

# Diagnostic Model Maturation

Shane Strasser & John Sheppard  
Montana State University

## 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. Some of the most common diagnostic algorithms used are rule based, fault dictionaries, and fault trees [2], [3], [4]. Model-based methods provide another common set of approaches to performing system level diagnosis, which compare observations from the system with those expected from a model [5], [6].

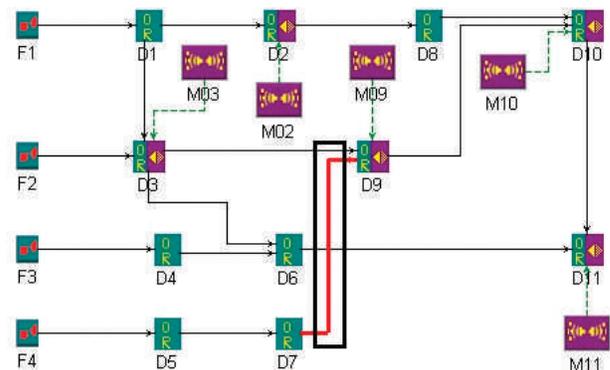
One model-based approach to system-level diagnosis that has been receiving attention is the timed failure propagation graph (TFPG), developed at Vanderbilt University [6], [7]. The TFPG specifies causal relationships between faults and discrepancies, which are irregular conditions that are the effects of the faults (whether monitored or unmonitored). TFPG diagnostic algorithms are then able to diagnose faults solely from knowing which alarms (monitored discrepancies) were or were not triggered. The diagnosis process, however, can be improved by incorporating the temporal information of when the alarms occurred [5], [8].

Figure 1 shows an example TFPG. Nodes labeled with F1, F2, F3, and F4 represent faults in the TFPG. The labels D1 through D11 denote nodes that represent discrepancies. Monitored discrepancies are represented by nodes labeled M2, M3, M9, M10, and M11. If fault F1 were to occur, then the signal would propagate to D1 and then split and propagate to D2 and D3, which would be detected by the alarms M2 and M3, respectively. The signal would then continue to propagate to the rest of the system and would trigger alarms M9, M10, and M11. In the diagnostic setting, the algorithm would only see alarms M2, M3, M9, M10, and M11 fire. Because all of those alarms indicate fault F1, F1 would be reported as the most likely diagnosis.

TFPGs are constructed by a domain expert of the system, and the modeling process is difficult because the expert must accurately represent causal relationships given an often incomplete understanding of actual system behavior, therefore

introducing error into the model. A poorly designed TFPG will result in inaccurate diagnosis from the reasoner, causing an increase in time, money, parts, and labor in the maintenance of the modeled system [9]. For example, suppose that in creating the TFPG in Figure 1, the engineer overlooked the causal relationship between the discrepancies D7 and D9. In this erroneous model, the reasoner will not be able to diagnose fault F4, as the model does not have any relationship between F4 and any alarm other than M11. If F4 does occur in the physical system, alarms M9, M10, and M11 will all occur, and the reasoner will likely diagnose F2 because it is the simplest explanation as to what fault would trigger M9. The result is an increase in maintenance time because the true fault has to be located by alternative means [10].

Because creating error-free diagnostic models is difficult, a process is needed to mature the diagnostic models over time. If the system user knows when the model is misdiagnosing a fault, he or she should be able to use that information to mature the model, resulting in a more accurate diagnosis. To determine whether the reasoner diagnosed the correct fault, one must compare the reasoner's diagnosis with the actual fault found by alternative means. By storing the past maintenance history, detailed engineering analysis can often be performed to determine the actual fault that occurred. The maintenance information can then be compared to the reasoner history and searched for any discrepancies between the two data sources. If there is a discrepancy be-



**Figure 1.** An erroneous TFPG. Nodes labeled with F1, F2, F3, and F4 represent faults in the TFPG. The labels D1 through D11 denote nodes that represent discrepancies. Monitored discrepancies are represented by nodes labeled M2, M3, M9, M10, and M11. The circled link from discrepancy D7 to discrepancy D9 should be included in the TFPG but was overlooked when the model was created, resulting in a poor diagnosis of faults.

Authors' current address: S. Strasser, J. Sheppard, Department of Computer Science, 347 EPS Building, Montana State University, Bozeman, Montana 59717. E-mail address: shane.strasser@cs.montana.edu; john.sheppard@cs.montana.edu. Manuscript SYSAES-21200037 was received February 14, 2012, and ready for publication October 4, 2012. Review was handled by R. Wang. 0885/8985/13/\$26.00 © 2013 IEEE

tween the two histories, then the user knows that the reasoner misdiagnosed a fault that can then be used to modify the model such that the reasoner is more likely to diagnosis the correct fault in the future [9].

Unfortunately, diagnostic model maturation is also a difficult problem [11]. To perform maturation, the reasoner diagnosis history and maintenance history are needed to be able to locate where the reasoner is misdiagnosing a fault. However, these data sources are often stored in heterogeneous systems, making retrieval and analysis of the data difficult [11]. In previous work, Wilmering and Sheppard suggested an approach using domain ontologies as a means to focus and filter data analysis in knowledge discovery [12]. The specific focus of that work showed how ontologies could be used to guide the process by which diagnostic models could be matured over time. That paper proposed using a method such as the Apriori algorithm [13] to discover new relationships within historical maintenance data that could be used to determine diagnostic relationships, improve probability estimates, or provide better specification of test processes [12].

In our previous work, we extended the work of Wilmering and Sheppard to map diagnostic models and historical diagnostic session data to two ontologies derived from IEEE standards [14]. In that work we mapped the reasoner information and D-matrix model from the Fault-Adaptive Control Technology (FACT) TFPG reasoner to the IEEE Std 1232 Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE) and the maintenance history to IEEE Std 1632.2 Software Interface for Maintenance Information Collection and Analysis (SIMICA): Maintenance Action Information (MAI) [15], [16], [17]. The IEEE models, which were standardized using the EXPRESS language, were redefined using the Web Ontology Language (OWL) [18], [19]. This allowed us to use graph-theoretic representations of the models and sessions to determine statistical discrepancies between what was expected by the models and what had been encountered in practice. From this, we were able to recommend changes to TFPGs which matured the relationships between faults and alarms.

That work was further extended by looking at how to mature the relationships between the different alarms in the TFPGs [20]. Given a set of maintenance events, we modeled all of the alarm sequences using a probabilistic transition matrix similar to a Markov chain. This probabilistic transition matrix was then compared to the TFPG to find discrepancies between the two. These recommendations were then reported back to an engineer or analyst for further review. Not only could these recommendations be used in the maturation of relationships between alarms, but also in the maturation of relationships between alarms and faults.

Our previous work focused on the ontological aspect of the maturation process and the maturation algorithms for TFPGs [14], [20]. In this paper, we discuss ModelMat, the tool we developed to perform diagnostic model maturation. In addition, we also discuss how the maturation algorithms can be extended to other diagnostic models, such as Bayes-

ian networks. The paper is organized as follows. Section 2 discusses various diagnostic models in greater detail while Section 3 discusses related work. The maturation algorithms are discussed in Section 4. Section 5 gives a description of ModelMat, an example of how the software is used, and a discussion of how the maturation algorithms can be extended to other diagnostic models. Finally, Section 6 concludes the paper and discusses future work.

## 2 FAULT DIAGNOSTIC MODELS

There are several different ways to perform fault diagnosis [1]. Some of the most common diagnostic algorithms are rule-based algorithms, such as those in [4]. Fault dictionaries have successfully been applied as another diagnostic method, especially with digital systems [3]. Fault trees have also been merged with rule-oriented reasoning, which allowed failure location and identification of the failure cause [2]. Fault Isolation Manuals (FIMs), which are a set of procedures for diagnosing faults in complex systems based on decision graphs, are another common way to perform fault diagnosis.

Model-based methods provide another common set of approaches to performing system level diagnosis, which compare observations from the system with those expected from a model such as the TFPG [5], [6], [7]. Since TFPGs have been introduced, there have been two primary algorithms that have been developed that use TFPGs: one based on parsimony and another one based on consistency [5], [6].

### 2.1 TIMED FAILURE PROPAGATION GRAPHS

A TFPG is a directed graph where each vertex represents a failure or a discrepancy [6], [7]. Failure nodes represent faults in the target system, and discrepancies are causal nodes that are affected by failures. Discrepancy nodes can be monitored or unmonitored. Monitored discrepancies are often referred to as alarms. The edges between the nodes represent the effect of failure propagation over time in the underlying system that is being modeled. Formally, this is represented as  $TFPG = (F, D, E, ET, DS)$ , where

- ▶  $F$  is a set of failure nodes
- ▶  $D$  is a set of discrepancy nodes
- ▶  $E = V \times V$  is a set of directed edges, where  $V = F \cup D$
- ▶  $ET: E \rightarrow (Int, Int)$  is a mapping that associates each edge in  $E$  with a finite time interval
- ▶  $DS: D \rightarrow \{M, N\}$  is a mapping that defines whether a discrepancy is monitored ( $M$ ) or not monitored ( $N$ ).

The set of discrepancies that are monitored are defined by the map  $DS$ . The map  $ET$  associates with each edge  $e$  in  $E$ , a minimum and maximum time for the failure to propagate along the edge. The goal of a diagnostic algorithm is to find a hypothetical state that tries to explain the physical system based on observed alarms [5]. Note that TFPGs are used in

an online setting, and therefore the alarms are asymmetric. This means that we only know the state of an alarm when it fires and that an alarm not firing does not mean the alarm is automatically in the off state.

Since TFPGs have been introduced, several online diagnostic algorithms have been developed to use these models by determining the most likely fault occurrence given a set of alarms that have been triggered [5], [8]. TFPGs have also been extended by Abdelwahed in 2004 to include model dependency constraints on the propagation links [21]. These extended models, referred to as a hybrid failure propagation graphs (HFPG), allow the model to operate in various operational modes. Thus, alarms can be either enabled or disabled, depending on the associated mode.

## 2.2 BAYESIAN NETWORKS

Another model based approach is the Bayesian network. This type of model allows the incorporation of uncertainty into the diagnostic procedure and is useful in scenarios where causality appears to be a factor, but the understanding of the system is incomplete [22].

A Bayesian network is a graphical model that represents a probability distribution in a compact way [23]. In addition, it enables a person to view relationships between random variables explicitly in the probability distribution. Given a set of random variables  $\mathbf{X} = \{X_1, \dots, X_n\}$ , we want to represent a joint probability  $\mathbf{P}(\mathbf{X}) = \mathbf{P}(X_1, \dots, X_n)$  as efficiently as possible. Using the product rule, we can factor  $\mathbf{P}(\mathbf{X})$  as

$$\mathbf{P}(X_1, \dots, X_n) = \mathbf{P}(X_1) \prod_{i=2}^n \mathbf{P}(X_i | X_1, \dots, X_{i-1}). \quad (1)$$

The problem with representing a probability distribution in this way is that each factor of the joint distribution is represented by a probability table that has a size 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

$$\mathbf{P}(X_i | X_j, X_k) = \mathbf{P}(X_i | X_k). \quad (2)$$

This means that the event represented by the random variable  $X_j$  has no affect on the knowledge of the event  $X_i$  if we know something about the event  $X_k$ .

Bayesian networks are directed acyclic graphs that model the conditional independences in a probability distribution [23]. In the graph, the set of vertices correspond to the random variables of the distribution while the set of directed edges represents conditional dependence relationships where destination nodes are conditionally dependent on the source node. We can denote the set of nodes Parents( $X_i$ ) where each node  $X_j \in \text{Parents}(X_i)$  has an edge to  $X_i$ . Using

the conditional independences represented in the Bayesian network, we can rewrite  $\mathbf{P}(\mathbf{X})$  as the following:

$$\mathbf{P}(X_1, \dots, X_n) = \prod_{X_i \in \mathbf{X}} \mathbf{P}(X_i | \text{Parents}(X_i)). \quad (3)$$

The structure of the Bayesian network is dependent on the conditional dependencies being modeled.

A 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 [23], [24]. These algorithms will return the exact probability of the event that is being queried given the evidence that has been set. However, exact inference in Bayesian networks is NP-complete [22]. For large networks, one can use approximate inference algorithms, such as stochastic sampling. These algorithms generate several example instantiations of variables in the network based on the evidence and use these samples to estimate probabilities over the variables of interest. While not exact, these estimates are often very accurate [23], [24].

Bayesian networks have been used extensively for performing fault diagnosis [25]. Traditionally, these networks are based on the QMR-DT scheme, which consists of two types of nodes: fault nodes and test nodes [26]. Each test node has a set of fault nodes as it parents such that if there is a relationship between the test and a fault, then the test indicates the fault. In addition, each test node has an associated probability table. Figure 2 shows an example Bayesian network that might be used for fault diagnosis [27].

In the example, there are four different faults, F1, F2, F3, and F4. In addition, there are four different tests M1, M2, M3, and M4. To diagnose a fault, a person would run a subset of the tests and record the outcome of each test. Then, these outcomes would be applied as evidence to the Bayesian network. Finally, the user could query the nodes F1, F2, F3, and F4 to see what fault had the highest probability of occurring, which would be reported as the diagnosis. Note that there is no requirement on how many tests must be run. However, running more tests will often lead to a higher confidence in the final diagnoses.

## 2.3 D-MATRICES

One-way TFPG or Bayesian models can be represented is with D-matrices. A D-matrix, at least partially, relates the faults and the discrepancies that monitor or observe those faults. We can formally define it as the following: Let  $\mathbf{F}$  represent a set of faults. Let  $\mathbf{D}$  represent the set of discrepancies. Assume each  $F_i \in \mathbf{F}$  is a Boolean variable such that  $eval(F_i) \in \{0, 1\}$ , and each  $D_j \in \mathbf{D}$  is also a Boolean variable such that  $eval(D_j) \in \{0, 1\}$ . Then, a diagnostic signature is defined to be the vector

$$\mathbf{F}_i = [eval(D_1), \dots, eval(D_{|\mathbf{D}|})], \quad (4)$$

where

$$eval(D_j) = \begin{cases} 1 & \text{if } D_j \text{ detects } F_i \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

A D-matrix is then defined to be the set of diagnostic signatures  $F_i$  for all  $F_i \in F$  [28]. Rows represent faults and columns represent discrepancies. The  $i$ th column corresponds to discrepancy  $D_i$  in the diagnostic model. When representing a TFGP with a D-matrix, the only discrepancies that are typically included in the D-matrix are the monitored ones.

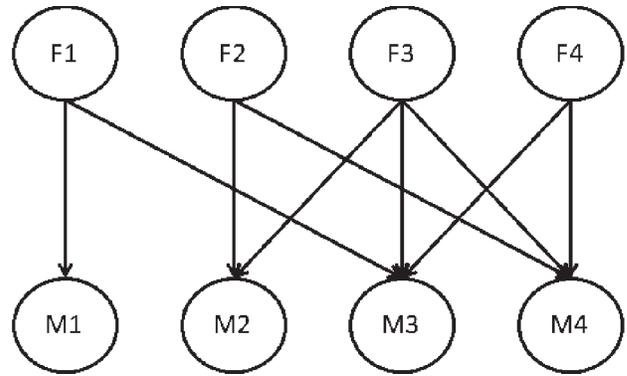
D-matrices do not fully represent TFGPs because they do not capture the temporal relationships. However, one can find the logical relationship between the alarms by computing the logical closure of the matrix [27]. This is done by determining which discrepancies detect fault set that is a subset of another discrepancy's detected fault set. Let  $D_i$  be a discrepancy that monitors faults  $F_i$  and let  $D_j$  be a discrepancy that monitors faults  $F_j$ . We can represent this as  $F_i \rightarrow D_i$  and  $F_j \rightarrow D_j$ . If  $F_i \subseteq F_j$ , then  $F_j \rightarrow D_i$ . If  $F_j$  is true,  $F_i$  must also be true. This means  $D_i$  must be true, and therefore  $D_i \rightarrow D_j$  [1], [27]. The result is a D-matrix representing the causal relationships between discrepancies, which are needed if a person is using a D-matrix to represent a TFGP and wants to perform sequence maturation.

After finding the logical closure, one can find the transitive reduction of the resulting matrix [29]. In taking the transitive reduction, all of the transitive links between discrepancies can be removed. Specifically, if there exists a link from discrepancy  $D_x$  to  $D_y$  and  $D_z$  and also from  $D_y$  to  $D_z$ , then we can remove the link from  $D_x$  to  $D_z$  because we know that  $D_z$  observes  $D_y$  and  $D_y$  observes  $D_x$ , then  $D_z$  observes  $D_x$ . This reduced D-matrix is then able to show the first-order dependencies between the discrepancies, which can be used in the Seq-Mat algorithm.

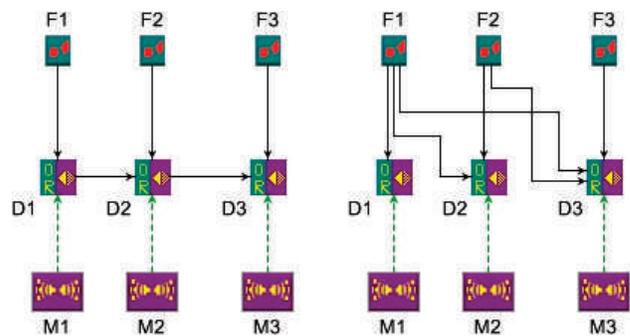
In our previous work, we focused on maturing the D-matrices of the TFGPs. One problem with using the D-matrix to represent the TFGP, however, is that the resulting unclosed D-matrix will not always represent the true causal relationships between alarms. For example, in Figure 3, after the logical closure and transitive reduction of the two models is found, the D-matrices will be the same.

Additionally, because the closure step depends on which faults the alarms monitor, the logically closed D-matrix will not be able to determine the correct temporal relationships in a sequence of alarms. Therefore, it is sometimes necessary to provide the maturation algorithms with the graph representation of the TFGP and then convert the graph to a matrix representation. This matrix will then correctly represent the relationships between all of the discrepancies and therefore allow for accurate sequence maturation.

Additionally, diagnostic Bayesian networks, such as the one in Figure 2, can be represented as a D-matrix. Table 1 shows the D-matrix for the Bayesian network in Figure 2. However, the D-matrix only captures structural relationships between diagnoses and tests and does not capture any of the probabilistic information [27], [30].



**Figure 2.** An example of a Bayesian network being used for fault diagnostics. The Fs represent a diagnosis while the Ms represent tests.



**Figure 3.** Two different TFGPs. If the logical closure and reduction of these two TFGPs is taken, the unclosed D-matrix of the two models will be the same.

### 3 RELATED WORK

The idea behind diagnostic maturation has been discussed in several papers, but to our knowledge, only one paper has published a formal algorithm for large amounts of corrective actions in which nondetects or false alarms could be occurring [14]. In [9], Wilmering points out there are unexpected and unplanned system interactions that can degrade the performance of the diagnostic design. To increase performance of the diagnostic model, historical maintenance actions will be used to help mature the model. Wilmering however, points out that the process requires ready access to the model, maintenance events, and any other information that could aid in the maturation process [9]. To use all of these resources, he proposes using an ontology to gather all the required data together in a meaningful way.

In [1], the authors discuss using explanation-based learning for the diagnostic model. If misdiagnosis occurs, then additional testing is done until a correct diagnosis has been made. This information can then be used to modify the structure of the model so that the correct diagnosis is consistent with testing. The authors also give a proof that, given a single misdiagnosis, the model can be modified so that the

Table 1.

A D-Matrix Representing the Diagnostic Bayesian Network in 2.				
	M1	M2	M3	M4
F1	1	0	1	0
F2	0	1	0	1
F3	0	1	1	1
F4	0	0	1	1

misdiagnosis never occurs again. This, however, did not include how to deal with nondetects or false alarms.

In our previous work, we presented an ontology-guided approach for D-matrix model maturation [14]. In that work we were limited to maturing the relationships between alarms and faults. Once the discrepancies between the maintenance and reasoner history were found, the algorithm would find the probability of an alarm  $M_i$  firing given a single ground truth (maintenance event) fault  $M_j$  was diagnosed as the true fault. These probabilities were then compared against the D-matrix. If an alarm occurred with a high probability and the D-matrix showed that the alarm was not observing the fault or if the alarm occurred with a low probability and the D-matrix showed that the alarm is observing the fault, then the relationship was flagged as possibly erroneous, and a recommendation made to the engineer.

The problem with that approach was that it did not give any information on how the erroneous relationship between the fault and alarm should be fixed, such as which link should be added or removed in the TFPG. We extended that previous work by using the sequences of alarms to inform which links should be removed or added. In addition, we were able to mature the relationships between discrepancies in the TFPG. Given a set of maintenance events, we modeled all of the alarm sequences using a post-occurrence probability matrix. This matrix was then compared to the TFPG to find discrepancies between the two. The discrepancies were then noted and reported to an engineer or analyst for further review. Details of these algorithms are presented in the next section.

## 4 MATURATION ALGORITHMS

In our previous work, we presented two maturation algorithms that performed D-matrix and alarm sequence maturation called D-Mat and Seq-Mat, respectively. If a certain alarm is monitoring a fault, but maintenance events are finding that the alarm never occurs when that fault occurs, then that alarm probably should not be modeled as monitoring that particular fault. Additionally, if another alarm is not monitoring a fault, but the alarm always occurs when the fault occurs, then that alarm should probably be modeled to monitor that fault. This

whole process falls under D-matrix model maturation. For alarm sequence maturation, suppose a TFPG has a sequence of alarms that diagnoses a particular fault; however, maintenance events are finding that when the fault occurs, the alarms are not occurring in the expected order. The sequence needs to be adjusted so that the alarms correctly identify the fault.

For both of these maturation scenarios, the algorithms need to take into account alarms that do not fire when they should (nondetects) and alarms that fire when they should not (false alarms). They need to be analyzed to gain an accurate picture of the alarms that should be occurring based on the maintenance events. Such analysis can also assist incorporating uncertainty measures into the diagnostic process. The remainder of the section describes the two maturation algorithms in detail.

### 4.1 D-MAT ALGORITHM

In the maturation of fault and alarm dependencies, or D-matrix maturation, we have a collection of alarm sequences from whenever maintenance locates a certain fault. The maturation process searches for any differences between the alarm sequences and signatures in the D-matrix. However, care needs to be taken when considering false alarms and nondetects because adjusting the dependencies based on those points of disagreement will likely have a negative effect on the performance of the model.

The maturation algorithm D-Mat (Algorithm 1) works as follows. First, we retrieve the alarm sequences recorded in logs of a repaired fault (whether the diagnosis was correct or not). An alarm sequence is represented as a bit string, where each position in the bit string corresponds to a different observable alarm in the TFPG. For example, a one at index  $i$  means that the  $i$ th alarm fired in the sequence, and a zero means that the  $i$ th alarm did not fire for that sequence. We then sum and normalize each bit over the set of log files, yielding a probability of firing, given the fault was diagnosed as the true fault. Finally, we compare the resulting vector of probabilities to the corresponding fault signature in the D-matrix. When there is a wide disparity between the bit positions in the D-matrix and the probabilities in the probability vector, we flag that as a relationship to be examined.

In D-Mat,  $F$  is the repaired fault the user queried,  $L$  is the set of logs corresponding to the reasoner history that caused the system to be repaired, and  $D$  is the D-matrix of the diagnostic model that was being used on the system. In the algorithm, lines 3 and 4 calculate the probability of an alarm occurring, while lines 5–8 compare the probabilities to the D-matrix and flag any discrepancies. Also, the user must set the threshold values  $\epsilon_1$  and  $\epsilon_2$ , which are compared against the probability vector to determine when the algorithm will flag discrepancies.

### 4.2 SEQ-MAT ALGORITHM

In performing alarm sequence maturation, we focus on maturing the causal relationships of discrepancies, or alarms,

with one another. In addition, the algorithm Seq-Mat (Algorithm 2) can aid a person in performing D-matrix maturation.

#### Algorithm 1 D-Mat

```

1: // F is a single fault
2: // L is a set of logs
3: // D is a D-matrix
4: SetOfAlarms A ← GetUniqueAlarms (L )
5: for all Alarms Ai ∈ A do
6:   x ← Number of logs L ∈ L containing Ai
7:   n ← x / |L|
8:   if D [F, Ai] = 1 and n < ε1 then
9:     E ← Flag F and Ai
10:  else if D [F, Ai] = 0 and n > ε2 then
11:    E ← Flag F and Ai
12:  end if
13: end for
14: return AllFlags E

```

Whenever a particular fault is found during post-maintenance analysis to be the true cause of a maintenance event, we analyze all of the reasoner logs that caused the unit to be pulled for maintenance. We then examine all of the reasoner logs for the corresponding maintenance event and extract the alarm sequences from the reasoner logs. Using these alarm sequences, we calculate a post-occurrence probability matrix  $M$  that gives the probability of an alarm occurring sometime after another alarm. For alarms  $A_i$  and  $A_j$ ,  $M[i, j]$  represents the probability that alarm  $A_j$  occurred sometime after alarm  $A_i$  with respect to the total number of alarm sequences that have occurred, given that fault  $F$  was diagnosed as the ground truth in the maintenance event. Note that alarm  $A_j$  does not have to occur directly after alarm  $A_i$ . This matrix represents the possible temporal occurrences of the alarms that are being observed.

Next, we determine the expected post-occurrence probability matrix. As noted in our previous work, this is a combinatorial problem and therefore has an exponential run time [14]. Therefore, we use the TFGP to find the current causal relationships between the various alarms. If there is a causal relationship between two alarms  $A_i$  and  $A_j$ , we would expect a high probability in  $M[i, j]$  from the post-occurrence probability matrix. Similarly, if there is no causal relationship between two alarms, the corresponding probability in the post-occurrence probability matrix should be low. If there is a discrepancy occurring between the TFGP and the post-occurrence probability matrix, we can then flag those two alarms to be observed in greater detail.

#### Algorithm 2 Seq-Mat

```

1: // M is the post-occurrence matrix
2: // T is a TFGP
3: SetOfAlarms A ← GetAlarms (T)
4: for all Alarms Ai ∈ A do
5:   for all Alarms Aj ≠ Ai ∈ A do
6:     if T [Ai, Aj] = 1 and M [i, j] < ε1 then
7:       E ← Flag Ai and Aj

```

```

8:     else if T [Ai, Aj] = 0 and M [i, j] > ε2 then
9:       if no common ancestor between Ai and Aj then
10:        E ← Flag Ai and Aj
11:      end if
12:    end if
13:  end for
14: end for
15: return AllFlags E

```

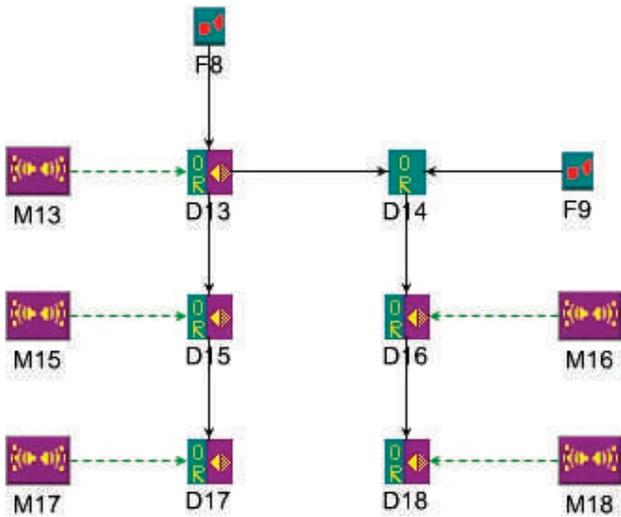
In Algorithm 2,  $M$  is the post-occurrence probability matrix,  $T$  is the TFGP, and the  $\epsilon$ s are the threshold values that are compared against the probabilities in the post-occurrence probability matrix. When the post-occurrence probability matrix indicates that there should not be a relationship between two alarms, but there is a causal relationship between the two alarms, we call it a Type I error. This is the *if* statement in line 4 of the pseudocode. We call a Type II error when the post-occurrence probability matrix says there is a relationship, but there is no causal relationship in the model, which is the *if* statement on line 6.

After all of the discrepancies are flagged, they are analyzed further to see if any of the discrepancies were falsely flagged. For example, suppose we have the TFGP in Figure 4, and this model is error free. After we calculate the post-occurrence probability matrix, suppose we find that the probability of alarm M18 occurring after alarm M15 is very high. Similarly, the probability that alarm M18 precedes alarm M17 is very high. Our algorithm will flag the relationships between M15 and M18 and between M17 and M18 because there is no causal relationship between those alarms. In this scenario, however, that is acceptable because of how the signal is split after discrepancy D13 and propagates through the two links. To account for this scenario, we added a step that checks to see if the alarms have a related “ancestor,” i.e., alarms that share a common causal discrepancy. This step is line 7 in the pseudocode of the Seq-Mat algorithm. If the alarms do have a common alarm or fault for which a causal relationship exists, then the alarms are not flagged as having an erroneous relationship.

The process to find common ancestors is straightforward. Using an adjacency matrix to represent the TFGP (using only faults and monitored faults), we find whether there is a path from alarm (or fault)  $i$  to alarm  $j$  by using Warshall’s algorithm [31]. We then take each of the Type II flagged relationships and check to see if there exists another alarm or fault for which there is a path to the two alarms that has also been flagged as having a false causal relationship. If there is a common ancestor, we then remove that flagged relationship. All of the remaining Type II flags, along with all of the Type I flags, are then recommended to an engineer as possible changes to be made to the TFGP.

## 5 MODEL MAT

We developed a tool set called ModelMat that implements both the D-Mat and Seq-Mat algorithms. ModelMat was written in C# and provides the user with an easy-to-use in-



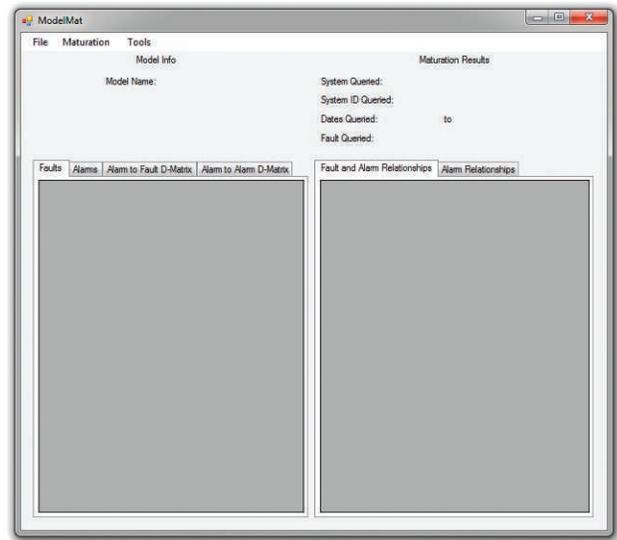
**Figure 4.** A correct TFPG. Note that alarm-sequence maturation algorithm may flag certain relationships as errors, unless we add a step to check it for relationships that were falsely flagged.

terface (Figure 5) for viewing the algorithm output and setting algorithm parameters.

In the left panel of Figure 5, ModelMat displays information about the diagnostic model that was loaded, including the list of faults, alarms, the D-matrix, and the matrices resulting from logical closure and transitive reduction. In the right panel, results from running the maturation algorithms are displayed. To analyze output from the D-Mat algorithm, the user clicks on the tab titled “Fault and Alarm Relationships.” Analyzing output from Seq-Mat is performed by pressing the tab titled “Alarm Relationships.” Once the user clicks the tab, the results of the maturation algorithm are instantly displayed. When there is no conflict found, the rows are colored blue, while potential conflicts that were discovered during the maturation process are highlighted in yellow.

ModelMat works as follows. First, the user selects a diagnostic model to open. Currently, the user can load D-matrices or TFPGs. If a TFPG is loaded, the software will first store the TFPG as a graph. Next, it will convert the TFPG to a D-matrix using transitive closure and then perform logical closure and transitive reduction. This allows the user to run the alarm sequence maturation algorithms on the original TFPG or D-matrix, which can then be used to determine how much information is lost when the TFPG is converted to a D-matrix.

Next, ModelMat connects to a MySQL database that contains entries listing what the reasoner diagnosed as the fault, what maintenance determined was the true fault, and the ordered list of alarms that fired. In addition, system information is also stored, such as the date of the maintenance event, the system ID, and the system type. ModelMat then allows the user to query the database based on ground truth, system type, system ID, and a range of dates. The query window is shown in Figure 6. Alarm sequences corresponding to the user’s query are pulled into the tool, which are then



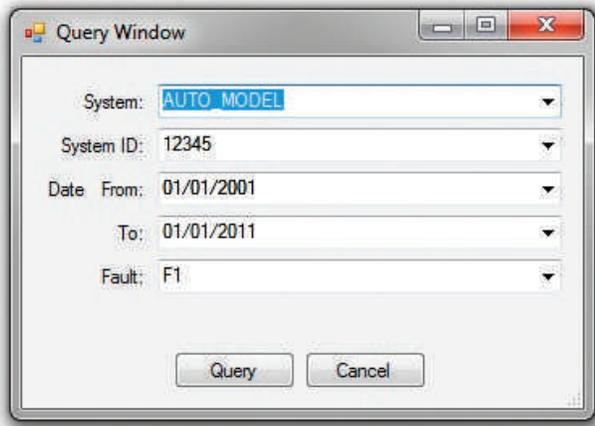
**Figure 5.** A screenshot of the software. The left side of the panel shows information of the diagnostic model, while the right panel shows results from the maturation algorithms.

used by the tool when running the selected analysis. If the user is running the alarm sequence maturation, the user can choose if they wish to run Warshall’s algorithm, which removes the falsely flagged relationships. The software also allows the user to set the various threshold values used by the algorithms.

### 5.1 DEMONSTRATION

To demonstrate how ModelMat is used, we created a simple scenario with user-generated data that also shows how both maturation algorithms can correctly identify erroneous links in the model. In the scenarios, we developed the simple TFPG in Figure 7 using the Generic Modeling Environment (GME) from Vanderbilt University. In the example, a particular alarm is not modeled as monitoring a fault that it should be. Specifically, the link circled from D3 to D4 in the figure is missing in the model, which will result in the reasoner not diagnosing fault F1 with high confidence.

Next, we populated the database with log files generated by the FACT reasoner, which was also developed by Vanderbilt University [15]. We used the correct model, i.e., the circled link was included, to build alarm sequences that would normally diagnose the fault F1. We also added false alarms and nondetects with a probability of 0.02 into the alarm sequences. Those alarm sequences were then used as input to the reasoner for both the correct and erroneous TFPGs. The diagnosis made using the correct model was then recorded as the ground truth, i.e., the true fault found during the maintenance event. The diagnosis from the erroneous model was then recorded as the diagnosis from the reasoner. The alarm sequences, along with the model name, date, ground truth, and reasoner diagnosis were then inserted into the database.



**Figure 6.** A screenshot of query window. The software allows the user to query the database based on aircraft, aircraft ID, range of dates, and ground truth (fault).

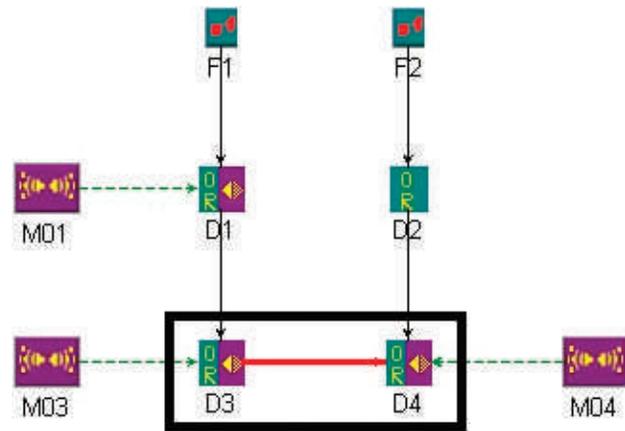
Using ModelMat, we first load the TFPG and query the database on the fault actually repaired. The results from D-Mat are displayed in Figure 8. Each row in the results pane corresponds to an alarm in the TFPG. The next column displays the probability that the alarm fired when the repaired fault occurred. The final column is the modeled fault signature for that fault. In this example, ModelMat had alarms M01 and M03 highlighted in blue because there is no error between those alarms and F1. On the other hand, alarm M04, which has been circled, observes F1 with a high probability, while the fault signature says the alarm should not fire when the fault occurs. Therefore, ModelMat highlighted the row in yellow.

Without knowing anything about the alarm sequences, we could hypothesize several different ways to add the causal relationship between M04 and F1, such as adding a link from F1 to D4 or D1 to D4. However, we can use Seq-Mat to help us correct the relationship between M04 and F1.

Figure 9 illustrates the results of alarm sequence analysis with our sample model. Each cell in the table presents the probability that the alarm in the column fired after the alarm in the row. If there is a discrepancy between the TFPG and the probabilities, the cell is highlighted in yellow. Cells that do not have errors are colored in blue. In the example in Figure 7, the erroneous model would not expect to find any relationship between either M01 or M03 and alarm M04. However, Seq-Mat discovered that M04 occurs after both M01 and M03 a high number of times; therefore, there is a conflict in the relationships. To show these conflicts (which have been circled in the figure), ModelMat has highlighted the error cells in yellow. To resolve this conflict, a link should be added from D3 to D4, which would also resolve the conflict between F1 and M04 described above.

## 5.2 MATURATION OF DIAGNOSTIC BAYESIAN NETWORKS

As discussed earlier, the algorithms were originally developed for maturation of TFPGs. However, because many of the al-



**Figure 7.** A TFPG with a missing link. The circled link should be in the TFPG but was not included when the model was created. Therefore, the reasoner will not diagnose the fault F1 with high confidence levels.

Maturation Results

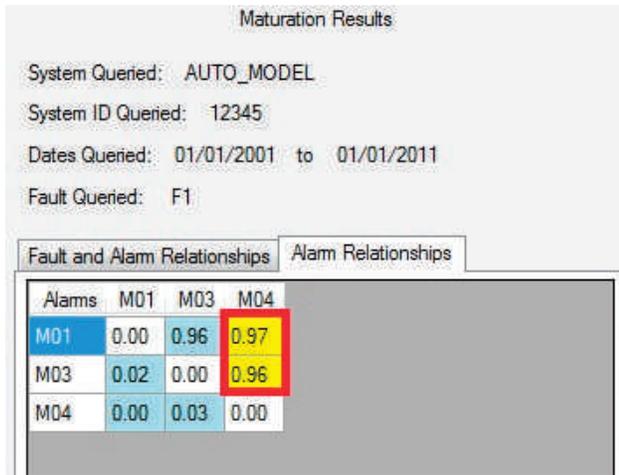
System Queried: AUTO\_MODEL  
 System ID Queried: 12345  
 Dates Queried: 01/01/2001 to 01/01/2011  
 Fault Queried: F1

Fault and Alarm Relationships		Alarm Relationships	
Alarms	Probability	Fault Sig	
M01	0.97	1	
M03	0.98	1	
M04	0.97	0	

**Figure 8.** The results from the D-matrix maturation algorithm. There is no conflict between fault F1 and the alarms M01 and M03; therefore, ModelMat highlights those rows in blue. However, the row that has been circled denotes a conflict in the relationship between fault F1 and alarm M04; therefore, ModelMat highlighted that row in yellow.

gorithms were designed to work on the D-matrices, they can easily be used when maturing other diagnostic models. For example, because a diagnostic Bayesian network can be represented as a D-matrix, the same algorithms can be used directly to mature network structure as well. To do so, we first need to make the assumptions that each test node has two outcomes, Yes and No and that a Yes outcome on the test node indicates its parents and a No outcome exonerates the parent nodes.

First, the user would need a D-matrix representation of the Bayesian network, log files that show that tests were run, the outcome of the each test, and finally, the ground truth. From the logs, the user would extract each test with an outcome of Yes and insert the tests into the database. Finally, the user



**Figure 9.** The results from the alarm sequence maturation algorithm. The cells that are circled in the figure represent an error between the alarms M01 and M03 with that of alarm M04. The software has therefore highlighted those cells in yellow to alert the user about the error in the model.

would be able use the D-Mat algorithm to perform maturing on the D-matrix representation of the Bayesian network. The results from D-Mat would give recommendations on relations that should be added or removed in the Bayesian network. The D-Mat algorithm, however, would give no information on how to mature the parameters for the conditional probability tables; therefore, an algorithm such as expectation-maximization would be needed to mature the tables.

For example, suppose that in Bayesian network in Figure 2, M3 should not observe F1. In the logs, we find that when F1 is the ground truth, the outcome of test M3 is No a high percentage of the time and, conversely, Yes a low percent of the time. In running D-Mat, we would then find out that because M3 is Yes a low percent of the time, there is a discrepancy between the D-Mat and the logs, and therefore the link between F1 and M3 would be flagged as a possible erroneous relation.

## 6 DISCUSSION

In this paper, we presented a tool set, ModelMat, which performs diagnostic model maturation. In addition, we described the algorithms that ModelMat uses in the maturation process, D-Mat and Seq-Mat. Both algorithms are able to identify erroneous relationships in D-matrix-based models, where the errors can be between faults and alarms and among a sequence of alarms. While these algorithms were originally designed for maturing TFPGs, they can be extended easily to any other diagnostic model that can be represented as a D-matrix, such as Bayesian networks.

For future work we plan to explore maturing the time intervals used on the propagation edges in the TFPG. Because FACT uses the time intervals to assist in diagnosing a fault, an incorrect time value could change how the result-

ing diagnosis. Again, using alarm sequences from historical maintenance records, one could flag suspicious values and adjust them. We will also explore the case when alarms are symmetric, i.e., when an alarm not firing is assumed to be in the off state. We are also exploring incorporating parameter estimation algorithms, which would enable adjusting the conditional probability tables in a Bayesian network. A user would then be able to mature many aspects of a Bayesian model.

Another area of future work that we want to explore is using the idea of significance testing in the maturation algorithms. Currently, both algorithms require the user to set a threshold value that is used to determine whether or not to flag a relationship as a possible error. For example, in D-Mat, the user is required to set the  $\epsilon$ s, which are compared against the probability vector representing the probability of an alarm firing given a fault occurred. However, a more flexible way to flag possible errors would be to use some form of significance testing to determine if the probability is significantly different than what is expected. More work, however, is needed to determine how to perform significance testing between a probability vector and D-matrix fault signature.

Finally, we are in the process of combining ModelMat with a companion tool for knowledge discovery in maintenance event sequences presented in [32]. In that work, a tool was developed that enables users to view transactional maintenance data as ontology-based event graphs. The tool allows users to filter sequences of maintenance events through time, making it easier to perform knowledge discovery. That tool is also being extended to display onboard data and its relationship to individual maintenance events. In follow-on work, knowledge discovery algorithms will be incorporated to find interesting relationships, such as the discrepancies between the reasoner diagnosis and the ground truth found in maintenance events needed for ModelMat. Currently, ModelMat assumes that the conflicts between the reasoner diagnosis and the maintenance events have already been located. However, finding those conflicts is itself difficult. Once integrated, the suite of tools could be connected to available maintenance databases giving engineers a seamless diagnostic maturation process. ◀

## ACKNOWLEDGMENTS

The authors would like to thank Boeing Research and Technology for their financial and technical support in this project. Additionally, we would like to thank the Institute of Software Integrated Systems (ISIS) at Vanderbilt University for providing us with their FACT TFPG reasoner and GME.

## REFERENCES

- [1] Simpson, W., and Sheppard, J. *System Test and Diagnosis*. Boston, MA: Kluwer Academic, 1994.
- [2] Narayanan, N. H., and Viswanadham, N. A methodology for knowledge acquisition and reasoning in failure analysis of sys-

- tems. *IEEE Transactions on Systems Man and Cybernetics*, Vol. 17, 2 (Mar. 1987), 274–288.
- [3] Ryan, P., and Kent Fuchs, W. Dynamic fault dictionaries and two-stage fault isolation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 6, 1 (Mar. 1998), 176–180.
- [4] Sheppard, J., and Simpson, W. Fault diagnosis under temporal constraints. In *Proceedings of the 1992 IEEE Systems Readiness Technology Conference AUTOTESTCON*, Dayton, OH, September 21–24, 1992, 151–157.
- [5] Abdelwahed, S., Karsai, G., and Biswas, G. A consistency-based robust diagnosis approach for temporal causal systems. In *Proceedings of the 16th International Workshop on Principles of Diagnosis*, Monterey, CA, 2005, 73–79.
- [6] Misra, A., Sztipanovitz, J., and Carnes, J. R. Robust diagnostic system: structural redundancy approach. *Proceedings of SPIE*, Vol. 2244, 1 (1994), 249–260.
- [7] Misra, A. Sensor-based diagnosis of dynamical systems. Ph.D. dissertation, Vanderbilt University, Nashville, TN, 1994.
- [8] Abdelwahed, S., G. Karsai, G., Mahadevan, N., and Ofsthun, S. C. Practical implementation of diagnosis systems using timed failure propagation graph models. *IEEE Transactions on Instrumentation and Measurement*, Vol. 58, 2 (Feb. 2009), 240–247.
- [9] Wilmering, T. Semantic requirements on information integration for diagnostic maturation. In *Proceedings of the 2001 IEEE Systems Readiness Technology Conference AUTOTESTCON*, Valley Forge, PA, 2001, 793–807.
- [10] Byington, C., Kalgren, P., and Donovan, B. Portable diagnostic reasoning for improved avionics maintenance and information capture continuity. In *Proceedings of the 2004 IEEE Systems Readiness Technology Conference AUTOTESTCON*, September 20–23, 2004, 518–524.
- [11] Wilmering, T. When good diagnostics go bad—why maturation is still hard. In *Proceedings of the 2003 IEEE Aerospace Conference*, March 8–15, 2003, 3137–3147.
- [12] Wilmering, T. J., and Sheppard, J. W. Ontologies for data mining and knowledge discovery to support diagnostic maturation. In *Proceedings of the 18th International Workshop on Principles of Diagnosis*, Nashville, TN, May 2007.
- [13] Agrawal R., and Srikant, R. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, 1994, 487–499.
- [14] Strasser, S., Sheppard, J., Schuh, M., Angryk, R., and Izurieta, C. Graph-based ontology-guided data mining for d-matrix model maturation. In *Proceedings of the 2011 IEEE Aerospace Conference*, Big Sky, MT, March 5–12, 2011, 1–12.
- [15] Institute for Software Integrated Systems. <http://www.isis.vanderbilt.edu/>.
- [16] IEEE Standards Association Press. Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE), Std 1232-2010, 2010.
- [17] IEEE Standards Association Press. Draft Trial-Use Standard for Software Interface for Maintenance Information Collection and Analysis (SIMICA): exchanging Maintenance Action Information (MAI) via the Extensible Markup Language (XML), Std P1636.2/D3, 2008.
- [18] International Organization for Standardization. Industrial automation systems and integration—product data representation and exchange—Part 11: description methods: The EXPRESS language reference manual. International Organization for Standardization, Std 10 303-11, 2004.
- [19] OWL 2 Web Ontology Language document overview. <http://www.w3.org/TR/owl2-overview/>, W3C, 2009.
- [20] Strasser, S., and Sheppard, J. Diagnostic alarm sequence maturation in timed failure propagation graphs. In *Proceedings of the 2011 IEEE Systems Readiness Technology Conference AUTOTESTCON*, September 2011, 158–165.
- [21] Abdelwahed, S., Karsai, G., and Biswas, G. System diagnosis using hybrid failure propagation graphs. In *Proceedings of the 15th International Workshop on Principles of Diagnosis*, Carcassonne, France, 2004.
- [22] Vachtsevanos, G., Lewis, F., Roemer, M., Hess, A., and Wu, B. *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. Hoboken, NJ: John Wiley & Sons, Inc., 2006.
- [23] Koller, D., and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA: MIT Press, 2009.
- [24] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann Publishers Inc., 1988.
- [25] Przytula, K., and Thompson, D. Construction of Bayesian networks for diagnostics. In *Proceedings of the 2000 IEEE Aerospace Conference*, Big Sky, MT, March 18–25, 2000, 193–200.
- [26] Shwe, M. A., Middleton, B., Heckerman, D. E., Henrion, M., Horvitz, E. J., Lehmann, H. P., et al. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, Vol. 30, 4 (1991), 241–255.
- [27] Wahl, S., and Sheppard, J. W. Extracting decision trees from diagnostic Bayesian networks to guide test selection. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*, 2010.
- [28] Sheppard, J. W., and Butcher, S. G. A formal analysis of fault diagnosis with d-matrices. *Journal of Electronic Testing: Theory and Applications*, Vol. 23, 4 (Aug. 2007), 309–322.
- [29] Aho, A. V., Garey, M. R., and Ullman, J. D. The transitive reduction of a directed graph. *Siam Journal on Computing*, Vol. 1, (1972), 131–137.
- [30] Wahl, S. Error-bounded probabilistic graphical model simplification. Ph.D. proposal, Montana State University, 2011.
- [31] Maurer, S. B., and Ralston, A. *Discrete Algorithmic Mathematics*. Wellesley, MA: A K Peters, 2004.
- [32] Schuh, M., Sheppard, J., Strasser, S., Angryk, R., and Izurieta, C. Ontology-guided knowledge discovery of event sequences in maintenance data. In *Proceedings of the 2011 IEEE Systems Readiness Technology Conference AUTOTESTCON*, Baltimore, MD, September 12–15, 2011, 279–285.