Hybrid Loopy Belief Propagation

Changhe Yuan
Department of Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
`cyuan@cse.msstate.edu`

Marek J. Druzdzel
Decision Systems Laboratory
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15260
`marek@sis.pitt.edu`

Abstract

We propose an algorithm called *Hybrid Loopy Belief Propagation* (HLBP), which extends the *Loopy Belief Propagation* (LBP) (Murphy et al., 1999) and *Nonparametric Belief Propagation* (NBP) (Sudderth et al., 2003) algorithms to deal with general hybrid Bayesian networks. The main idea is to represent the LBP messages with mixture of Gaussians and formulate their calculation as Monte Carlo integration problems. The new algorithm is general enough to deal with hybrid models that may represent linear or nonlinear equations and arbitrary probability distributions.

1 Introduction

Some real problems are more naturally modelled by hybrid Bayesian networks that contain mixtures of discrete and continuous variables. However, several factors make inference in hybrid models extremely hard. First, linear or nonlinear deterministic relations may exist in the models. Second, the models may contain arbitrary probability distributions. Third, the orderings among the discrete and continuous variables may be arbitrary. Since the general case is difficult, existing research often focuses on special instances of hybrid models, such as *Conditional Linear Gaussians* (CLG) (Lauritzen, 1992). However, one major assumption behind CLG is that discrete variables cannot have continuous parents. This limitation was later addressed by extending CLG with *logistic* and *softmax functions* (Lerner et al., 2001; Murphy, 1999). More recent research begin to develop methodologies for more general non-Gaussian models, such as *Mixture of Truncated Exponentials* (MTE) (Moral et al., 2001; Cobb and Shenoy, 2005), and *junction tree algorithm with approximate clique potentials* (Koller et al., 1999). However, most of these approaches rely on the junction tree algorithm (Lauritzen and Spiegelhalter, 1988). As Lerner et al. (Lerner et al., 2001) pointed out, it is important to have alternative solutions in case that junction tree algorithm-based methods are not feasible.

In this paper, we propose the *Hybrid Loopy Belief Propagation* algorithm, which extends the *Loopy Belief Propagation* (LBP) (Murphy et al., 1999) and *Nonparametric Belief Propagation* (NBP) (Sudderth et al., 2003) algorithms to deal with general hybrid Bayesian networks. The main idea is to represent LBP messages as *Mixtures of Gaussians* (MG) and formulate their calculation as Monte Carlo integration problems. The extension is far from trivial due to the enormous complexity brought by deterministic equations and mixtures of discrete and continuous variables. Another advantage of the algorithm is that it approximates the true posterior probability distributions, unlike most existing approaches which only produce their first two moments for CLG models.

2 Hybrid Loopy Belief Propagation

To extend LBP to hybrid Bayesian networks, we need to know how to calculate the LBP messages in hybrid models. There are two kinds of messages defined in LBP. Let X be a node in a hybrid model. Let Y_j be X's children and U_i

be X's parents. The $\lambda_X^{(t+1)}(u_i)$ message that X sends to its parent U_i is given by:

$$\lambda_X^{(t+1)}(u_i) =$$
$$\alpha \int_x \lambda_X(x) \prod_j \lambda_{Y_j}^{(t)}(x) \int_{u_k:k \neq i} P(x|u) \prod_{k \neq i} \pi_X^{(t)}(u_k) \quad (1)$$

and the message $\pi_{Y_j}^{(t+1)}(x)$ that X sends to its child Y_j is defined as:

$$\pi_{Y_j}^{(t+1)}(x) =$$
$$\alpha \lambda_X(x) \prod_{k \neq j} \lambda_{Y_k}^{(t)}(x) \int_u P(x|u) \prod_k \pi_X^{(t)}(u_k) \quad (2)$$

where $\lambda_X(x)$ is a message that X sends to itself representing evidence. For simplicity, I use only integrations in the message definitions. It is evident that no closed-form solutions exist for the messages in general hybrid models. However, we observe that their calculations can be formulated as Monte Carlo integration problems.

First, let us look at the $\pi_{Y_j}^{(t+1)}(x)$ message defined in Equation 2. We can rearrange the equation in the following way:

$$\pi_{Y_j}^{(t+1)}(x) =$$
$$\alpha \int_u \left\{ \overbrace{\lambda_X(x) \prod_{k \neq j} \lambda_{Y_k}^{(t)}(x) P(x|u) \prod_k \pi_X^{(t)}(u_k)}^{P(x,u)} \right\} .$$

Essentially, we put all the integrations outside of the other operations. Given the new formula, we realize that we have a joint probability distribution over x and u_is, and the task is to integrate all the u_is out and get the marginal probability distribution over x. Since $P(x, u)$ can be naturally decomposed into $P(x|u)P(u)$, the calculation of the message can be solved using a Monte Carlo sampling technique called *Composition method* (Tanner, 1993). The idea is to first draw samples for each u_is from $\pi_X^{(t)}(u_i)$, and then sample from the product of $\lambda_X(x) \prod_{k \neq j} \lambda_{Y_k}^{(t)}(x) P(x|u)$. We will discuss how to take the product in the next subsection. For now, let us assume that the computation is possible. To make life even easier, we make further

modifications and get

$$\pi_{Y_j}^{(t+1)}(x) =$$
$$\alpha \int_u \left\{ \frac{\lambda_X(x) \prod_k \lambda_{Y_k}^{(t)}(x) P(x|u) \prod_k \pi_X^{(t)}(u_k)}{\lambda_{Y_j}^{(t)}(x)} \right\} .$$

Now, for the messages sent from X to its different children, we can share most of the calculation. We first get samples for u_is, and then sample from the product of $\lambda_X(x) \prod_k \lambda_{Y_k}^{(t)}(x) P(x|u)$. For each different message $\pi_{Y_j}^{(t+1)}(x)$, we use the same sample x but assign it different weights $1/\lambda_{Y_i}^{(t)}(x)$.

Let us now consider how to calculate the $\lambda_X^{(t+1)}(u_i)$ message defined in Equation 1. First, we rearrange the equation analogously:

$$\lambda_X^{(t+1)}(u_i) = \alpha$$
$$\int_{x,u_k:k \neq i} \left\{ \overbrace{\lambda_X(x) \prod_j \lambda_{Y_j}^{(t)}(x) P(x|u) \prod_{k \neq i} \pi_X^{(t)}(u_k)}^{P(x,u_k:k \neq i|u_i)} \right\} (3)$$

It turns out that here we are facing a quite different problem from the calculation of $\pi_{Y_j}^{(t+1)}(x)$. Note that now we have $P(x, u_k : k \neq i|u_i)$, a joint distribution over x and $u_k(k \neq i)$ conditional on u_i, so the whole expression is only a likelihood function of u_i, which is not guaranteed to be integrable. As in (Sudderth et al., 2003; Koller et al., 1999), we choose to restrict our attention to densities and assume that the ranges of all continuous variables are bounded (maybe large). The assumption only solves part of the problem. Another difficulty is that composition method is no longer applicable here, because we have to draw samples for all parents u_i before we can decide $P(x|u)$. We note that for any fixed u_i, Equation 3 is an integration over x and $u_k, k \neq i$ and can be estimated using Monte Carlo methods. That means that we can estimate $\lambda_X^{(t+1)}(u_i)$ up to a constant for any value u_i, although we do not know how to sample from it. This is exactly the time when importance sampling becomes handy. We can estimate the message by drawing a set of importance samples as follows: sample u_i from a

chosen importance function, estimate $\lambda_X^{(t+1)}(u_i)$ using Monte Carlo integration, and assign the ratio between the estimated value and $I(u_i)$ as the weight for the sample. A simple choice for the importance function is the uniform distribution over the range of u_i, but we can improve the accuracy of Monte Carlo integration by choosing a more informed importance function, the corresponding message $\lambda_X^{(t)}(u_i)$ from the last iteration. Because of the iterative nature of LBP, messages usually keep improving over each iteration, so they are clearly better importance functions.

Note that the preceding discussion is quite general. The variables under consideration can be either discrete or continuous. We only need to know how to sample from or evaluate the conditional relations. Therefore, we can use any representation to model the case of discrete variables with continuous parents. For instance, we can use general softmax functions (Lerner et al., 2001) for the representation.

We now know how to use Monte Carlo integration methods to estimate the LBP messages, represented as sets of weighted samples. To complete the algorithm, we need to figure out how to propagate these messages. Belief propagation involves operations such as sampling, multiplication, and marginalization. Sampling from a message represented as a set of weighted samples may be easy to do; we can use resampling technique. However, multiplying two such messages is not straightforward. Therefore, we choose to use density estimation techniques to approximate each continuous message using a *mixture of Gaussians* (MG) (Sudderth et al., 2003). A K component mixture of Gaussian has the following form

$$M(x) = \sum_{i=1}^{K} w_i N(x; \mu_i, \sigma_i) , \qquad (4)$$

where $\sum_{i=1}^{K} w_i = 1$. MG has several nice properties. First, we can approximate any continuous distribution reasonably well using a MG. Second, it is closed under multiplication. Last, we can estimate a MG from a set of weighted

---

Algorithm: HLBP

1. Initialize the messages that evidence nodes send to themselves and their children as indicating messages with fixed values, and initialize all other messages to be uniform.

2. **while** (stopping criterion not satisfied)

   Recompute all the messages using Monte Carlo integration methods.

   Normalize discrete messages.

   Approximate all continuous messages using MGs.

   end while

3. Calculate $\lambda(x)$ and $\pi(x)$ messages for each variable.

4. Calculate the posterior probability distributions for all the variables by sampling from the product of $\lambda(x)$ and $\pi(x)$ messages.

Figure 1: The Hybrid Loopy Belief Propagation algorithm.

samples using a regularized version of *expectation maximization* (EM) (Dempster et al., 1977; Koller et al., 1999) algorithm.

Given the above discussion, we outline the final HLBP algorithm in Figure 1. We first initialize all the messages. Then, we iteratively recompute all the messages using the methods described above. In the end, we calculate the messages $\lambda(x)$ and $\pi(x)$ for each node X and use them to estimate the posterior probability distribution over X.

3 Product of Mixtures of Gaussians

One question that remains to be answered in the last section is how to sample from the product of $\lambda_X(x) \prod_{k \neq j} \lambda_{Y_k}^{(t)}(x) P(x|u)$. We address the problem in this section. If $P(x|u)$ is a continuous probability distribution, we can approximate $P(x|u)$ with a MG. Then, the problem becomes how to compute the product of several MGs. The approximation is often a reasonable thing to do because we can approximate any continuous probability distribution reasonably well using MG. Even when such approximation is poor, we can approximate the product of $P(x|u)$ with an MG using another MG. One example is that the product of Gaussian distribu-

tion and logistic function can be approximated well with another Gaussian distribution (Murphy, 1999).

Suppose we have messages $M_1, M_2, ..., M_K$, each represented as an MG. Sudderth et al. use Gibbs sampling algorithm to address the problem (Sudderth et al., 2003). The shortcoming of the Gibbs sampler is its efficiency: We usually have to carry out several iterations of Gibbs sampling in order to get one sample.

Here we propose a more efficient method based on the chain rule. If we treat the selection of one component from each MG message as a random variable, which we call a *label*, our goal is to draw a sample from the joint probability distribution of all the labels $P(L_1, L_2, ..., L_K)$. We note that the joint probability distribution of the labels of all the messages $P(L_1, L_2, ..., L_K)$ can be factorized using the chain rule as follows:

$$P(L_1, L_2, ..., L_K) = P(L_1) \prod_{i=2}^{K} P(L_i | L_1, ..., L_{i-1}) . \tag{5}$$

Therefore, the idea is to sample from the labels sequentially based on the prior or conditional probability distributions. Let w_{ij} be the weight for the jth component of ith message and μ_{ij} and σ_{ij} be the component's parameters. We can sample from the product of messages $M_1, ..., M_K$ using the algorithm presented in Figure 2. The main idea is to calculate the conditional probability distributions cumulatively. Due to the Gaussian densities, the method has to correct the bias introduced during the sampling by assigning the samples weights. The method only needs to go over the messages once to obtain one importance sample and is more efficient than the Gibbs sampler in (Sudderth et al., 2003). Empirical results show that the precision obtained by the importance sampler is comparable to the Gibbs sampler given a reasonable number of samples.

4 Belief Propagation with Evidence

Special care is needed for belief propagation with evidence and deterministic relations.

Algorithm: Sample from a product of MGs $M_1 \times M_2 \times ... \times M_K$.

1. Randomly pick a component, say j_1, form the first message M_1 according to its weights $w_{11}, ..., w_{1J_1}$.

2. Initialize cumulative parameters as follows
   $$\mu_1^* \leftarrow \mu_{1j_1}; \ \sigma_1^* \leftarrow \sigma_{1j_1}.$$

3. $i \leftarrow 2; wImportance \leftarrow w_{1j_1}.$

4. **while** $(i \leq K)$

   Compute new parameters for each component of ith message as follows
   $$\hat{\sigma}_{ik}^* \leftarrow ((\sigma_{i-1}^*)^{-1} + (\sigma_{ik})^{-1})^{-1},$$
   $$\hat{\mu}_{ik}^* \leftarrow (\frac{\mu_{ik}}{\sigma_{ik}} + \frac{\mu_{i-1}^*}{\sigma_{i-1}^*})\hat{\sigma}_{ik}^*.$$

   Compute new weights for ith message with any x
   $$\hat{w}_{ik}^* = w_{ik}\hat{w}_{i-1j_{i-1}}^* \frac{N(x;\mu_{i-1}^*,\sigma_{i-1}^*)N(x;\mu_{ik},\sigma_{ik})}{N(x;\hat{\mu}_{ik}^*,\hat{\sigma}_{ik}^*)}.$$

   Calculate the normalized weights $\bar{w}_{ik}^*$.

   Randomly pick a component, say j_i, from the ith message using the normalized weights.

   $\mu_i^* \leftarrow \hat{\mu}_{ij_i}^*; \ \sigma_i^* \leftarrow \hat{\sigma}_{ij_i}^*.$

   $i \leftarrow i + 1;$

   $wImportance = wImportance \times \bar{w}_{ij_i}^*.$

   end while

5. Sample from the Gaussian with mean μ_K^* and variance σ_K^*.

6. Assign the sample weight $\hat{w}_{Kj_K}^*/wImportance$.

Figure 2: Sample from a product of MGs.

In our preceding discussion, we approximate $P(x|u)$ using MG if $P(x|u)$ is a continuous probability distribution. It is not the case if $P(x|u)$ is deterministic or if X is observed. We discuss the following several scenarios separately.

Deterministic relation without evidence: We simply evaluate $P(x|u)$ to get the value x as a sample. Because we did not take into account the λ messages, we need to correct our bias using weights. For the message $\pi_{Y_i}(x)$ sent from X to its child Y_i, we take x as the sample and assign it weight $\lambda_X(x) \prod_{k \neq i} \lambda_{Y_k}^{(t)}(x)$. For the message $\lambda_X(u_i)$ sent from X to its parent U_i, we take value u_i as a sample for U_i and assign it weight $\lambda_X(x) \prod_k \lambda_{Y_k}^{(t)}(x)/\lambda_X^{(t)}(u_i)$.

Stochastic relation with evidence: The messages $\pi_{Y_j}(x)$ sent from evidence node X to its

children are always indicating messages with fixed values. The messages $\lambda_{Y_j}(x)$ sent from the children to X have no influence on X, so we need not calculate them. We only need to update the messages $\lambda_X(u_i)$, for which we take the value u_i as the sample and assign it weight $\lambda_X(e)\prod_k \lambda_{Y_k}^{(t)}(e)/\lambda_X^{(t)}(u_i)$, where e is the observed value of X.

Deterministic relation with evidence: This case is the most difficult. To illustrate this case more clearly, we use a simple hybrid Bayesian network with one discrete node A and two continuous nodes B and C (see Figure 3).
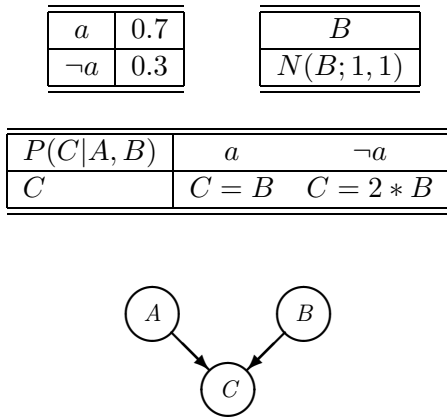
a	0.7
$\neg a$	0.3

B
$N(B;1,1)$

$P(C|A,B)$	a	$\neg a$
C	$C = B$	$C = 2*B$

Figure 3: A simple hybrid Bayesian network.

Example 1. Let C be observed at state 2.0. Given the evidence, there are only two possible values for B: 2.0 when $A = a$ and 1.0 when $A = \neg a$. We need to calculate messages $\lambda_C(a)$ and $\lambda_C(b)$. If we follow the routine and sample from A and B first, it is extremely unlikely for us to hit a feasible sample; Almost all the samples that we get would have weight 0.0. Clearly we need a better way to do that.

First, let us consider how to calculate the message $\lambda_C(a)$ sent from C to A. Suppose we choose uniform distribution as the importance function, we first randomly pick a state for A. After the state of A is fixed, we note that we can solve $P(C|A,B)$ to get the state for B. Therefore, we need not sample for B from $\pi_C(b)$, in this case $N(B;1,1)$. We then assign $N(B;1,1)$ as weight for the sample. Since A's two states are equally likely, the message $\lambda_C(A)$ would be

proportional $\{N(2;1,1),N(1;1,1)\}$.

For the message $\lambda_C(b)$ message sent from C to B, since we know that B can only take two values, we choose a distribution that puts equal probabilities on these two values as the importance function, from which we sample for B. The state of B will also determine A as follows: when $B = 2$, we have $A = a$; when $B = 1$, we have $A = \neg a$. We then assign weight 0.7 to the sample if $B = 2$ and 0.3 if $B = 1$. However, the magic of knowing feasible values for B is impossible in practice. Instead, we first sample for A from $\pi_C(A)$, in this case, $\{0.7, 0.3\}$, given which we can solve $P(C|A,B)$ for B and assign each sample weight 1.0. So $\lambda_C(b)$ have probabilities proportional to $\{0.7, 0.3\}$ on two values 2.0 and 1.0.

In general, in order to calculate λ messages sent out from a deterministic node with evidence, we need to sample from all parents except one, and then solve $P(x|u)$ for that parent. There are several issues here. First, since we want to use the values of other parents to solve for the chosen parent, we need an equation solver. We used an implementation of the *Newton's method for solving nonlinear set of equations* (Kelley, 2003). However, not all equations are solvable by this equation solver or any equation solver for that matter. We may want to choose the parent that is easiest to solve. This can be tested by means of a preprocessing step. In more difficult cases, we have to resort to users' help and ask at the model building stage for specifying which parent to solve or even manually specify the inverse functions. When there are multiple choices, one heuristic that we find helpful is to choose the continuous parent with the largest variance.

5 Lazy LBP

We can see that HLBP involves repeated density estimation and Monte Carlo integration, which are both computationally intense. Efficiency naturally becomes a concern for the algorithm. To improve its efficiency, we propose a technique called *Lazy LBP*, which is also applicable to other extensions of LBP. The tech-

nique serves as a summarization that contains both my original findings and some commonly used optimization methods.

After evidence is introduced in the network, we can pre-propagate the evidence to reduce computation in HLBP. First, we can plug in any evidence to the conditional relations of its children, so we need not calculate the π messages from the evidence to its children. We need not calculate the λ messages from its children to the evidence either. Secondly, evidence may determine the value of its neighbors because of deterministic relations, in which case we can evaluate the deterministic relations in advance so that we need not calculate messages between them.

Furthermore, from the definitions of the LBP messages, we can easily see that we do not have to recompute the messages all the time. For example, the $\lambda(x)$ messages from the children of a node with no evidence as descendant are always uniform. Also, a message needs not be updated if the sender of the message has received no new messages from neighbors other than the recipient.

Based on Equation 1, λ messages should be updated if incoming π messages change. However, we have the following result.

Theorem 1. *The λ messages sent from a non-evidence node to its parents remain uniform before it receives any non-uniform messages from its children, even though there are new π messages coming from the parents.*

Proof. Since there are no non-uniform λ messages coming in, Equation 1 simplifies to

$$
\begin{aligned}
\lambda_X^{(t+1)}(u_i) &= \alpha \int\limits_{x,u_k:k\neq i} P(x|u)\prod_{k\neq i}\pi_X^{(t)}(u_k) \\
&= \alpha \int\limits_{x,u_k:k\neq i} P(x,u_k:k\neq i|u_i) \\
&= \alpha .
\end{aligned}
$$

Finally for HLBP, we may be able to calculate some messages exactly. For example, suppose a discrete node has only discrete parents. We can always calculate the messages sent from this node to its neighbors exactly. In this case, we should avoid using Monte Carlo sampling.

6 Experimental Results

We tested the HLBP algorithm on two benchmark hybrid Bayesian networks: emission network (Lauritzen, 1992) and its extension augmented emission network (Lerner et al., 2001) shown in Figure 4(a). Note that HLBP is applicable to more general hybrid models; We choose the networks only for comparison purpose. To evaluate how well HLBP performs, we discretized the ranges of continuous variables to 50 intervals and then calculated the Hellinger's distance (Kokolakis and Nanopoulos, 2001) between the results of HLBP and the exact solutions obtained by a massive amount of computation (likelihood weighting with 100M samples) as the error for HLBP. All results reported are the average of 50 runs of the experiments.

6.1 Parameter Selection

HLBP has several tunable parameters. We have number of samples for estimating messages (number of message samples), number of samples for the integration (number of integration samples) in Equation 3. The most dramatic influence on precision comes from the number of message samples, shown as in Figure 4(b). Counter intuitively, the number of integration samples does not have as big impact as we might think (see Figure 4(c) with 1K message samples). The reason we believe is that when we draw a lot of samples for messages, the precision of each sample becomes less critical. In our experiments, we set the number of message samples to $1,000$ and the number of integration samples to 12. For the EM algorithm for estimating MGs, we typically set the regularization constant for preventing over fitting to 0.8, stopping likelihood threshold to 0.001, and the number of components in mixtures of Gaussian to 2.

We also compared the performance of two samplers for product of MGs: Gibbs sampler (Sudderth et al., 2003) and the importance sampler in Section 3. As we can see from Figure 4(b,c), when the number of message samples is small, Gibbs sampler has slight advantage over the importance sampler. As the num-
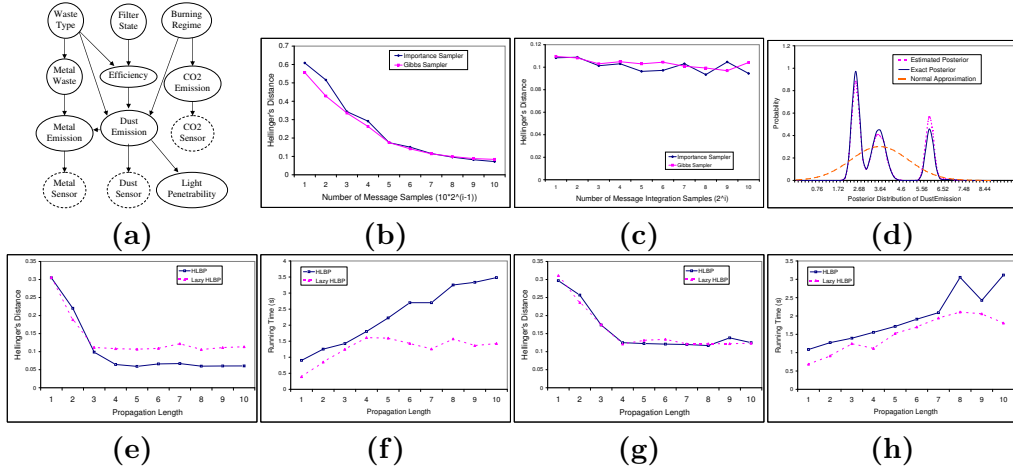
Figure 4: **(a)** Emission network (without dashed nodes) and augmented emission network (with dashed nodes). **(b,c)** The influence of number of message samples and number of message integration samples on the precision of HLBP on augmented Emission network CO2Sensor and DustSensor both observed to be true and Penetrability to be 0.5. **(d)** Posterior probability distribution of DustEmission when observing CO2Emission to be -1.3, Penetrability to be 0.5 and WasteType to be 1 in Emission network. **(e,f,g,h)** Results of HLBP and Lazy HLBP: (e) error on emission, (f) running time on emission, (g) error on augmented emission, (h) running time on augmented emission.

ber of message samples increases, the difference becomes negligible. Since the importance sampler is much more efficient, we use it in all our other experiments.

6.2 Results on Emission Networks

We note that mean and variance alone provide only limited information about the actual posterior probability distributions. Figure 4(d) shows the posterior probability distribution of node DustEmission when observing CO2Emission at -1.3, Penetrability at 0.5, and WasteType at 1. We also plot in the same figure the corresponding normal approximation with mean 3.77 and variance 1.74. We see that the normal approximation does not reflect the true posterior. While the actual posterior distribution has a multimodal shape, the normal approximation does not tell us where the mass really is. We also report the estimated posterior probability distribution of DustEmission by HLBP. HLBP seemed able to estimate the shape of the actual distribution very accurately.

In Figures 4(e,f), we plot the error curves of HLBP and Lazy HLBP (HLBP enhanced by Lazy LBP) as a function of the propagation length. We can see that HLBP needs only several steps to converge. Furthermore, HLBP achieves better precision than its lazy version, but Lazy HLBP is much more efficient than HLBP. Theoretically, Lazy LBP should not affect the results of HLBP but only improve its efficiency if the messages are calcualted exactly. However, we use importance sampling to estimate the messages. Since we use the messages from the last iteration as the importance functions, iterations will help improving the functions.

We also tested the HLBP algorithm on the augmented Emission network (Lerner et al., 2001) with CO2Sensor and DustSensor both observed to be true and Penetrability to be 0.5 and report the results in Figures 4(g,h). We again observed that not many iterations are needed for HLBP to converge. In this case, Lazy HLBP provides comparable results while improving the efficiency of HLBP.

7 Conclusion

The contribution of this paper is two-fold. First, we propose the *Hybrid Loopy Belief Propagation* algorithm (HLBP). The algorithm is general enough to deal with general hybrid Bayesian networks that contain mixtures of discrete and continuous variables and may represent linear or nonlinear equations and arbitrary probability distributions and naturally accommodate the scenario where discrete variables have continuous parents. Its another advantage is that it approximates the true posterior distributions. Second, we propose an importance sampler to sample from the product of MGs. Its accuracy is comparable to the Gibbs sampler in (Sudderth et al., 2003) but much more efficient given the same number of samples. We anticipate that, just as LBP, HLBP will work well for many practical models and can serve as a promising approximate method for hybrid Bayesian networks.

Acknowledgements

References

B. R. Cobb and P. P. Shenoy. 2005. Hybrid Bayesian networks with linear deterministic variables. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI–05)*, pages 136–144, AUAI Press Corvallis, Oregon.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society*, B39:1–38.

C. T. Kelley. 2003. *Solving Nonlinear Equations with Newton's Method.*

G. Kokolakis and P.H. Nanopoulos. 2001. Bayesian multivariate micro-aggregation under the Hellinger's distance criterion. *Research in official statistics*, 4(1):117–126.

D. Koller, U. Lerner, and D. Angelov. 1999. A general algorithm for approximate inference and its application to hybrid Bayes nets. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 324–333. Morgan Kaufmann Publishers Inc.

S. L. Lauritzen and D. J. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B (Methodological)*, 50(2):157–224.

S.L. Lauritzen. 1992. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108.

U. Lerner, E. Segal, and D. Koller. 2001. Exact inference in networks with discrete children of continuous parents. In *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 319–328. Morgan Kaufmann Publishers Inc.

S. Moral, R. Rumi, and A. Salmeron. 2001. Mixtures of truncated exponentials in hybrid Bayesian networks. In *Proceedings of Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-01)*, pages 156–167.

K. Murphy, Y. Weiss, and M. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–99)*, pages 467–475, San Francisco, CA. Morgan Kaufmann Publishers.

K. P. Murphy. 1999. A variational approximation for Bayesian networks with discrete and continuous latent variables. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 457–466. Morgan Kaufmann Publishers Inc.

E. Sudderth, A. Ihler, W. Freeman, and A. Willsky. 2003. Nonparametric belief propagation. In *Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR-03)*.

M. Tanner. 1993. *Tools for Statistical Inference: Methods for Exploration of Posterior Distributions and Likelihood Functions.* Springer, New York.