



A Formal Analysis of Fault Diagnosis with D -matrices

J. W. SHEPPARD AND S. G. W. BUTCHER

Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218, USA

jsheppa2@jhu.edu

stephyn@jhu.edu

Received May 15, 2006; Revised October 5, 2006

Editor: M. Abadir

Abstract. As new approaches and algorithms are developed for system diagnosis, it is important to reflect on existing approaches to determine their strengths and weaknesses. Of concern is identifying potential reasons for false pulls during maintenance. Within the aerospace community, one approach to system diagnosis—based on the D -matrix derived from test dependency modeling—is used widely, yet little has been done to perform any theoretical assessment of the merits of the approach. Past assessments have been limited, largely, to empirical analysis and case studies. In this paper, we provide a theoretical assessment of the representation power of the D -matrix and suggest algorithms and model types for which the D -matrix is appropriate. We also prove a surprising result relative to the difficulty of generating optimal diagnostic strategies from D -matrices. Finally, we relate the processing of the D -matrix with several diagnostic approaches and suggest how to extend the power of the D -matrix to take advantage of the power of those approaches.

Keywords: fault diagnosis, D -matrix, linear separability, linear discriminant fault tree

1. Introduction

Within the aerospace community and similar communities producing large, complex systems (e.g., the Department of Defense), considerable attention has been given to developing diagnostic systems based on a specific modeling paradigm—dependency modeling. Many available tools map their models into the so-called “ D -matrix” (derived from “dependency” matrix) and derive diagnostic strategies from this matrix. Recent research has even demonstrated a functional “equivalence” among a variety of graphical diagnostic models such as the behavioral Petri net, the bipartite Bayesian network, and the multi-signal flow model [17]. Multi-signal flow modeling is of particular interest because it is one example where tools have been developed that use the D -matrix [6]. Motivated by the widespread use of models based on the D -matrix, the IEEE published the derivative “diagnostic inference model” as a standard representation of this kind of model in IEEE Std 1232-2002 [15], and this standard is a candidate for inclusion in the DoD’s automatic test system framework [8] and the associated Automatic Test Markup Language (ATML) initiative [2].

Previously, Sheppard and Kaufman asserted that false alarms generally arise from multiple sources: human error, unpredictable or unmodeled environmental conditions, instrument uncertainty, or test design issues [26], and such false alarms can lead to false pulls and unnecessary maintenance actions. The test community’s desire to identify causes for false pulls during system maintenance motivates the work in this paper. In addition to false alarms, false pulls can be attributed to ineffective diagnostics arising from incomplete models, inaccurate models, or erroneous reasoning. We focus on incomplete diagnostic models in this paper.

In the following, we will consider the diagnostic problem from the perspective of pattern classification [9] and prove that a diagnostic model based upon the D -matrix instantiates a linearly separable classification problem. This is a significant result about the representational power of models based on the D -matrix. While similar results may be well known in the pattern classification community, the specific result for the D -matrix is new for the system test community. Because the D -matrix provides only linearly separable classification, a number of diagnostic inference algorithms based on the

D -matrix either have limited diagnostic power or are over-kill for the problem represented. We describe how the limitations arise and how they might be mitigated. We also prove two new results about the ability to derive optimal diagnostic strategies by showing that an optimal uniformly weighted linear discriminant fault tree can be constructed in polynomial time but that constructing an optimal non-uniformly weighted linear discriminant fault tree is NP -complete.

The organization of this paper is as follows. In the next section, we introduce the diagnostic inference model (DIM) and describe it as the foundation for many diagnostic approaches. Section 3 provides a summary of four different diagnostic algorithms and relates them to the DIM. Section 4 provides the first major result. In this section we define the structure of the D -matrix relative to the DIM and prove the linear separability of D -matrices. Section 5 then provides an optimal algorithm for constructing linear discriminant fault trees from D -matrices where the diagnoses are uniformly weighted. It then proves that this algorithm has polynomial complexity, but then shows that when there are non-uniform weights, finding an optimal linear discriminant fault tree is NP -complete. In Section 6, we discuss the implications of these results on the diagnostic algorithms presented in Section 3, and we conclude in Section 7.

2. Diagnostic Inference Models

The dependency model is a common form of model used in diagnostic systems. The dependency model is also known by other names such as the signal flow model, information flow model, the causal model, and the bipartite graphical model. One of the more complete descriptions of this model can be found in [31], and we summarize that description here. In the following, we will use the IEEE-standard term—the Diagnostic Inference Model, or DIM [15]. Note, however, that the DIM was derived as a generic representation of models based on the D -matrix, and the D -matrix will be defined formally in Section 4.

A DIM is built upon two fundamental model objects—tests (and their associated outcomes) and diagnoses (or diagnostic conclusions). A particular test can be any source of information that indicates the health state of the system (including symptoms, safe-to-turn-on tests, readiness tests, and diagnostic tests), and a particular diagnosis can be any diagnostic conclusion one wishes to draw about the system (including no fault found).

Inference relationships between tests and diagnoses as well as among tests are represented with a directed graph capturing information “flow” through the system. Specifically, let $V = T \cup D$ be a set of vertices, where T represents the set of tests and D represents the set of diagnoses. We define the set of directed edges to be de-

pendence relationships between tests and diagnoses (d_i, t_j) indicating a logical relationship corresponding to $d_i \Rightarrow t_j$. In other words, if diagnosis d_i is true, then test t_j will also be true and thereby detect the diagnosis. The directed graph corresponding to the logical relationships between tests and diagnoses can be represented in a bit-wise adjacency matrix, which has come to be called the D -matrix. An example of such a matrix is given in Table 1. In this matrix, a cell having a value of 1 indicates the corresponding logical relationship; and a value of 0 indicates no corresponding logical relationship. For example, this matrix shows $d_1 \Rightarrow t_1$ but $d_1 \not\Rightarrow t_2$.¹

3. Diagnostic Algorithms

A wide variety of inference algorithms have been proposed for fault diagnosis, many of which operate (or can operate) on the D -matrix. In the following, we will provide a brief overview of four such algorithms—rule based inference, set partitioning, Bayesian inference, and case based reasoning.

3.1. Rule Based Inference

Traditional rule based inference has often been used to diagnose faults using the D -matrix when diagnosis-to-test and test-to-test relationships are specified. In these cases, the rule $d_i \Rightarrow t_j$ is reversed using the logically equivalent form $\neg t_j \Rightarrow \neg d_i$, and algorithms such as forward chaining and backward chaining are applied [23]. If we let t_j denote test j failing and $\neg t_j$ denote test j passing then the challenge comes from considering the effect of multiple tests. Specifically, we find

$$\begin{aligned} d_i &\Rightarrow (t_j \wedge t_k \wedge \dots) \\ \neg(t_j \wedge t_k \wedge \dots) &\Rightarrow \neg d_m \\ (\neg t_j \vee \neg t_k \vee \dots) &\Rightarrow \neg d_m. \end{aligned}$$

This rule form becomes tricky for a chaining-type inference system and must be coupled with corresponding rules of the type $t_j \Rightarrow (d_i \vee d_k \vee \dots)$, which fall out from the complete set of rules by disjuncting the rules with common consequents.

Table 1. Bit-wise adjacency Matrix— D -matrix.

	t_1	t_2	t_3	t_4	t_5	t_6
d_1	1	0	0	1	1	0
d_2	0	1	0	1	0	0
d_3	0	1	1	0	1	1
d_4	0	0	0	1	0	1
d_5	0	0	1	0	1	1

3.2. Set Partitioning

The most widely used algorithm for processing D -matrices is based on set partitioning and set intersection [31]. Usually, these algorithms impose a single fault assumption to reduce the computational complexity; however, recent tools (e.g., QSI TEAMATE and DSI eXpress) relax this requirement. Fundamentally, diagnosis operates by observing that tests indict or clear the diagnoses attached to them, as indicated by entries in the D -matrix. Sets of cleared and suspected diagnoses are maintained and updated as tests are performed. Whenever a test fails, the set of candidate diagnoses is updated as follows:

Let S be the set of suspect diagnoses already identified. Let C be the set of cleared diagnoses (identified when prior tests have passed). Let I_j be the set of diagnoses indicted by a test t_j failing. Then we update the set of suspect diagnoses using $S \leftarrow S \cap I_j$. The set of cleared diagnoses is updated when test t_j passes as $C \leftarrow C \cup c_j$. The process continues until some termination criterion is met such as reducing S to a set of diagnoses sufficient to apply a maintenance action.

The partitioning process arises from a natural approach to fault diagnosis and is applied in many tools. For example, many model-based tools construct decision trees or paths based on test results. The most common approach to constructing such a tree is by choosing tests that maximize information gain [20, 31]. By considering the possible test outcomes, the set of possible diagnoses is partitioned, and a new subtree is constructed for each partition. This process continues recursively until the termination criterion is satisfied, and the result is a fault tree (see Section 5).

Recent approaches in constructing decision trees have also considered performing multiple tests at a particular node of the tree to reduce the size of the overall tree. Constructing such “oblique” trees² also has advantages for building general decision trees because of the ability to consider tests that are correlated in some way [19].

3.3. Bayesian Inference

Recently, Bayesian methods have gained popularity, and a widely used Bayesian model is the bipartite network [26, 28, 29]. Using this model, we assume the random variables in D (i.e., the diagnoses) are independent, as are the random variables in T (i.e., the tests). Now the characteristics of conditional independence allow for simple propagation of the probabilities from the tests to the diagnoses.

Given the conditional independence of the diagnoses, one can compute the posterior probabilities of each of the diagnoses given the test results as follows. First, we will assume that we are using the network form presented in Fig. 1 and partition the random variables into three sets: D (the diag-

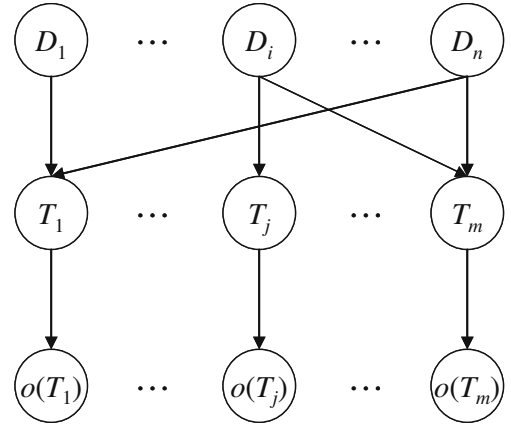


Fig. 1. Bipartite diagnostic Bayesian network.

noises), T (the true test states), and O (the test observations). The evidence variables will be restricted to O .

$$\begin{aligned} \Pr(d_i|O) &= \alpha \Pr(O|d_i) \Pr(d_i) \\ &= \alpha \Pr(d_i) \sum_{t_j \in T} \Pr(o(t_j)|t_j) \Pr(t_j|d_i) \end{aligned}$$

Here, α is a normalizer over the set D , equal to

$$\alpha = \sum_{d_i \in D} \Pr(d_i) \sum_{t_j \in T} \Pr(o(t_j)|t_j) \Pr(t_j|d_i).$$

Assuming we are able to generate the probability distributions for nominal and faulty behavior, we consider the effects of locating the decision boundaries. For this discussion, we will draw on results from Bayes decision theory and its derivative, signal detection theory [9].

Observe that $\Pr(o(t_j) | d_i) \in \{0, 1\}$, so the members of the sum are restricted only to those tests that observe d_i . Because this corresponds exactly to the D -matrix, we only need to consider two things: $\Pr(d_i)$, which corresponds to the prior probability for d_i based on failure rate, and $\Pr(o(t_j) | t_j)$, which corresponds to the confidence value assigned to the observed test result. Using the Bayes’ maximum *a posteriori* hypothesis, we determine the most likely diagnosis simply as

$$D_{\text{MAP}} = \arg \max_{d_i \in D} \{\Pr(d_i|O)\}$$

3.4. Case Based Reasoning

Case based reasoning (CBR) is a method of reasoning that combines elements of instance-based learning and database query processing [1]. Test and diagnosis can use CBR in several ways. The simplest method involves defining a case as a collection of test results and attempting to determine an appropriate diagnosis given these results. The retrieval process is very simple. All of the cases are nothing more than feature vectors with an associated diagnosis $\langle t_1, t_2, \dots, t_n; d_i \rangle$. The features in the feature vector correspond to test results and may be unknown. Retrieval then

consists of “matching” the new case with all of the cases stored in the case base and selecting the most similar case.

When considering possible similarity metrics, numerical features are frequently compared using a member of the family of L_p norms. Let \mathbf{x}' and \mathbf{x}'' be two feature vectors where any given x'_i corresponds to the i th feature in \mathbf{x}' (similarly for \mathbf{x}''). These features could correspond, for example, to test results. An L_p norm is defined to be

$$L_p(\mathbf{x}', \mathbf{x}'') = \left(\sum_i (x'_i - x''_i)^p \right)^{1/p}.$$

The most common values for p are 1, 2, and ∞ and yield Manhattan distance, Euclidean distance, and max-norm distance, respectively. Specifically, these metrics can be computed as:

$$\text{Manhattan : } L_1(\mathbf{x}', \mathbf{x}'') = \sum_i |x'_i - x''_i|$$

$$\text{Euclidean : } L_2(\mathbf{x}', \mathbf{x}'') = \sqrt{\sum_i (x'_i - x''_i)^2}$$

$$\text{Max-norm : } L_\infty(\mathbf{x}', \mathbf{x}'') = \max_i |x'_i - x''_i|$$

If we were using pass/fail results for testing, we would use either L_1 or L_2 (note that both essentially reduce to Hamming distance). This is exactly what is done with fault dictionary-based diagnosis. With real values, we would most likely use L_2 . Symbolic results (beyond pass and fail) are a bit more complicated and would require something like Stanfill and Waltz’s “value difference metric” [32]. Regardless of the metric, retrieval would be done for some test vector \mathbf{x}

$$\text{case} = \underset{\forall c \in \text{CASE_BASE}}{\text{arg min}} \{L_p(\mathbf{x}, \mathbf{c})\}.$$

As alluded to above, the digital fault dictionary is an application of the case based approach that matches the D -matrix representation explicitly [25]. Fault dictionaries define a mapping from combinations of input vectors and output vectors to faults. Formally, this is represented as $FD : I \times O \rightarrow F$ where FD is the fault dictionary, I is the space of input vectors, O is the space of output vectors, and F is the space of faults. At a more basic level, this can be represented as $FD : \{0, 1\}^n \times \{0, 1\}^m \rightarrow F$.

We convert the fault dictionary into a D -matrix by comparing test results in the presence of a fault to expected test results when the circuit is not faulty. As an example, consider the circuit in Fig. 2. The corresponding fault dictionary is in Table 2 where \mathbf{v}_i corresponds to a particular test vector. The columns identify stuck-at faults at the labeled location on the circuit. In this case, the fault dictionary is simply the transpose of the D -matrix. To derive the D -matrix, we place a 1 in the corresponding cell of the matrix if these values are different and a 0 in the cell if they are the same. If the value in the cell is 1, we say the associated failure mode “causes” the given test to fail. If the value is 0, the presence

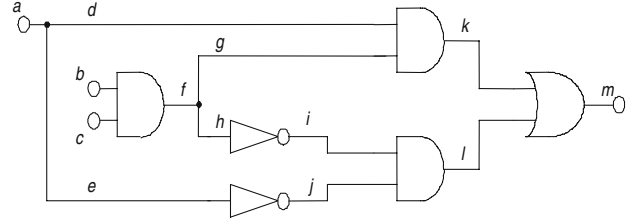


Fig. 2. Sample combinational circuit.

of the associated failure mode will not be detected by the given test. Given a complete row in the matrix, we say that if the associated failure mode is present, then all of the tests associated with the failure mode (i.e., whose cells have a value of 1) must fail. Conversely, if any of those tests pass, then the failure mode must not be present.

3.5. Application of D -matrices

Diagnostic models with D -matrices have been used extensively in several real-world applications. Results ranging from reduction in required test assets, optimized diagnostic processes, increased diagnostic accuracy, and decreased false alarm rates have been reported [30]. A recent detailed case study at the University of Connecticut showed how D -matrices were applied to diagnosing power quality problems on a power distribution system [3]. The referenced paper includes a portion of a D -matrix in which 49 different fault conditions are monitored. For this D -matrix, fault categories and associated detection events are defined, and their relationships are captured in the D -matrix. For example, a fault category of “Closing Power Factor Correction Capacitor” can be detected by two events—a low frequency oscillatory transient and a medium frequency oscillatory transient. This signature then defines one of the 49 rows in the matrix. Azam et al. then proceed to illustrate how to use tests looking for different types of transients to diagnose this specific fault.

A second example of a model based on the D -matrix is the Temporal Causal Graph. The TCG has been applied, for example, to the diagnosis of a reverse osmosis system, which was part of a larger water recovery system [21]. In this example, specific fault signatures of components are defined in terms of whether there is a positive change in flow, a negative change in flow, or no change. The TCG depends on the definition of “weakly independent” fault signatures. These signatures correspond directly with the fault signatures in the D -matrix, except that the cell entries can take on one of three states. In other words, there still must be a single, unique signature for each fault.

There is no question that, if there is a single unique signature identifying a particular fault or fault category, the D -matrix will provide an excellent way of finding that fault. As we will show in the next section, the added assumption that a particular fault only needs to have one signature to characterize it (which is typical of these types of models) is problematic.

Table 2. Fault dictionary for sample circuit.

Test	Fault signatures														
	a_0	a_1	b_1	c_1	d_1	f_0	f_1	g_1	i_0	i_1	j_1	k_0	k_1	m_0	nf
v_1	0	1	0	0	1	1	0	0	0	1	0	0	1	0	0
v_2	1	0	0	1	0	0	1	1	0	0	1	0	1	0	0
v_3	1	0	1	0	0	0	1	1	0	0	1	0	1	0	0
v_4	0	1	1	1	1	0	1	1	1	1	1	0	1	0	1
v_5	1	0	0	1	1	1	0	1	0	1	1	1	1	0	1
v_6	1	0	1	1	1	1	0	1	0	1	1	1	1	0	1
v_7	1	0	1	0	1	1	0	1	0	1	1	1	1	0	1
v_8	1	0	0	0	0	0	1	1	0	0	1	0	1	0	0

4. Linear Separability

In the following, we will discuss the relationship between linear separability and diagnostic systems based on the D -matrix and extend upon the treatment of linear separability in [27]. First, we formally define that a D -matrix is a form of knowledge representation supporting fault diagnosis equivalent to a classification problem. Then we define linear separability for diagnostic models and show that the D -matrix as defined is linearly separable. This is significant because, if the type of model can only support linearly separable diagnosis problems (e.g., AND), but the complexities of the underlying system are not linearly separable (e.g., XOR), then the model is not capable of accurate diagnosis for all of the faults in the system.

We begin this discussion by showing that the diagnostic problem can be posed as a classification problem. Formally, we define a Boolean classifier as follows.

Definition 1 Let C represent a set of concepts or classes. Let A represent a set of attributes (or features) of some object or concept. Assume each $c_i \in C$ is a Boolean variable such that $\text{eval}(c_i) \in \{0,1\}$. Assume each $a_j \in A$ is also a Boolean variable such that $\text{eval}(a_j) \in \{0,1\}$. Then a *feature vector* is defined to be the vector

$$\mathbf{c}_i = [\text{eval}(a_1), \dots, \text{eval}(a_{|A|})],$$

where

$$\text{eval}(a_j) = \begin{cases} 1 & \text{if } a_j \text{ is an attribute of } c_i \\ 0 & \text{otherwise.} \end{cases}$$

Definition 2 A *Boolean classifier* is a mapping $f: A \rightarrow C$

Definition 3 Let D represent the set of diagnoses to be considered. Let T represent the set of tests to be

considered. Assume each $d_i \in D$ is a Boolean variable such that $\text{eval}(d_i) \in \{0,1\}$. Assume each $t_j \in T$ is also a Boolean variable such that $\text{eval}(t_j) \in \{0,1\}$. Then a *diagnostic signature* is defined to be the vector $\mathbf{d}_i = [\text{eval}(t_1), \dots, \text{eval}(t_{|T|})]$, where

$$\text{eval}(t_j) = \begin{cases} 1 & \text{if } t_j \text{ detects } d_i \\ 0 & \text{otherwise.} \end{cases}$$

and $\mathbf{d}_i[j]$ is the j th element in vector \mathbf{d}_i .

Definition 4 A D -matrix is defined to be the set of diagnostic signatures \mathbf{d}_i for all $d_i \in D$.

By combining the definitions above with the following assignments

$$A = T, C = D, \text{ and } c_i = d_i,$$

we see that the diagnosis problem is a form of a classification problem.

There are some very specific restrictions in the above definition, however. First, we are assuming all attributes are Boolean (i.e., we are not permitting nominal or real-valued attributes at this point). Second, we are assuming there is a single feature vector sufficient to characterize each class. Third, although this is not really a restriction, we are assuming that diagnoses with equivalent signatures (i.e., ambiguous diagnoses) can be collapsed to a single row in the D -matrix representing the entire ambiguity group.

Consider this restriction on feature vectors. We will see that this is a significant restriction that rarely, if ever, occurs in practice. In fact, the general classification problem does not assume a corresponding “ D -matrix” but generates the classifier from a set of training instances where multiple, varying feature vectors can exist for each class label. Nevertheless, most graphical model-based diagnostic systems (e.g., dependency models, multi-signal flow models,

and fault dictionary models) are based on the D -matrix formalism.

From this point forward, we will use the language of diagnosis rather than classification unless we need to apply a result from classification theory. At that point, the association to diagnosis will be made explicit.

First, consider the case where we have only two diagnoses. Arguably, this is the simplest diagnostic problem since only one diagnosis would be trivially true (assuming a closed set).

Definition 5 Let \mathbf{w} be a column vector of weights, \mathbf{a} a column vector of attributes, and θ a threshold. Two concept classes are *linearly separable* if and only if there exists a linear function $\mathbf{w}^T \mathbf{a} - \theta = 0$ such that one concept is identified when $\mathbf{w}^T \mathbf{a} - \theta > 0$ and the other concept is identified when $\mathbf{w}^T \mathbf{a} - \theta \leq 0$.

Theorem 1: *Given two diagnoses d_1 and d_2 with distinct diagnostic signatures d_1 and d_2 (i.e., $d_1 \neq d_2$), then d_1 and d_2 are linearly separable.*

Proof: The logical representation of the associated Boolean classifier is given as

$$d_i \Leftrightarrow \prod_j (\text{eval}(t_j) = d[j])$$

where the product symbol, Π corresponds to logical AND. We can convert this into a linear expression as follows. First, let $\mathbf{w} = [1 \dots 1]^T$. This permits us to “ignore” the weight vector in the following derivation since it has no effect on the result. Next, let

$$a_j = 1 - |\text{eval}(t_j) - d[j]| + \varepsilon$$

where ε is any positive constant. Now

$$\prod_j (\text{eval}(t_j) = d[j]) = \prod_j a_j$$

and this product is maximized if and only if the signature d_i is matched by the set of test outcomes $T = [\text{eval}(t_1), \dots, \text{eval}(t_{|T|})]$ for some set of actual test evaluations t_i . Then define the following:

$$C(T) = \log \prod_j a_j = \sum_j \log a_j > \theta.$$

We can use this linear function describing diagnosis d_i as a discriminant by setting $\theta = (|T| - 1) \log(1 + \varepsilon) + \log(\varepsilon)$. Specifically, when $C(T) > \theta$, the test vector T

will be classified as diagnosing d_i . Thus, for a pair of diagnostic signatures, classification (i.e., diagnosis) is linearly separable. ■

What happens when there are more than two distinct diagnostic signatures? As we will see, the result is the same.

Corollary 1: *Given a set of diagnoses D with distinct diagnostic signatures $d_1, \dots, d_{|D|}$, then the set D is linearly separable.*

Proof: This is evident from the fact that our linear discriminant $C(T) > \theta$ is satisfied whenever T matches the corresponding vector. Given the way θ was defined, even one mismatch will cause $C(T)$ to be less than θ . ■

5. Finding Fault Trees

Consider the problem of finding a fault tree from a D -matrix to be used to perform diagnosis. It is well known that the general problem of finding an optimal binary decision tree (where optimality is defined as minimizing the expected number of tests required to classify some object) is NP -complete [14]. Since fault trees are simply binary decision trees where the classes correspond to diagnoses, it is reasonable to expect that finding optimal fault trees is also NP -complete. In the following, we will show that when constructing “linear discriminant” fault trees from a D -matrix, this is not necessarily the case. We begin by defining some terms to assist us in understanding the implications of these results.

Definition 6 A *decision tree* is a tree data structure where

- Each interior (i.e., non-leaf) node is identified by a single attribute;
- Each subtree of an interior node is identified by a specific value of the node’s attribute;
- Each leaf node is identified by a class label.

As we see, a decision tree is a data structure that supports classification as described in Section 4. Specifically, classification corresponds to starting at the root of the tree and evaluating the attributes at each of the nodes encountered in the tree. The tree is traversed along the edges to the appropriate subtree based on the values of the attributes at each interior node. When the leaf node is reached, the class label associated with that node is returned as a classification of the instance.

Definition 7 A *fault tree* is a decision tree where the attributes correspond to tests, and the class labels correspond to diagnoses.

Definition 8 A *linear discriminant decision tree* is a decision tree where the interior nodes of the tree are identified by linear combinations of all attributes. (Note that single-attribute decision trees are a special case of linear discriminant decision trees where the coefficients of all but one attribute are set to zero.)

Definition 9 A *linear discriminant fault tree* is a fault tree where the interior nodes of the tree are identified by linear combinations of all tests. (Note that, just as linear discriminant decision trees are generalizations of single-attribute decision trees, linear discriminant fault trees are generalizations of single-test fault trees.)

5.1. Uniform Linear Discriminant Fault Trees

Examples of single-test and linear discriminant fault trees are shown in Fig. 3.

Lemma 1: *Given a set of diagnoses D with distinct diagnostic signatures $d_1, \dots, d_{|D|}$, then there exist at least $\lceil \lg |D| \rceil$ tests in the D -matrix capable of uniquely differentiating those diagnostic signatures.*

Proof: Suppose we have $|D|$ unambiguous diagnoses but $|T| < \lceil \lg |D| \rceil$ tests. To start, let $|T|$ be a power of 2. Then $\lceil \lg |D| \rceil = \lg |D|$. Removing one test from the D -matrix reduces the number of tests by one, thus satisfying the above condition. Now with only $\lg |D| - 1$ tests, there exist only $2^{\lg |D| - 1}$ distinct diagnostic signatures. But we then see that

$$2^{\lg |D| - 1} = 2^{\lg |D|} \cdot 2^{-1} = \frac{1}{2}|D|$$

so we can no longer uniquely differentiate all $|D|$ diagnoses. If $|D|$ is not a power of 2, we can still only represent

$$2^{\lceil \lg |D| - 1 \rceil} = \frac{1}{2}2^{\lceil \lg |D| \rceil} < |D|$$

distinct diagnoses. ■

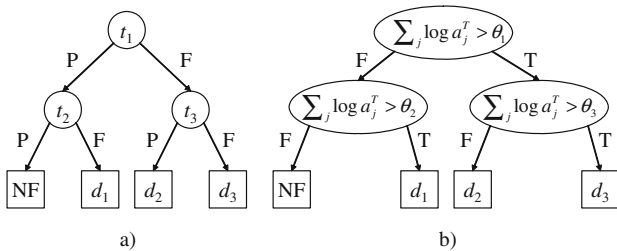


Fig. 3. Example fault trees. **a** A single-test fault tree where each interior node corresponds to a single pass/fail test. **b** A linear discriminant fault tree where each internal node compares a complete test signature (see Section 4) to a discriminant threshold θ_i .

Theorem 2: *Given a set of diagnoses D with distinct diagnostic signatures $d_1, \dots, d_{|D|}$, then there exists an optimal linear discriminant dividing the D -matrix into two submatrices.*

Proof: We note that an optimal linear discriminant is one that splits the set of diagnoses in half. Because there exist at least $\lceil \lg |D| \rceil$ tests in the D -matrix, and each test is represented as a Boolean attribute, we can convert the diagnostic signature (which is essentially a binary integer) for each d_i into a base-10 integer equivalent and find the median of the resulting set of integers. ■

Note that splitting based on the median corresponds to a linear combination of all of the tests. As mentioned earlier, research in decision tree learning refers to these splits as “oblique.” Splits based on single tests are called “axis-parallel” [19] because the corresponding linear discriminant is parallel with one of the axes defined by the set of tests.

Corollary 2: *Finding the optimal linear separator for a D -matrix can be performed in polynomial time.*

Proof: Follow the procedure described in Theorem 2 to find the median by converting the diagnostic signatures to their integer equivalents. This requires $O(|D| \cdot |T|)$ time. The median of n numbers can be found using the SELECT algorithm described in [5] in $O(n)$ time. That median can then be used to define a corresponding linear separator based on Theorem 1, and that separator can be found in $O(|T|)$ time. Since $n = |D|$, the total complexity to find the optimal linear separator in a D -matrix is $O(|D| \cdot |T|)$ time. ■

Theorem 3: *Given a set of diagnoses D with distinct diagnostic signatures $d_1, \dots, d_{|D|}$, then finding an optimal linear discriminant fault tree using the corresponding D -matrix can be performed in polynomial time.*

Proof: As shown in Corollary 2, each split in the fault tree can be found in $O(|D| \cdot |T|)$ time. Because that split is based on the median, it is also optimal. In addition, because the split is optimal, the maximum depth of the fault tree will be $O(\lg |D|)$. Therefore, the total complexity for building the optimal fault tree is $O(|D| \cdot |T| \lg |D|)$, which is polynomial. ■

The algorithm described in Corollary 2 and Theorem 3 is shown in Table 3 and is called with $tree.ptr \leftarrow OPTIMALFAULTTREE(D, T)$. To interpret the algorithm, we see that the tree is built recursively. The recursion “bottoms out” when the set of diagnoses D has been reduced to a single item (recall the assumption we have distinct diagnoses). Otherwise, the first step is to find the integer equivalents to each signature and then find the median using the SELECT algorithm. That median becomes the root discriminant, and the left hand partition for the tree (*Left*) is determined by testing each member

of D against the discriminant function (Linear Discriminant) and then deriving the actual partition by testing for values less than or equal to the split. The optimal subtree is then generated by calling Optimal Fault Tree on the left-hand partition. The same process is followed on the right-hand partition to complete the tree.

5.2. Non-uniform Linear Discriminant Fault Trees

The algorithm in Table 3 makes an implicit assumption that the probability distribution of the diagnoses is uniform. In other words, to apply the median principle to construct the fault tree, we must first assume that all diagnoses are equally likely to occur. In practice, this rarely (if ever) happens. A reasonable question to ask is what impact a non-uniform probability distribution would have on the ability to find optimal fault trees. In fact, as we will show, the simple transition from uniform to non-uniform probability distributions causes a significant problem.

Prior to proving just how significant a problem having non-uniform probabilities is, we will focus on an interesting characteristic of the OPTIMALFAULTTREE algorithm as defined. Recall, we are creating linear discriminant trees to diagnose faults. These linear discriminant trees can be related to “binary space partition trees” from computer graphics [33]. In a BSP-tree, a d -dimensional space is hierarchically partitioned with linear hyperplanes. Clearly, that is what our linear discriminant fault tree is doing; however, we are constraining the linear discriminant fault tree by limiting the set of possible hyperplanes to those defined by the numeric representation on the signatures and median hyperplanes separating adjacent signatures.

Table 3. Optimal fault tree algorithm.

```

Algorithm OPTIMALFAULTTREE(D, T)
  local: DInt, Left, Right,  $i$ ,  $split$ ,  $root$ ;

  if  $|\mathbf{D}| = 1$  then
    return  $ptr.\mathbf{D}[1]$ ;
  for  $i \leftarrow 1$  to  $|\mathbf{D}|$  do
     $\mathbf{DInt}[i] \leftarrow \text{INTEGERCONVERT}(\mathbf{D}[i])$ ;
   $split \leftarrow \text{SELECT}(\mathbf{DInt}, |\mathbf{D}|/2)$ ;
   $root.test \leftarrow \text{BINARYCONVERT}(split)$ ;
   $\mathbf{Left} \leftarrow \text{LINEARDISCRIMINANT}(root, “\leq”)$ ;
  if  $|\mathbf{Left}| \geq 1$  then
     $root.right \leftarrow \text{OPTIMALFAULTTREE}(\mathbf{Left}, \mathbf{T})$ ;
   $\mathbf{Right} \leftarrow \text{LINEARDISCRIMINANT}(root, “>”)$ ;
  if  $|\mathbf{Right}| \geq 1$  then
     $root.right \leftarrow \text{OPTIMALFAULTTREE}(\mathbf{Right}, \mathbf{T})$ ;
  return  $root$ ;

```

The imposition of this numeric representation on the fault signatures suggests an alternative algorithm for constructing optimal fault trees—the discovery of optimal binary search trees. In a binary search tree, we store numerically keyed objects in a binary tree data structure under the assumption that a total-order has been defined over these numeric keys. This is exactly what we are doing in the linear discriminant fault tree; therefore, it would seem reasonable to look for an algorithm to optimize the search through such a tree. Cormen et al. describe a dynamic-programming algorithm for constructing optimal binary search trees when the probabilities of querying the key values are known [5]. These probabilities correspond exactly to the failure probabilities we would impose on the diagnoses in our model and need not be uniform.

To utilize the algorithm in [5], we need to make one change in the structure of the fault tree. Specifically, the interior nodes of the tree must correspond to fault signatures rather than tests that partition the fault signatures. However, this is not a problem because the fault signatures themselves can be used as tests because they too partition the space. With this change, it can be proven that for a particular ordering of tests and a particular failure distribution defined over the diagnoses, an optimal linear discriminant fault tree can be constructed in polynomial time using the optimal binary search tree algorithm.

While promising, there is still a problem that we will demonstrate with an example. Suppose we have a very simple system with three diagnoses—*no fault*, d_1 , and d_2 . Suppose, further, that there are two tests, t_1 and t_2 , where t_1 observes only d_1 and t_2 observes only d_2 . This leads to the following signatures:

1. 00 (*no fault*)
2. 01 (d_1)
3. 10 (d_2)

Now let’s assume we apply a probability distribution of

$$(\pi(\text{no fault}) = 0.05, \pi(d_1) = 0.05, \pi(d_2) = 0.90).$$

Since the tests have no natural order, t_1 and t_2 are simply convenient labels and their ordering in the test signature is similarly one of convenience. If we were to take t_2 first and t_1 second as a result of swapping their columns in the D -matrix, we would get the following signatures:

1. 00 (*no fault*)
2. 10 (d_1)
3. 01 (d_2)

Using the same probability distribution for the diagnoses from above, we end up with the two trees shown in Fig. 4. In this figure, the “a” side shows the D -matrix,

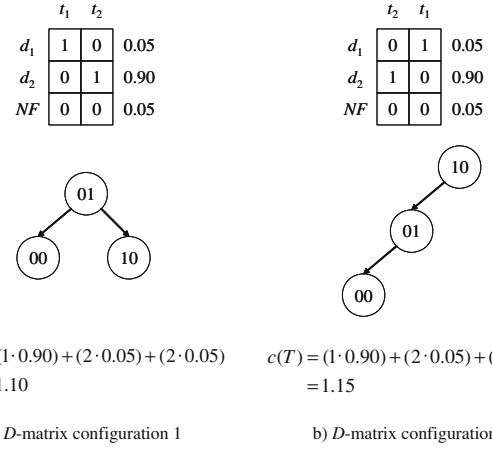
fault tree, and expected cost calculation for the first configuration (t_1 “first”), and the “b” side shows the same information for the second configuration (t_2 “first”). As this figure demonstrates, two optimal binary search trees are derived from logically equivalent D -matrices yet yield different expected costs. Therefore, while it is true that the fault trees are indeed optimal, they are optimal binary search trees with respect to their own particular ordering of the tests. As fault trees, there is clearly one that is “better” than the other simply because the columns of the D -matrix were re-ordered. In fact, we can demonstrate that this phenomenon occurs even when the interior nodes do not match fault signatures but simply partition the signatures as in Table 3.

What Fig. 4 demonstrates is that we have changed the nature of the optimization problem from one of finding the best fault tree based on the probabilities of the faults to one of finding the best permutation of the tests for determining the diagnostic signatures to derive the corresponding optimal fault tree. But the second is a much more difficult problem involving evaluating, in the worst case, all possible permutations of the tests. This leads to the final major result. In the following, we assume interior nodes of the tree correspond to combinations of tests but do not match diagnostic signatures directly. In other words, diagnoses appear only at the leaves.

Theorem 4: *Given a set of diagnoses D with distinct diagnostic signatures $d_1, \dots, d_{|D|}$ and a set of non-uniform probabilities $\pi(d_1), \dots, \pi(d_{|D|})$, then finding an optimal linear discriminant fault tree using the corresponding D -matrix is NP -complete.*

Proof: First, we must transform the optimal linear discriminant fault tree problem into a decision problem (which we denote LDFT). The LDFT decision problem is to determine whether there exists a linear discriminant fault tree with an expected path length from the root to a diagnosis less than or equal to some value $\Lambda > 0$. Given this transformation, it is straightforward to show that LDFT is in NP . Specifically, an oracle can “guess” a fault tree and determine if that fault tree’s expected path length is less than or equal to Λ simply by summing the lengths of each path from the root to a leaf, multiplying that path length by the probability of the diagnosis at the leaf, and averaging those weighted path lengths.³ Since each path is at most $O(|D|)$ in length and there are $|D|$ such paths, the complexity of such a calculation is $O(|D|^2)$, which is polynomial in D .

Now, to show that LDFT is NP -complete, the next step is to show that LDFT is NP -hard by providing a polynomial-time reduction of another problem known to be NP -hard to this problem. We will do this by identifying an NP -complete problem that is a subproblem



$$c(T) = (1 \cdot 0.90) + (2 \cdot 0.05) + (2 \cdot 0.05) = 1.10$$

$$c(T) = (1 \cdot 0.90) + (2 \cdot 0.05) + (3 \cdot 0.05) = 1.15$$

Fig. 4. Alternative optimal fault trees of logically equivalent D -matrices.

of the LDFT problem (as opposed to reducing PARTITION to LDFT), namely the PARTITION problem.

Recall that the PARTITION problem is defined to be the decision problem seeking to answer the question, “Given a set of integers Z , is there a way to divide the set into independent subsets, Z_1 and Z_2 such that $Z_1 \cup Z_2 = Z$, $Z_1 \cap Z_2 = \emptyset$, and $\sum_{z \in Z_1} z = \sum_{z \in Z_2} z$ Let $Z = \{ \pi(d_1), \dots, \pi(d_{|D|}) \}$ [12]. Recall that we are treating all of the diagnoses as if they are independent; therefore, the probabilities of failure are also independent. Let us define $\pi(D') = \sum_{d_i \in D'} \pi(d_i)$, where $D' \subseteq D$. For LDFT, if we can find the above partition, then $\pi(Z_1) = \pi(Z_2)$. In other words, the probabilities of the two respective subsets are equal, which leads to an optimal split of the diagnoses. Thus finding the optimal split is NP -hard.

The solution to the PARTITION problem identifies an ideal split of the current set of diagnoses based on their probabilities. After finding this ideal split, the tests need to be permuted to find the correct linear separator to ensure the subtree partitions agree with the probability partitions. Specifically, all diagnoses in one partition must be less than the test split, and all diagnoses in the other partition must be greater than the test split. Even assuming the new permutation can be found in polynomial time, finding the optimal split is still NP -hard. Because the problem is repeated at each branch of the tree, finding the optimal tree is also NP -hard.⁴ ■

Note that finding an approximately even split is the same problem, since any particular split equates to the PARTITION problem. Note also that this result is similar to the result given by Hyafil and Rivest in [14] except that Hyafil and Rivest made no assumption that there was exactly one path in the decision tree for each class label x_i (using their notation). Their result relates to ours in that their x_i is equivalent to our d_i , and their decision tree is

the same as our fault tree. In addition, Heath proved that finding an optimal split in an arbitrary oblique decision tree is also *NP*-complete [13]. Our result strengthens both the results in [13] and [14] by identifying a limiting case for intractability and showing intractability still holds for a small extension to that limiting case.

6. Implications for Diagnostic Algorithms

The immediate conclusion to be drawn from the above analysis is that the *D*-matrix provides a simplistic view of the diagnostic problem. As defined, the *D*-matrix is a “conjunctive” model in that it specifies the logical-AND of the test results associated with a particular diagnosis. The advantage is that most diagnostic algorithms are able to process such models easily, and we described several example algorithms in Section 3.

Now, if such a simple model can address real-world fault diagnosis requirements, then the underlying concept class so modeled must be linearly separable. In such cases, a variety of “simple” classifiers can be constructed to learn the concept, ranging from naïve Bayes classifiers [16] to single-layer perceptrons [18]. On the other hand, if such a simple model cannot address real-world fault diagnosis requirements, then the underlying concept is probably not linearly separable. This means a more complex model and associated classifier must be used (e.g., decision tree [20], augmented Bayes classifier [11, 27] or multi-layer perceptron [22]) to represent the concept. In fact, simply adding noise to the model can result in the need for handling non-linearly separable models.

6.1. Example: A Non-linearly Separable Problem

To illustrate the limitations of linearly separable models, suppose we wish to construct a system-level model where each signature in the *D*-matrix corresponds to a single subsystem that might be faulty.⁵ As a specific example, suppose we are attempting faulty isolate the stability augmentation system (SAS) of a helicopter. This example was chosen because it corresponds to a real-world problem, it is easy to conceptualize, and the corresponding model is not particularly complex. The SAS consists (at a minimum) of three gyros (one each for roll, pitch, and yaw), three servos (again, one each for roll, pitch, and yaw), and a mixing circuit. The purpose of the mixing circuit is to compare digital and analog inputs (see below) and to determine the appropriate mix of roll, pitch, and yaw corrections.

Consider the mixing circuit, which one could argue serves as a potential single-point failure for the SAS. The mixing circuit will determine differential corrections based on offset from a target orientation and apply, for example, *PID* control laws to signal the appropriate controls to the effectors.⁶ Fischer and Sivahop describe a

SAS design consisting of redundant gyros feeding the mixing circuit to drive the servos [10]. For their design, they have six gyros where three of the gyros provide analog signals, and three provide digital signals to the mixing circuit. Now suppose that tests are constructed that only compare the aircraft orientation as inputs to the corrections generated by the servos. Three tests, one for roll, pitch, and yaw, are assumed. This, of course, yields an underspecified diagnostic model in that the three tests are only capable of differentiating at most eight diagnoses. The SAS example based on [10] has 11 diagnoses (six gyros, three servos, the mixing circuit, and no-fault) and considerable ambiguity.

In spite of the fact the model is underspecified, let us focus on failure of the mixing circuit. Clearly, this mixing circuit embodies several possible failure modes; however, our assumptions have reduced the model to consider a single, global failure mode of the mixing circuit as a whole. Consider the *D*-matrix in Table 4. There are two different approaches to using this matrix for performing fault diagnosis—1) perform all of the tests and compare the result to the fault signatures or 2) run the tests incrementally and reduce the space of possible faults as we go. Consider the first approach. The problem with this matrix is that failure of the mixing circuit requires all three tests to detect that failure to correctly diagnose the problem. Unfortunately, it is possible that the mixing circuit may fail in several ways that do not involve all three dimensions of control. For example, certain failure modes could result in canceling effects between the redundant gyros and the associated servo, thus leading to complex dependence relationships between the components of the SAS that are not linearly separable. In fact, if we only have two tests that fail, the best we could do is to find the nearest match (unfortunately, all other faults are equally distant), guess based on failure rate information, or declare an error. Notice further that removing any of the tests from the signature

Table 4. *D*-matrix for stability augmentation system.

	Roll test	Pitch test	Yaw test
Roll Gyro 1	1	0	0
Roll Gyro 2	1	0	0
Pitch Gyro 1	0	1	0
Pitch Gyro 2	0	1	0
Yaw Gyro 1	0	0	1
Yaw Gyro 2	0	0	1
Roll servo	1	0	0
Pitch servo	0	1	0
Yaw servo	0	0	1
Mixing circuit	1	1	1
No fault	0	0	0

would end up missing failure modes involving the associated axis.

The other approach involves an incremental evaluation of the tests. This approach shows some promise. For example, suppose we run the Roll Test and the Pitch Test and they both fail. At this point, an incremental diagnostic system would halt and declare the mixing circuit as faulty, and this is probably correct. On the other hand, suppose the fault involves an incorrect mixing of roll and yaw corrections. This time, the Roll Test fails but the Pitch Test passes. At this point, the diagnostic system would halt and declare the roll section (one of the roll gyros or the roll servo) as being faulty.

6.2. Non-linear Separability and Multiple Faults

So far, the work performed has focused on the expressiveness of the D -matrix when diagnosing single faults. One of the outcomes of this analysis was a new algorithm for generating linear discriminant fault trees. A natural question to ask is, “What are the impacts of multiple faults on the analysis?” This question should be applied, both to the linear separability question and the resulting algorithmic analysis.

To answer this question, we need to consider the approaches taken to performing fault diagnosis with multiple faults. Generally, two approaches are applied—one in which explicit multiple faults are included in the D -matrix (in which case, the multiple fault simply becomes another single conclusion in the model, and the complexity of deriving linear discriminant fault trees does not change), and one in which a set covering algorithm is applied to match the tested signature with a combined fault in the D -matrix. The former approach changes nothing in the analysis presented so far except to require a potentially exponential increase in the number of rows in the D -matrix. The more interesting case is the latter.

When considering the potential search space for multiple fault diagnosis with set covering, we note that three situations arise:

1. A specific multiple fault has a unique signature.
2. A specific multiple fault has a signature that is distinct from all other single fault signatures but is ambiguous with one or more other multiple faults.
3. A specific multiple fault is ambiguous with one or more other single faults (due to the single faults masking the manifestation of the multiple fault).

The first case can be handled using the linear discriminant fault tree by traversing the tree to a leaf, recognizing that the signature at the leaf does not match the fault signature, and then incrementally identifying faults that do not conflict with the fault signature. This can be done in polynomial time because the multiple fault is not ambiguous with any other conclusion (either single or multiple). As a result, the

complexity of the LDFT is not affected. Unfortunately, we have lost any guarantees of optimality with uniform LDFT since the number of faults making up the fault signature may vary across the set of possible multiple faults. In other words, the number of operations per path through the tree may not be balanced.

The second case can also be handled using the LDFT; however, once the leaf of the tree is reached, a general multiple-fault analysis must be performed. The problem is complicated by the fact that several multiple fault groups exist matching the signature. As proven in [24], the problem of finding a minimum-sized multiple-fault group to explain the fault signature is NP -complete because the algorithm is reducible from set covering. Therefore, even with a LDFT, the multiple fault diagnosis problem for this case is hard.

The final case can also be handled using the LDFT; however, the returned diagnosis will correspond to the single fault signature contained within the D -matrix. Some form of repair action or supplemental information will be required to distinguish the multiple fault from the single fault. Diagnosis to the ambiguity group is still optimal, but this optimality does not include the complexity associated with identifying the true fault or faults within the ambiguity group.

6.3. Addressing Non-linear Separability

To address the general problem of linear separability, consider a seemingly simple extension to the D -matrix where we do not require the diagnoses to have one and only one signature. Without loss of generality, consider a particular diagnosis d_i . If we permit two (or more) signatures to appear in the D -matrix, each labeled with diagnosis d_i , we have

$$d_i^1 = [\text{eval}^1(t_1), \dots, \text{eval}^1(t_{|T|})]$$

as well as

$$d_i^2 = [\text{eval}^2(t_1), \dots, \text{eval}^2(t_{|T|})]$$

This is equivalent to

$$d_i = [\text{eval}^1(t_1), \dots, \text{eval}^1(t_{|T|})] \\ \vee [\text{eval}^2(t_1), \dots, \text{eval}^2(t_{|T|})].$$

Thus we are now able to represent disjunctive concepts as well. Notice that by permitting “disjuncted” test signatures, any model we build will be able to represent a full disjunctive normal form (DNF). Since DNF can represent any propositional logic expression, this “simple” extension introduces considerable complexity into such models, including the ability to represent nonlinearly separable concepts (i.e., diagnoses).

Returning to the example of the stability augmentation system, note that this extension is exactly what is required to refine the diagnosis. In this case, multiple signatures would be added to the D -matrix, all with the same class label. This is contrary to the traditional D -matrix, but addition of these signatures (a) improves the “resolution” of diagnosis by modeling the relevant failure modes and (b) enables that improved “resolution” to cover potential interdependencies between the failure modes. In other words, the D -matrix is no longer limited to modeling linearly separable classes.

Can the four inference algorithms we discussed previously handle this increased complexity? Looking at each in turn, we see that they can or are able to be suitably modified to do so.

Rule-based Inference Combinations of rules can be combined and converted into clause form (or even Horn clause form) for diagnosis. Therefore, current rule-based systems and satisfiability solvers can adapt to cover the more complex rules [7].

Set Partitioning Decision trees inherently form a partitioning of the set of diagnoses as tests are performed. Incorporating additional signatures into the D -matrix offers no increased difficulty in constructing decision trees and can, in fact, provide useful information for simplifying the structure of the tree. Alternative splitting criteria and the use of oblique and nonlinear splits also increase the power of the overall approach [19]. Of course, both the original NP -completeness result on deriving optimal decision trees and the NP -completeness result in this paper apply [14].

Bayesian Inference The mapping of the D -matrix to a Bayesian model reduces to applying the naïve Bayes assumption to the dependencies. This is a natural fit for the D -matrix since naïve Bayesian inference is only able to solve linearly separable problems. Extending the Bayesian approach can be handled like the set partitioning approach; however, this is not efficient. An alternative approach is to augment the naïve Bayes network to capture resulting dependencies in the model [9, 27].

Case Based Reasoning CBR naturally handles nonlinearly separable concepts. The primary deficiency with the CBR approach relative to fault dictionaries or the D -matrix representation (even augmented with additional signatures) is the compactness assumption. Specifically, CBR requires the case base to approximate the underlying distribution of the data; otherwise, noisy or missing data will lead to misclassification [9].

As a final comment, we should point out that the approach described for expanding the D -matrix seems to solve our problem, but potentially that solution is at a great cost.

Specifically, we are suggesting that all we need to do is introduce another layer of specificity in the model where diagnosis d_i has more than one signature (corresponding to each of the failure modes of d_i). Unfortunately, it is possible that the more detailed failure mode may also have nonlinearities being rolled into the signature. This would suggest further refinement. In the limit, we would see a system level model being required to include all possible failure modes, thus eliminating any benefit attributed to the D -matrix for creating hierarchical models.

In each of the algorithms discussed above, except set partitioning, it is possible that models can be constructed based on available performance data to capture these nonlinearities in a more compact fashion. This, in fact, is the primary area of current and future research for the authors. See [4, 27, 28] for related work on this topic.

7. Conclusion

Throughout this paper, we considered the applicability of several diagnostic inference strategies to a common diagnostic model based on the D -matrix. We examined the theoretical properties of the D -matrix and proved that, under normal assumptions, the D -matrix inherently supports only linearly separable diagnostic signatures.

We are interested in identifying potential causes for test and diagnostic error in the maintenance process. Tools based on D -matrices are pervasive and provide the foundation for determining diagnostic strategies for maintenance manuals and test programs. Given our conclusions, if models have been constructed such that only a single conjunctive fault signature is provided for generating these strategies, even though associated inference techniques can support nonlinearly separable diagnoses, the resulting model effectively cripples these techniques by ignoring the nonlinear dependencies in the system. However, the D -matrix can be extended quite easily to model nonlinearly separable diagnoses. We noted that such nonlinearities arise whenever there are multiple correct signatures that lead to the same associated diagnosis. By permitting each of these signatures to be included in the model, the ability to find efficient diagnostic strategies is now much more difficult. Nevertheless, such diagnostic strategies can be constructed using existing heuristic techniques, and we are now able to overcome the representational limitations of the traditional D -matrix.

Acknowledgments

The authors would like to thank Greg Bowman and Mark Kaufman for their reviews and comments on several earlier versions of this paper. We would also like to thank

the US Navy for motivating this research and providing data for related work. Finally, we thank the anonymous reviewers and the JETTA Associate Editor, Dr. Magdy Abadir, for comments that helped tighten and improve the discussion in this paper.

Notes

1. Similar implications can be specified between tests. Due to a subsumption property defined within [31] that enables derivation of these relationships from the edges (d_i, t_j) , we will ignore those relationships in this paper.
2. In Section 5, we introduce the “linear discriminant” tree, which is a variant on the oblique tree.
3. Note that, since the averaging involves dividing the sum by $|D|$ and $|D|$ is constant for a particular problem, the cost function can be reduced to calculating the “external path length” by ignoring the averaging step. In this case, the cost function becomes $\sum_{d_i \in D} c(d_i)$, where $c(d_i)$ is the length of the path from the root to the leaf identifying d_i .
4. This result can be extended to cover the case where diagnoses are at interior nodes as well. Specifically, at each node, we consider each diagnosis as a potential split node and then permute the tests to obtain the optimal partition.
5. The D -matrix has long been proposed as an excellent candidate for hierarchical, system-level modeling. The point of this discussion is to identify a necessary condition for this proposal to hold.
6. PID control refers to a control system utilizing corrections “proportional” to the error with corrections dampened through a “derivative” term and stability maintained through an “integral” term. Thus, PID stands for “Proportional-Integral-Derivative” control.

References

1. A. Aamodt and E. Plaza, “Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches,” *AI Commun.*, vol. 7, no. 1, pp. 39–59, 1994.
2. Automatic Test Markup Language Consortium website, <http://www.atml.org>.
3. M. Azam, F. Tu, K. Pattipati, and R. Karanam, “A Dependency Model-Based Approach for Identifying and Evaluating Power Quality Problems,” *IEEE Trans. Power Deliv.*, vol. 19, no. 3, pp. 1154–1166, July 2004.
4. S. Butcher, J. Sheppard, M. Kaufman, H. Ha, and C. MacDougall, “Experiments in Bayesian Diagnostics with IUID-Enabled Data,” *IEEE AUTOTESTCON Conference Record*, New York: IEEE Press, September 2006, pp. 605–614.
5. T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, second edn., Cambridge, MA: MIT, 2001.
6. S. Deb, K. Pattipati, V. Raghavan, M. Shakeri, and R. Shrestha, “Multi-Signal Flow Graphs: A Novel Approach for System Testability Analysis and Fault Diagnosis,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 10, no. 5, pp. 14–25, May 1995.
7. R. Dechter, *Constraint Processing*, San Francisco: Morgan Kaufman, 2003.
8. *DoD Automatic Test System Framework Roadmap*, May 2005.
9. R. Duda, P. Hart, and D. Stork, *Pattern Classification*, New York: Wiley 2001.
10. W.C. Fischer and A. Sivahop, *Tested, Dissimilar Stability Augmentation System*, United States Patent #4,313,201, January 26, 1982.
11. N. Friedman, D. Geiger, and M. Goldszmidt, “Bayesian Network Classifiers,” *Mach. Learn.*, vol. 29, pp. 131–163, 1997.
12. M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, New York: Freeman, 1979.
13. D. Heath, *A Geometric Framework for Machine Learning*, PhD dissertation, Department of Computer Science, The Johns Hopkins University, 1992.
14. L. Hyafil and R. Rivest, “Constructing Optimal Binary Decision Trees in NP-Complete,” *Inf. Process. Lett.*, vol. 5, no. 1, pp. 15–17, May 1976.
15. IEEE Std 1232-2002. *Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE)*, Piscataway, NJ: IEEE Standards Association Press, 2002.
16. C. Ling and H. Zhang, “The Representational Power of Discrete Bayesian Networks,” *J. Mach. Learn. Res.*, vol. 3, pp. 709–721, 2002.
17. J. Luo, H. Tu, K. Pattipati, L. Qiao, and S. Chigusa, “Diagnosis Knowledge Representation and Inference,” *IEEE Instrum. Meas. Mag.*, vol. 9, no. 4, pp. 45–52, August 2006.
18. M. Minsky and S. Papert, *Perceptrons*, Expanded edn., Cambridge, MA: MIT, 1988.
19. S. Murthy, *On Growing Better Decision Trees from Data*, PhD Dissertation, Department of Computer Science, The Johns Hopkins University, 1996.
20. J. Quinlan, “Induction of Decision Trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.
21. I. Roychoudhury, G. Biswas, X. Koutsoukos, and S. Abdelwahed, “Designing Distributed Diagnosticians for Complex Systems,” *Proceedings of the 16th International Workshop on Principles of Diagnosis*, Monterey, CA, June 2005, pp. 31–36.
22. D. Rumelhart, G. Hinton, and R. Williams, “Learning Internal Representations by Error Propagation,” in D. Rumelhart and J. McClelland (eds.), *Parallel Distributed Processing*, Cambridge, MA: MIT, 1986, pp. 318–362.
23. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd edn., Upper Saddle River, NJ: Prentice Hall, 2003.
24. J. Sheppard and W. Simpson, “Multiple Fault Diagnosis,” *IEEE AUTOTESTCON Conference Record*, New York: IEEE Press, September 1994, pp. 381–389.
25. J. Sheppard and W. Simpson, “Improving the Accuracy of Diagnostics Provided by Fault Dictionaries,” *Proceedings of the 14th IEEE VLSI Test Symposium*, Los Alamitos, CA: IEEE Computer Society Press, 1996, pp. 180–185.
26. J. Sheppard and M. Kaufman, “A Bayesian Approach to Diagnosis and Prognosis Using Built-in Test,” *IEEE Trans. Instrum. Meas.*, Special Section on Built-In Test, vol. 54, no. 3, pp. 1003–1018, June 2005.
27. J. Sheppard and S. Butcher, “On the Linear Separability of Diagnostic Models,” *IEEE AUTOTESTCON Conference Record*, New York: IEEE Press, September 2006, pp. 626–635.
28. J. Sheppard, S. Butcher, M. Kaufman, and C. MacDougall, “Not-So-Naïve Bayesian Networks and Unique Identification in Developing Advanced Diagnostics,” *Proceedings of the IEEE Aerospace Conference*, New York: IEEE Press, March 2006.
29. M. Shwe and G. Cooper, “An Empirical Analysis of a Likelihood-weighting Simulation on a Large, Multiply-connected Medical Belief Network,” *Comput. Biomed. Res.*, vol. 24, pp. 453–475, 1991.
30. W. Simpson and J. Sheppard, “System Complexity and Integrated Diagnostics,” *IEEE Design and Test of Computers Magazine*, September 1991, pp. 16–30.
31. W. Simpson and J. Sheppard, *System Test and Diagnosis*, Norwell, MA: Kluwer, 1994.
32. C. Stanfill and D. Waltz, “Toward Memory-based Reasoning,” *Commun. ACM*, vol. 29, no. 12, pp. 1213–1228, 1986.
33. W.C. Thibault and B.F. Naylor, “Set Operations on Polyhedra Using Binary Space Partitioning Trees,” *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, New York: ACM, pp. 153–167, 1987.

John W. Sheppard is an Assistant Research Professor in the Department of Computer Science, The Johns Hopkins University. Dr. Sheppard started his career as a research computer scientist for ARINC and attained the rank of Fellow. His research interests include algorithms for diagnostic and prognostic reasoning, machine learning and data mining in temporal systems, and reinforcement learning. Dr. Sheppard holds a BS in computer science from Southern Methodist University and an MS and PhD in computer science from Johns Hopkins University. He has over 100 publications in artificial intelligence and diagnostics, including an authored book and an edited book. He is a fellow of the IEEE and currently serves as Vice Chair of the IEEE Standards Coordinating Committee 20 (SCC20) on Test and Diagnosis for Electronic Systems, Secretary and Past Chair of the Diagnostic and Maintenance Control subcommittee of SCC20, member at large of the IEEE Computer

Society Standards Activities Board (CS SAB), and Official Liaison of the CS SAB to SCC20.

Stephyn G. W. Butcher received his MS in computer science at the Whiting School of Engineering, The Johns Hopkins University and is currently pursuing his PhD in computer science at Johns Hopkins. He has served as a Lecturer in economics and grader in computer science. He received his BA in economics from the California State University, Sacramento and his MA in economics from The American University, Washington, DC. His research interests are mainly in machine learning and include Bayesian networks and evolutionary computation with emphases on problems in classification and differential games.