Extracting Decision Trees from Diagnostic Bayesian Networks to Guide Test Selection

Scott Wahl [1], John W. Sheppard [1]

[1] *Department of Computer Science, Montana State University, Bozeman, MT, 59717, USA*
wahl@cs.montana.edu
john.sheppard@cs.montana.edu

ABSTRACT

In this paper, we present a comparison of five different approaches to extracting decision trees from diagnostic Bayesian nets, including an approach based on the dependency structure of the network itself. With this approach, attributes used in branching the decision tree are selected by a weighted information gain metric computed based upon an associated D-matrix. Using these trees, tests are recommended for setting evidence within the diagnostic Bayesian nets for use in a PHM application. We hypothesized that this approach would yield effective decision trees and test selection and greatly reduce the amount of evidence required for obtaining accurate classification with the associated Bayesian networks. The approach is compared against three alternatives to creating decision trees from probabilistic networks such as ID3 using a dataset forward sampled from the network, KL-divergence, and maximum expected utility. In addition, the effects of using χ^2 statistics and probability measures for pre-pruning are examined. The results of our comparison indicate that our approach provides compact decision trees that lead to high accuracy classification with the Bayesian networks when compared to trees of similar size generated by the other methods, thus supporting our hypothesis.

1. INTRODUCTION

Proper support of a system is wisely regarded as vital to its function. In general, support of a system involves both corrective and preventative maintenance. The primary goal of corrective maintenance is to repair faults while preventative maintenance attempts to avoid faults or improve the useful lifetime of parts. Satisfying these goals requires isolating what faults have occurred or are most likely to occur.

Decision trees have been used extensively in performing fault diagnosis during corrective maintenance. This procedure is a natural extension of the general process used in troubleshooting systems. Given no prior knowledge, tests are performed sequentially, continuously narrowing down the ambiguity group of likely faults. Resulting decision trees are called "fault trees" in the system maintenance literature.

In recent years, tools have emerged that apply an alternative approach to fault diagnosis using diagnostic Bayesian networks. One early example of such a network can be found in the creation of the the QMR knowledge base (Shwe et al., 1991), used in medical diagnosis. Bayesian networks provide a means for incorporating uncertainty into the diagnostic process; however, Bayesian networks by themselves provide no guidance on which tests to perform when. Rather, test information is applied as evidence whenever it becomes available. In this paper, we compare approaches to using Bayesian networks to derive decision trees to integrate the advantage of each approach.[1] Our approach weights the classes based on prior probability distributions and then derives a weighted decision tree using the associated D-matrix characterizing the structure of the Bayesian network. We hypothesize that this method will yield compact decision trees (thus reducing the amount of evidence required to be evaluated) that result in high classification accuracy relative to alternative methods that we evaluated.

Historically, creating decision trees is usually based around a static set of data (Casey & Nagy, 1984; Heckerman, Geiger, & Chickering, 1995; Martin, 1997; Murthy, 1997; Quinlan, 1986). It has been noted by such previous work that information theory and other methods are based upon the assumption that the data set accurately represents the true underlying distribution of the data. To completely do so requires an infinitely large data set. However, by using diagnostic Bayesian networks directly, it is possible to use the distributions and information provided by the network to create a decision tree directly. Without adequate stopping criteria, however, the resulting trees are likely to be extraordinarily

[1] While this paper focuses on deriving static decision trees, the algorithms used can be adapted easily to an online setting where tests are recommended dynamically.

large. Another potential method for creating decision trees from probabilistic networks is based upon using the structure of the network. The work here uses a specialized adjacency matrix used for diagnostic networks, called a D-matrix, to accomplish this task.

In describing the results of our study, this paper has been organized as follows. Section 2 provides motivation behind the current study. Section 3 describes related work with decision trees and Bayesian networks. Section 4 provides background information on probabilistic models and decision trees. Section 5 specifies the approach used in creating the decision trees. Sections 6 and 7 give the results and discussion, respectively. Finally, we conclude in section 8.

2. MOTIVATION

Recent trends in developing diagnostic tools have focused on providing online methods for selecting tests to evaluate and on incorporating uncertainty into the reasoning process. Legacy diagnostic systems rely on static fault trees to guide the test and diagnosis process. The primary issue with using static trees is that they are not able to adapt to changing conditions such as the loss of a test resource. Even so, evaluating the performance of an adaptive system is difficult in that the reasons for selecting certain sources of evidence may be unclear.

Our research is focused on developing Bayesian diagnostic systems that can be assessed through sound empirical analysis. In an attempt to reduce the amount of evidence to be evaluated and maintain control, we found it beneficial to use static decision trees in the evaluation process. Such trees can be used to control the amount of evidence evaluated, can be examined to justify the test choices, and can be applied consistently across all methods studied. Unfortunately, as we started looking for existing approaches to generate decision trees to be used with Bayesian networks, we found very little direct, comparative work had been done in this area. Therefore, we sought to evaluate alternative approaches with the goal of finding one that had minimal computational burden (both in generating the tree and in using the tree) and still yielded accurate results once evidence was evaluated.

In order to do so, three existing methods were selected based upon information theory and value of information: forward sampling and ID3, maximum expected utility, and maximum KL-divergence. Due to the structure of the network, a structure guided search which was weighted by the probability of the classes was also implemented. In order to provide a baseline for this method, the DM-based approach was decided upon followed by a simplification of the probability weighted D-matrix approach, creating the marginal weighted D-matrix method.

3. RELATED WORK

Considerable work has been performed in various methods for inducing decision trees from data as well as creating probabilistic models from decision trees. As described by Murthy (Murthy, 1997), many different heuristic techniques have been applied to inducing decision trees. Most of the top-down methods for generating decision trees fall into three categories: information theory, distance measure, and dependence measures. Some of these measures are used directly in the generation of the decision trees in this paper. Specifically, information theory is used in generating the decision tree from a D-Matrix and by forward sampling the diagnostic network.

Much of the prior work in creating decision trees and test selection from Bayesian networks focuses on reducing the expected cost of performing a series of actions, both tests and repairs (Heckerman, Breese, & Rommelse, 1995; Mussi, 2004; Skaanning, Jensen, & Kjærulff, 2000; Jensen et al., 2001; Vomlel, 2003). In the SACSO system, tests and actions are recommended based upon the expected cost of taking any action or performing of tests using the probability the problem will be corrected and a cost function. Given the nature of the problem, the SACSO system relies on a set of networks where each reported problem, e.g. spots on a page, is the root of its own tree. A similar approach was used by Mussi for the generation of sequential decision support systems. These approaches are similar to what is defined by Koller for measures of maximum expected utility (MEU) (Koller & Friedman, 2009). Given that performing the inference for such measures is intractable for large problems, Zheng, Rish, and Beygelzimer developed an algorithm for calculating approximate entropy (Zheng, Rish, & Beygelzimer, 2005).

Use of probabilities in creating decision trees has been analyzed before by Casey and Nagy for use in optical character recognition (Casey & Nagy, 1984). Based upon a data set, the probability of a single pixel having a specific color was determined for each possible letter. This information was then used in creating the decision tree.

Additional work has been performed in optimal creation of AND/OR decision trees. Research by Pattipati and Alexandridis has shown that using a heuristic search with AND/OR graphs and the Gini approach can be used to create optimal and near-optimal test sequences for problems (Pattipati & Alexandridis, 1990).

There has also been research performed in extracting rules and decision trees from neural networks in an attempt to provide context to learned neural networks (Craven. & Shavlik, 1996). Additionally, combining the two approaches, decision trees and Bayesian networks, has been performed by Zhou, Zheng, and Chen (Zhou, Zhang, & Chen, 2006). Their research associated a prior distribution to the leaves of the decision tree and utilized a Monte-Carlo method to perform inference. Jordan also used statistical approaches in creating a probabilistic decision tree (Jordan, 1994). Parameters for the system were estimated by an expectation-maximization algorithm. In addition to the work above, Liang, Zhang, and Yan analyzed various decision tree induction strategies when using conditional log likelihood in order to estimate the probability of classes in the leaf nodes of a tree (Liang, Zhang, & Yan, 2006).

Although their application did not specifically address test selection, Frey et. al. used a forward sampling method for generating decision trees from Bayesian networks for use in identifying the Markov blanket of a variable within a data set (Frey, Tsamardinos, Aliferis, & Statnikov, 2003).

Przytula and Milford (Przytula & Milford, 2005) created a system for converting a specific kind of decision tree, a fault tree, into a Bayesian network. Their implementation created an initial network of the same structure as the original fault tree and inserted additional observation nodes. The conditional probability distributions of the system are calculated by a given prior or by a specified fault rate.

Especially relevant for potential future work, Kim and Valtorta performed work in the automatic creation of approximate bipartite diagnostic Bayesian networks from supplied Bayesian networks (Kim & Valtorta, 1995).

4. BACKGROUND

The work in this paper primarily depends on principles from top-down induction of decision trees and diagnostic networks, a specialized form of a Bayesian network. To fully understand the metrics used in developing the decision trees, some information on both is required.

4.1 Bayesian Networks

Bayesian networks are a specific implementation of a probabilistic graphical model (Heckerman, Geiger, & Chickering, 1995; Koller & Friedman, 2009; Pearl, 1988). It takes the form of a directed, acyclic graph used to represent a joint probability distribution. Consider a set of variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ with a joint probability $\mathbf{P}(\mathbf{X}) = \mathbf{P}(X_1, \ldots, X_n)$.

Definition 1 *Given a joint probability distribution over a set of variables* $\{X_1, \ldots, X_n\}$*, the product rule provides a factoring of the joint probability distribution by the following*

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

The main issue with this representation is that each "factor" of the network is represented by a table whose size is exponential in the number of variables. Bayesian networks exploit the the principle of conditional independence to reduce the complexity.

Definition 2 *A variable* X_i *is conditionally independent of variable* X_j *given* X_k *if*

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

Given these definitions, a more compact representation of the joint probability can be calculated by the set of conditional independence relations. Bayesian networks encapsulate this representation by creating a node in the graph for each variable in $\mathbf{X}$. For all nodes X_i and X_j within the network, X_j is referred to as a parent of X_i if there is an edge from X_j to X_i. In order to complete the representation of the joint probability distribution, each node has an associated conditional probability distribution denoted by $\mathbf{P}(X_i \mid Parents(X_i))$. Thus a Bayesian network represents the joint distribution as

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

As an example, consider a joint probability destribution given by $\mathbf{P}(A, B, C, D)$ and suppose the conditional independence relations allow it to be factored to

$$\mathbf{P}(\mathbf{X}) = \mathbf{P}(A)\, \mathbf{P}(B)\, \mathbf{P}(C \mid A, B)\, \mathbf{P}(D \mid C).$$

Assuming binary variables, the full joint probability in table form would require $2^4 = 16$ entries whereas the factored form only requires $2^0 + 2^0 + 2^2 + 2^1 = 8$, halving the size of the representation. The Bayesian network resulting from this factorization is shown in Figure 1.

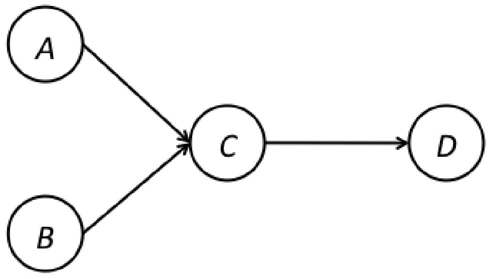

Figure 1: Example Bayesian Network

4.2 Diagnostic Networks

A diagnostic network is a specialized form of a Bayesian network which is used as a classifier for performing fault diagnosis (Simpson & Sheppard, 1994). This network consists of two types of nodes: class (i.e., diagnosis) and attribute (i.e., test) nodes. When performing diagnosis, each test performed is an indicator for a possible set of faults. As a simple example, consider a simple test of the state of a light bulb with the results on or off. With no other information, this is an indicator for a number of potential problems such as a broken filament or damaged wiring. By this principle, every test node in the network has a set of diagnosis nodes as parents. As in the standard Bayesian network, every node has an associated conditional probability distribution. For a specific diagnosis node, this distribution represents the probability of failure. For test nodes, this distribution represents the probability of a test outcome given the parent (i.e. causing) failures.

Constructing the network in this manner results in a bipartite Bayesian network. Because of this feature, it is possible to represent the structure of the network with a specialized adjacency matrix referred to as a D-Matrix. Consider a diagnostic network with the set of diagnoses $\mathbf{D} = \{d_1, \ldots, d_n\}$ and the set of tests $\mathbf{T} = \{t_1, \ldots, t_m\}$. In the simplest interpretation, every row of the D-Matrix corresponds to a diagnosis from $\mathbf{D}$ while each column corresponds to an test from $\mathbf{T}$. This leads to the following definition.

Definition 3 *A D-Matrix is an* $n \times m$ *matrix* $\mathbf{M}$ *such that for every entry* $m_{i,j}$*, a value of* 1 *indicates that* d_i *is a parent of* t_j *while a value of* 0 *indicates that* d_i *is not a parent of* t_j*.*

One important concept from this is that of the equivalence class. Given any two diagnoses, d_i and d_j, the two belong to the same equivalence class if the two rows in the D-Matrix corresponding to the classes are identical. In diagnostic terminology, such an equivalence class corresponds to an ambiguity group in that no tests exist capable of differentiating the classes. (Note that this is actually an over-simpification since structure in a Bayesian network alone is not sufficient to identify the equivalence classes. For our purposes, however, we will generate networks that ensure this property does hold.)

Figure 2 provides an example diagnostic network with four classes labeled d_1, d_2, d_3, and d_4 as well as four test attributes labeled t_1, t_2, t_3, and t_4. The corresponding D-matrix for this network is shown in Table 1.

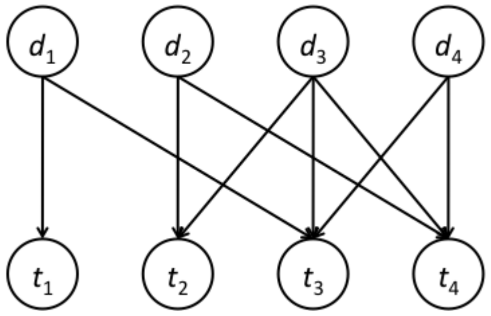

Figure 2: Example Diagnostic Network

	t_1	t_2	t_3	t_4
d_1	1	0	1	0
d_2	0	1	0	1
d_3	0	1	1	1
d_4	0	0	1	1

Table 1: D-Matrix of the example network

4.3 Decision Trees

Although there are multiple methods for creating decision trees, the method of interest for this paper is top-down induction of decision trees such as Quinlan's ID3 algorithm (Murthy, 1997; Quinlan, 1986). This type of decision tree is very popular for performing classification. Under ID3, the classification problem is based on a universe of classes that have a set of attributes.

The induction task is to utilize the attributes to partition the data in the universe. ID3 performs this partitioning and classification by iteratively selecting an attribute (i.e., evidence variable) whose value assignments impose a partitioning on the subset of data associated with that point in the tree. Every internal node in the tree represents a comparison or test of a specific attribute. Every leaf node in the tree corresponds to a classification. Thus, a path from the root of the tree to a leaf node provides a sequence of tests to perform to arrive at a classification. Fault trees used in diagnosis follow this principle. When creating the tree, the choice of attribute to use in partitioning the data is determined in a greedy fashion by some measure such as information gain.

For a more formal definition of the ID3 algorithm, consider a labeled training set of examples $\mathcal{D}$. Each individual in the training set is given as the set of evidence $\mathbf{E} = \{e_1, \ldots, e_m\}$ and a single class from the set of classes $\mathbf{C} = \{c_1, \ldots, c_n\}$. For this discussion, it will be assumed that the classification problem is binary (i.e., examples are labeled either as positive or negative), though this process is easily extended to multiple classes. Based on information theory, the information contained in the set of examples is given as

$$I(p,\, n) = -\frac{p}{p+n}\lg\frac{p}{p+n} - \frac{n}{p+n}\lg\frac{n}{p+n}$$

where p is the number of positive classes and n is the number of negative classes in the partition. At the root of the tree, this value is calculated for the entire data set. To determine the attribute to use to partition the data, the information gain is calculated for each possible attribute. To do so, the expected entropy of the partitions made by performing the test is calculated as

$$E(e) = \sum_{i=1}^{arity(e)} \frac{p_i + n_i}{p+n} I(p_i,\, n_i)$$

where p_i and n_i are the number of positive and negative classes in the partitions created by branching on the ith value of e. From this, the information gained by this test is given as

$$gain(e) = I(p,\, n) - E(e)\,.$$

As this is a greedy algorithm, the choice of attribute to branch on is the attribute which provides the greatest information gain. Once this attribute is selected, each individual in the data set that matches a specific value of that attribute is placed into a separate child of the root. This process is then repeated for the created partitions until a stopping criterion is met. Frequently, this is based on a minimum threshold of information gain, when a partition contains only one class, or when all attributes have been used in a path.

5. APPROACH

In this work, five separate heuristics for generating decision trees from diagnostic Bayesian networks were examined: information gain on a sampled data set, KL-divergence, maximum expected utility, information gain on the D-matrix, and a weighted information gain on the D-matrix. These five methods are either based on standard practice for selecting evidence nodes dynamically (e.g., maximum expected utility, or MEU) or are based on historical approaches to inducing decision trees (e.g., information gain). One exception is the approach using the D-matrix, which is a novel approach developed for this study with the intent of reducing overall computational complexity.

For every method, the class attributed to the leaf nodes of the decision tree is determined by the most likely fault indicated by the Bayesian network given the evidence indicated in the path of the tree. A direct consequence of this is that large trees effectively memorize the joint probability distribution given by the Bayesian network. Finally, a pre-pruning process was applied to simplify the trees. Pre-pruning was selected over post-pruning due to the fact that the maximum expected utility and KL-divergence methods create full trees, which are impractical in most situations. The pruning process aids in reducing this issue of memorization. Pre-pruning the trees created by this method relies on the multi-class modification of Quinlan's equations provided later in this section.

5.1 Induction by Forward Sampling

Performing induction by forward sampling is directly analogous to the method used in ID3. With forward sampling induction, a database of training instances $\mathcal{D}$ is created from the Bayesian network. Since this approach uses the ID3 algorithm, the database must contain individuals with a set of tests $\mathbf{T}$ which are associated with a single diagnosis from $\mathbf{D}$. This can be accomplished by assuming a single fault in the diagnostic Bayesian network and sampling the attributes based on the associated conditional probability distributions. To maintain the approximate distribution of classes, diagnoses are chosen based upon the marginal probabilities of the class nodes, assuming these correspond to failure probabilities derived from failure rate information.

Forward sampling is straightforward in a diagnostic Bayesian network. Once a single fault has been selected (according to the associated failure distributions), that diagnosis node is set to TRUE while all other diagnosis nodes are set to FALSE. Since the network is bipartite, all attributes can now be sampled based upon the conditional probabilities given by the Bayesian network. Each such sample is added to the database $\mathcal{D}$ to generate the training set.

The data sets used here are not based upon binary classification; therefore, we needed to modify the basic ID3 algorithm to handle the multi-class problem. Given a set of classes $\mathbf{C} = \{c_1, \ldots, c_n\}$, the information gain over a set of samples can be determined using the equations

$$I(c_1, \ldots, c_n) = -\sum_{i=1}^{n} \frac{c_i}{c_1 + \cdots + c_n} \lg \frac{c_i}{c_1 + \cdots + c_n}$$

and

$$E(e) = \sum_{i=1}^{arity(e)} \frac{c_{i,1} + \cdots + c_{i,n}}{c_1 + \cdots + c_n} I(c_{i,1}, \ldots, c_{i,n})$$

where e is the evidence variable being evaluated.

5.2 Induction by KL-Divergence

The underlying principle of induction by KL-divergence is to select attributes that greedily maximize the KL-divergence of the resulting marginalized networks after applying evidence. Calculating the KL-divergence of a child node from its parent node requires performing inference over the network based upon the evidence previously given in the tree to determine

$$KL(P||Q) = \sum_{c \in C} P(c \mid \mathbf{e}_k) \lg \frac{P(c \mid \mathbf{e}_k)}{Q(c \mid \mathbf{e}_k)}$$

where $P(c \mid \mathbf{e}_k)$ is the conditional probability of a class given evidence on the path to the parent node and $Q(c \mid \mathbf{e}_k)$ is the conditional probability of a class given evidence on the path to the child node. The task then becomes to select the attribute that maximizes the average KL-divergence of the resulting distributions, given by

$$KL = \frac{KL(P||Q_T) + KL(P||Q_F)}{2}$$

where Q_T is the conditional distribution where $e_{k+1} =$ TRUE and Q_F is the conditional distribution where $e_{k+1} =$ FALSE.

5.3 Induction by MEU

The heuristic of maximum expected utility closely follows the approach used by Casey and Nagy (Casey & Nagy, 1984). In their work, the equation for information was modified to

$$I(e) = -\sum_{c_i \in \mathbf{C}} P(c_i \mid e) \lg P(c_i \mid e).$$

Modifying this to realize maximum expected utility, the entropy of a system is calculated by multiplying the information of each partition by the conditional probability $P(e_{k+1} = \text{FALSE} \mid \mathbf{e}_k)$ and $P(e_{k+1} = \text{TRUE} \mid \mathbf{e}_k)$ where e_{k+1} is the evidence variable being evaluated, and $\mathbf{e}_k$ is the set of evidence collected so far along the path in the tree to the current node. In addition, a utility is associated with each attribute and class which are used in the entropy and information calculations. This results in

$$I(e) = -\sum_{c_i \in \mathbf{C}} U(c_i) P(c_i \mid e) \lg P(c_i \mid e)$$

and

$$\begin{aligned} E(e) &= cost(e) [P(e = \text{FALSE} \mid \mathbf{e}) I(\mathbf{e} \cup \{e\}) \\ &\quad + P(e = \text{TRUE} \mid \mathbf{e}) I(\mathbf{e} \cup \{e\})] \end{aligned}$$

where $U(\cdot)$ is the utility of a class and is assumed to be inversely proportional to its cost. For the tests performed here, it is also assumed that $U(\cdot)$ and $cost(\cdot)$ are uniform for all tests and classes; however, given a more realistic model, available utilities and costs can be used directly.

5.4 Induction by D-Matrix

Creating decision trees based upon the D-matrix is very similar to the approach taken by ID3. However, instead of using a set of classified examples, the rows of the D-matrix are treated as the dataset. Thus, there is only a single instance of a class within the data set. Consider again the example D-matrix in Table 1. By using every row as an element in the data set, attributes are selected that maximize the information gain using the same equation in forward sampling.

Since there is only one element of a class in each data set, the equation for information used in ID3 can be simplified to

$$\begin{aligned} I &= -\sum_{i=1}^{n} \frac{c_i}{c_1 + \cdots + c_n} \lg \frac{c_i}{c_1 + \cdots + c_n} \\ &= -\sum_{i=1}^{n} \frac{1}{n} \lg \frac{1}{n} \\ &= -\lg \frac{1}{n} \end{aligned}$$

Entropy is similarly modified to

$$E = \sum_{i=1}^{arity(e)} \frac{m_i}{n} I(\mathbf{c}_i)$$

where m_i is the size of partition i, n is the size of the original partition, and $\mathbf{c}_i$ is the set of classes placed into partition i. The result of this is that the algorithm attempts to select attributes that will split the data set in half, creating shallow trees with a single class in each leaf node. Because of this, χ^2 pre-pruning is unnecessary.

5.5 Induction by Weighted D-Matrix

The weighted D-matrix approach attempts to overcome the obvious problems of the regular D-matrix approach.

That is, the simple approach fails to consider the probability a class will occur in the resulting partitions. This method attempts to improve on this slightly by estimating the probability-weighted information of each partition by the probability $P\left(c_i \mid \mathbf{e}\right)$ using the equation

$$I\left(c_1, \ldots, c_n\right) = -\sum_{i=1}^{n} P\left(c_i \mid \mathbf{e}\right) \lg P\left(c_i \mid \mathbf{e}\right).$$

We looked at two different approaches to determining $P(c_i|\mathbf{e})$. In one case, we used the SMILE inference engine to infer the probability based on evidence applied so far. In the second, as evidence was selected, we partitioned the class space as we do with the simple D-matrix approach and renormalized the marginals over the smaller space. We refer to these two approaches as probability-weighted D-matrix and marginal-weighted D-matrix respectively.

5.6 Pre-Pruning the Trees

For the first three methods below, a pre-pruning procedure is used based upon a χ^2 test (Quinlan, 1986). Since there is no natural stopping criteria for the KL-divergence and MEU based induction rules, some form of prepruning is necessary to prevent creating a fully populated decision tree. Such a tree would be exponential in size with regard to the number of tests, and is thus infeasible. To ensure comparisons between the methods are fair, prepruning is used for the forward sampling induction rule as well. Furthermore, chi^2 was selected due to its basis for Quinlan's original original algorithms. Under Quinlan's adaptation of the χ^2 statistic from (Hogg & Craig, 1978), given an attribute under consideration e, a set of positive and negative examples p and n, and the partitions $p_1, \ldots, p_k$ and $n_1, \ldots, n_k$ with $k = arity\left(e\right)$, the χ^2 statistic is calculated by

$$\chi^2 = \sum_{j=1}^{arity(e)} \frac{\left(p_j - p'_j\right)}{p'_j} + \frac{\left(n_j - n'_j\right)}{n'_j}$$

where

$$p'_j = p\frac{p_j + n_j}{p + n}.$$

The value for n'_i is calculated in a similar manner. Extending this for multiple classes results in the equations

$$\chi^2 = \sum_{j=1}^{arity(e)} \sum_{i=1}^{n} \frac{\left(c_{i,j} - c'_{i,j}\right)^2}{c'_{i,j}}$$

and

$$c'_{i,j} = c_i \frac{\sum_{i=1}^{n} c_{i,j}}{\sum_{i=1}^{n} c_i}.$$

This χ^2 statistic can be used to test the hypothesis that the distribution within the partitions is equivalent to the original set of examples. Unfortunately, this equation requires modification when dealing with the parameters of the networks directly. The change occurs in the calculation of c_i and $c_{i,j}$ where $c_i = \mathbf{P}\left(c_i \mid \mathbf{e}_k\right)$, $c_{i,j} = \mathbf{P}\left(c_i \mid \mathbf{e}_k, e_{k+1}\right)$, $\mathbf{e}_k$ is the set of evidence leading to the root node, and e_{k+1} is the evidence gathered by applying the next attribute.

Another issue with this approach is that the χ^2 statistic is dependent upon the size of the example set. Partitions of large sets are likely to be deemed significant by this test. However, by using probabilities, the values which correspond to the size of the set are only in the range of $[0, 1]$. Therefore, tests are performed such that the χ^2 statistic is tested against a threshold parameter τ. By supplying a threshold τ, the new branch is created only if $\chi^2 > \tau$. In order to perform the same experiment with the sampling based approach, the χ^2 value is normalized by the size of the data set in the partition. Originally, the methods were tested based on the classical use of χ^2 under the approximate 5 percent significance test with $k = 10 - 1 = 9$ degrees of freedom from the classes. However, the resulting decision trees were significantly larger than the D-matrix based methods, motivating the threshold method.

Another potential method for performing pre-pruning is based on the probabilities themselves. Specifically, nodes in the tree are not expanded if the probability of reaching the node is less than a specific threshold. This procedure lends itself easily to the KL-divergence and maximum expected utility methods since they already perform inference over the network. Calculating the probability of reaching a node is straightforward. At the root node, prior to any branching, the probability of reaching the node is set to 1.0. Once an attribute e_{k+1} has been selected for branching, inference is performed to determine

$$\mathbf{P}\left(e_{k+1} = \textsc{True} \mid \mathbf{e}_k\right)$$

and

$$\mathbf{P}\left(e_{k+1} = \textsc{False} \mid \mathbf{e}_k\right)$$

where $\mathbf{e}_k$ is the set of evidence gathered in the path to the parent node. For the children nodes, this value is multiplied by the probability of the parent node which is given by $\mathbf{P}\left(\mathbf{e}_k\right)$. For the child created for the $e_{k+1} = \textsc{True}$ partition, this yields the equation

$$\mathbf{P}\left(e_{k+1} = \textsc{True} \mid \mathbf{e}_k\right) \mathbf{P}\left(\mathbf{e}_k\right)$$

by the rules of probability, giving the probability of reaching the child node. This provides a simple method for iteratively determining the probability as nodes are created. In the experiments, this method was applied to both KL-divergence and maximum expected utility.

These procedures are not necessary for the last two methods since the limitations on branching imposed by the structure provide pre-pruning. Early results using χ^2 pre-pruning in addition to the structure requirements failed to provide any benefit for those two methods.

Simple post-pruning was also applied to the decision trees in order to reduce their size. Since it is possible for these trees to branch by an attribute where the resulting leaves indicate the same most-likely fault as the parent, these leaf nodes in the tree are unnecessary for the purposes of matching the most-likely fault. Therefore, once the trees have been created, any subtree where all nodes within that subtree indicate the same most-likely fault is pruned to just include the root of that subtree. This procedure does not modify the accuracy of any tree, only reducing its size.

	BN	FS	KL	MEU	PWDM	MWDM	DM	FS (pruned)	KL (pruned)	MEU (pruned)
BN01_01	0.991	0.987 (45)	0.989 (271)	0.991 (512)	0.984 (22)	0.984 (43)	0.983 (10)	0.983 (23)	**0.988 (137)**	0.987 (151)
BN02_01	0.999	0.999 (63)	0.999 (535)	0.998 (544)	0.984 (17)	0.984 (35)	0.982 (10)	0.988 (21)	0.996 (219)	**0.996 (203)**
BN03_01	0.999	0.999 (53)	0.997 (420)	0.999 (575)	0.988 (18)	0.991 (36)	0.986 (10)	0.991 (19)	**0.995 (197)**	0.995 (238)
BN04_01	1.000	0.998 (56)	0.998 (467)	1.000 (585)	0.988 (25)	0.987 (38)	0.983 (10)	0.995 (25)	**0.997 (213)**	0.996 (290)
BN05_01	0.935	0.935 (61)	0.935 (385)	0.935 (499)	0.924 (20)	0.927 (41)	0.919 (9)	0.927 (21)	**0.930 (114)**	0.930 (164)
BN06_10	0.924	0.924 (194)	0.910 (300)	0.924 (458)	**0.854 (20)**	0.826 (47)	0.795 (10)	0.800 (20)	0.681 (21)	0.743 (21)
BN07_10	0.969	0.969 (113)	0.959 (368)	0.969 (452)	0.930 (25)	0.938 (40)	0.867 (10)	**0.945 (25)**	0.872 (25)	0.596 (33)
BN08_10	0.838	0.838 (83)	0.834 (298)	0.838 (377)	0.785 (31)	0.669 (41)	0.639 (8)	**0.824 (34)**	0.771 (49)	0.731 (32)
BN09_10	0.957	0.959 (146)	0.954 (349)	0.957 (533)	0.868 (19)	0.892 (43)	0.827 (10)	**0.918 (23)**	0.877 (20)	0.900 (19)
BN10_10	0.986	0.986 (106)	0.972 (491)	0.986 (548)	0.937 (25)	0.954 (42)	0.908 (10)	**0.955 (25)**	0.660 (40)	0.658 (36)
BN11_20	0.939	0.946 (145)	0.907 (196)	0.939 (498)	0.832 (27)	0.844 (35)	0.761 (10)	**0.892 (27)**	0.778 (29)	0.822 (28)
BN12_20	0.897	0.902 (307)	0.894 (411)	0.897 (458)	0.769 (30)	0.767 (40)	0.690 (10)	**0.802 (30)**	0.776 (35)	0.653 (39)
BN13_20	0.895	0.895 (204)	0.889 (374)	0.895 (432)	0.769 (29)	0.751 (36)	0.721 (10)	**0.836 (29)**	0.767 (39)	0.653 (29)
BN14_20	0.860	0.866 (169)	0.855 (375)	0.860 (451)	0.779 (26)	0.769 (41)	0.683 (10)	**0.782 (26)**	0.719 (27)	0.715 (39)
BN15_20	0.875	0.876 (104)	0.868 (177)	0.875 (240)	0.818 (27)	0.760 (48)	0.666 (10)	**0.857 (28)**	0.827 (27)	0.749 (27)
BN16_30	0.787	0.787 (308)	0.778 (306)	0.787 (319)	0.665 (37)	0.669 (45)	0.615 (8)	0.676 (39)	0.518 (43)	**0.710 (42)**
BN17_30	0.887	0.891 (257)	0.877 (416)	0.887 (542)	0.731 (33)	0.743 (44)	0.654 (9)	**0.812 (55)**	0.746 (33)	0.748 (33)
BN18_30	0.813	0.814 (198)	0.788 (267)	0.813 (339)	0.692 (27)	0.653 (41)	0.590 (9)	**0.775 (32)**	0.639 (32)	0.674 (28)
BN19_30	0.814	0.822 (253)	0.807 (404)	0.814 (491)	0.653 (37)	0.608 (51)	0.563 (10)	**0.786 (37)**	0.677 (42)	0.648 (47)
BN20_30	0.850	0.849 (318)	0.840 (340)	0.850 (414)	0.660 (35)	0.540 (44)	0.512 (10)	**0.800 (40)**	0.699 (39)	0.708 (42)
BN21_40	0.646	0.647 (225)	0.632 (404)	0.647 (320)	0.459 (43)	0.466 (52)	0.409 (9)	**0.623 (65)**	0.564 (108)	0.587 (46)
BN22_40	0.742	0.742 (253)	0.740 (222)	0.742 (250)	0.550 (44)	**0.663 (42)**	0.613 (8)	0.651 (46)	0.645 (48)	0.647 (48)
BN23_40	0.616	0.617 (320)	0.614 (374)	0.616 (415)	0.525 (29)	0.500 (43)	0.481 (9)	**0.548 (30)**	0.482 (36)	0.491 (32)
BN24_40	0.739	0.745 (161)	0.694 (297)	0.739 (399)	0.607 (35)	0.571 (43)	0.565 (10)	**0.683 (36)**	0.605 (35)	0.632 (37)
BN25_40	0.651	0.655 (94)	0.611 (145)	0.651 (210)	0.576 (31)	0.584 (33)	0.512 (10)	**0.603 (34)**	0.553 (37)	0.569 (31)

Table 2: Accuracies for each individual network with χ^2 pruning with the values in paranthesis representing the number of leaf nodes, or classification paths, in the resulting tree

5.7 Data Sets

To test these five methods for deriving the decision tree from the Bayesian network, simulated diagnostic Bayesian networks were created with varying degrees of determinism. In total, twenty-five networks were generated each with ten diagnoses and ten tests. Constructing the topologies for these networks involved multiple steps. First, for every diagnosis node in the network, a test (i.e., evidence node) was selected at random and added as a child node to that diagnosis node. Afterwards, any test that did not have a parent diagnosis node was given one by selecting a diagnosis node at random. Finally, for every pair of diagnosis and test nodes d_i and t_j that were currently not connected, an edge $d_i \to t_j$ was added to the network with probability $P \in [0.2, 0.4]$.

Once the 25 network topologies were generated, five different types of probability distributions were generated, each with an associated level of determinism in the tests. In other words, given a diagnosis d_i that we assume to be true, the probability of any test attribute $t_j = \textsc{True}$ with d_i as a parent would be required to be within the range $[0.001, 0.40]$. The actual values for these parameters (i.e., probabilities) were determined randomly using a uniform distribution over the given range. Following this, to better illustrate the effects of determinism, the networks were copied to create 100 additional networks with their parameters scaled to fit into smaller ranges: $[0.001, 0.01]$, $[0.001, 0.10]$, $[0.001, 0.20]$, $[0.001, 0.30]$. In the following, each network is referred to by the number representing the order in which it was created, as well as the largest value the parameters can receive: BN##_01, BN##_10, BN##_20, BN##_30, and BN##_40.

6. RESULTS

To test the five approaches to deriving decision trees, a dataset of 100,000 examples was generated by forward sampling each Bayesian network. This dataset was subdivided into two equal-sized partitions, one for creating the forward sampling decision tree, and the other for use in testing accuracy of all the methods. For each network, the ID3 algorithm was trained on its partition while the other decision trees were built directly from the network. Every resulting tree was tested on the test partition. Testing pre-pruning was performed by repeating the above procedure for multiple threshold levels. Initial tests on χ^2 pruning set the threshold to 0.0. Each subsequent test increased the threshold by 0.01 up to and including a final threshold level of 2.0. For the probability-based pre-pruning, the initial threshold was set to 0.0 and increased to 0.2 by intervals of 0.005. Accuracy from using the evidence selected by the trees is determined by applying the evidence given by a path in the tree and determining the most-likely fault. The accuracy was then calculated as follows:

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n} I(\hat{d}_i = d_i)$$

where n is the size of the test set, $\hat{d}_i$ is the most-likely fault as predicted by the evidence, d_i is the true fault as indicated by the test set, and $I(\cdot)$ is the indicator function.

The best (over the various χ^2 thresholds) average accuracy achieved over all tests, regardless of size, is given in Table 2. Instead of providing results for all 125 trials, a subsection of the results is given. To ensure the data selection is fair, the original 25 topologies are grouped in the order they were created. The results for parameters in the $[0.001, 0.01]$ range are shown for the first five networks, $[0.001, 0.10]$ for the next five, and so on. Included in the first column of Table 2 are the accuracies obtained by performing inference with the Bayesian network (BN) using all of the evidence.[2] This was done to provide a useful baseline since it provides the theoretical best accuracy for the given set of evidence and the given set of examples generated. The results of applying all evidence are shown in the first column. The next six columns refer to forward sampling (FS), KL-divergence (KL), maximum expected utility (MEU), probability-weighted D-matrix (PWDM), marginal-weighted D-matrix (MWDM), and D-matrix (DM) methods. As can be seen, forward sampling, KL-divergence, and MEU all perform similarly to the original Bayesian network. As indicated earlier, this is due to the trees with no pruning being large and memorizing the Bayesian network.

In all cases, without pruning, the trees created by DM and PWDM are significantly smaller than the other two methods. In some cases, MWDM is somewhat larger than PWDM but still tends to be quite a bit smaller than the other methods. To better illustrate the efficacy of each method when constrained to similar sizes, the last three columns in Table 2 show the accuracies of the methods when the trees are pruned to a similar size as the trees created by PWDM. We noticed that there were several cases where directly comparable trees were not generated. Therefore, in an attempt to maximize fairness, we selected the trees for each pre-pruning threshold as close as possible but no smaller than the trees generated by PWDM. Doing this in an active setting is unlikely to be practical, however, as tuning the prepruning threshold to a fine degree is a costly procedure. Regardless, the best accuracy of the pruned networks is bolded in the table. In the table, the number in parentheses represents the number of leaf nodes, or possible classification paths, in the tree. As can be seen from the table, in most cases PWDM and forward sampling have similar performance. However, in networks with low determinism, forward sampling performs significantly better than PWDM.

In all but the near-deterministic networks, the performance of KL-divergence and MEU is drastically lower than the other methods. However, using the probability-based pruning method improves the accuracy of more compact trees and is shown in Table 3. Once again, results applying all evidence to the Bayesian network are shown in the first column (BN).

As the amount of evidence supplied in PHM settings is frequently fewer than 50,000, additional tests were performed to determine the sensitivity of this method to the sample size. The results of some of these experiments are shown in Table 4. For comparison, the results from the previous table for PWDM are repeated here. The remaining columns represent the results of training the decision tree with 75, 250, 1000, and 50,000 samples respectively. Like before, the trees created by forward

[2]We used the SMILE inference engine developed by the Decision Systems Laboratory at the University of Pittsburgh (SMILE, 2010).

	BN	KL	MEU
BN01_01	0.991	0.924 (22)	0.917 (22)
BN02_01	0.999	0.886 (17)	0.982 (17)
BN03_01	0.999	0.896 (21)	0.895 (18)
BN04_01	1.000	0.869 (26)	0.986 (29)
BN05_01	0.935	0.880 (20)	0.925 (19)
BN06_10	0.924	0.742 (20)	0.812 (20)
BN07_10	0.969	0.880 (25)	0.911 (25)
BN08_10	0.838	0.760 (32)	0.748 (31)
BN09_10	0.957	0.842 (21)	0.849 (20)
BN10_10	0.986	0.925 (27)	0.909 (26)
BN11_20	0.939	0.844 (27)	0.862 (27)
BN12_20	0.897	0.778 (30)	0.812 (31)
BN13_20	0.895	0.755 (30)	0.775 (32)
BN14_20	0.860	0.747 (28)	0.740 (26)
BN15_20	0.875	0.805 (28)	0.815 (28)
BN16_30	0.787	0.663 (37)	0.712 (39)
BN17_30	0.887	0.763 (35)	0.779 (36)
BN18_30	0.813	0.704 (30)	0.753 (28)
BN19_30	0.814	0.715 (38)	0.698 (37)
BN20_30	0.850	0.738 (35)	0.741 (35)
BN21_40	0.646	0.543 (48)	0.600 (43)
BN22_40	0.742	0.701 (44)	0.713 (52)
BN23_40	0.616	0.491 (31)	0.524 (29)
BN24_40	0.739	0.634 (36)	0.673 (35)
BN25_40	0.651	0.570 (33)	0.604 (31)

Table 3: Accuracies for networks using KL-divergence, MEU, and probability-based pruning with the values in paranthesis representing the number of leaf nodes, or classification paths, in the resulting tree

sampling were pruned to a similar size as that obtained by PWDM. However, with so few samples, some of the trees were smaller without any prepruning required, explaining the small tree sizes shown in the table. As can be seen in the results, for many of the networks, PWDM outperforms forward sampling when few samples are available for training. The addition of training samples usually increases performance, but it can be seen on many networks that the addition of training samples can cause a decrease in performance.

To better show the performance of the resulting trees with respect to the pre-pruning process, Figures 3–7 show the average accuracy over each network series relative to the size of the trees when averaged by threshold level. Since PWDM, MWDM, and DM trees are not pruned, there is only a single data point for each of these methods in the graphs to represent the results from that method. These graphs incorporate χ^2 pruning for forward sampling and the probability pruning for KL-divergence and MEU. As can be seen from these graphs, across all networks, the simple D-matrix based approach performs quite well in strict terms of accuracy by size of the network. Additionally, PWDM and MWDM perform similarly across all ranges of determinism, with the gap narrowing as determinism decreases.

While the size of the tree helps determine its complexity, the average number of tests selected is also an important measure. Similar to the graphs relative to size, the graphs in Figures 8–12 show the accuracy in comparison to the average number of tests recommended for evaluation. Under this measure, forward sampling per-

	PWDM	FS (75)	FS (250)	FS (1000)	FS (50000)
BN01_01	0.984 (22)	0.974 (9)	0.975 (9)	**0.988 (23)**	0.983 (23)
BN02_01	0.984 (17)	0.963 (9)	0.981 (10)	0.983 (12)	**0.988 (21)**
BN03_01	0.988 (18)	0.987 (10)	0.985 (12)	**0.992 (18)**	0.991 (19)
BN04_01	0.988 (25)	0.984 (10)	0.989 (11)	0.967 (15)	**0.995 (25)**
BN05_01	0.924 (20)	0.920 (11)	0.919 (11)	0.922 (16)	**0.927 (21)**
BN06_10	0.854 (20)	0.812 (14)	**0.855 (20)**	0.765 (20)	0.800 (20)
BN07_10	0.930 (25)	0.881 (11)	0.937 (21)	**0.951 (26)**	0.945 (25)
BN08_10	0.785 (31)	0.775 (12)	0.813 (23)	0.822 (31)	**0.824 (34)**
BN09_10	0.868 (19)	0.885 (13)	0.905 (22)	**0.922 (22)**	0.918 (23)
BN10_10	0.937 (25)	0.902 (9)	0.910 (12)	**0.968 (28)**	0.955 (25)
BN11_20	0.832 (27)	0.840 (19)	**0.893 (27)**	0.889 (29)	0.892 (27)
BN12_20	0.769 (30)	0.729 (17)	0.768 (30)	**0.841 (30)**	0.802 (30)
BN13_20	0.769 (29)	0.723 (27)	0.802 (29)	0.815 (32)	**0.836 (29)**
BN14_20	0.779 (26)	0.760 (19)	**0.814 (26)**	0.785 (29)	0.782 (26)
BN15_20	0.818 (27)	0.848 (21)	0.849 (31)	0.825 (28)	**0.857 (28)**
BN16_30	0.665 (37)	0.664 (21)	**0.708 (37)**	0.671 (43)	0.676 (39)
BN17_30	0.731 (33)	0.764 (20)	0.793 (33)	**0.819 (47)**	0.812 (55)
BN18_30	0.692 (27)	0.707 (21)	0.752 (34)	0.767 (31)	**0.775 (32)**
BN19_30	0.653 (37)	0.706 (21)	0.767 (39)	0.771 (50)	**0.786 (37)**
BN20_30	0.660 (35)	0.738 (28)	0.783 (43)	0.742 (38)	**0.800 (40)**
BN21_40	0.459 (43)	0.526 (30)	0.579 (44)	0.541 (47)	**0.623 (65)**
BN22_40	0.550 (44)	0.663 (21)	**0.702 (50)**	0.702 (52)	0.651 (46)
BN23_40	0.525 (29)	0.518 (27)	0.480 (33)	0.405 (25)	**0.548 (30)**
BN24_40	0.607 (35)	0.623 (26)	0.692 (38)	**0.693 (53)**	0.683 (36)
BN25_40	0.576 (31)	0.551 (28)	0.601 (33)	0.601 (33)	**0.603 (34)**

Table 4: Accuracy for differing sample sizes for forward sampling and ID3 over individual networks with χ^2 pruning where the numbers in parentheses indicate the number of leaf nodes in the tree

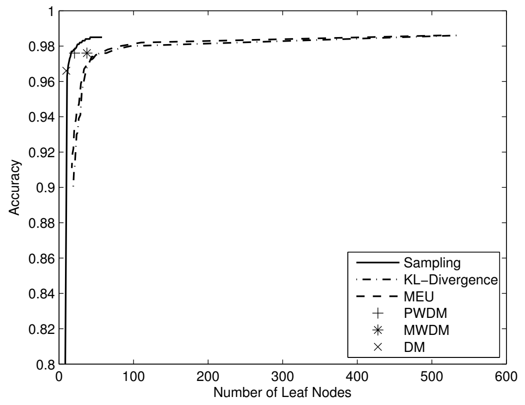

Figure 3: Average accuracy for networks with parameters in the range [0.001,0.01] with respect to size

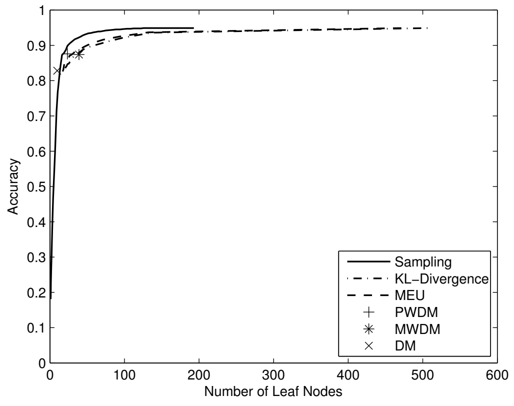

Figure 4: Average accuracy for networks with parameters in the range [0.001,0.10] with respect to size

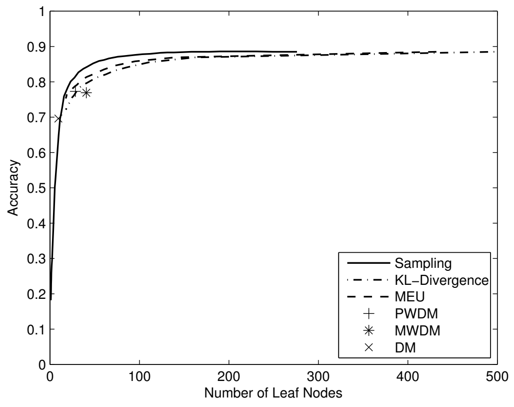

Figure 5: Average accuracy for networks with parameters in the range [0.001,0.20] with respect to size

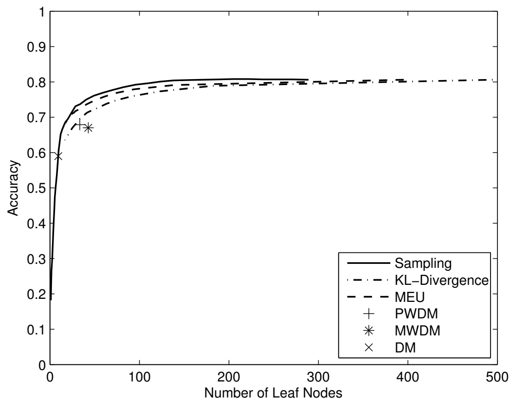

Figure 6: Average accuracy for networks with parameters in the range [0.001,0.30] with respect to size

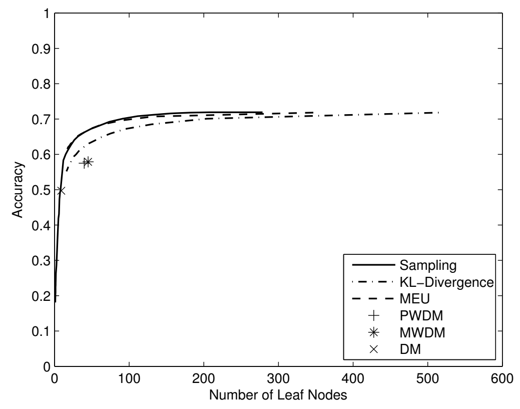

Figure 7: Average accuracy for networks with parameters in the range [0.001,0.40] with respect to size

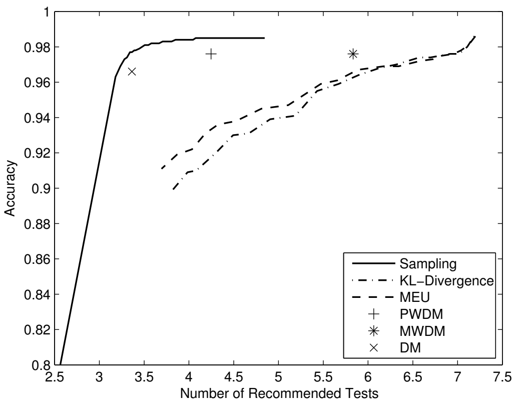

Figure 8: Average accuracy for networks with parameters in the range [0.001,0.01] with respect to the number of recommended tests

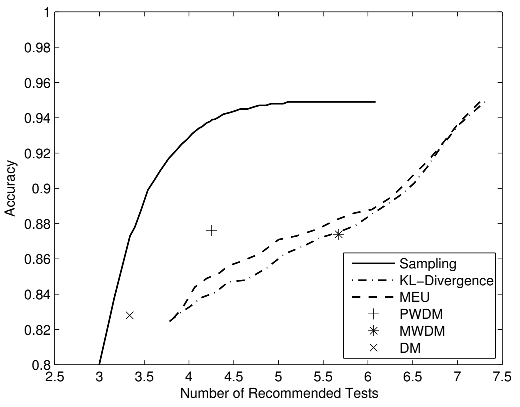

Figure 9: Average accuracy for networks with parameters in the range [0.001,0.10] with respect to the number of recommended tests

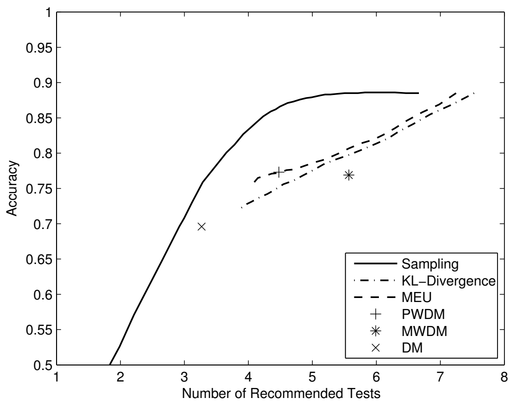

Figure 10: Average accuracy for networks with parameters in the range [0.001,0.20] with respect to the number of recommended tests

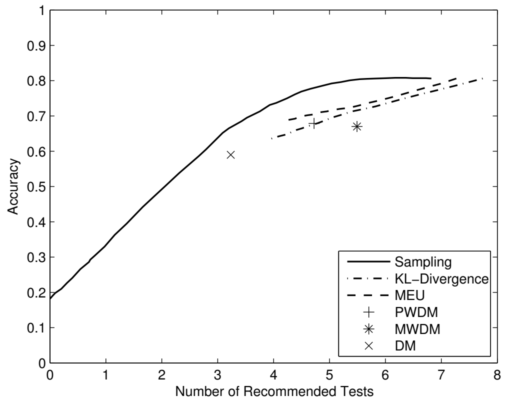

Figure 11: Average accuracy for networks with parameters in the range [0.001,0.30] with respect to the number of recommended tests

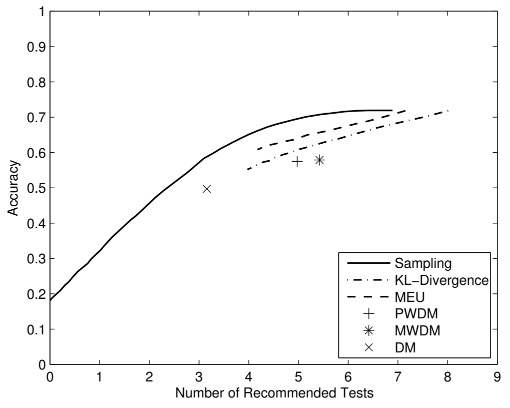

Figure 12: Average accuracy for networks with parameters in the range [0.001,0.40] with respect to the number of recommended tests

forms quite well across all levels of determinism. Also, while both PWDM and MWDM are similar in overall size, MWDM tends to recommend more tests for evaluation. Once again, in strict terms of accuracy compared to the number of recommended tests, the simple D-matrix approach performs well.

7. DISCUSSION

In networks with high determinism, all of the methods perform similarly when pruned to compact trees. In networks with low determinism, the performance of PWDM and MWDM degrades in comparison to the other three methods. Across all levels of determinism, forward sampling performs well, provided it is given an adequate training set size. With smaller training sets, the accuracy of the trees can be erratic. We also note that the decision trees generated by the DM algorithm are consistently much smaller than those generated by all other methods. Given the nature of the algorithm, this is not a surprise. What is particularly interesting, however, is that, while the accuracy was typically less than both PWDM and MWDM, in many cases it was still comparable. Thus, the DM approach could provide an excellent "first approximation" for a decision tree, to be refined with the more complex PWDM- or MWDM-generated trees, or other methods, if higher accuracy is necessary.

When accuracy is compared to the number of tests recommended by a method, forward sampling performs quite well across all of the networks. Additionally, in this measure PWDM and MWDM are significantly different with PWDM recommending fewer tests. However, both number of tests and the size measurement for forward sampling is highly dependent on the size of the data set. As the size of the data set increases, the size of the trees created by the method increases, until reaching a point where the data set accurately represents the underlying distribution. This introduces yet another parameter to be optimized in order to maximize the performance of forward sampling.

Finally, we note that the processes by which the decision trees are generated vary considerably in computational complexity. While we do not provide a formal complexity analysis here, we can discuss the intuition behind the complexity. Forward sampling first requires generating the data (whose complexity depends on the size of the network) and then applying the ID3 algorithm. ID3 requires multiple passes through the data, so complexity is driven by the size of the data set. The complexity of all other methods depend only on the size of the network. Both the KL-divergence method and the MEU method are very expensive in that they each require performing multiple inferences with the associated network, which is NP-hard to perform exactly. Suitable approximate inference algorithms may be used to improve the computational complexity of this problem, but the accuracy of the method will suffer. PWDM likewise requires inference to be performed during creation of the decision tree. However, since determination of an appropriate pruning parameter is not required in order to limit the size of the network, PWDM requires fewer inferences to be performed over the network, resulting in higher performance in regards to computation speed over MEU and KL-divergence. Neither DM nor MWDM require data to be generated or inference to be performed. Instead, the trees are generated from a simple application of the ID3 algorithm to a compact representation of the network. Due to this, the trees generated by these networks can be created very quickly, even for larger networks. Like PWDM, these two methods also do not require learning parameters for sample size or prepruning in order to generate compact trees, resulting in these two methods having the smallest computational burden.

In addition to the computational complexity, forward sampling, KL-divergence, and MEU all require significant pruning of the resulting trees. However, early results showed that performing a χ^2 test at the 5% significance level failed to significantly prune the trees. Since KL-divergence and MEU do not have set data sizes, partition sizes were estimated based upon the probability of reaching a node. Thus, pruning to a size that is competitive with PWDM and MWDM requires an additional parameter to be learned, adding complexity to the system. Pruning to a level where the average number of recommended tests is as low as PWDM and MWDM is similarly difficult, though can be guided by calculating the probability of reaching nodes in the network.

The results indicate that, while PWDM and MWDM are not necessarily the most accurate methods, PWDM and MWDM yield compact trees with reasonably accurate results, and they do so efficiently. The difference in performance between the two is negligible in most cases, suggesting that MWDM is most likely the most cost-effective approach in complexity. PWDM is however more cost-effective in the number of tests performed.

8. CONCLUSION

Comparing accuracy in relation to the size of the tree as well as number of tests evaluated, forward sampling performs well across all levels of determinism tested when given adequate samples. However, in terms of complexity, PWDM and MWDM yield compact trees while maintaining accuracy. The unweighted D-matrix based method provides the best results when comparing complexity, size, and number of tests evaluated when compared to the level of accuracy from trees with similar properties created by other methods. This indicates that the D-matrix based methods are useful in determining a low-cost baseline for test selection which can be expanded upon by other methods if a higher level of accuracy is required. Further work in this area will compare these methods against networks specifically designed to be difficult to solve for certain inference methods, as noted by Mengshoel, Wilkins, Roth, Ide, Cozman, and Ramos (Mengshoel, Wilkins, & Roth, 2006; Ide, Cozman, & Ramos, 2004).

ACKNOWLEDGMENTS

The research reported in this paper was supported, in part, by funding through the RightNow Technologies Distinguished Professorship at Montana State University. We wish to thank RightNow Technologies, Inc. for their continued support of the computer science department at MSU and of this line of research. We also wish to thank John Paxton and members of the Numerical Intelligent Systems Laboratory (Steve Butcher, Karthik Ganesan Pillai, and Shane Strasser) and the anonymous reviewers for their comments that helped make this paper stronger.

REFERENCES

Casey, R. G., & Nagy, G. (1984). Decision tree design using a probabilistic model. *IEEE Transactions on Information Theory*, *30*, 93-99.

Craven., M. W., & Shavlik, J. W. (1996). Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems* (Vol. 8, pp. 24–30).

Frey, L., Tsamardinos, I., Aliferis, C. F., & Statnikov, A. (2003). Identifying markov blankets with decision tree induction. In *In icdm 03* (pp. 59–66). IEEE Computer Society Press.

Heckerman, D., Breese, J. S., & Rommelse, K. (1995). Decision-theoretic troubleshooting. *Communications of the ACM*, *38*, 49–57.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, *20*(3), 20–197.

Hogg, R. V., & Craig, A. T. (1978). *Introduction to mathematical statistics* (4th ed.). Macmillan Publishing Co.

Ide, J. S., Cozman, F. G., & Ramos, F. T. (2004). Generating random bayesian networks with constraints on induced width. In *Proceedings of the 16th european conference on artificial intelligence* (pp. 323–327).

Jensen, F. V., Kjærulff, U., Kristiansen, B., Lanseth, H., Skaanning, C., Vomlel, J., et al. (2001). The sacso methodology for troubleshooting complex systems. *AI EDAM Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *15*(4), 321–333.

Jordan, M. I. (1994). A statistical approach to decision tree modeling. In *In m. warmuth (ed.), proceedings of the seventh annual acm conference on computational learning theory* (pp. 13–20). ACM Press.

Kim, Y. gyun, & Valtorta, M. (1995). On the detection of conflicts in diagnostic bayesian networks using abstraction. In *Uncertainty in artificial intelligence: Proceedings of the eleventh conference* (pp. 362–367). Morgan-Kaufmann.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models*. The MIT Press.

Liang, H., Zhang, H., & Yan, Y. (2006). Decision trees for probability estimation: An empirical study. *Tools with Artificial Intelligence, IEEE International Conference on*, *0*, 756-764.

Martin, J. K. (1997). An exact probability metric for decision tree splitting and stopping. *Machine Learning*, *28*, 257-291.

Mengshoel, O. J., Wilkins, D. C., & Roth, D. (2006). Controlled generation of hard and easy bayesian networks: Impact on maximal clique size in tree clustring. *Artificial Intelligence*, *170*(16–17), 1137–1174.

Murthy, S. K. (1997). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, *2*, 345–389.

Mussi, S. (2004). Putting value of information theory into practice: A methodology for building sequential decision support systems. *Expert Systems*, *21*(2), 92-103.

Pattipati, K. R., & Alexandridis, M. G. (1990). Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Transactions on Systems, Man and Cybernetics*, *20*(44), 872–887.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Przytula, K. W., & Milford, R. (2005). An efficient framework for the conversion of fault trees to diagnostic bayesian network models. In *Proceedings of the ieee aerospace conference* (pp. 1–14).

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Shwe, M., Middleton, B., Heckerman, D., Henrion, M., Horvitz, E., Lehmann, H., et al. (1991). Probabilistic diagnosis using a reformulation of the internest-1/qmr knowledge base: 1. the probabilistic model and inference algorithms. *Methods of Information in Medicine*, *30*(4), 241-255.

Simpson, W. R., & Sheppard, J. W. (1994). *System test and diagnosis*. Norwell, MA: Kluwer Academic Publishers.

Skaanning, C., Jensen, F. V., & Kjærulff, U. (2000). Printer troubleshooting using bayesian networks. In *Iea/aie '00: Proceedings of the 13th international conference on intustrial and engineering applications of artificial intelligence and expert systems* (pp. 367–379).

Smile reasoning engine. (2010). University of Pittsburgh Decision Systems Laboratory. (http://genie.sis.pitt.edu/)

Vomlel, J. (2003). *Two applications of bayesian networks.*

Zheng, A. X., Rish, I., & Beygelzimer, A. (2005). Efficient test selection in active diagnosis via entropy approximation. In *Uncertainty in artificial intelligence* (pp. 675–682).

Zhou, Y., Zhang, T., & Chen, Z. (2006). Applying bayesian approach to decision tree. In *Proceedings of the international conference on intelligent computing* (pp. 290–295).

Scott Wahl is a PhD student in the department of computer science at Montana State University. He received his BS in computer science from MSU in December 2008. Upon starting his PhD program, Scott joined the Numerical Intelligent Systems Laboratory at MSU and has been performing research in Bayesian diagnostics.

John W. Sheppard is the RightNow Technologies Distinguished Professor in the department of computer science at Montana State University. He is also the director of the Numerical Intelligent Systems Laboratory at MSU. Dr. Sheppard holds a BS in computer science from Southern Methodist University as well as an MS and PhD in computer science, both from Johns Hopkins University. He has 20 years of experience in industry and 15 years in academia (10 of which were concurrent in both industry and academia). His research interests lie in developing advanced algorithms for system level diagnosis and prognosis, and he was recently elected as a Fellow of the IEEE for his contributions in these areas.