm probabilities are 0.01, 0.04, 0.02, and 0.01. The structure is determined by the D-matrix, which is shown below. The corresponding logic diagram is shown in Figure 3. Going forward, we refer to this network as "Synthetic".

$$D = \begin{array}{c} \\ F_1 \\ F_2 \\ F_3 \\ F_4 \end{array} \begin{array}{c} \begin{array}{cccc} T_1 & T_2 & T_3 & T_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

The second dataset used in our experiments is referred to as "SimpleModel" and originates from *System Test and Diagnosis* [1]. The model consists of nine main faults and seven main tests. In addition, there is an NF fault that represents no fault occurring, and an $FINT_1$ and $TINT_1$ that are the fault and test that describe the input to the model as a whole. The following D-matrix determines the structure of the SimpleModel network. The associated logic diagram is shown in Figure 4.

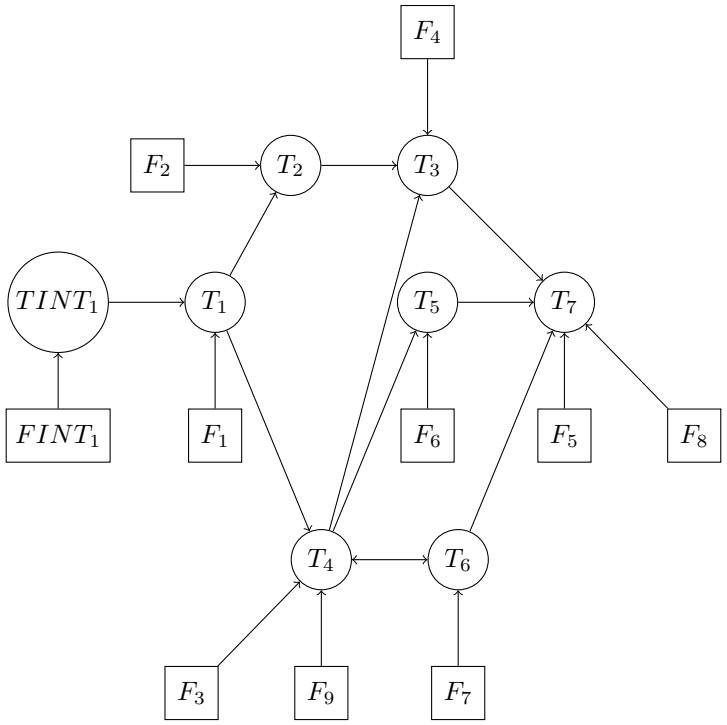

Fig. 4. Logic Diagram for SimpleModel

$$D = \begin{array}{c} \\ FINT_1 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \\ NF \end{array} \begin{array}{c} \begin{array}{cccccccc} TINT_1 & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

SimpleModel was chosen to provide more complexity in terms of the fault-test relationships than the Synthetic model. Furthermore, SimpleModel has been used in related D-matrix literature to generate the structure of Bayesian networks, which is a similar task to our own [4].

Unlike previous work, however, we have proposed a method for parameterizing the constructed network using MTBF, false alarm, and non-detect information. SimpleModel as originally presented does not include this data, so we have augmented the model with the necessary information. The MTBFs for the nine faults in order, measured in thousands of

hours, are 1, 2, 4, 8, 1, 2, 4, and 8. The non-detect probabilities for the seven tests are 0.02, 0.01, 0.01, 0.02, 0.02, 0.01, and 0.01, while the false alarm probabilities are 0.04, 0.02, 0.01, 0.01, 0.04, 0.02, and 0.01.

B. Parameterization Experiment

In the Synthetic model, T_1 monitors only a single fault: F_1. The parameterization experiment constructs and parameterizes a CTBN based on the D-matrix and reliability information. The inference algorithm is used to predict the state distribution for the nodes T_1 and F_1 in the constructed network starting at time 0 and increasing by 100 hour time slices until a final time of 10 thousand hours, resulting in a total of 100 queries per node. These query results were then compared to the expected state distribution for T_1 and F_1, based on the MTBF, non-detect and false alarm values.

Recall from Section VI-A that the MTBF for F_1 in the Synthetic model is 1 thousand hours. Since there is no repair rate and the fault initially starts in a working state, the probability of a failure at time t is fully specified by the exponential distribution defined by the first row of the intensity matrix for F_1. More formally, $P(F_1 = f_1^1) = \lambda e^{-\lambda t} = e^{-t}$, since the MTBF is 1.0. For example, the expected probability of F_1 occurring at 2 thousand hours is e^{-2}. We computed this expected value for each of the 100 time steps that were queried for node F_1 and compared them to the value returned by inference. By taking the mean error of these 100 points, we find that inference over the network results in an error of 0.0002 ± 0.0005.

We can also compute the expected values for T_1 at these time steps based on the non-detect and false alarm probabilities. The non-detect probability for T_1 is 0.02, while the false alarm probability is 0.01. Given that test T_1 only monitors F_1, the state distribution for T_1 at time t can be written in terms of F_1. Specifically, $P(T_1 = t_1^1|F_1) = P(F_1 = f_1^1)(1-ND) + P(F_1 = f_1^0)(FA) = P(F_1 = f_1^1)(0.98) + P(F_1 = f_1^0)(0.01)$. Here, the probability of each fault is computed as before by using an exponential distribution with a rate of 1.0. Just as before, the expected state distribution for T_1 is computed for each of the 100 time slices and compared to the inferred values. In this case, we find that the mean error when inferring the state of T_1 is 0.0111 ± 0.0030. Figure 5 shows the queried and expected state distribution for T_1 over the 10 thousand hours of interest.

The results of this experiment verify that the construction and parameterization of the CTBN network correctly describes the expected behavior of the model. The faults are relatively simple to model, as their behavior does not depend on any other faults or tests. This is reflected by the extremely low error produced by the inference over F_1. Tests are somewhat more complex, as their behavior is dependent on a subset of the faults in the model, meaning any error that occurs while inferring the state of a fault will in turn produce error when inferring the test. Despite this, the error produced during inference on T_1 is relatively low, and can expected when using an approximate inference algorithm. Figure 5 illustrates how closely the inference results match the expected results for T_1. This affirms the mathematical justification for the parameterization process from Section V-C.

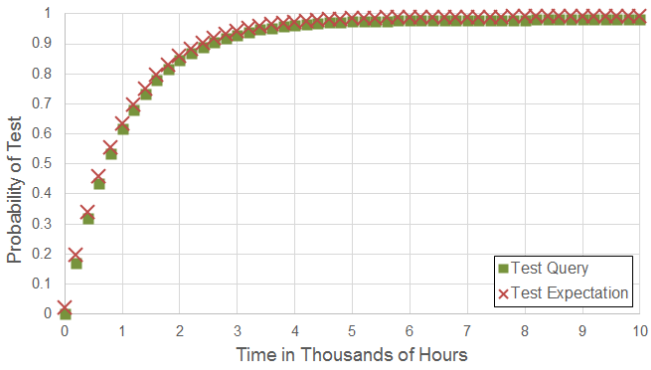

Fig. 5. Probability of F_1 through time in the constructed and parameterized synthetic network.

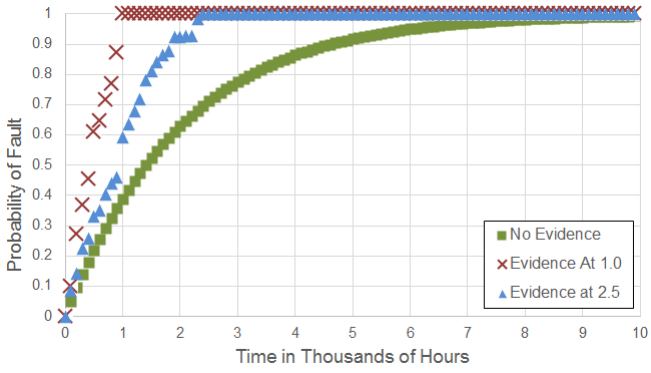

Fig. 6. Probability of F_2 through time in the synthetic network where evidence is applied for T_3 and T_4 at different points in time.

C. Evidence Application Experiment

The next experiment was designed to investigate how evidence affects the probability distributions returned by the model. This experiment again worked with the Synthetic model, but in this case the nodes of interest were T_3, T_4, and F_2. The evidence that was applied set $T_3 = t_3^0$ and $T_4 = t_4^1$. Note that based on the D-matrix for the Synthetic model, the only fault that would produce such assignments to the tests is F_2. To observe how this evidence affects the network, we consider three cases: one where evidence is set at time 2.5, another where evidence is set at time 1.0, and finally a baseline case where no evidence is set. The results are shown in Figure 6.

As expected, evidence significantly changes the state distribution for F_2 over time. The natural behavior of F_2 with no evidence applied is shown by the green squares, which equates to an exponential distribution with a rate parameter of 2, corresponding to the MTBF for F_2. The blue triangles show how this distribution changes over time when evidence is applied at time 2.5. Specifically, the probability of F_2 occurring is forced to one by time 2.5 thousand hours, which is necessary to account for the evidence applied to T_3 and T_4. Similarly, when the same evidence is applied at time 1.0, the probability of F_2 is forced to one even faster, as shown by the red Xs. These results demonstrate the expected behavior of the model, and again support the construction and parameterization

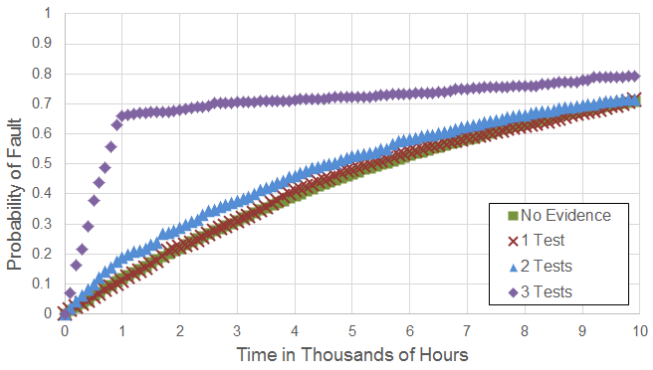

Fig. 7. Probability of F_4 through time in the SimpleModel as tests are incrementally performed to diagnose the fault.

process.

D. Incremental Diagnosis and Prognosis Experiment

The final experiment is meant to show the applicability of the constructed model by demonstrating a potential use-case. For this experiment the SimpleModel is used, which incorporates a larger number of faults and tests than the Synthetic model. Using the D-matrix, MTBF, non-detect and false alarm values from Section VI-A, a CTBN network with eight tests and eleven faults is constructed using the methods described in this paper. The assumption is that this D-matrix and the corresponding parameters are available for a real–world system, and upon construction of the CTBN network the goal is to use the model to diagnose and prognose a failure.

Suppose that the system has been running for 1 thousand hours, at which point something appears to be functioning incorrectly. Without any further knowledge, inference can be run over the network to determine the natural tendency of each of the faults in the network. For F_4, this distribution is shown by the green squares in Figure 7, which is defined by an exponential distribution with a rate of 8, based on the MTBF for F_4.

After viewing the probability of each of the faults, the maintenance aid (or technician) would choose a test to run that will most effectively identify the suspected faults while weighing the cost to run the test itself. In this scenario, let us assume that the maintenance aid chooses to run test 2 and the technician finds that the test passes ($T_2 = t_2^0$). The model can be queried again after applying this new evidence, and in the case of F_4, we see by the red Xs in Figure 7 that there is not much change in the distribution.

Testing may proceed by running test 3 and finding that it failed ($T_3 = t_3^1$), which slightly increases the probability of F^4 occurring, as indicated by the blue triangles in Figure 7. Finally, the technician runs test 4 and sees that the test passes ($T_4 = t_4^0$). This results in a drastic change in the probability of F_4, shown in the figure as the purple diamonds.

By inspection of the D-matrix, fault F_4 is the only fault that can produce the three test results seen by the technician. For this reason, it may seem as though the model is underestimating the probability of F_4, since even after all three tests are run the probability of F_4 at 1 thousand hours is still less than 70%, and steadily increases to a probability of 80% by time 10 thousand hours. What the D-matrix does not consider however, is the MTBF, non-detect, and false alarm values. Although F_4 may in theory be the only fault that produces the test results that the technician encounters, that might not necessarily be the case if there is a non-detect or false alarm event. Furthermore, F_4 is less likely to occur than the other faults in the network due to the larger MTBF associated with it, and the constructed model takes this into account. Even this relatively simple example demonstrates the usefulness of the constructed model in a real–world scenario, where fault-test relationships may be initially misleading.

VII. Conclusion

We have demonstrated that the graphical network structure for a bipartite CTBN consisting of fault and test nodes can be constructed directly from the relationships identified by a D-matrix. Furthermore, we showed how the CIMs for the fault nodes in the network can be parameterized using information about the mean time between failures. Next we presented a method for parameterizing the test node CIMs using the non-detect and false alarm values associated with each fault-test relationship and for the test alone. Finally, a special case of the test node parameterization was presented where all non-detect and false alarm values for fault-test relationships are set to 0.0. This greatly simplifies the parameterization process, and is suitable for many real–world scenarios where non-detect and false alarm information is not available for fault-test relationships, but rather for just the test itself. It also provides a method to field a working model and gather historical data to estimate the non-detect and false alarm values.

We demonstrate the correctness and effectiveness of our approach with three experiments. The first experiment compared the results returned by inference to the expected value of the system. The error introduced by querying the network was minimal in the case of tests, and negligible in the case of faults. The second experiment showed how the application of evidence to the model affects the distributions returned by inference. This verifies that if evidence is applied that forces the behavior of another node in the network, the distribution of the node is adjusted to enter the state by the time of evidence. Finally, our third experiment demonstrates how a CTBN constructed and parameterized using our approach can be used in a real–world diagnostic environment. Specifically, a scenario was described where a technician incrementally performs three tests in an attempt to diagnose a fault in the system. Although the structure provided by the D-matrix indicated with absolute certainty that a particular fault had occurred, the query results instead reported about a 70% confidence that a fault had occurred based on MTBF, non-detect and false alarm information. This further demonstrates how CTBNs, which include probabilistic information, can be more informative than a D-matrix alone, which can only describe basic dependence relationships.

VIII. Future Work

One possible avenue for future work is the introduction of logical closure and transitive reduction to the network construction process. Existing test nodes in the network are assumed to follow the Noisy-OR model, which allows for

a set of CIMs that are linear in the number of parents rather than exponential. Although this is a drastic reduction, at times there are still a significant number of faults that are monitored by a single test. For this reason, it may be beneficial to reduce the number of parents that a node has altogether. Logical closure provides a method for introducing dependencies between tests, which in turn allows transitive reduction to remove dependencies from faults to tests. The result is a network that is potentially less complex from a computational standpoint, while also being more readily understandable to a technician. While the network structure can be modified easily to accommodate the altered dependencies from logical closure and transitive reduction, more work is needed to properly parameterize the new network.

Another interesting area of research involves the use of fault trees for the purpose of CTBN network construction. The work we have presented uses D-matrices to construct a bipartite network between faults and tests. While useful, many models also include the effects that occur as a result of the faults. This effects might include Mission Failure, Loss of Equipment, or Loss of Life. The relationship between the faults and these effects are typically represented as a fault tree, where each node is connected to its children using logic gates. We are currently in the process of deriving a bipartite CTBN consisting of fault and effect nodes. Here the structure is obtained directly from the structure of the fault tree, and the CIMs for the nodes are parameterized based on the logic gates associated with the nodes. Further work in this area involves reducing the complexity of the resulting network by pruning unnecessary dependencies. Network construction from fault trees, in combination with the work presented in this paper, would allow for a more complete representation of a system involving tests, faults, and effects.

ACKNOWLEDGMENTS

This paper was developed in part due to work that was performed under the NASA STTR T13.01-9887 contract. Without the support and guidance from our contacts at NASA and the people at Qualtech Systems, Inc, this work would not have been possible. We would also like to thank the members of the Numerical Intelligent Systems Laboratory (NISL) at Montana State University for their input during the development stages of this paper.

REFERENCES

[1] W. R. Simpson and J. W. Sheppard, *System Test and Diagnosis*. Norwell, MA: Kluwer Academic Publishers, 1994.

[2] J. W. Sheppard and W. R. Simpson, *Research Perspectives and Case Studies in System Test and Diagnosis*. Norwell, MA: Kluwer Academic Publishers, 1998.

[3] J. Sheppard and S. Butcher, "A formal analysis of fault diagnosis with D-matrices," *Journal of Electronic Testing*, vol. 23, no. 4, pp. 309–322, 2007.

[4] S. Strasser and J. Sheppard, "An empirical evaluation of Bayesian networks derived from fault trees," in *Proceedings of the IEEE Aerospace Conference*, March 2013, pp. 1–13.

[5] U. Nodelman, C. R. Shelton, and D. Koller, "Continuous time Bayesian networks," in *Proceedings of the Eighteenth conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 378–387.

[6] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[7] F. J. Díez, "Parameter adjustment in Bayes networks. The generalized noisy or-gate," in *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1993, pp. 99–105.

[8] S. Srinivas, "A generalization of the noisy-or model," in *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1993, pp. 208–218.

[9] S. Parsons and J. Bigham, "Possibility theory and the generalised noisy or model," in *Proceedings of the Sixth International Conference on Information Processing and the Management of Uncertainty*, 1996, pp. 853–858.

[10] L. Perreault, S. Strasser, M. Thornton, and J. W. Sheppard, "The noisy-or gate model for continuous time Bayesian networks," Department of Computer Science, Montana State University, Bozeman, Montana, Tech. Rep. MSU-NISL-15-1, July 2015.

[11] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla, "Improving the analysis of dependable systems by mapping fault trees into Bayesian networks," *Reliability Engineering & System Safety*, vol. 71, no. 3, pp. 249–260, March 2001.

[12] G. Hulten, "Learning Bayesian networks from dependency networks: A preliminary study," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[13] K. McNaught and A. Zagorecki, "Using dynamic Bayesian networks for prognostic modelling to inform maintenance decision making," in *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on*, 2009, pp. 1155–1159.

[14] D. Tobon-Mejia, K. Medjaher, and N. Zerhouni, "CNC machine tool's wear diagnostic and prognostic by using dynamic Bayesian networks," *Mechanical Systems and Signal Processing*, vol. 28, pp. 167 – 182, 2012.

[15] M. van Gerven, B. G. Taal, and P. J. F. Lucas, "Dynamic Bayesian networks as prognostic models for clinical patient management," *Journal of Biomedical Informatics*, vol. 41, no. 4, pp. 515–529, 2008.

[16] IEEE Std. 1232-2010, *IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE)*, Piscataway, NJ: IEEE Standards Association Press, 2010.

[17] H. King, N. Fortier, and J. W. Sheppard, "An AI-ESTATE conformant interface for net-centric diagnostic and prognostic reasoning," in *Proc. IEEE AUTOTEST 2014*, 2014, pp. 226–232.

[18] H. King, N. Fortier, and J. W. Sheppard, "An AI-ESTATE conformant interface for net-centric diagnostic and prognostic reasoning," *IEEE Instrumentation and Measurement Magazine*, vol. 18, no. 4, pp. 18–24, 2015.

[19] J. W. Sheppard and M. A. Kaufman, "A Bayesian approach to diagnosis and prognosis using built-in test," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 3, pp. 1003–1018, 2005.