Belief Updating in MSBNs without Repeated Local Propagations

Y. Xiang
Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2, yxiang@cs.uregina.ca

April 16, 1998

Abstract

We redefine inference operations for multiply sectioned Bayesian networks (MSBNs). When two adjacent subnets exchange belief, previous operations require repeated belief propagations within the receiving subnet. The new operations require such propagation only twice. We prove that the new operations do not compromise the coherence while improving the efficiency.

A MSBN must be initialized before inference can take place. The initialization involves special operations not shared by inference computation. We show that the new inference operations unify inference and initialization. Therefore, the new operations not only are more efficient, but also are simpler. They speed up inference as well as ease practical implementation.

1 Introduction

Multiply sectioned Bayesian networks (MSBNs) provides a coherent framework for probabilistic inference in a large domain [8]. It can be applied under a single agent paradigm [7] or a cooperative multi-agent paradigm [5]. It supports object-oriented probabilistic inference [2].

We focus on the inference computation of MSBNs. Global consistency among multiple subnets in a MSBN can be achieved by communication. During communication, each subnet exchanges belief twice with each adjacent subnet in a half-duplex fashion. Each exchange requires repeated belief propagations within the receiving subnet according to previously proposed inference operations.

In this work, we redefine these operations such that each exchange of belief requires belief propagation in the receiving subnet only twice. We prove that the new operations do not compromise the coherence while improving the efficiency.

A MSBN must be initialized before evidential inference can take place. The initialization involves several special operations not shared by inference computation. In this work, we show that the newly proposed inference operations unify inference and initialization. Therefore, the new operations not only are more efficient, but also are simpler. They allow faster run time inference as well as simplify the practical implementation.

We review the basics of MSBNs in Section 2. In Section 3, we establish the syntactic and semantic properties of linkage trees, the interface between subnets, which has not been treated formally before. In Section 4, we redefine the messages to be passed between subnets. The inference operations are redefined in Section 5 based on the new form of messages, and their coherence are proven. We discuss the efficience gain through the new operations in Section 6, and discuss the unification of inference and initialization in Section 7.

2 Overview of MSBNs

A BN S is a triplet (N, D, P) where N is a set of domain variables, D is a DAG whose nodes are labeled by elements of N, and P is a joint probability distribution (jpd) over N. A MSBN M is a collection of Bayesian subnets that together defines a BN. These subnets should satisfy certain conditions to permit coherent distributed inference. One condition requires that nodes shared by two subnets form a *d-sepset*, as defined below.

Let $G_i = (N_i, E_i)$ $(i = 0, 1)$ be two graphs. The graph $G = (N_0 \cup N_1, E_0 \cup E_1)$ is referred to as the *union* of G_0 and G_1, denoted by $G = G_0 \sqcup G_1$.

Definition 1 *Let $D_i = (N_i, E_i)$ $(i = 0, 1)$ be two DAGs such that $D = D_0 \sqcup D_1$ is a DAG. The intersection $I = N_0 \cap N_1$ is a* `d-sepset` *between D_0 and D_1 if for every $x \in I$ with its parents π in D, either $\pi \subseteq N_0$ or $\pi \subseteq N_1$. Each $x \in I$ is called a* `d-sepnode`.

Just as the structure of a BN is a DAG, the structure of a MSBN is a multiply sectioned DAG (MSDAG) with a hypertree organization:

Definition 2 *A* `hypertree MSDAG` *$\mathcal{D} = \bigsqcup_i D_i$, where each D_i is a connected DAG, is a connected DAG constructible by the following procedure:*

Start with an empty graph (no node). Recursively add a DAG D_k, called a `hypernode`*, to the existing MSDAG $\bigsqcup_{i=0}^{k-1} D_i$ subject to the constraints:*
[d-sepset] For each D_j $(j < k)$, $I_{jk} = N_j \cap N_k$ is a d-sepset when the two DAGs are isolated.
[local covering] There exists D_i $(i < k)$ such that, for each D_j $(j < k; j \neq i)$, we have $I_{jk} \subseteq N_i$. For an arbitrarily chosen such D_i, I_{ik} is the `hyperlink` *between D_i and D_k which are said to be* `adjacent`.

A MSBN is then defined as follows:

Definition 3 *A MSBN M is a triplet $M = (\mathcal{N}, \mathcal{D}, \mathcal{P})$. $\mathcal{N} = \bigcup_i N_i$ is the* `total universe` *where each N_i is a set of variables. $\mathcal{D} = \bigsqcup_i D_i$ (a hypertree MSDAG) is the* `structure` *where nodes of each DAG D_i are labeled by elements of N_i. $\mathcal{P} = \prod_i P_{D_i}(N_i) / \prod_k P_{I_k}(I_k)$ is the* `jpd`. *Each $P_{D_i}(N_i)$ is a distribution over N_i such that whenever D_i and D_j are adjacent in $\mathcal{D}$, the marginalizations of $P_{D_i}(N_i)$ and $P_{D_j}(N_j)$ onto their d-sepset are identical. Each $P_{I_k}(I_k)$ is such a marginal distribution over a hyperlink I_k of $\mathcal{D}$. Each triplet $S_i = (N_i, D_i, P_i)$ is called a* `subnet` *of M. S_i and S_j are* `adjacent` *if D_i and D_j are adjacent.*

Inference in a MSBN can be performed more effectively on a compiled representation, called linked junction forest (LJF) of belief universes (LJFBU). Each D_i is converted into a junction tree (JT) T_i over N_i [1]. A *belief table* is a non-normalized probability distribution. A belief table $B_{T_i}(N_i)$ associated with T_i is defined as follows:

Definition 4 *Let T be a JT over a set N of variables. The belief table of T, denoted by $B_T(N)$, is defined as $B_T(N) = \prod_C B_C(C) / \prod_S B_S(S)$ where each C is a clique in T, each S is a sepset in T, $B_C(C)$ is a belief table over C and $B_S(S)$ is a belief table over S.*

A triplet $\mathcal{T}_i = (N_i, T_i, B_{T_i}(N_i))$ is called a junction tree of belief universes (JTBU) [1]. Proposition 5 states the semantics of a JTBU and is needed later. See [3] for the definition of I-maps and [6] for JTs as I-maps.

Proposition 5 *Let $P(N)$ be a probability distribution over N. Let a JT T over N be an I-map of P. Then $B_T(N)$ is equivalent to $P(N)$ if for each clique and each sepset in T, the corresponding belief table is equivalent to the marginalization of $P(N)$ over the corresponding subset of variables.*

A LJFBU has the same hypertree organization as its deriving MSBN. Each hypernode is a JTBU converted from its deriving subnet. Each hyperlink includes a linkage tree converted from its deriving d-sepset:

Definition 6 *[5] Let I be the d-sepset between JTs T_a and T_b in a LJF. A* `linkage tree` *L of T_a with respect to T_b is constructed as follows:*

First perform the following recursively:
(1) Remove each leaf clique C of T_a if $C \cap I = \phi$.
(2) Remove each leaf clique C if $C \cap I$ is a subset of its adjacent clique.

Then repeat the following until no variable can be removed:
(3) Remove a variable $x \notin I$ if x is contained in a single clique C.
(4) If C becomes a subset of an adjacent clique D after (3), union C into D.

Each clique l in L is a `linkage`*. Define a clique in T_a that contains l as its* `linkage host` *and break ties arbitrarily.*

A triplet $\mathcal{L}_i = (I, L, B_L(I))$ is called a linkage tree of belief universes (LTBU), where $B_L(I)$ is a belief table associated with L. A LJFBU is then defined as:

Definition 7 *Let M be a MSBN. A LJFBU F derived from M is a triplet $F = (\mathcal{T}, \mathcal{L}, \mathcal{P}')$. $\mathcal{T}$ is a set of JTBUs each of which is derived from a subnet in M. The JTBUs are organized into a hypertree isomorphic to the hypertree MSDAG of M. $\mathcal{L}$ is a set of LTBUs each of which is derived from a pair of adjacent JTBUs in the hypertree. $\mathcal{P}' = \prod_i P_{T_i}(N_i) / \prod_k P_{L_k}(I_k)$ is the* `joint system belief` *(JSB). Each $P_{T_i}(N_i)$ is the belief table of a JTBU and each $P_{L_k}(I_k)$ is the belief table of a LTBU.*

The structure of a LJFBU is a LJF consisting of its JTs and linkage trees.

3 Properties of linkage trees

In this section, we formally establish the syntactic and semantic properties of linkage trees. A linkage tree is a JT as shown in Proposition 8:

Proposition 8 *A linkage tree constructed according to Def.6 is a junction tree.*

Proof:

After removal of leaf cliques, the remaining subgraph of a JT is still a JT. Hence the subgraph obtained after steps (1) and (2) in Def. 6 is a JT.

After removal of a variable contained in a single clique C of a JT, the resultant graph is still a JT. If such removal renders C a subset of an adjacent clique D, then union of C into D neither changes any sepset between C and its neighbor cliques (other than D), nor changes any sepset between D and its neighbor cliques (other than C). Hence the graph obtained after steps (3) and (4) is a JT. $\Box$

Furthermore, the linkage tree preserves the I-mapness as shown in Proposition 9:

Proposition 9 *Let L be a linkage tree between a pair of JTs in a LJF and I be the d-sepset. Then L is an I-map over I with respect to the distribution of either JT.*

Proof:

Let T be one of the JTs. We show that the graphical separation between variables in I portrayed by T is unchanged during construction of L from T.

In steps (1) and (3) of Def. 6, the removal of C and x, respectively, is irrelevant to the graphical separation among elements of I.

In step (2), if $C \cap I$ is a subset of its adjacent clique, then removal of C does not change the fact that $C \cap I$ is contained in a clique. Similarly in step (4), union of C into D still leaves C contained in a clique. Thus removal (union) of C does not alter the graphical separation among elements of I. $\Box$

Definition 7 does not specify how a belief table for a linkage tree is defined. It can be specified as follows:

Definition 10 *Let $(N, T, B_T(N))$ be a JTBU and $I \subset N$ be its d-sepset with another JTBU. Let L be a linkage tree over I obtained from T. For each linkage l in L of host C in T, define its belief table $B_l(l) = \sum_{C \setminus l} B_C(C)$. For each sepset q in L, define its belief table $B_q(q) = \sum_{l \setminus q} B_l(l)$, where l is any one of the two linkages whose sepset is q. Then the belief table of L is $B_L(I) = \prod_l B_l(l) / \prod_q B_q(q)$.*

The semantics of a LTBU is established by Proposition 11. A JTBU is *internally consistent* if $\sum_{C \setminus S} B_C(C)$, $\sum_{Q \setminus S} B_Q(Q)$ and $B_S(S)$ are equivalent for every adjacent cliques C and Q with sepset S.

Proposition 11 *Let $(N, T, B_T(N))$ be an internally consistent JTBU and $(I, L, B_L(I))$ be a LTBU obtained from $(N, T, B_T(N))$. Then $B_L(I)$ is a marginalization of $B_T(N)$.*

Proof:

By Proposition 8, L is a JT. By Proposition 9, L is an I-map over I. From Proposition 5, the result follows. $\Box$

4 Extending linkage belief

In this section, we extend the linkage belief defined in Def. 10 such that more efficient inference can be supported. First, we introduce the peer sepset of a linkage:

Definition 12 *Let L be a linkage tree between a pair of JTs in a LJF. For each node l in L, assign an adjacent sepset q as the* `peer` *sepset of l such that every sepset is assigned to a node and every node except one has a peer.*

The following algorithm assigns peer sepsets to linkages:

Algorithm 1
while L has more than one node, do
  select a leaf node l in L;
  assign sepset between l and its adjacent node as the peer of l;
  remove l from L;

The effect of the algorithm is established as follows:

Proposition 13 *For any linkage tree L, the peer sepsets of linkages are well defined by Algorithm 1.*

Proof:
If L has a single node/linkage l, no peer is to be defined for l by definition.
Assume that L has $k > 1$ linkages. Since L is a tree, it has at least one leaf node. Hence the while loop will iterate at least once with the peer of one leaf defined and with the leaf and the link/sepset incident to it removed. Since L is still a tree after the removal of a leaf and a tree of k nodes has $k-1$ links, the loop will iterate exactly $k-1$ times. □

We now extend the linkage belief defined in Def. 10:

Definition 14 *Let L be a linkage tree with linkage and sepset belief defined as Def. 10, and linkage peers defined as Def. 12. For each node l in L with peer q, the* `extended linkage belief` *is $B_l^*(l) = B_l(l)/B_q(q)$, and for the node l without peer, define $B_l^*(l) = B_l(l)$.*

The semantics of extended linkage belief is shown in Proposition 15. The proof is trivial.

Proposition 15 *Let L be a linkage tree. Then $B_L(I)$, as defined in Def. 10, can be expressed in terms of extended linkage belief as $B_L(I) = \prod_l B_l^*(l)$, where each l is a linkage in L.*

We shall see that if we use extended linkage belief tables as messages passed between JTBUs during inference, the computation can be performed more efficiently. Hence we assume explicit storage of extended linkage belief $B_l^*(l)$, while $B_l(l)$ will only be used as a conceptual object in our analysis.

5 Inference operations

In this section, we redefine all inference operations in [8, 5] based on extended linkage belief. First, we redefine the operation AbsorbThroughLinkage. The objective of the operation is to bring the two linkage hosts involved into a form of consistency.

Operation 16 (AbsorbThroughLinkage) *Let l be a linkage in a linkage tree L between JTBUs $\mathcal{T}_a$ and $\mathcal{T}_b$. Let C_a and C_b be the corresponding linkage host of l in T_a and T_b. Let $B_l^*(l)$ be the extended linkage belief associated with l, and $B_{C_b}^*(l)$ be the extended linkage belief on l defined in C_b.*

When AbsorbThroughLinkage is called on C_a to absorb from C_b through l, perform the following:
(1) Updating host belief: $B'_{C_a}(C_a) = B_{C_a}(C_a) * B_{C_b}^*(l)/B_l^*(l)$.
(2) Updating linkage belief: $B_l^{*'}(l) = B_{C_b}^*(l)$.

Due to the use of extended linkage belief, the normal concept of consistency as used in [8] does not apply any more. We extend it to define the concept of e-consistency:

Definition 17 *Let l be a linkage between JTBUs $\mathcal{T}_a$ and $\mathcal{T}_b$. Let C_a be the linkage host of l in T_a. C_a and l are said to be* `e-consistent` *if $\sum_{C\setminus l} B_{C_a}(C_a) = B_l(l)$.*

Note that $B_l(l)$ is not the belief table associated with l. Instead, $B_l^*(l)$ is. We show several properties of AbsorbThroughLinkage:

Proposition 18 *After AbsorbThroughLinkage is performed, the following hold:*
(1) The joint system belief is invariant.
(2) C_b and l are e-consistent.
(3) If C_a and l were e-consistent before AbsorbThroughLinkage is performed, then C_a and l are also e-consistent after.

Proof:
(1) Denote the JSB by $B_F(\mathcal{N})$. After AbsorbThroughLinkage, the new JSB is

$$\begin{aligned} B'_F(\mathcal{N}) &= B_F(\mathcal{N}) * [B'_{C_a}(C_a)/B_{C_a}(C_a)]/[B_l^{*'}(l)/B_l^*(l)] \\ &= B_F(\mathcal{N}) * B'_{C_a}(C_a) * B_l^*(l)/[B_{C_a}(C_a) * B_l^{*'}(l)] \\ &= B_F(\mathcal{N}) * \frac{[B_{C_a}(C_a) * B_{C_b}^*(l)/B_l^*(l)] * B_l^*(l)}{B_{C_a}(C_a) * B_{C_b}^*(l)} = B_F(\mathcal{N}). \end{aligned}$$

(2) This is true from the definition of AbsorbThroughLinkage.
(3) After the operation, we have

$$\begin{aligned} \sum_{C\setminus l} B'_{C_a}(C_a) &= \sum_{C\setminus l} B_{C_a}(C_a) * B_{C_b}^*(l)/B_l^*(l) \quad \text{(definition of AbsorbThroughLinkage)} \\ &= [B_{C_b}^*(l)/B_l^*(l)] * \sum_{C\setminus l} B_{C_a}(C_a) \quad \text{(Proposition 4.1 [1])} \\ &= [B_{C_b}^*(l)/B_l^*(l)] * B_l(l) \quad \text{(e-consistency assumption)} \\ &= B_{C_b}(l) = B'_l(l) \end{aligned}$$

□

The operation UpdateBelief brings adjacent JTBUs to consistency. It uses the operation UnifyBelief [8] which brings a JTBU internally consistent by two local propagations.

Operation 19 (UpdateBelief) *Let T_a and T_b be adjacent JTBUs, and L be the linkage tree between them. When* `UpdateBelief` *is called on T_a relative to T_b, perform the following:*
(1) For each linkage l in L, call the host of l in T_a to perform AbsorbThroughLinkage.
(2) Perform UnifyBelief at T_a.

The effects of UpdateBelief are shown as follows:

Proposition 20 *Let T_a and T_b be locally consistent JTBUs of a LJFBU F. After UpdateBelief is performed in T_a relative to T_b, the following hold:*
(1) T_a is internally consistent.
(2) The joint system belief of F is invariant.
(3) L is consistent with T_b.
(4) If T_a and L were consistent before UpdateBelief, they are also consistent after.

Proof:

(1) This holds due to UnifyBelief at the end of UpdateBelief.

(2) It holds since neither AbsorbThroughLinkage nor UnifyBelief changes the joint system belief.

(3) It is implied by Propositions 15 and 18 (2).

(4) It follows from Proposition 18 (3). □

CollectBelief recursively propagates belief inwards on the hypertree of a LJFBU:

Operation 21 (CollectBelief) *Let T be a JTBU. Let* `caller` *by an adjacent JTBU or the LJFBU. When caller calls T to* `CollectBelief`*, T performs the following:*
(1) If T as no neighbor except caller, it performs UnifyBelief and return.
(2) Otherwise, for each adjacent JTBU Y except caller, call CollectBelief in Y. After Y finishes, T performs UpdateBelief relative to Y.

Note that Y is always internally consistent when T performs UpdateBelief relative to Y due to UnifyBelief in step (1) and in UpdateBelief.

DistributeBelief recursively propagates belief outwards on the hypertree of a LJFBU:

Operation 22 (DistributeBelief) *Let T be a JTBU. Let* `caller` *by an adjacent JTBU or the LJFBU. When caller calls T to* `DistributeBelief`*, T performs the following:*
(1) If caller is a JTBU, performs UpdateBelief relative to caller.
(2) For each adjacent JTBU Y except caller, call DistributeBelief in Y.

CommunicateBelief combines the previous two operations to bring a LJFBU into consistency:

Operation 23 (CommunicateBelief) *When CommunicateBelief is initiated at an LJFBU, CollectBelief is called at any JTBU T, followed by a call of DistributeBelief at T.*

CommunicateBelief brings a LJFBU into global consistence as defined below. It is shown in Theorem 25.

Definition 24 *A LJFBU F is* `globally consistent` *if each JTBU is internally consistent and each linkage tree is consistent with each of the two corresponding JTBUs.*

Theorem 25 *After CommunicateBelief in a LJFBU F, F is globally consistent.*

Proof:

Let Y be any JTBU in F other than T as referred in Operation 23. Let Y' be the adjacent JTBU of Y on the path between Y and T in the hypertree. Let L be the linkage tree between Y' and Y.

After CollectBelief at T, each JTBU Y is internally consistent (due to Proposition 20 (1)), and is consistent with L (due to Proposition 20 (3)).

After DistributeBelief at T, each JTBU Y' is internally consistent (due to Proposition 20 (1)), is consistent with L (due to Proposition 20 (3)), and the corresponding JTBU Y is also consistent with L (due to Proposition 20 (4)). □

As discussed in [5], CommunicateBelief is performed once for a while after evidence has been entered into different JTBUs. The operation ensures that local belief at each JTBU is consistent with evidence accumulated in the entire LJFBU.

6 Efficiency gain from new operations

What efficiency gain does the new operations provide?

According to the definition of CommunicateBelief, UpdateBelief is performed twice for each hyperlink of the LJFBU, and consumes a major portion of the communication computation. In the original version of UpdateBelief [8], a local belief propagation (called DistributeEvidence) is performed in the receiving JTBU after each AbsorbThroughLinkage[1]. Hence as many propagations as the number $|L|$ of linkages in the linkage tree L are performed for each execution of UpdateBelief.

The UpdateBelief defined in Operation 19 performs UnifyBelief once (two local propagations) no matter how many linkages are contained in the linkage tree. It improves the efficiency by a factor of $|L/2|$ relative to the original UpdateBelief [8]. The savings in computation is significant when each JTBU is large.

Alternative improvement over the original UpdateBelief has been proposed in [4]. There $|L|-1$ propagations are first performed with each restricted to a subgraph (a chain) of the JTBU, and finally a DistributeEvidence is performed. The control of the first $|L|-1$ propagations, however, are more sophisticated.

The UnifyBelief performed in the new UpdateBelief can be improved similarly: The first propagation in UnifyBelief can be restricted to the subgraph of the JTBU that terminates at linkage hosts. The second propagation is the same (DistributeEvidence). The amount of computation in the first propagation will be less than or equal to that in the first $|L|-1$ propagations in the alternative UpdateBelief, and the control needed is simpler than the

[1] UnifyBelief consists of two local propagations and DistributeEvidence is one of them.

alternative. The less amount of computation can be seen by observing that the $|L|-1$ propagations may repeat over certain sepsets in the JTBU. But the improved new UpdateBelief does not. The amount of computation of the two versions become equal if and only if the subgraph terminated by linkage hosts is a chain. Therefore, the new UpdateBelief with such modification will be superior than that in [4].

7 Belief initialization

Before inference can be performed in a LJFBU, its belief tables need to be initialized such that marginal probabilities of each variable x can be computed locally in any clique of any JTBU that contains x. The process involves first assigning conditional probability tables in subnets of the MSBN to cliques in JTBUs of the LJFBU, and then making the LJFBU globally consistent.

The first step is similar to what is performed in compilation of a BN to a JTBU [1] with exception for each d-sepnode. Since a d-sepnode occurs in multiple subnets, only the subnet that contains its most parents should transfer its table to the derived JTBU. This ensures equivalence of the JSB of LJFBU with the jpd of MSBN [8].

The second step was achieved by a special operation BeliefInitialization in [8]. It in turn is supported by some special operations not shared by inference computation (e.g., NonRedundancyAbsorption and ExchangeBelief). These operations dedicated to initialization complicates the theory of MSBNs as well as the practical implementation. The question is whether they are necessary. Below we answer the question negatively:

Note that Theorem 25 does not assume any previous state of consistence in F (compare with Theorem 14 in [5]). Therefore, it can be used both for inference as well as for belief initialization. In other words, after the probability table assignment (the first step above), the belief initialization can be completed by performing CommunicateBelief. A separate set of initialization operations is thus no longer needed.

Acknowledgements

This work is supported by the Research Grant OGP0155425 from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] F.V. Jensen. *An introduction to Bayesian networks.* UCL Press, 1996.

[2] D. Koller and A. Pfeffer. Oject-oridented Bayesian networks. In D. Geiger and P.P. Shenoy, editors, *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*, pages 302–313, Providence, Rhode Island, 1997.

[3] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[4] Y. Xiang. Optimization of inter-subnet belief updating in multiply sectioned Bayesian networks. In *Proc. 11th Conf. on Uncertainty in Artificial Intelligence*, pages 565–573, Montreal, 1995.

[5] Y. Xiang. A probabilistic framework for cooperative multi-agent distributed interpretation and optimization of communication. *Artificial Intelligence*, 87(1-2):295–342, 1996.

[6] Y. Xiang. Models learnable by belief net learning algorithms equipped with single-link search. Technical Report CS-97-03, University of Regina, 1997. Submitted for publication.

[7] Y. Xiang, B. Pant, A. Eisen, M. P. Beddoes, and D. Poole. Multiply sectioned Bayesian networks for neuromuscular diagnosis. *Artificial Intelligence in Medicine*, 5:293–314, 1993.

[8] Y. Xiang, D. Poole, and M. P. Beddoes. Multiply sectioned Bayesian networks and junction forests for large knowledge based systems. *Computational Intelligence*, 9(2):171–220, 1993.