Belief Updating in Multiply Sectioned Bayesian Networks without Repeated Local Propagations

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

Abstract

Multiply sectioned Bayesian networks (MSBNs) provide a coherent and flexible formalism for representing uncertain knowledge in large domains. Global consistency among subnets in a MSBN is achieved by communication. When a subnet updates its belief with respect to an adjacent subnet, existing inference operations require repeated belief propagations (proportional to the number of linkages between the two subnets) within the receiving subnet, making communication less efficient. We redefine these operations such that two such propagations are sufficient. We prove that the new operations, while improving the efficiency, do not compromise the coherence.

A MSBN must be initialized before inference can take place. The initialization involves dedicated operations not shared by inference operations according to existing methods. We show that the new inference operations presented here unify inference and initialization. Hence the new operations are not only more efficient but also simpler. The new results are presented such that their connection with the common inference methods for single Bayesian networks is highlighted.

`keywords`: Bayesian networks, probabilistic reasoning, multi-agent inference, distributed inference, uncertain knowledge representation.

1 Introduction

Bayesian networks (BNs) [14, 7] provide a coherent and effective framework for decision support systems that must function with uncertain knowledge. However, as the problem domains become larger and more complex, modeling a domain as a single BN and conducting inference in it becomes increasingly more difficult and expensive.

Multiply Sectioned Bayesian Networks (MSBNs) [24] provide one alternative to meet this challenge by relaxing the single BN paradigm. The framework allows a large domain to be modeled modularly and the inference to be performed distributively, while maintaining the coherence. The framework can be applied under the single agent paradigm [23] as well as the multi-agent paradigm [19]. It supports hierarchical model based diagnosis [16, 18] and modeling large systems with the object-oriented paradigm [10].

Several other frameworks for decomposition of probabilistic knowledge under a single agent paradigm has been proposed. Lam [11] proposed *abstract network* which replaces fragments of a BN by *abstract arcs* to improve inference efficiency. Geiger and Heckerman [4] presented *similarity network* and *Bayesian multinet* for representation of asymmetric independence relations. Kjaerulff [9] proposed *nested junction trees* to exploit independence relations induced by incoming messages of a cluster.

The focus of this paper is twofold. The first is on the inference computation in MSBNs. Evidence propagation 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. According to existing inference operations [24, 18], each exchange requires repeated belief propagations in the receiving subnet. The repeated local propagation was viewed as the unavoidable price to trade communication bandwidth.

That view has proved to be limited by the new results to be presented below. In this work, we redefine these operations such that each exchange of belief requires only two belief propagations in the receiving subnet. We prove that the new operations, while improving the efficiency, do not compromise the coherence.

A MSBN needs to be initialized before evidential inference takes place. According to existing method [24], the initialization involves several operations that are 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 computation as well as simplify the prototype implementation.

The second focus of this paper is on the unification of frameworks for inference in single BNs and in MSBNs. Inference in a BN can be performed effectively using its junction tree (JT) representation. Shafer [15] gives a unified presentation of Shafer-Shenoy, Lauritzen-Spiegelhalter [12] and HUGIN [8] methods.

The MSBN framework is an extension of these JT based inference methods with the HUGIN [8] method the most relevant. The theory of MSBNs and our new results can be better understood by following their connection with these methods. In our overview of MSBNs and presentation of the new results, we highlight such a connection.

We present the basic ideas underlying the MSBN framework in Section 2 with an emphasis on how they relate to JT based inference methods for BNs. A more formal review of the framework is given in Section 3. In Section 4, we establish the syntactic and semantic properties of linkage trees (the interface between subnets) which have not been treated formally before. In Section 5, we redefine the messages to be passed between subnets. The inference operations are redefined in Section 6 based on the new form of messages, and their coherence are proven. We discuss the efficience gain from the new operations in Section 7, and discuss the unification of inference and initialization in Section 8. About a dozen abbreviations frequently used in the paper are listed in Appendix.

2 Extending junction trees beyond single BNs

In this section, we present intuitively the basic ideas behind the MSBN framework with an emphasis on how it relates to junction tree based inference methods for Bayesian networks (BNs). We assume that readers are familiar with the basics about representation of probabilistic knowledge using BNs and the common inference methods in BNs [14, 12, 7, 15].

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. D encodes the assumption that each variable x is independent of its nondescendants given its parents $\pi(x)$. This allows P to be expressed as $P(N) = \prod_{x \in N} P(x|\pi(x))$. A BN can be used to model our uncertain knowledge about a domain, e.g., medical diagnosis [5], equipment trouble-shooting [6], financial forecasting [1], automated vehicles [3], etc.

Figure 1 (a) shows a digital circuit and the DAG of a BN that models the circuit is shown in (b). An example conditional probability distribution associated with the variable f (output of a *not* gate) is given below:

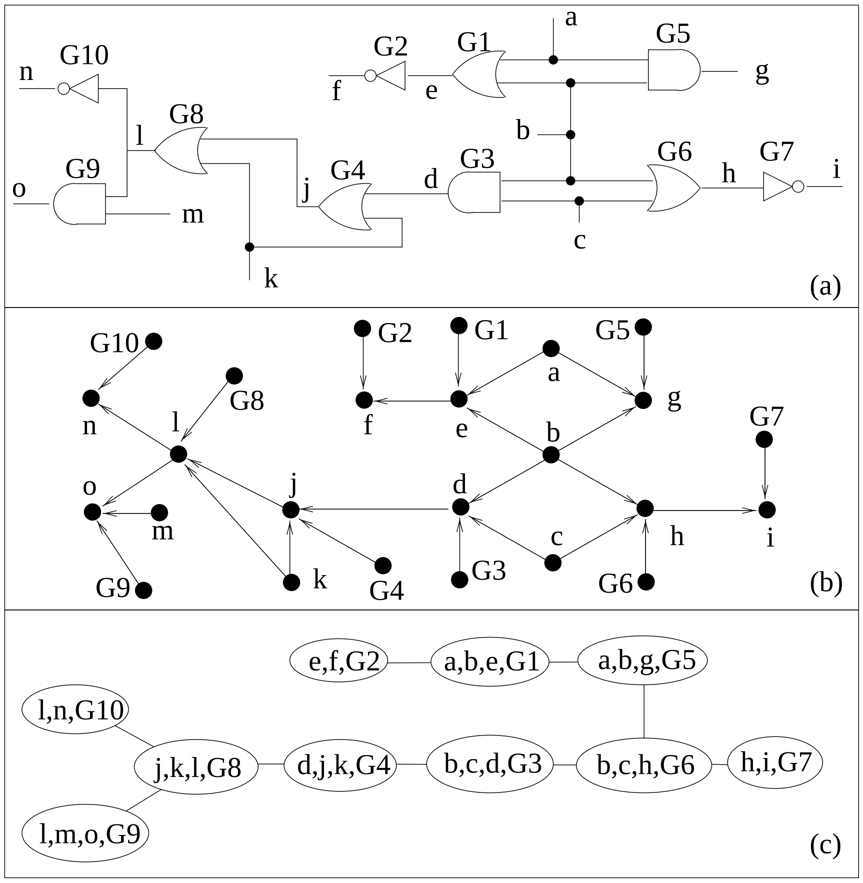

Figure 1: (a) A digital circuit. (b) The DAG of a BN to model the circuit. (c) A JT of the BN.

$$P(f=0|G_2=normal, e=0)=0 \quad P(f=0|G_2=normal, e=1)=1.0$$
$$P(f=0|G_2=faulty, e=0)=0.3 \quad P(f=0|G_2=faulty, e=1)=0.8$$

Once observation on the domain is available, inference can be performed using the BN to estimate the states of unobserved variables. For example, we can compute the *posterior* probability $P(G_1 = faulty|a = 0, b = 1, f = 1)$ from the above BN. Well-known methods for computing such posteriors *exactly* include those by Lauritzen-Spiegelhalter [12], HUGIN [8] and Shafer-Shenoy [15]. These methods base their inference computation on a *junction tree* (JT) representation of the domain. For example, variables in the above BN can be organized into a JT of *clusters* in Figure 1 (c). During inference, message passing is performed first inward and then outward along the tree structure. After message passing, the posteriors for each variable can be obtained locally at any cluster that contains it. As explained by Shafer [15] (p64), the message passing can be equivalently controlled in an asynchronous fashion or a synchronous fashion initiated from a root cluster. In the HUGIN method (synchronous control), a single message passing from a cluster to an adjacent cluster is called Absorption, the inward message passing along the entire JT is called CollectEvidence and outward passing is called DistributeEvidence.

As the problem domain becomes larger and more complex, modeling such a domain as a single BN and conducting inference in it becomes increasingly more difficult and expensive. The approach taken by multiply section Bayesian networks (MSBNs) is to explore modularity and distribution, two important factors that motivate distributed artificial intelligence (DAI) [2] and multi-agent systems [17]. The key issue then is how to determine the units for distribution such that the coherence of inference is not compromised by distribution. The junction tree representation of a single BN provides useful hints:

In a JT, each cluster consists of a subset of the domain variables. Each cluster acts as a unit/object in message passing during inference. Similarly, a MSBN partitions a large domain into a *hypertree* (that can be proven to be a JT) of some natural subdomains. Such subdomains become the units for distribution. Based on such a partition, the top level inference in the large domain, called CommunicateBelief (Section 6), can be performed similarly to what is performed in the JT of a single BN, namely, by an inward message passing through subdomains along the hypertree, called CollectBelief (Section 6), followed by an outward message passing, called DistributeBelief (Section 6). Note that these operations are named to correspond to the HUGIN operations.

We illustrate the idea using the above circuit example. We choose to use a digital circuit as no special domain knowledge is required. Readers should keep in mind that the example is an *over-simplified* one, and a MSBN is *not* needed in practice unless the domain is much larger than this example.

Suppose the circuit in Figure 1 (a) is organized into three components (shown as dotted boxes in Figure 2 (a)) which are spatially distributed. Hence U_i $(i = 0, 1, 2)$ form a natural partition of the domain, where $U_1 = \{a, b, c, g, h, i, G5, G6, G7\}$ for example. The hypertree in this case is the hyperchain $U_2 - U_0 - U_1$.

We have seen that a MSBN partitions a large domain into a hypertree which is analogous to a JT of a single BN. This is the *first* level of application of the JT representation in MSBNs. On the other hand, a cluster (e.g., $\{a, b, g, G5\}$ in Figure 1 (c)) in a JT has no internal structure (saving for a recent development [13]). The belief over a cluster is represented as a potential (non-normalized probability distribution) over all variables in the cluster. Since a subdomain in a large domain is itself large in general, representing it as a cluster is neither feasible nor necessary. Instead, a MSBN represents each subdomain as a Bayesian network called a *subnet*. For example, the circuit

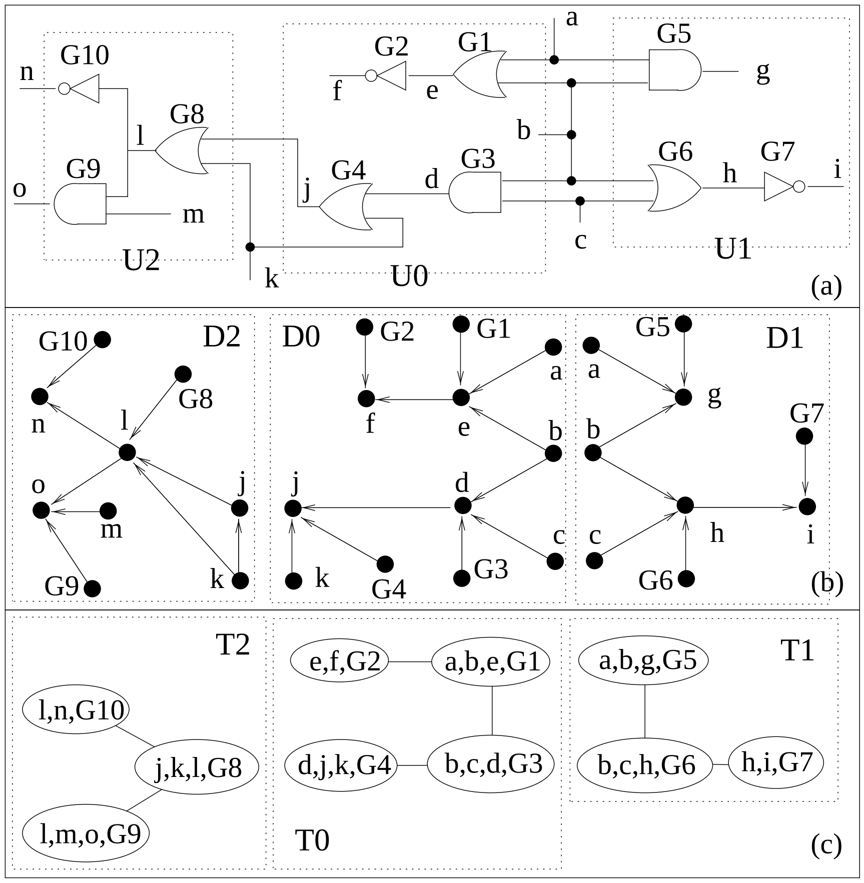

Figure 2: (a) A digital circuit organized as three components. (b) The DAGs of three subnets of a MSBN. (c) JTs converted from the subnets.

in Figure 2 (a) can be represented by the three subnets in (b).

Since each subnet is itself a BN, inference within a subdomain can be performed in the same way as if the subnet is a normal BN. Hence in the MSBN framework, a subnet is converted into a JT and inference in it is performed by CollectEvidence and DistributeEvidence if *only* local observations in its subdomain are involved. For example, the three subnets in Figure 2 (b) are converted into the three JTs in (c) for local inference. This is the *second* level of application of the JT representation in MSBNs.

In a JT of a single BN, a message sent by a cluster C to an adjacent cluster C' is a *belief table* over their intersection $C \cap C'$, called *sepset* (which labels the link between the clusters). For example, the sepset between clusters $\{a, b, g, G5\}$ and $\{a, b, e, G1\}$ (Figure 1 (c)) is $\{a, b\}$. Like a cluster in a JT, a sepset has no internal structure (saving for a recent development [13]). In a large domain, the intersection of two subdomains, called a *d-sepset*, is also large in general. Hence, more compact representation of the d-sepset is desired. The MSBN framework represents each d-sepset also as a JT, called a *linkage tree*, which allows a more efficient representation of the message passed between subdomains. This is the *third* level of application of the JT representation in MSBNs. Figure 3 expresses the three JTs as three boxes. Each band between a pair of boxes illustrates a d-sepset and is labeled accordingly. The d-sepset between T_0 and T_1 is represented as a linkage tree of two clusters, and that between T_0 and T_2 is represented as a trivial linkage tree of a single cluster.

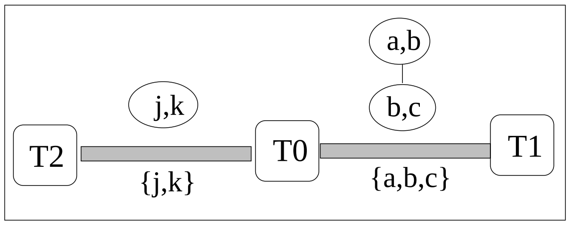

Figure 3: Linkage trees for JTs of the circuit MSBN.

In a JT of a single BN, the inward/outward message passing are performed by a series of Absorptions, each of which passes a message over one sepset. In the MSBN framework, CollectBelief and DistributeBelief are performed by a series of message passings each of which is over one linkage tree and is called UpdateBelief (Section 6). A key result presented in this paper is a redesign of UpdateBelief for better conceptual clarity as well as computational efficiency.

3 Overview of the MSBN framework

In this section, we present briefly the formal theory of the MSBN framework. A MSBN M is a collection of Bayesian subnets that together defines a BN. M represents probabilistic dependence of a *total universe* partitioned into multiple *subdomains* each of which is represented by a subnet. The partition 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`*.*

For example, in Figure 2 (b) the intersection $\{a, b, c\}$ between D_0 and D_1 is a d-sepset, so is $\{j, k\}$ between D_0 and D_2. A d-sepset is a sufficient information channel for passing all relevant evidence from one subnet to another. Formally, a pair of subnets are conditionally independent given their d-sepset.

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 only* D_j *and* D_k *are considered.*
[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`*.*

It can be proven [21] that if each hypernode D_k of a hypertree MSDAG is replaced by the cluster N_k and each hyperlink between D_j and D_k is replaced by the d-sepset I_{jk}, then the resultant is a JT. The DAGs in Figure 2 (b) is organized into the trivial hypertree MSDAG in Figure 4 (a) where each hypernode is labeled by a DAG and each hyperlink is labeled by a d-sepset. Figure 4 (b) depicts a more general hypertree MSDAG. A hyperlink is a sufficient information channel for passing all relevant evidence from one side of hyperlink to the other. Formally, given a hyperlink, the two subtrees connected through the link are conditionally independent.

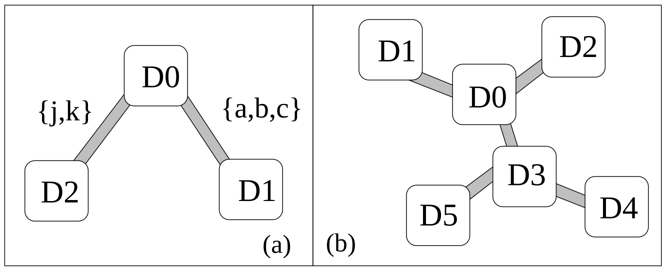

Figure 4: (a) The hypertree MSDAG of the circuit MSBN. (b) A MSDAG of a more general topology.

In a MSDAG, a non-d-sepnode occurs only once, and a d-sepnode has multiple occurrences one at each DAG involved. For each d-sepnode, at least one occurrence in one DAG has all its parents in the entire MSDAG, which is ensured by the d-sepset condition. A MSBN is 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. For each $x \in \mathcal{N}$, its occurence with the most parents (breaking ties arbitrarily) $\pi(x)$ is associated with a probability distribution $P(x|\pi(x))$, and each other occurrence is associated with a constant (trivial) distribution. $\mathcal{P} = \prod_i P_{D_i}(N_i)$ is the* `jpd`*, where $P_{D_i}(N_i) = \prod_{x \in N_i} P(x|\pi(x))$ is a* `local` *distribution over N_i. Each triplet $S_i = (N_i, D_i, P_{D_i}(N_i))$ 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) [7] T_i over N_i. A *junction tree* T over N is a tree whose nodes are labeled by subsets (clusters) of N such that the intersection of any two clusters is contained in every cluster between them. Each link in T is labeled by the intersection (sepset) of the end clusters. D_i is converted into T_i by *moralization* and *triangulation*. How to perform these operations is presented in [24] and is improved in [22]. The JTs obtained from DAGs in Figure 2 (b) are shown in (c).

Each cluster and each sepset in a JT is associated with a *belief table*: a non-normalized (hence equivalent) probability distribution. How to assign these tables will be detailed in Section 8. A belief table $B_{T_i}(N_i)$ associated with a JT T_i is defined below.

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 cluster with the belief table $B_C(C)$ and each S is a sepset with the belief table $B_S(S)$.*

A triplet $\mathcal{T}_i = (N_i, T_i, B_{T_i}(N_i))$ is called a junction tree of belief universes (JTBU) [7]. We shall sometimes refer to a JTBU as simply a JT if no confusion may arise. Proposition 5 states the semantics of a JTBU from one perspective and is needed later. Let $P(N)$ be a probability distribution over N and T be a JT over N. T is an *I-map* of P if for any disjoint subsets X, Y, Z of N, that X and Y are independent given Z according to P implies that clusters containing X and Y are separated in T by sepsets contained in Z. See [14] for a general discussion on I-maps and [20] 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 cluster 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. Here we give a definition equivalent to (but computationally less efficient than) that in [19]. The proof of equivalence is trivial.

Definition 6 *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:*

Initialize L to T_a. Repeat the following on clusters of L until no variable can be removed:

(1) Remove a variable $x \notin I$ if x is contained in a single cluster C.

(2) If C becomes a subset of an adjacent cluster D after (1), union C into D.

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

For the circuit MSBN, the linkage trees L_1 between T_0 and T_1 and L_2 between T_0 and T_2 are shown in Figure 5. The thick grey links illustrate how each linkage relates to its two linkage hosts.

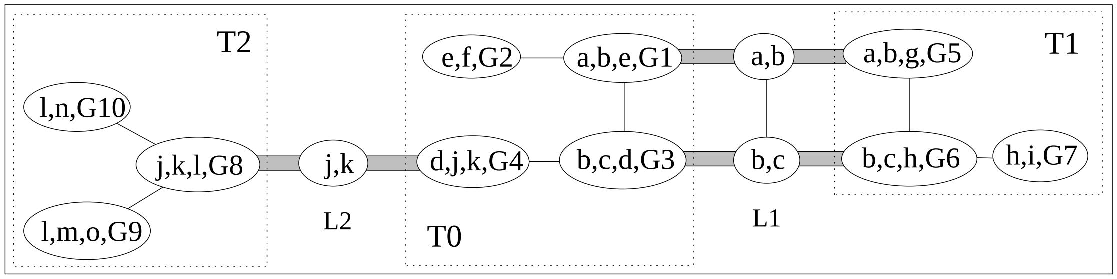

Figure 5: Linked junction forest for the circuit MSBN.

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. How to assign belief tables for clusters and sepsets of a LTBU is detailed in Sections 4 and 8.

A common question on MSBN is whether the JTs in a linked junction forest can be merged into a single JT by simply adding links between clusters in different JTs. The JTs can certainly be constructed such that they can be merged. However, this implies that each d-sepset will be represented as a single unit/cluster (without explicit internal structure). The consequence is that clusters of each JT will be larger and the inference computation will be more expensive.

When a d-sepset is represented as a LTBU, such as L_1 in Figure 5, it allows more compact representation of belief over the d-sepset, smaller clusters of JTs being linked, and more efficient inference. On the other hand, the JTs so constructed *cannot* be merged into a single JT. For example, T_i $(i = 0, 1, 2)$ in Figure 5 cannot be merged into one JT by adding links between clusters in different JTs.

More discussion on $\mathcal{L}_i$ follows in Section 4. 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), where 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. In Section 8, we will detail how to assign belief tables such that the JSB of a LJFBU is equivalent to the jpd of its deriving MSBN.

4 Properties of linkage trees

In this section, we formally establish the syntactic and semantic properties of linkage trees.

A linkage tree is an alternative representation of the d-sepset. The procedure in Definition 6 may not be able to remove all the non-d-sepnodes and in that case a linkage tree is undefined. The condition under which a linkage tree is well defined and how to satisfy that condition are presented in [22]. Here, we assume that a linkage tree is well defined when the procedure in Definition 6 terminates.

Proposition 8 shows that a linkage tree is a JT:

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

Proof:

After removal of a variable contained in a single cluster C of a JT, the resultant graph is still a JT. If such removal renders C a subset of an adjacent cluster D, then union of C into D neither changes any sepset between C and its neighbor clusters (other than D), nor changes any sepset between D and its neighbor clusters (other than C). Hence the graph obtained after steps (1) and (2) is a JT. $\square$

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 step (1) of Definition 6, the removal of x is irrelevant to the graphical separation among elements of I.

In step (2), union of C into D still leaves C contained in a cluster. Thus removal (union) of C does not alter the graphical separation among elements of I. $\square$

Definition 7 does not specify how a belief table for a linkage tree is defined. It is defined 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 $B_L(I) = \prod_l B_l(l) / \prod_q B_q(q)$.*

For example, the belief of L_1 in Figure 5 can be defined from belief tables in T_1. For linkage $\{b, c\}$, its belief table is obtained from the belief table of its host cluster $\{b, c, h, G_6\}$ through marginalization. For linkage $\{a, b\}$, its belief table is obtained from that of $\{a, b, g, G_5\}$.

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 clusters 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. $\square$

5 Extending linkage belief

In this section, we extend the linkage belief defined in Definition 10 such that more efficient belief propagation (than the existing methods) between JTBUs can be supported. The extended belief for each linkage is a combination of the original linkage belief with the belief of a sepset in the linkage tree. First, we introduce the peer sepset of a linkage used to signify which sepset belief should be combined with which linkage belief:

Definition 12 *Let L be a linkage tree between a pair of JTs in a LJF. Convert L into a rooted tree by select a node l arbitrarily as the root and direct links away from it. For each node $l' \neq l$ in L, assign its sepset with its parent node as the* `peer` *sepset of l'.*

For example, in Figure 5, there are two linkages in L_1. If we select linkage $\{a, b\}$ as the root, then $\{a, b\}$ has no peer assigned to it, and the sepset $\{b\}$ becomes the peer of linkage $\{b, c\}$. We extend the linkage belief from Definition 10 as follows:

Definition 13 *Let L be a linkage tree with linkage and sepset belief defined as Definition 10, and linkage peers defined as Definition 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 $B^*_l(l) = B_l(l)$.*

As an example, consider L_1 in Figure 5 using the above peer assignment. The extended belief for linkage $\{b, c\}$ will be $B_{\{b,c\}}(b, c)/B_{\{b\}}(b)$, and the extended belief for linkage $\{a, b\}$ will be $B_{\{a,b\}}(a, b)$.

The semantics of extended linkage belief is shown in Proposition 14. The proof is trivial.

Proposition 14 *Let L be a linkage tree. Then $B_L(I)$, as defined in Definition 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.*

The linkage belief by Definition 10 is equivalent to the HUGIN belief representation. In this representation, the belief on each sepset is repeated in the linkage belief tables. During evidence propagation between JTBUs, we have to remove this redundant information, which is a main contributing factor that causes the complication of existing inference operations for MSBNs. The extended linkage belief removes this redundancy before propagation. Hence it is similar to the Shafer-Shenoy belief representation (although no link buffer storage is used as S-S scheme does). We shall see that by using extended linkage belief tables as messages between JTBUs during inference, belief propagation between JTBUs can be performed more efficiently than the existing methods. 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.

6 Inference operations

In this section, we redefine inference operations in [24, 19] based on extended linkage belief. First, we redefine the operation AbsorbThroughLinkage. The effect of the operation is to propagate belief from one linkage host to the other.

Operation 15 (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 [24] does not apply any more. We extend it to define the concept of e-consistency:

Definition 16 *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 $\sum_{C_a \setminus l} B_{C_a}(C_a) = B_l(l)$.*

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 17 *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

$$\begin{aligned} \sum_{C_a \setminus l} B'_{C_a}(C_a) &= \sum_{C_a \setminus l} B_{C_a}(C_a) * B^*_{C_b}(l)/B^*_l(l) \quad \text{(def. of AbsorbThroughLinkage)} \\ &= [B^*_{C_b}(l)/B^*_l(l)] * \sum_{C_a \setminus l} B_{C_a}(C_a) \quad \text{(Proposition 4.1 [7])} \\ &= [B^*_{C_b}(l)/B^*_l(l)] * B_l(l) \quad \text{(e-consistency assumption)} \end{aligned}$$

$$
\begin{aligned}
&= \begin{cases} \frac{B_{C_b}(l)/B_q(q)}{B_l(l)/B_q(q)} * B_l(l) & \text{[if } l \text{ has peer } q\text{]} \\ \frac{B_{C_b}(l)}{B_l(l)} * B_l(l) & \text{[otherwise]} \end{cases} \quad \text{(def. of extended linkage belief)} \\
&= B_{C_b}(l) = B'_l(l)
\end{aligned}
$$

□

As shown by Jensen et al., the operations CollectEvidence and DistributeEvidence [8] bring a JTBU internally consistent. As they are called by several operations defined below, we combine the two into a single operation UnifyBelief as in [24] for simplicity.

Operation 18 (UnifyBelief[24]) *Let T be a JTBU and C be any cluster in T. When* `UnifyBelief` *is called on T, initiate CollectEvidence [8] at C followed by DistributeEvidence [8] from C.*

The operation UpdateBelief propagates belief from a JTBU to another adjacent JTBU through multiple linkages (a hyperlink) between them. In the HUGIN method for inference in a JT of a single BN, evidence is propagated from a cluster to an adjacent one through a sepset by an operation called Absorption [7]. UpdateBelief is analogous to Absorption but the sender and the receiver are JTBUs, and the channel is a d-sepset/hyperlink.

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 in the following proposition. The consistency between a linkage tree and one of its deriving JTBU is defined in the normal way.

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 14 and 17 (2).

(4) It follows from Propositions 14 and 17 (3). □

CollectBelief recursively propagates belief inwards (from leaves towards an initiating JTBU) on the hypertree of a LJFBU. Just as UpdateBelief is analogous to Absorption at a higher abstraction level, CollectBelief is analogous to CollectEvidence in the HUGIN method but at the hypertree level.

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 has 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 (from an initiating JTBU towards leaves) on the hypertree of a LJFBU. DistributeBelief is analogous to DistributeEvidence in the HUGIN method but at the hypertree level.

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. CommunicateBelief is analogous to UnifyBelief (at the JTBU level) but at the LJFBU/hypertree level.

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 consistency 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. See Figure 6 for illustration.

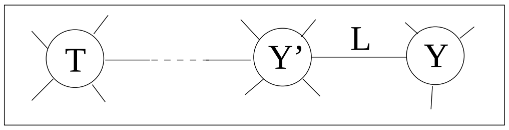

Figure 6: Illustration of proof for Theorem 25.

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)). $\square$

As discussed in [19], 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.

7 Efficiency gain from new operations

What efficiency gain do 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 [24], a local belief propagation (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

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

factor of $|L|/2$ relative to the original UpdateBelief [24]. The savings in computation are significant when each JTBU is large.

Alternative improvement over the original UpdateBelief has been proposed in [18]. There $|L|-1$ propagations are first performed each of which is along a chain in the JTBU, and a DistributeEvidence is performed at the end. The control of the first $|L|-1$ propagations, however, is more sophisticated in that each chain is terminated by a different pair of clusters.

The UnifyBelief performed in the new UpdateBelief can be improved similarly: The first propagation (CollectEvidence) in UnifyBelief can be restricted to the subgraph of the JTBU that terminates at linkage hosts. The second propagation (DistributeEvidence) is the same. 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 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 (with respect to efficiency and simplicity in control) than that in [18].

8 Belief initialization

Before inference can be performed in a LJFBU, its belief tables need to be set up such that marginal probabilities of each variable x can be computed locally in any cluster of any JTBU that contains x. In other words, the joint system belief (JSB) of the LJFBU should be assigned *equivalently* to the jpd of its deriving MSBN and the LJFBU should be made *globally consistent*.

Definition 7 did not detail how belief tables for clusters/sepsets in the JTBUs and LTBUs are initially assigned. We present the assignment here:

The beliefs for clusters of JTBUs are assigned in the same way as common methods of inference in JTs of single BNs: For each subnet S_i, assign the probability table of each node x to a unique cluster C in T_i such that C contains x and its parents in S_i. Then the belief table of each cluster is the product of all tables assigned to it. Each sepset in a JTBU is assigned a constant table. For each LTBU, all clusters and sepsets are assigned constant tables of proper dimensions. Then from Definitions 3 and 7, it is trivial to show the following:

Proposition 26 *The JSB defined in Definition 7 is equivalent to the jpd defined in Definition 3.*

Next, we consider the issue of consistence. Clearly the LJFBU, with its JSB assigned as above, is *not* globally consistent. The process of rendering the LJFBU globally consistent is called *initialization.*

In the early work on MSBNs [24], initialization is achieved by a special operation BeliefInitialization. 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.

We note that Theorem 25 does not assume any previous state of consistency in F (compare with Theorem 14 in [19]). Therefore, it can be used both for inference as well as for initialization. In other words, after belief tables are assigned, initialization can be completed by performing CommunicateBelief. A separate set of initialization operations is thus no longer needed. We summarize this in the following corollary:

Corollary 27 *CommunicateBelief (Operation 23) performed in a LJFBU before any evidence is entered is equivalent to the operation BeliefInitialization as defined in [24].*

9 Conclusion

MSBNs allow effective local inference by representing each subnet as a JTBU and by representing the d-sepset between a pair of subnets as a linkage tree. Given a linkage tree with $|L|$ linkages, previous inference operations require $|L|$ belief propagation in order to propagate new evidence from one JTBU to an adjacent one. Hence communication among subnets is slowed down by the use of multiple linkages. A separate set of operations different from that for inference was also used to initialize a MSBN before inference can take place. These operations complicate the theory of MSBNs and hinders its practical application.

In this paper, we redefined operations for inference in MSBNs. Using the new operations, two local propagations are sufficient for propagating evidence from one JTBU to an adjacent one no matter how many linkages there are between the two JTBUs. Thus they improve the efficiency of communication by a factor of $|L|/2$. The computational savings are particularly significant when each subnet in the MSBN is large.

In our presentation, we have emphasized the connection between the MSBN/LJFBU representation and the standard JT representation of single BNs. At the top level, a MSBN partitions a large domain into a hypertree (a JT) of subdomains. At the next level, each subdomain is represented as a JT for local inference computation. At the intersubdomain level, each d-sepset is represented as a linkage tree (a JT). These representations are crucial in order to perform inference in a large domain distributively, coherently, and effectively.

It has long been a puzzle to us why inference as well as initialization in JTs of single BNs can be performed using the same set of operations (CollectEvidence and DistributeEvidence) but two different sets of operations are needed for inference and initialization in MSBN/LJFBU. The new operations presented unify operations for inference and those for initialization, which simplifies the theory of MSBNs and facilitates practical implementation. These operations have been implemented in WEBWEAVR-III (freely available at "http://cs.uregina.ca/ yxiang/ww3/index.html") and tested experimentally.

The new set of operations presented in the paper is directly suited for inference in MSBNs under the multi-agent paradigm. By replacing the operation CommunicateBelief with the operation ShiftAttention as defined in [24], the modified set will be suited for inference in MSBNs under the single-agent paradigm. All the benefits as indicated above will still apply.

Acknowledgements

This work is supported by the Research Grant OGP0155425 from the Natural Sciences and Engineering Research Council (NSERC) of Canada. Partial writing was completed while the author was visiting Aalborg University, Denmark.

References

[1] B. Abramson. Arco1: an application of belief networks to the oil market. In B. Dambrosio, P. Smets, and P. Bonissone, editors, *Proc. 7th Conf. on Uncertainty in Artificial Intelligence*, pages 1–8, Los Angeles, CA, 1991.

[2] A.H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, 1988.

[3] J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The batmobile: towards a bayesian automated taxi. In *Proc. Fourteenth International Joint Conf. on Artificial Intelligence*, pages 1878–1885, Montreal, Canada, 1995.

[4] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82:45–72, 1996.

[5] D. Heckerman. *Probabilistic Similarity Networks.* PhD thesis, Stanford University, 1990.

[6] D. Heckerman, J.S. Breese, and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, pages 49–57, 1995.

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

[8] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, (4):269–282, 1990.

[9] U. Kjaerulff. Nested junction trees. In *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*, pages 294–301, Providence, Rhode Island, 1997.

[10] 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.

[11] W. Lam. Abstraction in Bayesian belief networks and automatic discovery from past inference sessions. In *Proc. of AAAI*, pages 257–262, 1994.

[12] S.L. Lauritzen and D.J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society, Series B*, (50):157–244, 1988.

[13] A.L. Madsen and F.V. Jensen. Lazy propagation in junction trees. In *Proc. 14th Conf. on Uncertainty in Artificial Intelligence*, 1998.

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

[15] G. Shafer. *Probabilistic Expert Systems.* Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[16] S. Srinivas. A probabilistic approach to hierarchical model-based diagnosis. In *Proc. 10th Conf. Uncertainty in Artificial Intelligence*, pages 538–545, Seattle, Washington, 1994.

[17] M. Wooldridge and N.R. Jennings. Intelligent agents: theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[18] 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.

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

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

[21] Y. Xiang. Cooperative multiagent distributed interpretation: is multiply sectioned Bayesian networks necessary? Technical Report CS-97-05, University of Regina, 1998. Submitted for publication.

[22] Y. Xiang. Cooperative triangulation in MSBNs without revealing subnet structures. Technical Report CS-98-02, University of Regina, 1998. Submitted for publication.

[23] 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.

[24] 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.

Appendix: Frequently used abbreviations

BN: Bayesian network

DAG: directed acyclic graph

jpd: joint probability distribution

JSB: joint system belief

JT: junction tree

JTBU: junction tree of belief universes

LJF: linked junction forest

LJFBU: linked junction forest of belief universes

LTBU: linkage tree of belief universes

MSBN: multiply sectioned Bayesian network

MSDAG: multiply sectioned DAG