Comparing probabilistic inference for mixed Bayesian networks

K.C. Chang and Wei Sun
Dept. of Systems Engineering & Operations Research
School of Information Technology and Engineering
George Mason University
Fairfax, VA 22030, USA

ABSTRACT

Bayesian Networks are graphical representation of dependence relationships between domain variables. They have been applied in many areas due to their powerful probabilistic inference such as data fusion, target recognition, and medical diagnosis, etc. There exist a number of inference algorithms that have different tradeoffs in computational efficiency, accuracy, and applicable network topologies. It is well known that, in general, the exact inference algorithms are either computationally infeasible for dense networks or impossible for mixed discrete-continuous networks. However, in practice, mixed Bayesian Networks are commonly used for various applications. In this paper, we compare and analyze the trade-offs for several inference approaches. They include the exact Junction Tree algorithm for linear Gaussian networks, the exact algorithm for discretized networks, and the stochastic simulation methods. We also propose an almost instant-time algorithm (AIA) by pre-compiling the approximate likelihood tables. Preliminary experimental results show promising performance.

Keywords: Bayesian Networks, Gaussian Mixture Model (GMM), probabilistic inference

1. INTRODUCTION

A Bayesian Network is a directed acyclic graph (DAG), in which each node represents a random variable; each arc represents the probabilistic relationship between the parent and the children nodes. For each node, there is a conditional probability distribution (CPD) specified, which represents the parametric dependence relationship with its parents. Bayesian networks have been applied in many areas due to their powerful probabilistic inference [1]. The most common problem we wish to solve using Bayesian networks is how to compute the posterior distribution of the target node given observed evidence. This process is generally called probabilistic inference. A graphical model specifies a complete joint probability distribution (JPD) over all the variables. Given the JPD, we can answer all possible inference queries by marginalization (summing out over irrelevant variables). Various inference methods differ in how they sum over the irrelevant variables [2-11].

In the past two decades, much effort has been focused on the development of efficient probabilistic inference algorithms. These algorithms, for the most part, have been designed to efficiently compute the posterior probability of each node or the result of simple arbitrary queries. However, most of the research about exact inference is on Bayesian networks with either all discrete-valued variables or mixed discrete continuous variables with “linear Gaussian” assumption. It is well known that the algorithms for exact inference are either computationally infeasible for dense networks or impossible for the networks containing mixed (discrete and continuous) variables with nonlinear or non-Gaussian CPD. In those cases, one either has to discretize all the continuous variables in order to apply an exact algorithm or rely on approximate algorithms such as stochastic simulation methods.

In practical situations, hybrid Bayesian Networks are commonly used for various applications [8][12-13][23]. It is therefore important to understand the performance of different approximate algorithms applied in such situations.
this paper, we first compare and analyze the trade-offs for several existing inference approaches for mixed networks. Particularly, in order to compare the performance, we design a hybrid linear Gaussian network (see Figure 1) for testing. The algorithms to be compared include: (1) an exact algorithm (eg., Junction tree) on the original network, (2) an exact algorithm (eg., Junction tree) on the discrete network obtained by discretizing all continuous nodes of the original network, and (3) an approximate algorithm based on stochastic simulation with likelihood weighting \[10-11\] on the original network.

Since inference is generally computationally intensive, we then propose an Almost Instant-time Algorithm (AIA). The idea is to pre-compute and store the likelihood tables of the relevant evidences given the target node. When evidence arrives, the inference process becomes trivial since one only needs to extract the proper likelihoods from the table and conduct simple multiplication and normalization operations according to Bayes' Rule. However, in general, the likelihood tables could be huge, we propose several methods to approximate the likelihood functions and demonstrate their performances.

This paper is organized as follows: Section 2 defines the Bayesnet inference problem and Section 3 presents the results of the comparison between the three algorithms. Section 4 describes the AIA algorithm and shows the preliminary experimental results. Some special issues in implementing AIA algorithm, such as continuizing discrete nodes and choosing number of components in Gaussian mixture, are also discussed. Concluding remarks are given in Section 5.

## 2. PRELIMINARIES ON BAYESIAN INFERENCE

Consider a generic Bayesian network with both discrete and continuous nodes. The target node is assumed to be discrete (say, for classification purpose, we have a Bayesnet with a node being the target class and the rest of the network are direct or indirect observables). A common inference problem is to compute the posterior probability of the target node \(x\) given the evidence set \(E\),

\[
p(x \mid E) = \int p(x, \Omega \mid E) d\Omega = \frac{1}{c} \int p(E \mid x, \Omega) p(\Omega \mid x) p(x) d\Omega,
\]

where the \(c\) is the normalization constant, \(\Omega\) is the set of nodes other than the target node and observable set \(E\) that may exist in the network.

There are a few algorithms available for exact inference (SPI, Junction tree, etc) [4][6]. In order to compare different algorithms, we design the network given in Figure 1 for testing. In this network, there are two discrete nodes (SDA, SDB) each has eight and ten state values respectively, the rest of the nodes are continuous. The relationships between continuous nodes and discrete nodes are defined as conditional Gaussian (CG) and the relationships between continuous nodes are linear Gaussian. Therefore the distribution of each continuous node can be represented as a Gaussian mixture. Without loss of generality, assuming the target node of interest is SDB and the observable nodes are COA – COF.

The three inference approaches being compared are as follows:

1. Exact inference with Junction tree algorithm (Conditional Gaussian) on the original mixed network.
2. Exact inference with Junction tree algorithm (Discrete) on the modified discrete network obtained by discretizing all the continuous nodes of the original network.
3. Approximate inference with likelihood weighting \[10-11\] on the original mixed network.
In this work, we use MATLAB Bayesian Networks Toolbox [14] for simulation and inference. We first generate 1,000 random sets of the six evidence nodes given each target state from the original Bayesnet using forward sampling. These sets of evidences are used to test all three algorithms. The probability of correct decision (Pd), defined as the percentage of trials where the inferred target state is identical to the true target state, is used as the performance measure.

The results are summarized in Table 1 and Figure 2. Figure 2 shows the performances (in terms of Pd) of the three methods for each target state. For discretized network, we used uniform discretization for each continuous node with either 10 or 40 bins. As can be seen from Table 1, the performance with the discretized network could degrade (compared to the Junction Tree exact algorithm on the original network) as much as 25% for the case with 10 bins and 10% for the case with 40 bins. For simulation approach, we conducted 10,000 trials for each observed evidence set using likelihood weighting (LW) technique [10-11]. The results from the LW is about 2% below the exact algorithm. Table 1 shows the normalized run time (NRT) of the three algorithms. Note here that the accuracy and run time for the simulation algorithm depends on the number of trials we used in each test.

From the results, it is clear that one needs to be very careful in discretizing a continuous node when converting a hybrid network into a discrete one. Many approaches have been proposed to intelligently partition the continuous space when discretizing a continuous node [13][19]. The focus of this paper is not to discuss the trade-offs of different discretization approaches or to develop a newer and better discretization method, rather is to develop an algorithm which could potentially avoids the problem caused by the discretization and at the same time provide an efficient real time inference. We will propose such an algorithm in the next section.

### Table 1. Average Pd and Normalized Run Time Comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Junction Tree</th>
<th>LW</th>
<th>Discrete/10</th>
<th>Discrete/40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pd</td>
<td>0.8852</td>
<td>0.8640</td>
<td>0.6665</td>
<td>0.8001</td>
</tr>
<tr>
<td>NRT</td>
<td>1.0</td>
<td>11.8</td>
<td>0.045</td>
<td>0.05</td>
</tr>
</tbody>
</table>

1 The inferred ID is the one with the highest posterior probability
In this section, we propose a real time algorithm called the Almost Instant-time Algorithm (AIA). The idea is to pre-compute and store the likelihood tables before the evidence arrives. When evidence is received, the inference becomes a quick table look-up and a set of simple operations. This is so because Bayes’ Rule states:

\[ p(x | E) = \frac{1}{Z} p(E | x) p(x) \]

where the \( p(E | x) \) is the likelihood of the observed evidence given the target node \( x \), \( p(x) \) is the prior probability of \( x \). When \( p(E, x) \) is readily available in a table, the only calculations needed are a few multiplications and normalization.

However, in general the observable set \( E \) could include a large number of evidence nodes and therefore \( p(E | x) \) would be a high dimensional distribution. Besides, each evidence node in \( E \) could be a discrete variable with a large set of values or a continuous variable. This could make the joint space of \( E \) very large. In order to handle the potentially huge tables, we propose using the Gaussian mixture model (GMM) to approximate the distribution \( p(E | x) \). It is well known that one can approximate any arbitrary distribution with a GMM of sufficient components to any desirable accuracy [15]. Besides, due to the fact that typically the model constructed by a Bayesnet is not precise, the resulting small error in posterior probability due to the approximation makes little difference in decision making.

In order to approximate the likelihood function with a GMM, we need to specify the number of components, or mixture size, for the Gaussian mixture. How should we select this mixture size? Basically, this depends on the joint state space.
of evidence set and the utility function making a trade-off between accuracy and computational cost. One approach is to use relative entropy (Kullback-Liebler distance) \[16][20\] as a measure to decide the optimal number of Gaussian components. At first, we begin to learn the Gaussian mixture by choosing an estimated mixture size, and then increase the number of components gradually until the decrease of relative entropy is less than a pre-defined threshold \[17\].

If the likelihood function being approximated is a joint Gaussian distribution, then the learning process is trivial. In our experiments, 10 components Gaussian mixture is sufficient enough to make the AIA perform almost identical to the exact Junction Tree algorithm. When some or all of the evidence nodes are discrete variables, instead of building the potentially huge discrete or conditional Gaussian tables, we propose to first “continunize” the discrete nodes by adding proper Gaussian noises to “fuzzify” the discrete variables. To do so, we first convert the discrete state into numerical representation, i.e., using numerical values (e.g. integers) to represent different discrete states. We then add a proper Gaussian noise to each numerical value when generating the evidence samples. In order to keep the authenticity of discrete states, the variance of added noise needs to be chosen carefully. Typically, the standard deviation of the added Gaussian noise for each discrete state should not be greater than one third of the numerical distance between two neighboring discrete states. Once all evidence samples are available, we then proceed to approximate the joint likelihood function with a GMM distribution. One benefit of making discrete nodes continuous is that we have the flexibility of using a smaller number of higher dimensional Gaussian components to approximate the original joint distribution.

The key advantage of the AIA approach is the ability to handle arbitrary networks including mixed variables as well as nonlinear non-Gaussian models. The fact that only forward sampling is needed to generate the data for learning the likelihood tables makes this approach very general and attractive. Preliminary experiments demonstrate the feasibility and effectiveness of this approach. To test the performance of the algorithm, we use the same example shown in Figure 1. The following steps summarize the procedures:

1. Generate random samples for evidence nodes given each target state as training data. Specifically, we generate 100,000 samples for each target state using forward sampling where each sample consists of a six-dimensional vector of real values.

2. Learn the Gaussian mixture from the training data to approximate the joint distribution \( p(E|x) \) by using EM algorithm. We use the MATLAB software package Netlab toolbox \[17\] to implement this procedure. Each component of the Gaussian mixture consists of a mean vector and a full covariance matrix.

3. For each target state, a GMM will be created and stored. In our example, we have shown that a GMM of 10 components is sufficiently accurate to represent the likelihood function.

4. Use the same 1,000 evidence sets as the one described in section n3 to do testing.

The resulting performance is shown in Table 2 as well as in Figure 3. It can be seen that in this case, the performance of the AIA algorithm is almost the same as the exact Junction tree algorithm with orders of magnitude faster in computation.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Junction Tree</th>
<th>AIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pcd</td>
<td>0.8852</td>
<td>0.8857</td>
</tr>
<tr>
<td>NRT</td>
<td>1.0</td>
<td>0.0054</td>
</tr>
</tbody>
</table>

Table 2. Average Pcd and Run Time Comparison
To compile the likelihood tables, another approach is to use the classification tree [18] to partition the joint space into discrete “hyper cubes”. Similar to the one described in [21], we propose to represent the high dimensional joint evidence space with an asymmetric discretized space where each cell is induced by a path in the decision tree. The difference here is that we are not simplifying the network topology by aggregating nodes [22], rather we are focusing only on simplifying the joint evidence space $E$ in order to produce the likelihood tables of smaller sizes.

To train the decision tree, we use the random samples obtained by the forward sampling from the Bayesnet as described earlier. The algorithm such as C4.5/C5.0 [27] is used to derive the tree structure. The target node is used as the classification node and all the evidence nodes are used as the feature variables. Preliminary results show that the joint state space representation could be reduced by several orders of magnitude. For example, for the discretized network, with 10 bins on each of the evidence node, the size of the likelihood table $p(E|x)$ is $10^6 \times 10 = 10^7$. With the classification tree, we could reduce the size of the joint likelihood table to approximate $10^3$ each cell corresponds to a leaf in the tree.

It is common that given a Bayesnet model, the target nodes to be queried are usually a small subset of the networks. To answer the queries, one only needs to build the relevant likelihood tables for those target nodes. Besides, evidence nodes can typically be partitioned into groups where only evidences within each group can be observed simultaneously\(^2\).

\(^2\) It is easy to identify for each target node the relevant evidence nodes using simple procedure [25]

\(^3\) For example, in multisensor fusion applications, evidences from different sensors are typically observed asynchronously.
Constructing the likelihood tables within each group can reduce their sizes significantly. Note that another approach is to use a naïve model to describe the likelihood table where explicit conditional independence assumption between evidence is made. In that case, only likelihood tables of each individual evidence node given the target node are needed. However, our preliminary results show that the performance could be degraded significantly.

5. CONCLUDING REMARKS

A number of inference algorithms exist for Bayesian Networks with different trade-offs in computational efficiency, accuracy, and applicable network topologies. It is well known that the exact inference algorithms are not possible for general non-Gaussian hybrid Bayesian networks. However, in practice, hybrid Bayes nets are commonly used for various applications. In this paper, we first compare and analyze the trade-offs of three algorithms: the Junction Tree algorithm for linear Gaussian network, the exact algorithm on discretized network, and a stochastic simulation method. We then present a new computational method, the almost instant-time algorithm (AIA) for general hybrid Bayesnet inference. Preliminary experimental results demonstrate the feasibility and effectiveness of this approach.

Although there exist several exact inference algorithms, they are only applicable to limited types of network (discrete or mixed linear Gaussian). Based on our preliminary experiment results, the proposed AIA performs comparably to the exact algorithm in accuracy for the linear Gaussian case. Furthermore, there is no limitation on applicable networks for AIA. Most importantly, AIA is a real time algorithm with almost instant-time inference. By shifting the burden of likelihood computation off-line, AIA is especially useful in the applications where real-time requirement is critical. We believe the proposed AIA method represents an important step in realizing practical inference for complex hybrid systems.

6. REFERENCE


