Latent Tree Models and Approximate Inference in Bayesian Networks *

Yi Wang and **Nevin L. Zhang** and **Tao Chen**
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong, China
{wangyi,lzhang,csct}@cse.ust.hk

Abstract

We propose a novel method for approximate inference in Bayesian networks (BNs). The idea is to sample data from a BN, learn a latent tree model (LTM) from the data offline, and when online, make inference with the LTM instead of the original BN. Because LTMs are tree-structured, inference takes linear time. In the meantime, they can represent complex relationship among leaf nodes and hence the approximation accuracy is often good. Empirical evidence shows that our method can achieve good approximation accuracy at low online computational cost.

Introduction

Latent tree models (LTMs) are tree-structured Bayesian networks in which leaf nodes are observed while internal nodes are hidden. They are also known as hierarchical latent class models (Zhang 2004). We call the leaf nodes *manifest variables* and the internal nodes *latent variables*. In this paper, we do not distinguish between nodes and variables.

Pearl (1988) was the first to identify LTMs as a potentially useful class of models. There are two reasons. First, inference in LTMs takes time linear in the number of nodes, while it is intractable in general BNs. Second, the latent variables capture complex relationships among manifest variables. In an LTM, the manifest variables are mutually independent given the latent variables, while eliminating all the latent variables results in a completely connected BN.

We study the possibility of exploiting those two merits for approximate inference in BNs. Here is the most natural idea:

1. *Offline*: Obtain an LTM $\mathcal{M}$ that approximates a BN $\mathcal{N}$ in the sense that the joint distribution of the manifest variables in $\mathcal{M}$ approximately equals the joint distribution of the variables in $\mathcal{N}$.
2. *Online*: Use $\mathcal{M}$ instead of $\mathcal{N}$ to compute answers to probabilistic queries.

The cardinalities of the latent variables play a crucial role in the approximation scheme. They determine the inferential complexity and influence the approximation quality. At one extreme, we can represent a BN exactly using an LTM by setting the cardinalities of the latent variables large enough. In this case, the inferential complexity is very high. At the other extreme, we can set the cardinalities of the latent variables at 1. In this case, the manifest variables become mutually independent. The inferential complexity is the lowest and the approximation quality is the poorest. We seek a middle point between those two extremes.

We assume that there is a predetermined constraint on the cardinalities of the latent variables to control the inferential complexity. We develop an algorithm for finding an LTM that satisfies the constraint and approximates the original BN well. The idea is to sample data from the BN, and learn an LTM from the data. The model structure is determined by hierarchically clustering manifest variables. In each step, two closely correlated sets of manifest variables are grouped, and a new latent variable is introduced to account for the relationship between them. The cardinalities of the latent variables are set at the predetermined value. The parameters are optimized using the Expectation-Maximization (EM) algorithm (Dempster, Laird, & Rubin 1977).

We have empirically evaluated our inference method on an array of networks. The possibility to tradeoff between the inferential complexity and accuracy has been demonstrated by adjusting the cardinality constraints. It turns out that our method is able to achieve good approximation accuracy before the cardinality becomes too high.

We also compared our method with other approximate inference methods. One of them is loopy belief propagation (LBP) (Pearl 1988), a standard approximate inference method which has been successfully used in many real world domains (Murphy, Weiss, & Jordan 1999). In one particular setting, the accuracy of our method is competitive with or higher than that of LBP in most of the networks. In the meantime, our method is faster than LBP by up to two orders of magnitude.

Our inference scheme exploits the computational simplicity and the strong expressive capability of LTMs. One can of course obtain new schemes by replacing LTMs with other models that bear those merits. Two straightforward choices are Chow-Liu (CL) trees (Chow & Liu 1968) and latent class models (LCMs) (Lowd & Domingos 2005). The former are tree-structured BNs without latent variables, while the latter are a special case of LTM with only one latent variable. In

*Research on this work was supported by Hong Kong Grants Council Grant #622307, and the National Basic Research Program of China (aka the 973 Program) under project No. 2003CB517106.

machine learning community, LCM is also known as naive Bayes model with latent variable. We refer to the schemes that use CL trees and LCMs as the CL-based method and the LCM-based method, and compared our method with them. The results show that our method can always achieve higher approximate accuracy.

It should be noted that our approximate scheme needs a lot of time in the offline phase. Therefore, our method is suitable for applications that demand good online performance while allowing a long offline phase.

The remainder of this paper is organized as follows. We first review LTMs. Then we present our method of constructing LTMs to approximate BNs, followed by the formal description of our scheme for approximate inference. Empirical results are reported next, and the relationship between our approach and existing work is discussed. Finally, we conclude this paper.

Latent Tree Model

An LTM is a pair $\mathcal{M} = (m, \boldsymbol{\theta}_m)$. The first component m denotes the rooted tree and the set of cardinalities of the latent variables. We refer to m as the model, and the rooted tree as the model structure. The second component $\boldsymbol{\theta}_m$ denotes the collection of parameters in $\mathcal{M}$. It contains a conditional probability table for each node given its parent. Let $\mathbf{X}$ and $\mathbf{Y}$ be the set of manifest variables and the set of latent variables in $\mathcal{M}$, respectively. We will use $P(\mathbf{X}, \mathbf{Y}|m, \boldsymbol{\theta}_m)$, or $P_{\mathcal{M}}(\mathbf{X}, \mathbf{Y})$ in short, to denote the joint distribution represented by $\mathcal{M}$.

Let $|Z|$ denote the cardinality of variable Z. For a node Z in m, we use $\mathbf{nb}(Z)$ to denote the set of its neighbors. A model m is *regular* if for any latent node Y, $|Y| \leq \frac{\prod_{Z \in \mathbf{nb}(Y)} |Z|}{\max_{Z \in \mathbf{nb}(Y)} |Z|}$, and the inequality strictly holds when Y has only two neighbors. In a regular model, a latent node Y is *saturated* if $|Y| = \frac{\prod_{Z \in \mathbf{nb}(Y)} |Z|}{\max_{Z \in \mathbf{nb}(Y)} |Z|}$. In this case, we say that Y *subsumes* all its neighbors except the one with the largest cardinality.

Approximating BNs with LTMs

In this section, we study the problem of approximating a BN with an LTM. Let $\mathcal{N}$ be the BN to be approximated. Let $\mathbf{X}$ be the set of variables in $\mathcal{N}$. For an LTM $\mathcal{M}$ to be an approximation of $\mathcal{N}$, it should use $\mathbf{X}$ as its manifest variables, and the cardinalities of its latent variables should not exceed a predetermined threshold C. Figures 1(b), 1(c), and 1(d) show three example LTMs that approximate the BN in Figure 1(a). They will be used for illustration in this section.

Let $P_{\mathcal{N}}(\mathbf{X})$ be the joint distribution represented by $\mathcal{N}$. An approximation $\mathcal{M}$ is of high quality if $P_{\mathcal{M}}(\mathbf{X})$ is close to $P_{\mathcal{N}}(\mathbf{X})$. We measure the quality of the approximation by the KL divergence (Cover & Thomas 1991)

$$D[P_{\mathcal{N}}(\mathbf{X}) \| P_{\mathcal{M}}(\mathbf{X})] = \sum_{\mathbf{X}} P_{\mathcal{N}}(\mathbf{X}) \log \frac{P_{\mathcal{N}}(\mathbf{X})}{P_{\mathcal{M}}(\mathbf{X})}.$$

Our objective is to find an LTM that minimizes the KL divergence, i.e.,

$$\mathcal{M}^\star = \arg\min_{\mathcal{M}} D[P_{\mathcal{N}}(\mathbf{X}) \| P_{\mathcal{M}}(\mathbf{X})].$$

An LTM $\mathcal{M}$ consists of two components, the model m and the parameters $\boldsymbol{\theta}_m$. Therefore, the optimization problem can be naturally decomposed into two subproblems.

1. Find an optimal model $m^\star$.
2. Optimize the parameters $\boldsymbol{\theta}_m$ for a given model m.

In the remainder of this section, we will discuss these two subproblems in details.

Parameter Optimization

We start by addressing the second subproblem. Given a model m, the target is to find

$$\boldsymbol{\theta}^\star_m = \arg\min_{\boldsymbol{\theta}_m} D[P_{\mathcal{N}}(\mathbf{X}) \| P(\mathbf{X}|m, \boldsymbol{\theta}_m)].$$

It turns out that, due to the presence of latent variables, the KL divergence is difficult to directly minimize. Therefore, we transform the problem into an asymptotically equivalent maximum likelihood estimation (MLE) problem:

1. Generate a data set $\mathcal{D}$ with N samples from $P_{\mathcal{N}}(\mathbf{X})$.
2. Find the MLE of $\boldsymbol{\theta}_m$ with respect to $\mathcal{D}$, i.e.,

$$\hat{\boldsymbol{\theta}}_m = \arg\max_{\boldsymbol{\theta}_m} P(\mathcal{D}|m, \boldsymbol{\theta}_m).$$

It is well known that $\hat{\boldsymbol{\theta}}_m$ converges almost surely to $\boldsymbol{\theta}^\star_m$ as the sample size N approaches infinity.

To implement this solution, we first need to generate $\mathcal{D}$ from $P_{\mathcal{N}}(\mathbf{X})$. This can be done by using logic sampling for BNs (Henrion 1988). Given $\mathcal{D}$, the next step is to find the MLE. Note that the values of latent variables $\mathbf{Y}$ are missing in $\mathcal{D}$. We thus use the EM algorithm (Dempster, Laird, & Rubin 1977). A practical issue is that EM can converge to local maxima on the likelihood surface, which could be poor approximations to $\boldsymbol{\theta}^\star_m$. In practice, one can use various techniques such as multiple restart (Chickering & Heckerman 1997) to alleviate this issue.

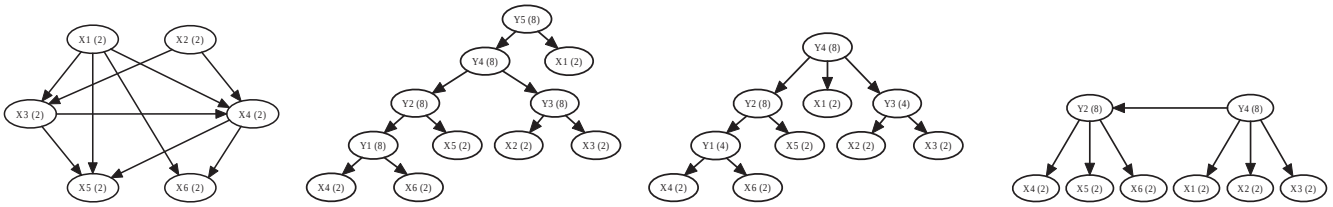
Note that EM takes a long time to converge, especially when the sample size N is large. This is why our algorithm has an expensive offline phase.

Exhaustive Search for the Optimal Model

We now consider the first subproblem, i.e., to find the best model $m^\star$. A straightforward way to solve this problem is to exhaust all possible models, find the optimal parameters $\boldsymbol{\theta}^\star_m$ for each model m, compute the KL divergence $D[P_{\mathcal{N}}(\mathbf{X}) \| P(\mathbf{X}|m, \boldsymbol{\theta}^\star_m)]$, and then return a model $m^\star$ with the minimum KL divergence. The problem with this solution is that the search space is super-exponentially large (Zhang 2004). Therefore, the exhaustive search is computationally infeasible. In the following three subsections, we will present a heuristic method.

Heuristic Construction of Model Structure

We first present a heuristic for determining the model structure. In an LTM, two manifest variables are called siblings if they share the same parent. Our heuristic is based on two ideas: (1) In an LTM $\mathcal{M}$, siblings are generally more closely correlated than variables that are located far apart; (2) If $\mathcal{M}$



(a) Bayesian network $\mathcal{N}$ (b) Latent tree model m (c) Regularized model m' (d) Simplified model m''

Figure 1: An illustrative example. The numbers within the parentheses are the cardinalities of the variables.

is a good approximation of $\mathcal{N}$, then two variables X_i and X_j are closely correlated in $\mathcal{M}$ if and only if they are closely correlated in $\mathcal{N}$. So we can examine each pair of variables in $\mathcal{N}$, pick the two variables that are most closely correlated, and introduce a latent variable as their parent in $\mathcal{M}$.

We measure the strength of correlation between a pair of variables X_i and X_j by the mutual information (Cover & Thomas 1991)

$$I(X_i; X_j) = \sum_{X_i, X_j} P_{\mathcal{N}}(X_i, X_j) \log \frac{P_{\mathcal{N}}(X_i, X_j)}{P_{\mathcal{N}}(X_i) P_{\mathcal{N}}(X_j)}.$$

To compute $I(X_i; X_j)$, one needs to make inference in $\mathcal{N}$. This could be computationally hard in the first place. So we use the sampling technique to address this issue. Specifically, we generate a data set $\mathcal{D}$ with N samples from the BN $\mathcal{N}$, and compute the empirical mutual information $\hat{I}(X_i; X_j)$ using the empirical distribution $\hat{P}(X_i, X_j)$ based on $\mathcal{D}$. By the strong law of large samples, $\hat{I}(X_i; X_j)$ converges almost surely to $I_{\mathcal{N}}(X_i; X_j)$ as N goes to infinity.

We now use the BN in Figure 1(a) as an example to illustrate the idea. It contains 6 binary variables X_1, X_2, ..., X_6. Suppose that the empirical mutual information based on some data set $\mathcal{D}$ is as presented at the top of Table 1. We find that X_4 and X_6 are the pair with the largest mutual information. Therefore, we create a latent variable Y_1 and make it the parent of X_4 and X_6.

	X_1	X_2	X_3	X_4	X_5
X_1	-	-	-	-	-
X_2	0.0000	-	-	-	-
X_3	0.0003	0.0971	-	-	-
X_4	0.0015	0.0654	0.0196	-	-
X_5	0.0017	0.0311	0.0086	0.1264	-
X_6	0.0102	0.0252	0.0080	0.1817	0.0486
Y_1	0.0102	0.0654	0.0196	-	0.1264

Table 1: Empirical mutual information

The next step is to find, among Y_1, X_1, X_2, X_3, and X_5, the pair of variables with the largest mutual information. There is one difficulty: Y_1 is not in the original BN and hence not observed in the data set. The mutual information between Y_1 and the other variables cannot be computed directly. We hence seek an approximation. In the final model, Y_1 would d-separate X_4 and X_6 from the other variables. Therefore, for any $X \in \{X_1, X_2, X_3, X_5\}$, we have $I(Y_1; X) \geq I(X_4; X)$ and $I(Y_1; X) \geq I(X_6; X)$. Therefore, we approximate $I(Y_1; X)$ using the lower bound

$$\max\{I(X_4; X), I(X_6; X)\}.$$

Back to our running example, the estimated mutual information between Y_1 and X_1, X_2, X_3, X_5 is presented at the bottom of Table 1. We see that the next pair to pick is Y_1 and X_5. We introduce a latent variable Y_2 as the parent of Y_1 and X_5. The process continues. The final model structure is a binary tree as shown in Figure 1(b).

Cardinalities of Latent Variables

After obtaining a model structure, the next step is to determine the cardinalities of the latent variables. We set the cardinalities of all the latent variables at a predetermined value C. In the following, we discuss how the choice of C influences the quality of approximation and inferential efficiency.

We first discuss the impact of the value of C on the approximation quality. We start by considering the case when C equals to the product of the cardinalities of all the manifest variables. In this case, each latent variable can be viewed as a joint variable of all the manifest variables. We can therefore set the parameters $\boldsymbol{\theta}_m$ so that $P(\mathbf{X}|m, \boldsymbol{\theta}_m) = P_{\mathcal{N}}(\mathbf{X})$. That is, m can capture all the interactions among the manifest variables.

What will happen if we decrease C? The following proposition answers this question.

Proposition 1 *Let m and m' be two models that share the same structure. Let the cardinalities of the latent variables in m and m' be C and C', respectively. If $C > C'$, then*

$$\min_{\boldsymbol{\theta}_m} D[P_{\mathcal{N}}(\mathbf{X}) \| P(\mathbf{X}|m, \boldsymbol{\theta}_m)] \leq \min_{\boldsymbol{\theta}_{m'}} D[P_{\mathcal{N}}(\mathbf{X}) \| P(\mathbf{X}|m', \boldsymbol{\theta}_{m'})].$$

As mentioned earlier, when C is large enough, model m can capture all the interactions among the manifest variables and hence can represent $P_{\mathcal{N}}(\mathbf{X})$ exactly. If C is not large enough, we can only represent $P_{\mathcal{N}}(\mathbf{X})$ approximately. According to the proposition, as C decreases, the approximation accuracy (in terms of KL divergence) will gradually degrade, indicating that model m can capture less and less interactions among the manifest variables. The worst case occurs when $C = 1$. In this case, all the interactions are lost. The approximation accuracy is the poorest.

The parameter C also determines the computational cost of making inference in m. We use the clique tree propagation (CTP) algorithm for inference. So we measure the cost

by the *inferential complexity*, which is defined to be the sum of the clique sizes in the clique tree of m. It is given by

$$(|\mathbf{X}| - 2) \cdot C^2 + \sum_{X \in \mathbf{X}} |X| \cdot C.$$

Note that $|\mathbf{X}|$ is the number of manifest variables, while $|X|$ is the cardinality of a manifest variable X. Therefore, one can control the inferential complexity by changing the value of C. The smaller the value of C, the lower the complexity.

In summary, one can achieve a tradeoff between the approximation quality and the inferential complexity of the resultant model m by tuning the parameter C.

Model Simplifications

Suppose that we have obtained a model m using the technique described in the previous two subsections. In this subsection, we will show that it is sometimes possible to simplify m without compromising the approximation quality.

We first notice that m could be irregular. Consider the model in Figure 1(b) as an example. It is constructed as an approximation to the BN $\mathcal{N}$ in Figure 1(a) with C=8. By checking the latent variables, we find that Y_1 violates the regularity condition because $|Y_1| > \frac{|X_4| \cdot |X_6| \cdot |Y_2|}{\max\{|X_4|, |X_6|, |Y_2|\}}$. It is the same case for Y_3 and Y_5. The following proposition suggests that irregular models should always be simplified until they become regular.

Proposition 2 *If m is an irregular model, then there must exists a model m' with lower inferential complexity such that*

$$\min_{\boldsymbol{\theta}_m} D[P_{\mathcal{N}}(\mathbf{X}) \| P(\mathbf{X}|m, \boldsymbol{\theta}_m)] = \min_{\boldsymbol{\theta}_{m'}} D[P_{\mathcal{N}}(\mathbf{X}) \| P(\mathbf{X}|m', \boldsymbol{\theta}_{m'})].$$

To regularize an irregular model, we identify all the latent variables that cause the irregularity and deal with them one by one. If the latent variable has only two neighbors, we remove it from the model and connect the two neighbors. Otherwise, we decrease its cardinality until it becomes saturated. For the aforementioned example model m, we need to simplify Y_1, Y_3, and Y_5. Y_1 and Y_3 both have three neighbors. So we decrease their cardinalities to $\frac{|X_4| \cdot |X_6| \cdot |Y_2|}{\max\{|X_4|, |X_6|, |Y_2|\}} = 4$ and $\frac{|X_2| \cdot |X_3| \cdot |Y_4|}{\max\{|X_2|, |X_3|, |Y_4|\}} = 4$, respectively. Y_5 has only two neighbors X_1 and Y_4. We thus remove it and connect X_1 and Y_4. The resultant regular model m' is shown in Figure 1(c).

After regularization, the model can be further simplified in some cases. The following proposition addresses this situation.

Proposition 3 *Let m be a model with two adjacent and saturated latent variables Y_1 and Y_2. Assume that Y_2 subsumes Y_1. Let m' be another model obtained by removing Y_1 from m and connecting Y_2 to the other neighbors of Y_1. Then m' has lower inferential complexity than m, and*

$$\min_{\boldsymbol{\theta}_m} D[P_{\mathcal{N}}(\mathbf{X}) \| P(\mathbf{X}|m, \boldsymbol{\theta}_m)] = \min_{\boldsymbol{\theta}_{m'}} D[P_{\mathcal{N}}(\mathbf{X}) \| P(\mathbf{X}|m', \boldsymbol{\theta}_{m'})].$$

Given a regularized model, we check each pair of adjacent latent variables and apply Proposition 3 to eliminate redundant latent variables. Take the model m' in Figure 1(c) as an example. There are two pairs of latent variables that satisfy the condition in the proposition: Y_1—Y_2 and Y_3—Y_4. All these variables are saturated. Because Y_2 subsumes Y_1, we remove Y_1 from the model, and connect Y_2 to X_4 and X_6. Similarly, we remove Y_3, which is subsumed by Y_4, and connect Y_4 to X_2 and X_3. The final model m'' is shown in Figure 1(d).

The Algorithm LTAB

To summarize, we outline the algorithm for approximating BNs using LTMs. We call the algorithm LTAB, a shorthand for **L**atent **T**ree **A**pproximation of **B**ayesian networks. It has three inputs: a BN $\mathcal{N}$, a predetermined cardinality C for latent variables, and a sample size N. The output of LTAB is an LTM that approximates $\mathcal{N}$. LTAB is briefly described as follows.

1. Generate a data set $\mathcal{D}$ of N samples from $P_{\mathcal{N}}(\mathbf{X})$.
2. Obtain an LTM structure by performing hierarchical clustering of manifest variables.
3. Set the cardinalities of the latent variables at C and simplify the model.
4. Optimize the parameters by running EM.
5. Return the resultant LTM.

LTM-based Approximate Inference

The focus of this paper is approximate inference in Bayesian networks. We propose the following two-phase method:

1. *Offline*: Given a BN $\mathcal{N}$, use LTAB to construct an approximation $\mathcal{M}$. The sample size N should be set as large as possible, while the cardinality C should be determined to meet the requirement on inferential complexity.
2. *Online*: Make inference in $\mathcal{M}$ instead of $\mathcal{N}$. More specifically, given evidence $\mathbf{E}$=$\mathbf{e}$ and a querying variable Q, return $P_{\mathcal{M}}(Q|\mathbf{E}{=}\mathbf{e})$ as an approximation to $P_{\mathcal{N}}(Q|\mathbf{E}{=}\mathbf{e})$.

Empirical Results

In this section, we empirically evaluate our approximate inference method. We first examine the impact of sample size N and cardinality C on the performance of our method. Then we compare our method with clique tree propagation (CTP), LBP, the CL-based method, and the LCM-based method.

We used eight networks in our experiments: ALARM, WIN95PTS, HAILFINDER, INSURANCE, CPCS54, WATER, MILDEW, BARLEY. They are available at http://www.cs.huji.ac.il/labs/compbio/Repository/. The last three networks WATER, MILDEW, and BARLEY are densely connected. Exact inference in them takes a long time.

For each network, we sampled 500 pieces of evidence on the leaf nodes according to the joint distribution. Then we used the CTP algorithm and the approximate inference methods to compute the posterior distribution of each non-leaf node conditioned on each piece of evidence. The accuracy of the approximate methods are measured by the average KL divergence between the exact and the approximate posterior distributions of the query variables.

All the algorithms were implemented in Java and run on a machine with an Intel Pentium IV 3.2GHz CPU.

Impact of N and C

We discussed the impact of N and C on the performance of our method in the previous two sections. This subsection empirically verifies the claims.

Three sample sizes were chosen in the experiments: $1k$, $10k$, and $100k$. For each network, we also chose a set of C. See the x-axes in the plots of Figure 2. LTMs were then learned using LTAB with different combination of the values of N and C. For parameter learning, we terminated EM either when the improvement in loglikelihoods is smaller than 0.1, or when the algorithm ran for two months. To avoid local maxima, we used multiple restart (Chickering & Heckerman 1997) with 16 starting points.

In general, the running time of LTAB increases with N and C, ranging from seconds to weeks. For several settings, EM failed to converge in two months. The details are reported in Figure 2. We emphasize that LTAB is executed offline and its running time should not be confused with the time for online inference, which will be reported next.

We used CTP to make inference in the learned LTMs. Figure 3 shows the approximation accuracy. There are four curves and two horizontal lines in each plot. For now, we only consider the three curves labeled as LTM. They are for our method with different sample size N. We first examine the impact of sample size by comparing the corresponding curves in each plot. We find that, in general, the curves for larger samples are located below those for smaller ones. This shows that the approximation accuracy increases with the sample size.

To see the impact of C, we examine each individual curve from left to right. According to our discussion, the curve is expected to drop monotonically as C increases. This is generally true for the results with $N{=}100k$. For $N{=}1k$ and $N{=}10k$, however, there are cases in which the approximation becomes poorer as C increases. See Figures 3(e) and 3(f). This phenomenon does not conflict with our claims. As C increases, the expressive power of the learned LTM increases. So it tends to overfit the data. On the other hand, the empirical distribution of a small data set may significantly deviate from the joint $P_{\mathcal{N}}(\mathbf{X})$. This also suggests that the sample size should be set as large as possible.

Finally, let us examine the impact of N and C on the inferential complexity. Figure 4 plots the running time for different methods to answer all the simulated queries. It can be seen that the three LTM curves overlap in all plots. This implies that the running time is independent of the sample size N. On the other hand, all the curves are monotonically increasing. This confirms our claim that the inferential complexity is positively dependent on C.

In the following, if not stated otherwise, we will only consider the results for $N{=}100k$ and the largest C. Under these settings, our method achieves the highest accuracy.

Comparison with CTP

We now compare our method with CTP, a state-of-the-art exact inference algorithm. The first concern is that how accurate our method is. By examining Figure 3, we argue that our method achieves good approximation accuracy: For HAILFINDER, CPCS54, WATER, the average KL divergence of our method is around or less than 10^{-3}; For the other networks, the average KL is around or less than 10^{-2}.

We next compare the inferential efficiency of our method and CTP. It can be seen from Figure 4 that our method is more efficient than the CTP algorithm. In particular, for the last five networks in which CTP spends a lot of time, our method is by two to three orders of magnitude faster.

To summarize, the results show that our method can achieve good approximation accuracy at low online cost.

Comparison with Other Approximate Methods

In this subsection, we compare our method with the following approximate methods.

- LBP: The termination condition is set as follows: (1) The change of every marginal distribution in two consecutive iterations is smaller than 10^{-4}, or (2) the number of iterations exceeds 100.
- CL-based method: For each network, a CL tree is learned from the $100k$ samples and used for inference.
- LCM-based method: For each LTM learned from the $100k$ samples, we also learn an LCM and use it for inference. The cardinality of the latent variable is determined such that the inferential complexity of the LCM is comparable with that of the LTM. The parameters are optimized using EM with the same setting as in the case of LTM.

The approximation accuracy and the running time of these methods are shown in Figures 3 and 4. In the following, we highlight some findings from the results.

- In terms of approximate accuracy, our method is competitive with LBP: our method significantly won in the 2 densest networks MILDEW and BARLEY; LBP significantly won in HAILFINDER and CPCS54; For the other networks, it was a tie.
- Our method is consistently more accurate than the CL-based method and the LCM-based method except for MILDEW. For MILDEW, our method is competitive with the CL-based method. Compared with the LCM-based method, our method is less accurate when C is small. But when C becomes large, it begins to win.
- In terms of running time, our method is faster than LBP by up to two orders of magnitude, but not as efficient as the CL-based method.

Based on these findings, we conclude that

- Our method is preferable to LBP in dense networks where exact inference is intractable.
- If the running time is very limited, the CL-based method will be a good choice; Otherwise, our method is more attractive because it can achieve more accurate results.
- LTMs are superior to LCMs when used for approximate inference.

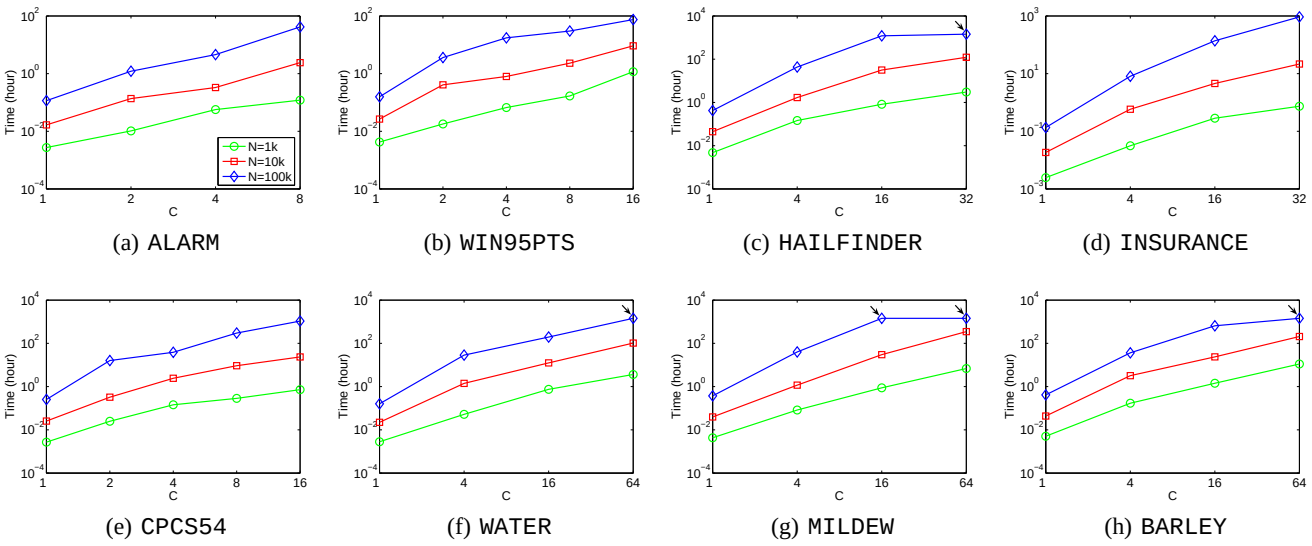


Figure 2: Running time of LTAB. Settings for which EM did not converge are indicated by arrows.

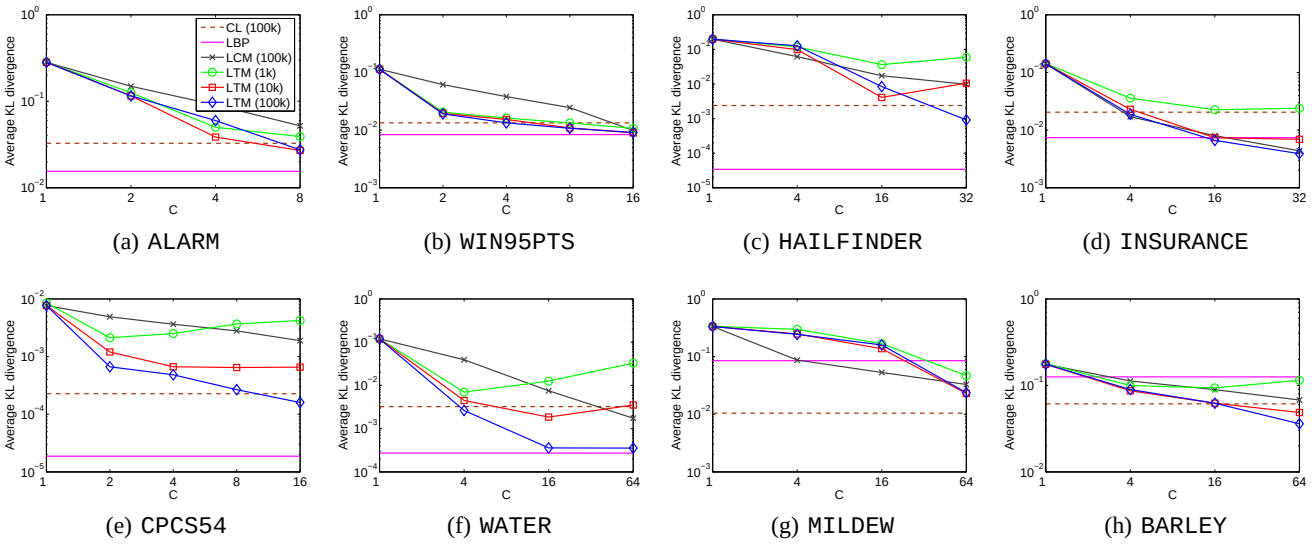


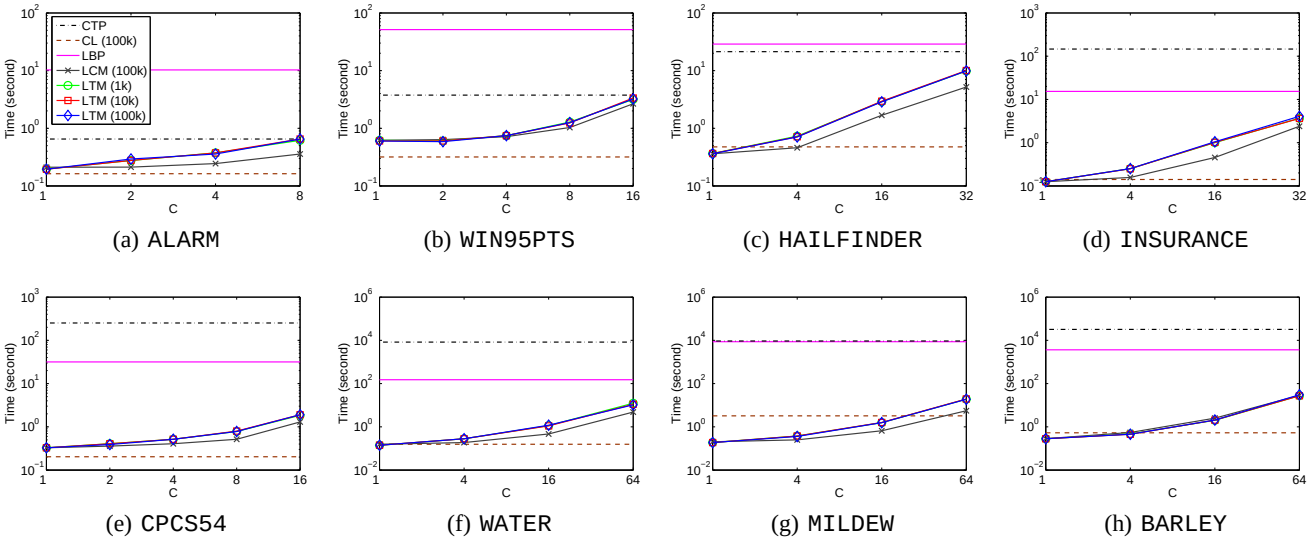
Figure 3: Approximation accuracy in terms of average KL divergence.

Related Work

In addition to the CL-based method (Chow & Liu 1968) and the LCM-based method (Lowd & Domingos 2005), this work is also related to variational methods. While we approximate the prior joint $P_{\mathcal{N}}(\mathbf{X})$ offline, variational methods assume that the evidence $\mathbf{E}{=}\mathbf{e}$ is known and build approximations to the posterior joint $P_{\mathcal{N}}(\mathbf{X}\backslash\mathbf{E}|\mathbf{E}{=}\mathbf{e})$ online. Examples include (Saul, Jaakkola, & Jordan 1996) which assumes that $\mathbf{X}\backslash\mathbf{E}$ are mutually independent, and (Bishop *et al.* 1997) which uses LCMs for approximation. These methods usually involve an iterative process for optimizing the variational parameters. Consequently, there is no guarantee on the online inference time. With our method, in contrast, one can determine the inferential complexity beforehand.

Concluding Remarks

We propose a novel scheme for BN approximate inference using LTMs. With our scheme one can trade off between the approximation accuracy and the inferential complexity. Our scheme achieves good accuracy at low online costs in all the networks that we examined. In particular, it is often more accurate and faster than LBP in dense networks. We also show that LTMs are superior to Chow-Liu trees and LCMs when used for approximate inference.



(a) ALARM (b) WIN95PTS (c) HAILFINDER (d) INSURANCE

(e) CPCS54 (f) WATER (g) MILDEW (h) BARLEY

Figure 4: Running time of the online inference.

References

Bishop, C.; Lawrence, N.; Jaakkola, T.; and Jordan, M. 1997. Approximating posterior distributions in belief networks using mixtures. In *Advances in Neural Information Processing Systems 10*, 416–422.

Chickering, D., and Heckerman, D. 1997. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning* 29:181–212.

Chow, C., and Liu, C. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* 14(3):462–467.

Cover, T., and Thomas, J. 1991. *Elements of Information Theory*. New York: Wiley-Interscience.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39(1):1–38.

Henrion, M. 1988. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Proceedings of the 4th Annual Conference on Uncertainty in Artificial Intelligence*, 317–324.

Lowd, D., and Domingos, P. 2005. Naive Bayes models for probability estimation. In *Proceedings of the 22nd International Conference on Machine Learning*, 529–536.

Murphy, K.; Weiss, Y.; and Jordan, M. 1999. Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, 467–475.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo: Morgan Kaufmann.

Saul, L.; Jaakkola, T.; and Jordan, M. 1996. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research* 4:61–76.

Zhang, N. 2004. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research* 5(6):697–723.