An Empirical Evaluation of Bayesian Networks Derived from Fault Trees

Shane Strasser and John Sheppard
Department of Computer Science
Montana State University
Bozeman, MT 59717
{shane.strasser, john.sheppard}@cs.montana.edu

***Abstract*—Fault Isolation Manuals (FIMs) are derived from a type of decision tree and play an important role in maintenance troubleshooting of large systems. However, there are some drawbacks to using decision trees for maintenance, such as requiring a static order of tests to reach a conclusion. One method to overcome these limitations is by converting FIMs to Bayesian networks. However, it has been shown that Bayesian networks derived from FIMs will not contain the entire set of fault and alarm relationships present in the system from which the FIM was developed. In this paper we analyze Bayesian networks that have been derived from FIMs and report on several measurements, such as accuracy, relative probability of target diagnoses, diagnosis rank, and KL-divergence. Based on our results, we found that even with incomplete information, the Bayesian networks derived from the FIMs were still able to perform reasonably well.**

TABLE OF CONTENTS

1. INTRODUCTION

Performing fault diagnosis is an extremely difficult problem [1]. Given a complex system and a set of tests or alarm outcomes, the user wants to determine the most likely faulty component. One diagnostic tool that is often utilized is the Fault Isolation Manual (FIM). FIMs are derived from a type of decision tree and allow users to isolate faults at the lowest component level [2]. The user performs the test corresponding to the root of the tree and follows the branch based on the test outcome to the next test specified in the FIM. This process is repeated until the user reaches a leaf in the tree corresponding to the diagnosed fault.

However, there are some drawbacks to using decision trees for maintenance, such as requiring a static order of tests to reach a conclusion. For example, suppose the maintainer only has access to a subset of the test resources required by a FIM (e.g., the oscilloscope is broken). Using a static FIM, determining which faults are diagnosed based on only the available tests can lead a difficult problem, especially if tests requiring unavailable resources occur higher in the tree. One way to overcome this limitation is to convert the FIM to an alternate type of model that allows for dynamically ordered test sequences, such as D-Matrices (i.e., the set of all the relationships between a set of alarms and faults) or Bayesian networks.

In this paper we look at converting FIMs to Bayesian networks. In addition to allowing for a dynamic ordering of tests, converting static FIMs to Bayesian networks has several other advantages. The first is that it allows for probabilistic reasoning over the system. Another is that certain processes, such as diagnostic model maturation, are more naturally suited for Bayesian networks. While there are algorithms that do allow for the incremental updating of decision trees, these algorithms assume that one has access to all of the data used to generate the trees, which will not always be the case, especially when using FIMs generated by domain experts [3].

Bayesian networks have been used extensively for fault diagnosis. Traditionally, there are two ways for building Bayesian networks; the first is to build the network from data using a learning algorithm. The other is to build the network by hand with the aid of a systems expert and manually define the structure and parameters of each random variable [4]. Additionally, one can use some combination of these two methods: First start with a network built by a domain expert and then use a learning algorithm to refine the Bayesian network [5].

There has also been work in deriving Bayesian networks from other models. Starting with a fault tree used in fault tree analysis, one can derive a Bayesian network representing the various logic gates in the fault tree [6], [7]. This work was later extended to allow dynamic fault trees to be converted into Bayesian networks by utilizing dynamic Bayesian networks [8]. However these fault trees are not decision trees. Dependency networks are another type of model that has been used to aid in constructing Bayesian networks. In [9], the authors first constructed dependency networks because such networks are often easier to learn than Bayesian networks. They then used an oracle to construct the Bayesian networks from the dependency networks. Our work is similar in that we are using a model containing relationships between tests and faults (i.e., dependencies) to construct Bayesian networks, except that in our case, we the dependencies are specified using D-Matrices derived from fault trees. In addition, this is the first paper to analyze the performance of these Bayesian networks derived from FIMs.

Given a FIM, one can construct a D-Matrix representing the fault and test relationships in the FIM. From the D-Matrix derived from the FIM, one can then construct a bipartite Bayesian network, similar to the Bayesian network presented

978-1-4673-1813-6/13/$31.00 ©2013 IEEE.
[1] IEEEAC Paper #2714, Version 03, Updated 29/01/2013 .

in QMR-DT [10], [11]. However, it has been shown that it is not feasible to derive a full D-Matrix from a FIM [2]; therefore, any Bayesian network derived from such a D-Matrix will not represent the entire set of relationships between faults and tests in the system.

Despite this limitation, we hypothesize that bipartite Bayesian networks derived from FIMs can still perform reasonably well. To test this hypothesis, we derive several FIMs, each from multiple D-Matrices. Using these FIMs, we then generate the resulting Bayesian network by converting the FIM to a partial D-Matrix and then the D-Matrix to a Bayesian network. Additionally, we create a Bayesian network from the original D-Matrix, treating that network as "ground truth." We then evaluate the Bayesian networks created from the FIMs by comparing them to the Bayesian network derived from the original D-Matrix and report on several measurements, such as accuracy, relative probability of target diagnoses, diagnosis rank, and KL-divergence. In our results, we found that even with the incomplete information derived from the FIM, fairly high diagnostic accuracy results when using the corresponding Bayesian networks.

The rest of the paper is organized as follows. Section 2 gives a formal definition of D-Matrices, FIMs, and Bayesian networks while Section 3 gives a formal algorithm on how to derive Bayesian networks from FIMs. The experiments in the paper are presented in Section 4 with the results in Section 5. Finally, analyses of the results are in Section 6 and conclusions in Section 7.

2. Diagnostic Models

There are several ways to perform fault diagnosis [1]. Some of the most common diagnostic algorithms are rule-based algorithms, such as those in [12]. Fault dictionaries are another diagnostic method that have successfully been applied to various systems [13]. Fault trees have also been merged with rule-oriented reasoning, which allows for locating and identifying failures [14]. As we have already discussed, Fault Isolation Manuals (FIMs) are a type of decision tree or fault tree that provide another means to perform fault diagnosis. Model-based methods, which compare observations from the system with those expected from a model, provide another common set of approaches to performing system-level diagnosis [15]. Another diagnostic algorithm that has been extensively used for fault diagnostics and prognostics are Bayesian networks, which allow for a compact representation of probability distributions [10], [16]. Diagnostic models and Bayesian networks can be represented as D-Matrices [1]. D-Matrices are matrix representations of the relationships between faults and tests (or alarms) in a diagnostic model [1]. The rest of this section gives a more detailed explanation of D-Matrices, FIMs, and Bayesian networks.

D-Matrices

One way diagnostic models can be represented is with D-Matrices. A D-Matrix relate the faults and the tests monitor or observe those faults. We can formally define it as the following: Let $\mathbb{F}$ represent a set of faults and $\mathbb{T}$ represent a set of tests. Assume each $\mathrm{F}_i \in \mathbb{F}$ is a Boolean variable such that $eval(\mathrm{F}_i) \in \{0, 1\}$ and each $\mathrm{T}_j \in \mathbb{T}$ is also a Boolean variable such that $eval(\mathrm{F}_i, \mathrm{T}_j) \in \{0, 1\}$. We define $eval(F_j, T_j)$ to then be the following:

$$eval(\mathrm{F}_i, \mathrm{T}_j) = \begin{cases} 1 & \text{if } \mathrm{T}_j \text{ detects } \mathrm{F}_i \\ 0 & \text{otherwise} \end{cases}$$

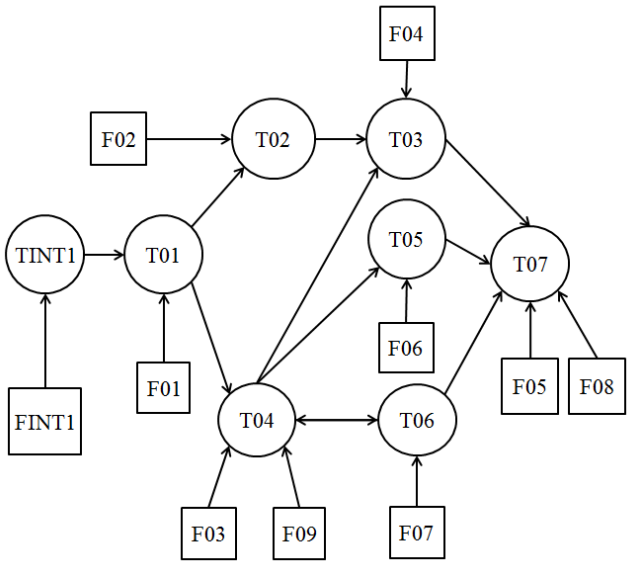


Figure 1: A logic diagram for the Simple Model from Simpson and Sheppard [1].

Table 1: The D-Matrix for Figure 1.

	TINT1	T01	T02	T03	T04	T05	T06	T07
FINT1	1	1	1	1	1	1	1	1
F01	0	1	1	1	1	1	1	1
F02	0	0	1	1	0	0	0	1
F03	0	0	0	1	1	1	1	1
F04	0	0	0	1	0	0	0	1
F05	0	0	0	0	0	0	0	1
F06	0	0	0	0	0	1	0	1
F07	0	0	0	1	1	1	1	1
F08	0	0	0	0	0	0	0	1
F09	0	0	0	1	1	1	1	1
NF	0	0	0	0	0	0	0	0

Then a diagnostic signature is defined to be the vector

$$\mathbf{F}_i = \left[eval(\mathrm{F}_i, \mathrm{T}_1), \ldots, eval(\mathrm{F}_i, \mathrm{T}_{|\mathbb{T}|})\right]$$

A D-Matrix is then defined to be the set of diagnostic signatures $\mathbf{F}_i$ for all $\mathrm{F}_i \in \mathbb{F}$ [17]. Rows represent faults and columns represent tests. The ith column corresponds to test T_i in the diagnostic model. Table 1 shows an example D-Matrix representing the logic diagram of the diagnostic model in Figure 1. In the Simple Model there are 7 tests, 10 faults including no fault found (which is not shown in the logic diagram), and 1 testable input.

Fault Isolation Manuals

Fault Isolation Manuals (FIMs) provide diagnostic strategies using structured sequences of tests to diagnose the root cause of fault codes that are reported by on-board diagnostic systems [1]. At each step, a test is performed and depending on the test outcome (pass or fail), the FIM then gives the next test that should be performed. This process is repeated until a single fault or ambiguity group is diagnosed as the cause for the fault code.

FIMs are often represented as fault trees, which is a particular kind of decision tree.[2] A decision tree is a tree data structure

[2]The term "fault tree" has also been used to define a tree-like structure in

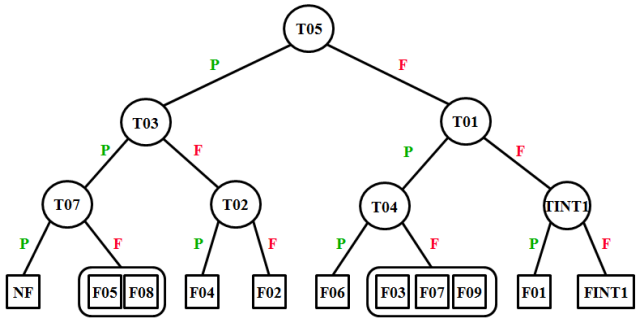


Figure 2: An example FIM derived from the D-Matrix in Table 1.

composed of internal decision nodes and terminal leaves [18]. At each decision node, there is a test function with a set of outcomes. Given an input at each decision node, the test function is used and a branch from the decision node is taken corresponding to the outcome of the test function. To classify a single data instance, the process begins at the root node and then evaluates the attributes by applying the test to the data point being classified. The tree is then traversed along the edges to the next appropriate test until a leaf node is reached. The leaf node's label is then assigned to the current instance that is being classified.

In machine learning, decision trees are often built using information gain or the Gini index [18]. Given a set of data, the learning algorithm selects tests to split members of a current ambiguity group using the expected amount of information the tests provide. Unless derived from a a system model such as a D-matrix, fault trees are usually built by a system expert who has an understanding of how the system being tested is designed, including the relationships between tests and faults. The downside to this approach is that a technician using the resulting tree could encounter difficulties if he or she wants to modify the fault tree when new data becomes available. This is due to the fact that the existing algorithms for incremental updating of decision trees need to keep track of entropy values for each test and modify the structure of the network if the entropy values require a reordering of the tests [3], [19]. With fault trees built by domain experts, there is no way to know what entropy values to start with; therefore, model maturation becomes a difficult problem. This is one of the motivations for converting FIMs to Bayesian networks.

An example FIM is shown Figure 2 built from the model shown in Figure 1. In this example, one test is not used (T06) and two diagnoses result in ambiguity groups (which have been circled). To use the FIM, the user would first run test T05. Based on whether test T05 Passes (P) or Fails (F), the user would run either test T03 or T01. Based on the outcome, the user would then run the next test designated in the FIM. The process would continue until a leaf in the tree is reached.

Bayesian Networks

A Bayesian network is a graphical model that can be used to represent a joint probability distribution in a compact way [20]. In addition, it enables a person to explicitly view relationships between random variables in the probability distribution. Specifically, let $\mathbf{X}= \{X_1, \ldots X_n\}$ be a set of random variables where we want to represent the joint probability $\mathbf{P(X)}= (X_1, \ldots X_n)$ as efficiently as possible. Using the product rule, we can factor **P(X)** as

$$\mathbf{P(X)} = (X_1, \ldots X_n) = \mathbf{P}(X_1) \prod_{i=2}^{n} \mathbf{P}\left(X_i | X_1, \ldots X_i - 1\right).$$

The problem with representing a probability distribution in this way is that each factor of the joint distribution is represented by a probability table whose size is exponential in the number of variables. However, we can make use of conditional independence between variables to represent the probability distribution in a more efficient way.

A variable X_i is conditionally independent of variable X_j given X_k if

$$\mathbf{P}(X_i, X_j | X_k) = \mathbf{P}(X_i | X_k)\mathbf{P}(X_j | X_k).$$

This means that the event represented by the random variable X_j has no effect on the knowledge of the event X_i if we know something about the event X_k. Using conditional independence, we can rewrite **P(X)** as the following:

$$\mathbf{P}(X_1, \ldots, X_n) = \prod_{X_i \in \mathbf{X}} \mathbf{P}\left(X_i | \text{Parents}(X_i)\right).$$

Bayesian networks are directed acyclic graphs that model the conditional independences in a probability distribution [20]. Formally, we can represent it as $\mathcal{B} = \langle \mathbf{X}, \mathbf{E}, \mathbf{P} \rangle$ where:

- **X** is a set of vertices corresponding to the random variables of the distribution,
- **E** is a set of directed edges (X_i, X_j), where the source of the edge corresponds to X_i, the destination of the edge corresponds to X_j, and the edge represents a conditional dependence relationship of X_j on X_i,
- **P** is a set of conditional probability tables $P(X_i|\text{Parents}(X_i))$, where each entry provides the probability of X_i given the set of parents of X_i.

Given a Bayesian network, a user can query the network for the probability of an event occurring by performing inference. In addition, the user can assign evidence to the graph and then perform inference. There exist several algorithms for performing exact inference in Bayesian networks, such as the clique tree algorithm or the message passing algorithm [20] [21]. These algorithms will return the exact probability of the event that is being queried, given the evidence that has been set.

Unfortunately, exact inference in Bayesian networks is NP-complete [20]. In cases where the Bayesian network is too large or complex, a person can use approximate inference algorithms, such as stochastic sampling. These algorithms use a sampling technique that, depending on how the samples were taken, may return a different probability each time the algorithm is run [20] [21]. While approximate algorithms may not return the exact probability when querying the Bayesian network, they have the advantage of not facing the computational complexity that exact inference algorithms may encounter on complex networks.

One of the most basic Bayesian networks used for fault diagnosis is a bipartite network. In these networks there are only two types of nodes: fault nodes and test nodes. In this network, faults that are detected by a test are represented as

which a series of inputs and Boolean logic gates are used to determine if a failure occurred in a system. In this paper we refer only to the definition of fault trees above [6].

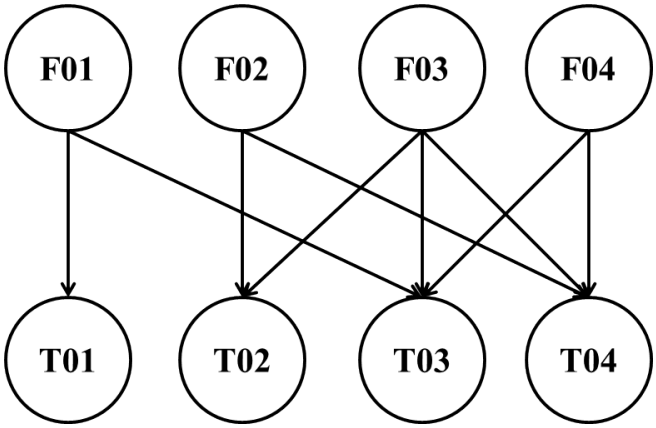


Figure 3: An example diagnostic Bayesian network. The F's represents faults and the T's represent tests.

Table 2: A D-Matrix corresponding to Figure 3.

	T01	F02	F03	F04
F01	1	0	1	0
F02	0	1	0	1
F03	0	1	1	1
F04	0	0	1	1

parents of the corresponding test node. If there is no link between a fault and a test then the test does not detect the presence of the fault. In addition, each test node has an associated probability table. The network in Figure 3 is a Bayesian network that might be used for fault diagnosis [11].

In this example, there are four faults: F01, F02, F03, and F04, and four tests: T01, T02, T03, and T04. To diagnose a fault, a person would run a subset of the tests and record the outcome of each test. The results of the tests would be applied as evidence to the Bayesian network. Finally, the user would query the fault nodes to see which had the highest probability of occurring. Note that there is no requirement of how many tests must be run; however, more tests will often lead to a higher confidence in the final diagnosis.

Diagnostic Bayesian networks, like the one in Figure 3, can also be represented as a D-Matrix; however, the D-Matrix only captures structural relationships between faults and tests and does not capture any of the probabilistic information [11]. Table 3 shows the D-Matrix for the Bayesian network in Figure 3.

Because of the computational complexity of developing and using Bayesian networks, Noisy-OR nodes are a special type of node that applies additional assumptions to simplify the network. First, a noisy-OR node is presumed FALSE if all of the node's parents are FALSE. This assumption is referred to as accountability. The second assumption, called exception independence, is that each of the node's parents influence the node independently. This also implies that the mechanism that inhibits the noisy-OR node for one parent is independent of the mechanism that inhibits all of the other parents of the node [20] [21]. The end result is that the size of the probability table for a noisy-OR node grows linearly instead of exponentially as the number of parents increases.

Figure 4 gives a schematic representation of a noisy-OR node [21]. In the figure the U_i's represent the parents of X.

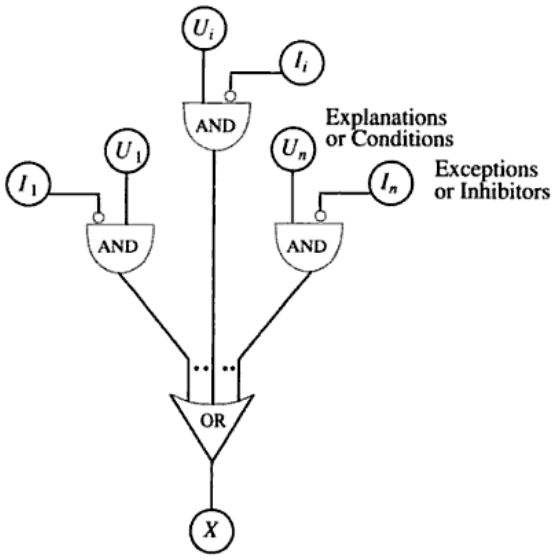


Figure 4: A Noisy-OR Node [21].

The I_i's represent exceptions that interfere with the normal relationship between the parents of X.

Let I_i represent the inhibitor of the parent U_i. We can then let the value q_i represent the probability of inhibitor I_i occurring. If only one parent U_i is TRUE while all others parents are FALSE, X will be TRUE if and only if the inhibitor I_i is FALSE.

$$P(X = \text{ TRUE } | U_i = \text{ TRUE}, U_k = \text{ FALSE }, k \neq i) = 1 - q_i$$

Therefore, the value $c_i = 1 - q_i$ represents the probability that an individual parent U_i is TRUE that also causes the node X to be TRUE. Let $T_{\mathbf{u}}$ represent any possible assignment $\mathbf{u}$ of TRUE to a subset of the parents of X.

$$T_{\mathbf{u}} = \{i : U_i = \text{TRUE}\}$$

X will be FALSE if and only if all of the inhibitors for the parent nodes which are in $T_{\mathbf{u}}$ are also TRUE. In other words, X will be FALSE if inhibitors for parents that are TRUE are also TRUE. Therefore, we can write the following to represent X being false given the assignment $\mathbf{u}$ as

$$P(X = \text{FALSE} \mid \mathbf{u}) = \prod_{i \in T_{\mathbf{u}}} q_i.$$

This means that the probability for X to be FALSE given a set of its parents being TRUE is simply the product of all of the inhibitors for the parents that are TRUE. Writing in a more general form, we have the following

$$P(X|\mathbf{u}) = \begin{cases} \prod_{i \in T_{\mathbf{u}}} q_i & \text{if } X = \text{FALSE} \\ 1 - \prod_{i \in T_{\mathbf{u}}} q_i & \text{if } X = \text{TRUE} \end{cases}$$

In addition, there is a Leak value λ for each noisy-OR node which accounts for the probability of X being in the TRUE state given no evidence from any of the other parents. We can then rewrite the above equations as the following.

$$P(X|\mathbf{u}) = \begin{cases} (1 - \lambda) \times \prod_{i \in T_{\mathbf{u}}} q_i & \text{if } X = \text{FALSE} \\ 1 - \left[(1 - \lambda) \times \prod_{i \in T_{\mathbf{u}}} q_i\right] & \text{if } X = \text{TRUE} \end{cases}$$

Once inference algorithms are adopted to account for noisy-OR nodes, Bayesian networks with noisy-OR nodes can be used as regular Bayesian networks. However, unlike inference in a regular Bayesian network, the complexity of exact inference in a Bayesian network with noisy-OR nodes grows exponentially with the number of nodes that are assigned positive evidence [22].

Noisy-OR nodes are often used for fault diagnosis because they correspond to assumptions typically made, such as there being only a single fault. For example, in the network in Figure 3, one could replace the test nodes with noisy-OR nodes. This makes defining the conditional probability tables of the test nodes much more manageable, especially if the there are a large number of faults being modeled.

3. CONVERTING FIMS

While there has been work on deriving D-Matrices from fault trees and building Bayesian networks from D-Matrices, there has not been any work that has formally defined how to generate Bayesian network directly from fault trees. In this section, we explain how this process can be performed following the two-step process. Our approach is derived from Timothy Bearse's paper that provides a general outline for deriving a D-Matrix from a FIM [2]. Using those matrices, we then build a bipartite Bayesian network with noisy-OR nodes to retain the semantics of the D-matrix. In the remainder of this section we give details for how these Bayesian networks are built, including how to set the conditional probability tables for each node.

Deriving D-Matrices From FIMs

In [2], Bearse gives an informal process for deriving the partial information present in a fault tree. Here we describe a formal algorithm that implements this process. The algorithm starts with an empty D-Matrix with all tests and faults from the FIM. Beginning at each leaf node, which in the FIM corresponds to a fault node, the algorithm traverses up to the parent node. Depending on whether the edge traversed up to the test corresponds to the parent test passing or failing, a 0 (Pass) or 1 (Fail) is inserted into the D-Matrix for the index for the corresponding fault and test. For example, in the FIM in Figure 2, starting at NF, the algorithm would traverse to the parent node T07 through test T07's Pass link. Therefore, in the D-Matrix a 0 is inserted into the D-Matrix for NF and T07. Next, the algorithm traverses to the next parent node, where the process is repeated. In our example, this would mean traversing from T07 up to T03 through T03's Pass link and therefore a 0 is inserted into the D-Matrix between T03 and NF. This is performed until the algorithm reaches the root node of the FIM and then this process is repeated for all fault nodes in the FIM. The completed D-Matrix derived from the FIM in Figure 2 is shown in Table 3. The D-Matrix shown here assumes symmetric inference.

As shown, after the algorithm has finished, there will still be empty entries in the D-Matrix. In these entries, it is unknown whether the test should detect the fault (1) or not (0). Bearse argues that because of these empty entries, asymmetric inference should be used instead of symmetric inference. However, no work has been done to empirically analyze how erroneous, if it all, the D-Matrices are when symmetric inference is used.

Table 3: The D-Matrix derived from Figure 2.

	TINT1	T01	T02	T03	T04	T05	T06	T07
FINT1	1	1				1		
F01	0	1				1		
F02			1	1		0		
F03		0			1	1		
F04			0	1		0		
F05				0		0		1
F06		0			0	1		
F07		0			1	1		
F08				0		0		1
F09		1			1	1		
NF				0		0		0

Table 4: The conditional probability table for test TINT1 using a noisy-OR node.

	Parent	FINT1	F01	Leak
	State	True	True	
TINT1	Pass	0.001	0.999	0.999
	Fail	0.999	0.001	0.001

Building Bayesian Networks From FIMs

Using the partial D-Matrix, we can build a complete D-Matrix similar to the bipartite network described above. To do so, we instantiate each fault as a regular random variable and each test as a noisy OR node. If a relationship does exist in the D-Matrix between a fault and test, a link is added from the fault to the test. Finally, the conditional probability tables for the nodes have to be set. For the priors of the fault nodes, one can set all of the probabilities based on the corresponding failure frequency. However, these probabilities are often very small, and in the Bayesian network after evidence is assigned to the test nodes, the probabilities of the fault occurring will still be very small. This is because in the network with priors set so low, there is still the possibility that there is no fault in the system. If we assume that there is a fault in the system, then we must adjust the priors by normalizing the probability of each fault over the entire set of faults. This means that the probability of each fault being True is set according to the formula:

$$\mathbf{P}(F_i) = \frac{\text{FailureFrequency}(F_i)}{\sum_{\forall F_j \in textbf{F}} \text{FailureFrequency}(F_j)}.$$

Uniform priors can also be used if there is no clear information on failure frequency.

For the test nodes, we want to define the probability of the test failing given a particular fault to be True to be relatively high. For this study, we used a value of 0.999 for the test failing if the particular test detects the fault. Likewise, we use a value of 0.999 for the test passing if the test does not detect the fault. Also, a Leak value must be set, which we set to 0.001, which means that the probability of the test Failing, given there is actually no fault in the system is 0.001 An example table for the test TINT1 is show in Table 4

While it is possible to first derive a D-Matrix from a FIM and then build a bipartite Bayesian network, it is also possible to build a Bayesian network directly from the FIM. We present the formula in Algorithm 1. The algorithm takes in a FIM and first instantiates all of the faults in the FIM as general random variables in the network (lines 3-7). This is also where the priors of the fault nodes are set. Next, the algorithm iterates over the entire set of test nodes in the FIM and instantiates the fault node as a noisy-OR node in the Bayesian network. For

Algorithm 1 Build Bayesian Network

1: // FT is the fault tree
2: // B is the Bayesian network
3: **for all** Faults $F_i \in FT$ **do**
4: $F_i \leftarrow$ CreateNode
5: $F_i \leftarrow$ SetCPT
6: FT.AddFaultNode(F_i)
7: **end for**
8: **for all** Tests $T_i \in FT$ **do**
9: $T_i \leftarrow$ CreateNoisyORNode
10: FT.AddTestNode(T_i)
11: **for all** Faults $F_j \in FT$**do**
12: **if** T_i indicts F_j by pass link **then**
13: T_i.AddParent(F_j)
14: T_i.ProbOfPassGivenFault(F_j) $= 0.999$
15: T_i.ProbOfFailGivenFault(F_j) $= 0.001$
16: **end if**
17: **if** T_i indicts F_j by fail link **then**
18: T_i.AddParent(F_j)
19: T_i.ProbOfPassGivenFault(F_j) $= 0.001$
20: T_i.ProbOfFailGivenFault(F_j) $= 0.999$
21: **end if**
22: **end for**
23: T_i.ProbOfLeak $= 0.001$
24: **end for**
25: **return** B

each test node, the algorithm then iterates over all fault nodes (line 11). If the fault is indicted by the test then a link is added from the fault to the test node. For example, in Figure 2, test T07 indicts NF, F05, and F08. The two if statements at lines 12 and 17 determine how to set the probabilities in the test node. In our example, since T07 indicts F05 and F08 through the failure link, the algorithm would execute lines 18-20 for test T07 and faults F05 and F08. Finally, the leak value for the test node is set.

4. EXPERIMENTS

To empirically evaluate the bipartite Bayesian networks derived from FIMs, we use two diagnostic models from Simpson and Sheppard [1]. The first model, which is referred to as the "Simple Model" is shown in Figure 1 that contains seven tests, nine faults, one testable input, and one feedback look. The second is a model of a hypothetical missile launcher shown in Figure 5 that contains 18 tests, 21 faults, 2 testable inputs, two untestable inputs, and two feedback loops. This model will be referred to as the "Missile Model" for the remainder of the paper.

Given the D-Matrices for the two models, we derive fault trees from each. We first use ID3 to build a fault tree that is fairly balanced. The second tree is also built using ID3. However, instead of choosing a test split that maximizes information gain, we set the algorithm to split on the tests that minimize the information gain, resulting in very biased trees. For example, the FIM in Figure 6 is derived from the D-Matrix in Table 1.

The reason for building the biased trees is because real FIMs will not always be perfectly balanced and this allows us to see how a Bayesian network derived from an unbalanced tree performs. In building the trees, there are cases where the biased and balanced trees contain different sets of tests. To make each tree as comparable to the other, we manually enforce that the biased and balanced trees contain the exact same set of tests. In addition, if there is an ambiguity group in the tree, we combine the faults into one node. Also, since in the Bayesian networks we always assume that a fault has occurred, we remove the NF from the set of faults in the Simple Model. For the balanced tree in Figure 2, we would remove the fault NF and test T07; faults F05 and F08 would then be linked directly to test T03.

Given the biased and balanced tree for each model, we then derive a biased and balanced Bayesian network for each model. In addition, for each full D-Matrix, we derive a Bayesian network with the same basic structure as the Bayesian networks derived from the FIMs. However, since these networks were derived from the full D-Matrix, there will be no missing relationships in the Bayesian network. This provides ground truth so that we can compare the FIM-based Bayesian networks. Using these networks, we then performed three different sets of experiments.

Accuracy of Test Sequences

In the first set of experiments, we look at how each tree performed over every possible sequence of tests. For each fault in the Bayesian network, we take all possible combinations of tests using the D-Matrix. For each combination of tests for the fault, we apply to the corresponding test the appropriate test outcome. We then query the Bayesian networks to see which fault is diagnosed and compare the results with the D-Matrix to see if the diagnosis is correct. In these experiments we plot the number of tests that were applied versus the following measurements:

- **Accuracy:** The ratio of the total number of times the DFIM correctly diagnosed a fault divided by the total number of test combinations for every fault. In certain cases the DFIM will diagnose several faults all with the same probabilities. In this case, if the correct test is still in the top ranked ambiguity group, we count that as being a correct diagnosis.
- **Rank:** The average rank of the correct fault. If there are ambiguity groups in the DFIM, the faults are grouped into ambiguity groups and the rank of the ambiguity group containing the correct fault is then reported.
- **Probability of Fault:** The posterior probability of the fault after the set of tests have been applied to the network.
- **Probability of Top Ranked Fault:** The probability of the fault that is ranked the highest by the DFIM.
- **Probability Difference:** The difference in probability between the correct diagnosis and the highest ranked diagnosis.
- **KL-Divergence:** A measure of the difference between two probability distributions P and Q, calculated as

$$D_{\mathrm{KL}}(P||Q) = \sum_i \log_2 \left(\frac{P(i)}{Q(i)} \right) P(i),$$

where P usually represents the "true" distribution and Q is the distribution being compared to P. In our experiments, P is based on the distribution derived from the full D-Matrix.

Degradation of Bayesian Networks

In the second set of experiments, we sought to observe how quickly the performance of networks derived from FIMs may degrade. To do so, we started with the Bayesian network derived from the full D-Matrix. We then randomly deleted links between faults and tests to simulate the effects of different paths being generated in a FIM and then ran the same set of experiments described in the previous section.

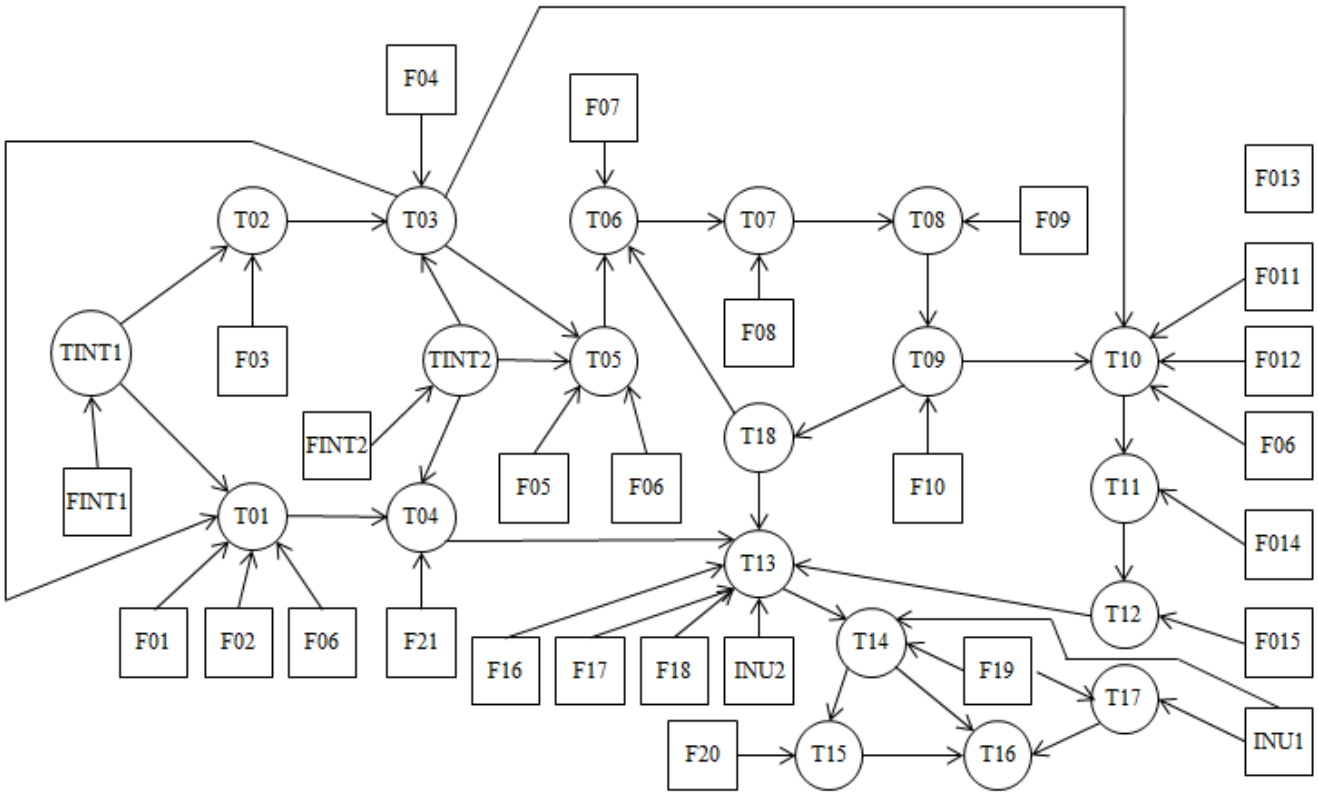


Figure 5: A logic diagram for the Anti-Tank Missile Launcher Model from Simpson and Sheppard [1].

In randomly deleting the links from the network, we enforced that none of the faults would become completely isolated, i.e., that there was always at least one test connected to the fault in the Bayesian network. We performed the above process 10,000 times and averaged the results. For the sake of brevity, we only present the results for accuracy, rank, and KL-divergence since they offer a good summary of performance of the network. Because of the exponential explosion in the number of different test combinations, running these experiments on the Missile Model was infeasible; therefore, only the results for the Simple Model are shown.

Costs of Test Sequences

Finally, we examined the average cost to run a set of tests for each network. For this set of experiments, we selected tests based on some cost criterion, assigned outcomes to the tests, and repeated until a diagnosis could be made. More specifically, we evaluated two different ways to select tests in for the networks. First, tests were selected that minimized the entropy (maximized information gain). Second, we incorporated a cost metric into the entropy calculation by dividing the gain by the test cost:

$$gainDivCost(T_i) = \frac{gain(T_i)}{\alpha * cost(T_i)}.$$

where α is a weight factor that controls how much influence cost has on the test selection.

To build a test sequence, we first select the test ranked the highest according to the given test selection method. We then assign that test a Pass outcome. The entropy for the tests are recalculated and the highest ranked test is selected and also assigned a Pass outcome. This process is repeated until there are no more tests to run or if the difference between the top ranked fault and second ranked fault reaches a user specified threshold. After reaching the stopping criterion, the faults are queried to find the top ranked fault. The sequence

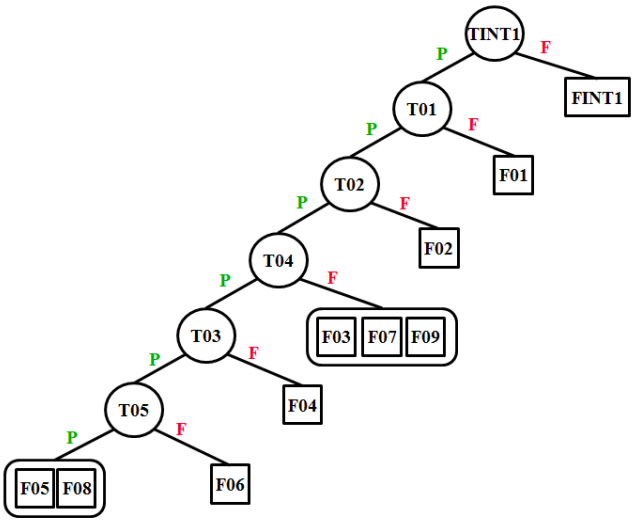


Figure 6: Biased tree derived from Table 1.

of tests and corresponding diagnosed fault are then stored to be evaluated later. To build another test sequence, the last selected test outcome that had a Pass outcome is changed to a Fail outcome. Also, all of the tests that were performed after the test outcome that was changed have the evidence removed. The process for selecting the highest ranked test is repeated and the outcome is set to Pass. Again, if the stopping criterion is met, the test sequence and highest ranked fault are recorded. In performing this test sequence generation, we enumerate all possible combinations of tests and test outcomes. However, in certain cases, the assignment of test outcomes will result in a fault signature that is inconsistent with the D-Matrix. In those cases, the test sequence will not traverse down that branch of the tree.

We assigned random values between 0 and 1 for the costs in the Simple Network. For the Missile Model, we used the costs specified in Simpson and Sheppard [1] and normalized the values to be between 0 and 1. Once we had all of the test sequences, we calculated the average cost of the diagnostic strategy.

For these experiments, we considered two additional variations. In the first, we assumed all the fault prior probabilities were uniform. For the second, we use non-uniform priors. For the Simple Model with non-uniform priors, we generated a single random set of failure rates, and the threshold for the difference between the top ranked fault and the next ranked fault was set to 0.50. For the Missile Model, we use the normalized failure rates specified by Simpson and Sheppard [1]. A higher threshold rate of 0.95 was used in this case. When lower threshold values were used, they resulted in not all faults being diagnosed in a test sequence. Below, we present the average test sequence cost, accuracy, and the length of the test sequences.

5. RESULTS

For all experiments we use the Java plug-in for SMILE [23]. Note that in SMILE, noisy-OR nodes are only used as a modeling tool; when inference is performed; the full conditional probability tables are generated from the user defined noisy-OR tables. For inference, we used the Lauritzen clique tree algorithm [24], which is the preferred algorithm in SMILE for performing exact inference [23].

Accuracy of Test Sequences Results

The accuracy-based results for the Simple Model are shown in Figure 7, and the corresponding results for the Missile Model are shown in Figure8. In the figures, "D-Matrix" denotes the Bayesian network derived from the full D-Matrix while "Biased" and "Balanced" denote the Bayesian networks derived from the based and balanced trees, respectively.

Here we see that the D-Matrix models always correctly diagnose the expected faults, thereby justifying their use for ground truth. This is because each model has all of the links from the D-Matrix, so any assignment of tests will always raise the probability of the correct fault. In the Simple Model, the Balanced network performed second best with the Biased network performing the worst. However, in the Missile Models, the performance of the Biased and Balanced networks are almost the same for test sequences of length 4 to 15. In the Simple Model, the performance of the Biased and Balanced networks degrades from running only 1 test to running 2 tests but then slowly improves. Meanwhile, the accuracy of the Biased and Balanced networks for the Missile Model decreases dramatically from about 1 to 4 tests in the sequence and slowly increases as more tests are performed.

We see a similar set of results for the average Rank in Figures 7b and 8b, where the D-Matrix always ranks the correct fault either first or in the top ambiguity group. In addition, the Biased network performs the worst while the Balanced network performs second best. For the Biased and Balanced networks derived from the Missile Model, the average rank of the correct fault increases as more tests are performed up to 8 tests, as which point the rank begins to decrease.

When considering the probability of the correct fault on the Simple network, the highest ranked fault, and the difference between the two are shown in Figures 7c, 7d, and 7e. The probability of the correct fault increases in a linear fashion for both the D-Matrix, Biased, and Balanced networks. For the Biased and Balanced networks, the probability of the top ranked fault is higher than that of the correct fault but then slowly levels out. In particular, this can be seen when looking at the probability difference. The results for the Missile Model, shown in Figures 8c, 8d, and 8e, are very similar to that of the Simple Model. The probability of the correct fault increases linearly for the D-Matrix, Biased, and Balanced networks while the probability for the top ranked fault increases in a logarithmic fashion for the Biased and Balanced networks. The probability difference results show an increase in the difference between the correct fault and true fault in test sequences of length 1 to 5, but then a steady decrease occurs in sequences from 6 to 16 to tests.

Finally, we have the KL-Divergence results for the Simple Model in Figure 7f and the Missile Model in Figure 8f. Based on these results, we can see that the Balanced network's distribution is closer to that of the D-Matrix network than that of the Biased network for both the Simple and Missile Models. In addition, the difference between the distributions increases as the number of tests run is increased.

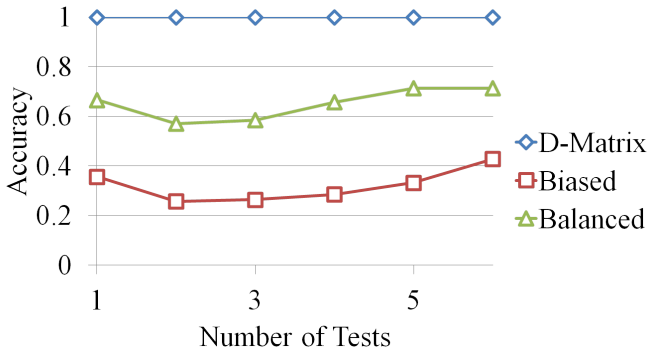
Degradation of Models Results

The results for the model degradation experiments are shown in Figure 9. Here, the x-Axis indicates the percent of the links that have been removed from the D-Matrix model. In Figure 9a we see that, as we remove links from the network, the accuracy decreases. In addition, the rank (Figure 9b) increases as the number of links are removed. Finally, we see that the KL-Divergence (Figure 9c) increases as the number of links are removed.

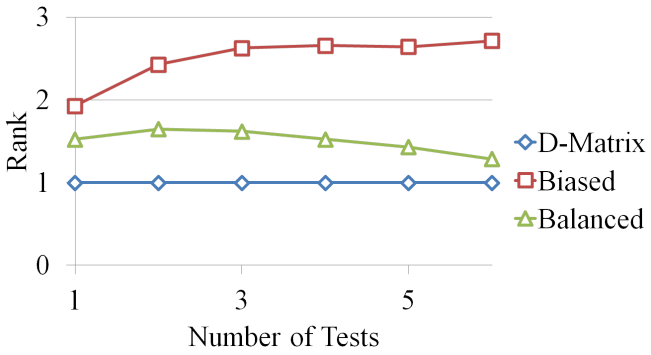
Average Costs

For the final set of experiments, we show the accuracy and the average cost of the test sequences. In addition, we report on the counts of the number of test sequence lengths, which allows a user to see the distribution of tests that need to be performed. Table 5 shows the sequence lengths and the number of sequences of length x for both test selection methods, along with the average test sequence cost and accuracy for the Simple Models. Table 6 shows the same results for these networks; however, these networks do not have equal priors on the faults.

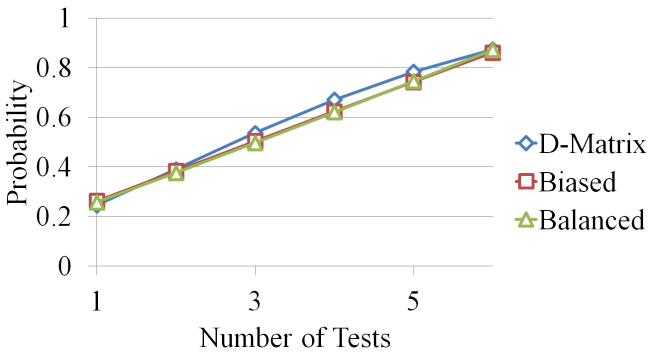
As shown in Table 5, the distribution of the sequence lengths does not appreciably when comparing the test selection methods. However, including test cost in the selection process does yield a slightly lower average cost (as one would expect).



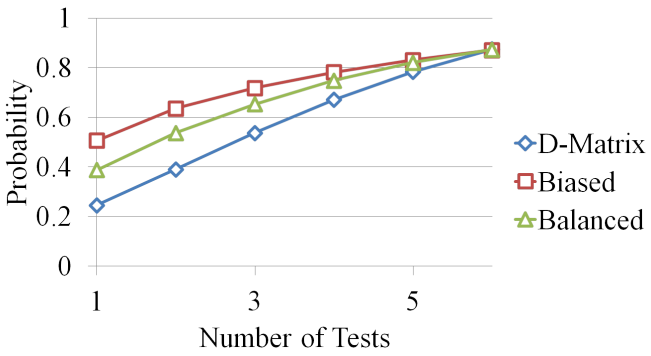
(a) Accuracy



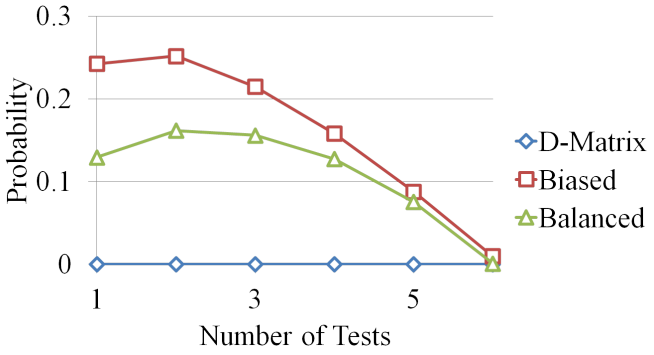
(b) Rank



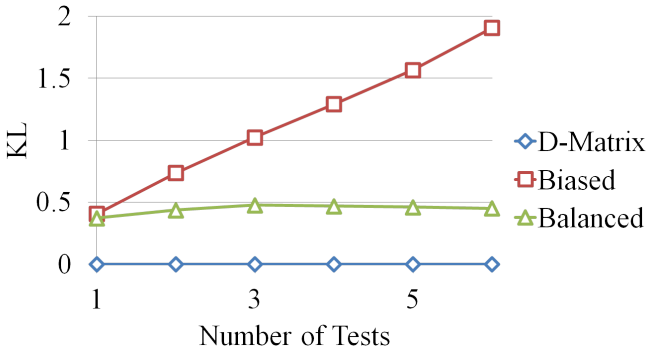
(c) Probability of Fault



(d) Probability of Top Ranked Fault



(e) Probability Difference

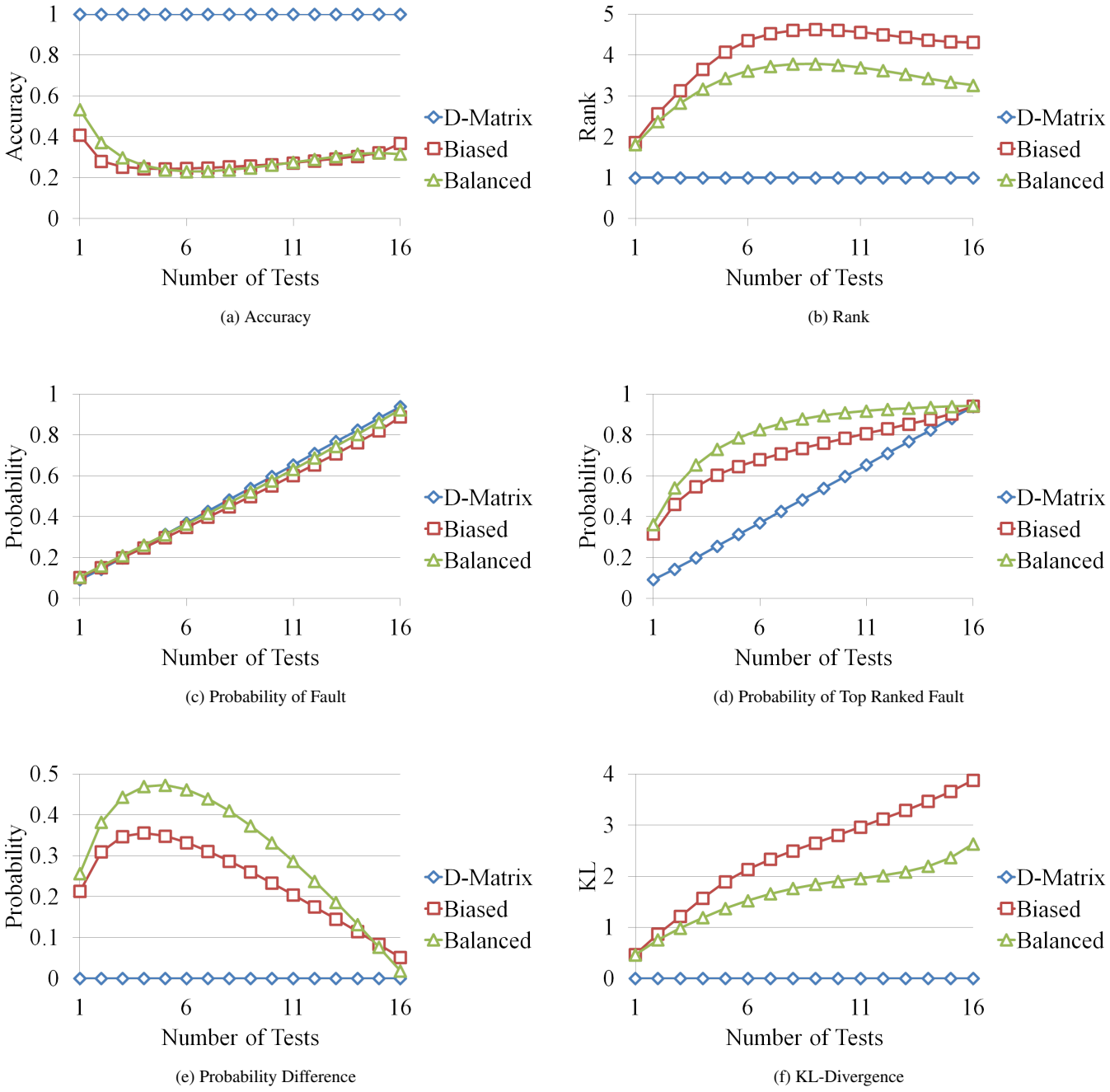


(f) KL-Divergence

Figure 7: Simple Model Results

Table 5: Number of Test Sequence Lengths for the Simple Model with Equal Priors

Number of Tests	No Costs			With Costs		
	D-Matrix	Biased	Balanced	D-Matrix	Biased	Balanced
1	1	1	1	1	1	1
2		1			1	1
3	3	1	3	3	1	
4		1			1	2
5	1	1	1	1	1	1
6	6	2	6	6	2	6
Avg Cost:	1.54	1.78	1.72	1.49	1.77	1.72
Accuracy:	1.00	1.00	1.00	1.00	1.00	1.00



(a) Accuracy

(b) Rank

(c) Probability of Fault

(d) Probability of Top Ranked Fault

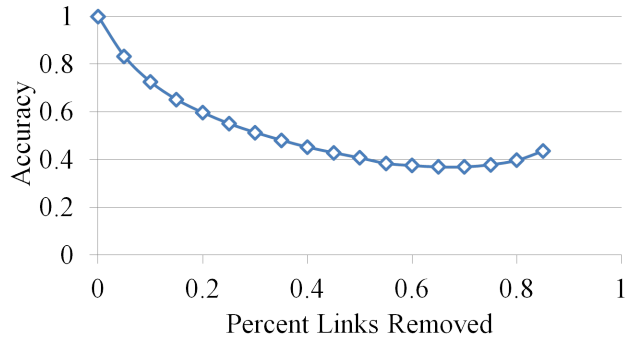
(e) Probability Difference

(f) KL-Divergence

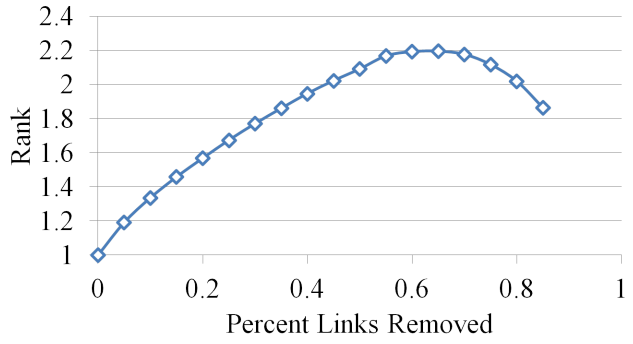
Figure 8: Missile Model Results

Table 6: Number of Test Sequence Lengths for the Simple Model with Unequal Priors

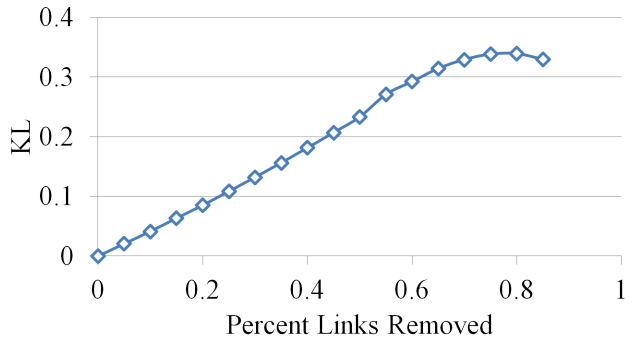
Number of Tests	No Costs			With Costs		
	D-Matrix	Biased	Balanced	D-Matrix	Biased	Balanced
1		1			1	1
2	1	1	1	2	1	
3	3	1	5	3	1	3
4	2	1		4	1	1
5		1	1	5	1	1
6	1	2	4	6	2	2
Avg Cost:	1.99	1.58	1.95	1.91	1.58	1.98
Accuracy:	1.00	0.80	0.86	1.00	0.80	0.86



(a) Accuracy



(b) Rank



(c) KL-Divergence

Figure 9: Degradation Results for the Simple Model.

When the priors of the faults are unequal (Table 6), the average cost of a test sequence increases for the D-Matrix and Balanced networks while decreasing for the Biased network. In addition, the Biased and Balanced networks no longer perform with 100 percent accuracy. Another interesting result is that the average cost of the test sequences for the Biased network is the same regardless of the test selection method. We also see that the difference between the average cost for the two test selection methods on the D-Matrix and Balanced is very small.

The results for the Missile Models with equal priors are shown in Table 7 while the Missile Model results with non-equal priors are in Table 8. With equal priors, there is no difference between the two test selection methods. In addition, for all of the models, all 16 tests had to be performed since the difference between the top ranked fault and the second top ranked fault never exceeded the threshold. When a lower threshold was used, we encountered a problem that some faults were never diagnosed because the stop criterion was met too early. Finally, the accuracy of the networks is lower for the networks with unequal priors because a fault with a low probability for a prior is harder to diagnose.

6. CONCLUSIONS

In this paper, we explored a procedure for deriving Bayesian networks from Fault Isolation Manuals (fault trees) and investigated the accuracy of this procedure. From our experiments, we can conclude that deriving Bayesian networks from fault trees will not perform as well as Bayesian networks derived from the full D-Matrix because there are links missing in the Bayesian network. This is not surprising; however, this is the first study to our knowledge that attempts to explore this issue in a systematic way.

Despite the fact that the Bayesian networks derived from FIMs do not perform as well as the Bayesian network derived from D-Matrices, there are several encouraging results. First, the average rank of the correct fault in the Simple Model was around 2.5 and 1.5 for the Biased and Balanced networks depending on how many tests were run. Additionally, the ranks for the Biased and Balanced networks for the Missile Model were between 2 and 4.5. While the ranks for Biased and Balanced networks in the Missile Model were worse than that for the Simple Model, there are more than twice as many faults in the Missile Model than there are in the Simple Model. This shows that, while the correct fault may not be ranked the highest, it is still ranked relatively high in the overall list. This fits with what we would expect since in the full D-Matrix, not all of the tests need to be performed to diagnose a fault.

Another encouraging result is that the probability of the highest ranked fault and the true fault was relatively small, especially as the number of tests performed was increased. This shows us that while the correct fault may be ranked in the top three, the difference between those top three faults is relatively small. If a maintainer was using a Bayesian network instead of a FIM, he or she would probably repair all three if the difference between the top candidates was so little (as if they belonged to an ambiguity group). As an alternative, the maintainer could choose to run more tests to further isolate the fault in the system.

We also noted some interesting results from the results on accuracy. The first is that the probability of the top ranked fault for the Biased and Balanced networks was on average higher than that of the D-Matrix network. This was because the Biased and Balanced networks were missing links from the faults to the test nodes. Because of these missing links, the influence of the tests performed were stronger relative to faults connected to those tests.

From the degradation results, we see that the performance of the Bayesian network degrades fairly quickly when the first set of links are removed (20 percent of the links are removed) and then performance seems to level out (from about 20 to 60 percent). We also noticed that the accuracy of the networks remains steady from 60 to 80 percent and even increases in performance at the end. We also found that the graphs for Rank and KL-Divergence show a similar behavior. The difference in KL-Divergence between the

Table 7: Number of Test Sequence Lengths for the Missile Model with Equal Priors

Number of Tests	No Costs			With Costs		
	D-Matrix	Biased	Balanced	D-Matrix	Biased	Balanced
16	19	19	19	19	19	19
Avg Cost:	22.50	22.50	22.50	22.50	22.50	22.50
Accuracy:	1.00	0.50	0.45	1.00	0.50	0.45

Table 8: Number of Test Sequence Lengths for the Missile Model with Unequal Priors

Number of Tests	No Costs			With Costs		
	D-Matrix	Biased	Balanced	D-Matrix	Biased	Balanced
16	19	19	19	19	19	19
Avg Cost:	22.50	22.50	22.50	22.50	22.50	22.50
Accuracy:	1.00	0.42	0.38	1.00	0.42	0.38

full D-Matrix network and the D-Matrix with links removed degrades fairly steadily until about 60 percent of the links have been removed, at which point the difference levels out and even starts to reduce at the end.

In the final set of experiments that evaluate test cost and sequence length, we see that while we can still get decent performance from the Biased and Balanced networks, the networks derived from the trees are not as cost efficient. While the distribution of the sequence lengths is fairly even, the average cost for the D-Matrix network was almost always lower than the two other networks, especially when costs were incorporated into the test selection process. We also see in the Missile Model that using a hard threshold of when to stop performing tests is not a good stopping criterion. In the Missile Model, all of the test sequences used all of the tests. However, in certain cases one could stop performing tests sooner, but determining when those cases occur is not a trivial task. This may suggest a variable approach to stopping such as that described in [25].

7. SUMMARY

In this paper, we empirically analyzed the performance of bipartite Bayesian networks that were derived from FIMs. While the derived Bayesian networks do not always accurately diagnose the correct fault, they still can perform reasonably well when looking at some of the other results, such as the average probability of the correct fault. However, the Bayesian networks derived from the FIMs will not be as cost effective.

Using Bayesian networks for diagnosis offer several advantages. The first is that they allow for a natural extension for diagnostic model maturation [26]. In our previous work we focused on maturing D-Matrices and TFPGs. That work can easily be extended to maturing bipartite Bayesian networks since we can use historical maintenance and test session data to learn new relationships and conditional probabilities between faults and tests. This will then allow us to learn both if there is an erroneous relationship between a fault and alarm or to fill in the gap where a relationship was not previously known to exist.

For future work, we plan to examine and develop approaches to mature the Bayesian networks derived from the FIMs. By developing an efficient approach to mature the networks, we would then be better able to justify converting FIMs to Bayesian networks in the first place. This way we can make use of the engineering expertise that generated the FIMs in the first place as well as providing a seamless approach to maturing the processes based on historical data.

ACKNOWLEDGMENTS

The authors would like to thank Boeing Research & Technology for their support and guidance in work related to this project. Without their guidance and direction this paper would not have been possible. The authors would also like to thank the members of the Numerical Intelligent Systems Laboratory (NISL) at Montana State University for their reviews and comments as the work unfolded as as earlier drafts of the paper were prepared.

REFERENCES

[1] W. Simpson and J. Sheppard, *System test and diagnosis*. Norwell, MA: Kluwer Academic Publishers, 1994.

[2] T. M. Bearse, "Deriving a diagnostic inference model from a test strategy," in *Research Perspectives and Case Studies in System Test and Diagnosis*, ser. Frontiers in Electronic Testing, J. W. Sheppard and W. R. Simpson, Eds. Springer US, 1998, vol. 13, pp. 55–67.

[3] P. Utgoff, "Incremental induction of decision trees," *Machine Learning*, vol. 4, no. 2, pp. 161–186, 1989.

[4] F. Sahin, M. Yavuz, Z. Arnavut, and Ö. Uluyol, "Fault diagnosis for airplane engines using bayesian networks and distributed particle swarm optimization," *Parallel Computing*, vol. 33, no. 2, pp. 124–143, 2007.

[5] Z. Yongli, H. Limin, and L. Jinling, "Bayesian networks-based approach for power systems fault diagnosis," *Power Delivery, IEEE Transactions on*, vol. 21, no. 2, pp. 634–639, 2006.

[6] 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, 2001.

[7] S. Montani, L. Portinale, and A. Bobbio, "Dynamic bayesian networks for modeling advanced fault tree features in dependability analysis," in *Proceedings of the sixteenth European conference on safety and reliability*, 2005, pp. 1415–22.

[8] S. Montani, L. Portinale, A. Bobbio, and D. Codetta-

Raiteri, "Automatically translating dynamic fault trees into dynamic bayesian networks by means of a software tool," in *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*. IEEE, 2006, pp. 6–pp.

[9] G. Hulten, D. Chickering, and D. Heckerman, "Learning bayesian networks from dependency networks: a preliminary study," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[10] T. Jaakkola and M. I. Jordan, "Variational Probabilistic Inference and the QMR-DT Network," *JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH*, vol. 10, pp. 291–322, 1999.

[11] S. Wahl and J. W. Sheppard, "Extracting decision trees from diagnostic bayesian networks to guide test selection," in *Annual Conference of the Prognostics and Health Management Society*, 2010.

[12] J. Sheppard and W. Simpson, "Fault diagnosis under temporal constraints," in *AUTOTESTCON Proceedings, 19921. IEEE Systems Readiness Technology Conference*, Sep. 1992, pp. 151–157.

[13] P. Ryan and W. Kent Fuchs, "Dynamic fault dictionaries and two-stage fault isolation," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 6, no. 1, pp. 176–180, Mar. 1998.

[14] N. H. Narayanan and N. Viswanadham, "A methodology for knowledge acquisition and reasoning in failure analysis of systems," *IEEE Trans. Syst. Man Cybern.*, vol. 17, pp. 274–288, March 1987.

[15] S. Abdelwahed, G. Karsai, and G. Biswas, "A consistency-based robust diagnosis approach for temporal causal systems," in *The 16th International Workshop on Principles of Diagnosis*, 2005, pp. 73–79.

[16] K. Przytula and D. Thompson, "Construction of bayesian networks for diagnostics," in *Aerospace Conference Proceedings, 2000 IEEE*, vol. 5. IEEE, 2000, pp. 193–200.

[17] J. W. Sheppard and S. G. Butcher, "A formal analysis of fault diagnosis with d-matrices," *J. Electron. Test.*, vol. 23, pp. 309–322, aug. 2007.

[18] E. Alpaydin, *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2010.

[19] P. Utgoff, N. Berkman, and J. Clouse, "Decision tree induction based on efficient tree restructuring," *Machine Learning*, vol. 29, no. 1, pp. 5–44, 1997.

[20] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[21] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[22] G. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artificial intelligence*, vol. 42, no. 2, pp. 393–405, 1990.

[23] D. S. L. of the University of Pittsburgh. [Online]. Available: http://genie.sis.pitt.edu/

[24] C. Huang and A. Darwiche, "Inference in belief networks: A procedural guide," *International journal of approximate reasoning*, vol. 15, no. 3, pp. 225–263, 1996.

[25] J. Sheppard, W. Simpson, and J. Graham, "Method and apparatus for diagnostic testing including a neural network for determining testing sufficiency," Jul. 14 1992, US Patent 5,130,936.

[26] S. Strasser and J. Sheppard, "Diagnostic alarm sequence maturation in timed failure propagation graphs," in *IEEE AUTOTESTCON Conference Record*, 2011, pp. 158–165.

BIOGRAPHY

Shane Strasser received a BS in computer science and mathematics from the University of Sioux Falls in Sioux Falls, South Dakota. Afterwards he went on to obtain an MS in Computer Science at Montana State University, where he is currently working on his PhD in Computer Science. While at Montana State, he has received several awards, such as the 2012 outstanding PhD MSU Computer Science researcher and the 2011 AUTOTESTCON Best Student Paper Award. His research interests are primarily in artificial intelligence and machine learning with a focus on prognostics and health management systems. In his free time he enjoys singing, running, and riding motorcycles.

John Sheppard was the inaugural RightNow Technologies Distinguished Professor in Computer Science at Montana State University and currently holds an appointment as Professor in the Computer Science Department at MSU. He is also an Adjunct Professor in the Department of Computer Science at Johns Hopkins University. He holds a BS in computer science from Southern Methodist University and an MS and PhD in computer science, both from Johns Hopkins University. In 2007, he was elected as an IEEE Fellow "for contributions to system-level diagnosis and prognosis." Prior to joining Hopkins, he was a Fellow at ARINC Incorporated in Annapolis, MD where he worked for almost 20 years. Dr. Sheppard performs research in Bayesian classification, dynamic Bayesian networks, evolutionary methods, and reinforcement learning. In addition, Dr. Sheppard is active in IEEE standards activities. Currently, he serves as a member of the IEEE Computer Society Standards Activities Board and is the Computer Society liaison to IEEE Standards Coordinating Committee 20 on Test and Diagnosis for Electronic Systems. He is also the co-chair of the Diagnostic and Maintenance Control Subcommittee of SCC20 and has served as an official US delegate to the International Electrotechnical Commission's Technical Committee 93 on Design Automation.