1 Lazy Inference in Multiply Sectioned Bayesian Networks Using Linked Junction Forests

Yang Xiang and Xiaoyun Chen

University of Guelph, Canada

Abstract. Lazy propagation reduces the space complexity from HUGIN inference. Multiply Sectioned Bayesian Networks extend Bayesian Networks a cooperative multiagent paradigm. To combine the benefits of the two, a framework was proposed earlier to apply lazy propagation to inference in MSBNs. We propose an alternative framework with a simpler compiled structure. The issues of lazy communication and observation entering in a multiagent setting are considered. We prove that the inference is exact.

1.1 Introduction

Multiply Sectioned Bayesian Networks (MSBNs) [8] extend BNs [4] to the multiagent paradigm. The inference method is an extension of HUGIN method for BNs using the junction tree (JT) representation. Lazy propagation [3] extends the applicability of HUGIN inference method to larger domains. It uses a factorized representation for belief, performs only the necessary multiplication and marginalization, and results in reduced space complexity.

A framework was proposed earlier [6] to apply lazy propagation to inference in MSBNs. The compiled runtime representation requires the maintenance of multiple local graphical structures for each subnet. In this work, we propose an alternative framework for multiagent systems where only a single local structure is needed. We propose a set of algorithms for local lazy inference at each agent, for lazy communication among agents, and for entering observations. We prove that the lazy inference is autonomous and exact.

The alternative framework has the following advantages: Its local structure is isomorphic to that for standard inference in MSBNs. Hence, the same set of structure compilation algorithms [8] are applicable and the same compilation software components (such as those in WEBWEAVR [1]) can be reused. It can also lead to space savings (one local structure versus several) and resultant simplified control. Experimental evidence is expected from our ongoing research.

We briefly overview the framework of MSBNs and lazy propagation in Sections 1.2 and 1.3. Our overview assumes the knowledge on HUGIN and Shafer-Shenoy inference methods in JT representations of BNs. Readers unfamiliar with these are directed to [2,5]. Readers who desire in-depth understanding of MSBNs are directed to [8]. The remaining sections develop

the lazy propagation based new inference scheme for MSBNs. MSBNs are intended for large and complex domains. However, many relevant concepts can and should be illustrated with simple examples. Readers are reminded of the discrepancy between the complexity of the examples in the paper and that of intended applications.

1.2 Overview of MSBNs

1.2.1 Multiply Sectioned Bayesian Networks

A BN [4] can be used to structure the knowledge of a single agent. What is its counterpart for a cooperative multiagent system? From a small set of assumptions, it has been shown [7] that the resultant representation of a cooperative multiagent system is an MSBN:

1. exact probability measure of belief,
2. communication by belief over small sets of shared variables,
3. a simpler organization of agents,
4. DAG domain structuring, and
5. joint belief admitting agents' beliefs on internal variables and combining their beliefs on shared variables.

Although an MSBN can be applied under the single agent paradigm, our presentation follows the multiagent paradigm.

An MSBN M is a collection of Bayesian subnets, one from each agent, 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. Agents cooperate to reason about what is going on [8]. Without confusion, we refer to an agent, its subdomain, and its subnet interchangeably from time to time. To ensure correct, distributed inference, subnets are required to satisfy certain conditions [7] described below:

Let $G_i = (N_i, E_i)$ $(i = 0, 1)$ be two graphs (directed or undirected). G_0 and G_1 are said to be *graph-consistent* if the subgraphs of G_0 and G_1 spanned by $N_0 \cap N_1$ are identical. Given two graph-consistent graphs $G_i = (N_i, E_i)$ $(i = 0, 1)$, 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 $G = G_0 \sqcup G_1$. Given a graph $G = (N, E)$, a partition of N into N_0 and N_1 such that $N_0 \cup N_1 = N$ and $N_0 \cap N_1 \neq \emptyset$, and subgraphs G_i of G spanned by N_i $(i = 0, 1)$, G is said to be *sectioned* into G_0 and G_1. Sectioning is useful in defining the dependence between variables shared by subdomains in a graphical model:

Definition 1 *Let $G = (N, E)$ be a connected graph sectioned into subgraphs $\{G_i = (N_i, E_i)\}$. Let the subgraphs be organized into an undirected tree Ψ where each node is uniquely labeled by a G_i and each link between G_k and G_m is labeled by the non-empty* `interface` *$N_k \cap N_m$ such that for each i and*

j, $N_i \cap N_j$ is contained in each subgraph on the path between G_i and G_j in Ψ. Then Ψ is a `hypertree` *over G. Each G_i is a* `hypernode` *and each interface is a* `hyperlink`*. A pair of hypernodes connected by a hyperlink is said to be* `adjacent`*.*

Each hyperlink serves as the information channel between subnets connected and is referred to as an agent *interface*. Agents communicate by exchanging beliefs over their interfaces. An interface must be a *d-sepset*, as defined below:

Definition 2 *Let G be a directed graph such that a hypertree over G exists. A node x contained in more than one subgraph with its parents $\pi(x)$ in G is a* `d−sepnode` *if there exists at least one subgraph that contains $\pi(x)$. An interface I is a* `d−sepset` *if every $x \in I$ is a d-sepnode.*

The overall structure of an MSBN is a hypertree MSDAG:

Definition 3 *A* `hypertree MSDAG` *$\mathcal{G} = \bigsqcup_i G_i$, where each G_i is a DAG, is a connected DAG such that (1) there exists a hypertree ψ over $\mathcal{G}$, and (2) each hyperlink in ψ is a d-sepset.*

Graphically, a hyperlink separates the hypertree MSDAG into two subtrees. Semantically, this corresponds to conditional independence given the d-sepset. An MSBN is then defined as follows:

Definition 4 *An MSBN M is a triplet $M = (\mathcal{N}, \mathcal{G}, \mathcal{P})$. $\mathcal{N} = \bigcup_i N_i$ is the* `total universe` *where each N_i is a set of variables. $\mathcal{G} = \bigsqcup_i G_i$ (a hypertree MSDAG) is the* `structure` *where nodes of each DAG G_i are labeled by elements of N_i. Let x be a variable and $\pi(x)$ be all the parents of x in G. For each x, exactly one of its occurrences (in a G_i containing $\{x\} \cup \pi(x)$) is assigned $P(x|\pi(x))$, and each occurrence in other DAGs is assigned a constant table. $\mathcal{P} = \prod_i P_i(N_i)$ is the* `jpd`*, where each $P_i(N_i)$ is the product of probability tables associated with nodes in G_i. Each triplet $S_i = (N_i, G_i, P_i)$ is called a* `subnet` *of M. Two subnets S_i and S_j are said to be* `adjacent` *if G_i and G_j are adjacent on the hypertree MSDAG.*

An example MSBN is shown in Fig. 1.1.

1.2.2 Linked Junction Forest

Inference in an MSBN is performed based on message passing. *Local inference* within each agent passes intra-subnet messages which bring a subnet into consistency. *Communication* among agents passes inter-subnet messages which brings the system into global consistency. These messages are marginal probability distributions. The key issue is to use messages over small subsets of variables so that inference is efficient.

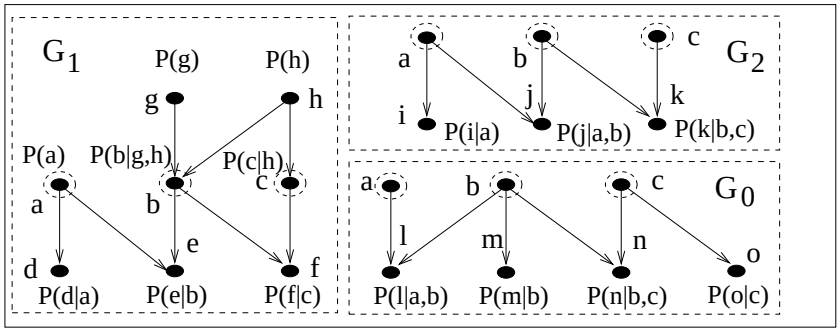

Fig. 1.1. A trivial MSBN where each d-sepnode is shown with a dashed circle. The hypertree has the structure $G_1 - G_0 - G_2$ and each d-sepset is $\{a, b, c\}$.

To compute intra-subnet messages and propagate them effectively, each agent compiles its subnet into a junction tree (JT), where variables are grouped into *clusters* with intersection of adjacent clusters referred to as *separators*. Note that the hypertree of a MSDAG is a JT if each hypernode is labeled by the corresponding subdomain N_i. Without confusion, we simply refer to this JT as hypertree.

Similarly, to facilitate computation of inter-subnet messages, agents compile each d-sepset into a JT, called a *linkage tree*. With local JTs and linkage trees combined, the resultant representation is called a *linked junction forest* (LJF). For details on compilation, see [8]. See Fig. 1.2 for linkage trees L_1 between T_0 and T_1 and L_2 between T_0 and T_2. Each cluster in a linkage tree is called a `linkage`. Linkage $\{b, c\}$ is an information channel between cluster $\{b, c, f\}$ in T_1 and cluster $\{b, c, n\}$ in T_0. They are referred to as the `linkage hosts` of $\{b, c\}$.

Parallel to the structure compilation, probability tables in MSBN are converted to *potentials* (non-normalized probability distributions) associated with clusters, separators and linkages. From them, the *joint system potential* of LJF is defined that is equivalent to the jpd $\mathcal{P}$ of the MSBN. When observations are available, each agent performs local inference in its local JT using the HUGIN method. Communication among agents is performed by propagation on hypertree along hyperlinks (technically along linkages). After the communication, probabilistic queries posed to any agent can be answered exactly relative to observations entered in the entire LJF. We refer to the inference method as *HUGIN-like inference with LJFs*.

1.3 Overview of Lazy Propagation

Lazy propagation [3] is performed using the JT structure of a BN. Each cluster is associated with a set of potentials from the BN. We refer to the cluster of current focus by C and its set of potentials by β. When no potential

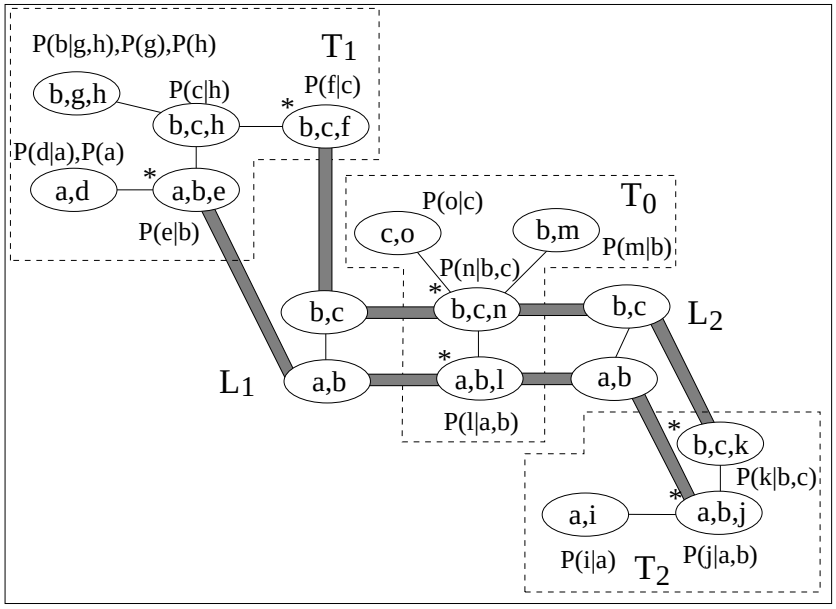

Fig. 1.2. JTs and linkage trees obtained from Fig. 1.1. Each linkage host is labeled by *. The thick links show the relation between each linkage and its hosts.

is assigned to a cluster, $\beta = \emptyset$. The *joint system potential* of the JT is then the product of all potentials in all clusters, denoted as $B(N)$.

Each separator S between two adjacent clusters C and C' is associated with two buffers. One buffer is used to store the message from C to C' and the other from C' to C. We formalize lazy propagation below as pseudo-code algorithms so that we can refer to them in the new inference algorithms for MSBNs. Given a cluster C, for each separator S, we shall refer to the two buffers *locally* as the *in-buffer* and the *out-buffer* relative to C.

A cluster executes the following algorithm to compute and send a message to an adjacent cluster, where $\backslash$ is the set difference operator.

Algorithm 1 (SendPotential) *Let C be a cluster with β. Let adjacent clusters be $C_1, ..., C_m$. Let β_i be the set of potentials in the in-buffer from C_i. When SendPotential relative to C_k is called in C, C does the following:*

(1) $\beta' = \beta \cup_{i \neq k} \beta_i$.

(2) Marginalize out variables $C \backslash C_k$ from β' . (To marginalize out variable x, multiply potentials with x in the domain and apply marginalization to the product.)

(3) Send the resultant set of potentials to the out-buffer to C_k.

In the following two algorithms, C is a cluster and `caller` is an adjacent cluster or the JT. The following algorithm is executed recursively by each cluster for inward message passing.

Algorithm 2 (CollectPotential) *When caller calls CollectPotential in cluster C, C does the following:*

(1) If caller is the only adjacent cluster, perform SendPotential relative to caller.

(2) Otherwise, for each adjacent cluster Q except caller, call CollectPotential in Q. After all calls are completed, perform SendPotential relative to caller if it is an adjacent cluster.

The following algorithm is executed recursively by each cluster for outward message passing.

Algorithm 3 (DistributePotential) *When caller calls DistributePotential in C, for each adjacent cluster Q except caller, C performs SendPotential relative to Q followed by a call of DistributePotential in Q.*

The following algorithm is executed by a JT for a full round of message passing.

Algorithm 4 (UnifyPotential) *Select a cluster C arbitrarily. Call CollectPotential in C. Call DistributePotential in C.*

The following proposition establishes the effect of UnifyPotential, where *const* denotes a constant:

Proposition 5 (Proposition 3.4 in [5]) *Let UnifyPotential be performed in a JT. For any cluster C with β and in-buffer messages β_i $(i = 1, ..., m)$ from separators R_i with adjacent clusters, denote the product of potentials in β as $\beta(C)$ and the product of potentials in β_i as $\beta_i(R_i)$. Then*

$$\beta(C) \prod_{i=1}^{m} \beta_i(R_i) = const \sum_{N \setminus C} B(N).$$

When observations are available, for each cluster, update each potential whose domain contains observed variables and remove the observed variables from the domain. Store the observed values for subsequent queries. The following algorithm is used to enter the observation on a variable to the JT.

Algorithm 5 (EnterObservation) *When a variable x is observed at value x_0, for each cluster C (with β) containing x, do the following:*

(1) Remove each potential $f(x)$ from β.

(2) For each potential $f(x, Y)$ in β, where $Y \neq \emptyset$, replace it by

$$g(Y) = f(x = x_0, Y).$$

The effect of EnterObservation is such that the new joint system potential corresponds to the posterior distribution given the observation. After EnterObservation is performed for each observed variable, followed by an UnifyPotential, the posterior probabilities for each variable can be obtained from any cluster that contains it.

1.4 Lazy Inference With LJFs

We apply lazy propagation to inference in MSBNs. The on-line message computation will be guided by LJFs, but factorized beliefs and messages will be used as in lazy propagation. Each agent A_i is associated with the subnet S_i and local JT T_i.

1.4.1 Potential Assignment

Conditional probability tables (CPTs) in an MSBN are assigned to clusters in its LJF as potentials: For each node x in each subnet S_i, if it is assigned with a non-constant CPT (see Def 4), then assign the CPT to a cluster in local JT T_i that contains x and its parents in S_i. The potential associated with a local JT T_i is then

$$B_{T_i}(N_i) = \prod_j \prod_k \beta_{i,j,k},$$

where j indexes clusters, $\beta_{i,j}$ denotes the set of potentials assigned to the jth cluster, and $\beta_{i,j,k}$ is the kth potential in the set. The *joint system potential* of the LJF is

$$B_F(\mathcal{N}) = \prod_i B_{T_i}(N_i).$$

$B_F(\mathcal{N})$ is identical to jpd of the MSBN.

1.4.2 Lazy Inference: An Example

Lazy inference consists of lazy communication among agents followed by local lazy propagation. During lazy communication, inter-subnet messages are sent through linkage trees. Messages are passed through a linkage tree in both directions. Hence, a linkage between subnets S and R is associated with two message buffers, one for each direction.

Fig. 1.3 illustrates inward propagation with root agent A_0. First, Unify-Potential is performed by A_1 and A_2. At T_1, it causes message

$$B(b,c) = \sum_h P(c|h)B(b,h)$$

to be sent from cluster $\{b,c,h\}$ to $\{b,c,f\}$, where

$$B(b,h) = P(h)\sum_g P(b|g,h)P(g).$$

Similarly, messages $P(a)$ and $B(b) = \sum_h B(b,h)$ are sent from clusters $\{a,d\}$ and $\{b,c,h\}$ to cluster $\{a,b,e\}$, respectively. At linkage host $\{b,c,f\}$, message to linkage $\{b,c\}$ is computed based on local potentials plus the message from

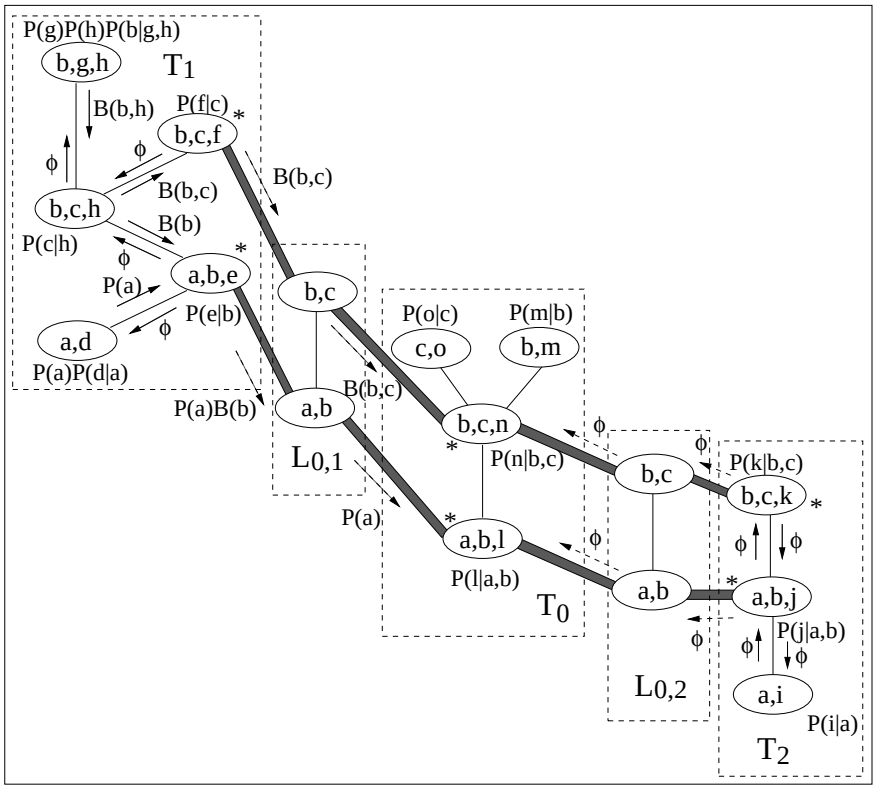

Fig. 1.3. Inward propagation in LJF.

cluster $\{b, c, h\}$. The resultant message is $B(b, c)$. At linkage host $\{a, b, e\}$, message $P(a)B(b)$ to linkage $\{a, b\}$ is computed. As a consequence, both linkages in $L_{0,1}$ contain information on variable b: a duplication. To remove the duplication, A_1 examines potentials at linkage $\{a, b\}$ and identify $B(b)$ as the duplicated information on b. After $B(b)$ is deleted, messages from $L_{0,1}$ to T_0 become $B(b, c)$ through linkage $\{b, c\}$ and $P(a)$ through linkage $\{a, b\}$.

At A_2, UnifyPotential generates only empty messages among clusters. Messages from linkage hosts $\{b, c, k\}$ and $\{a, b, j\}$ to linkages are also empty. This concludes inward propagation.

Outward propagation follows, during which A_0 sends messages to A_1 and A_2. To calculate messages to A_1, A_0 performs UnifyPotential using linkage messages (empty) from A_2 but not those from A_1. All messages (intra as well as inter-subnet) are empty in this case.

Fig. 1.4 shows outward propagation from A_0 to A_2. A_0 performs UnifyPotential using linkage messages from A_1 but not those from A_2. Message from cluster $\{b, c, n\}$ to $\{a, b, l\}$ is $B'(b) = \sum_c B(b, c)$ and all other intra-subnet messages are empty. Message from linkage host $\{b, c, n\}$ through linkage $\{b, c\}$

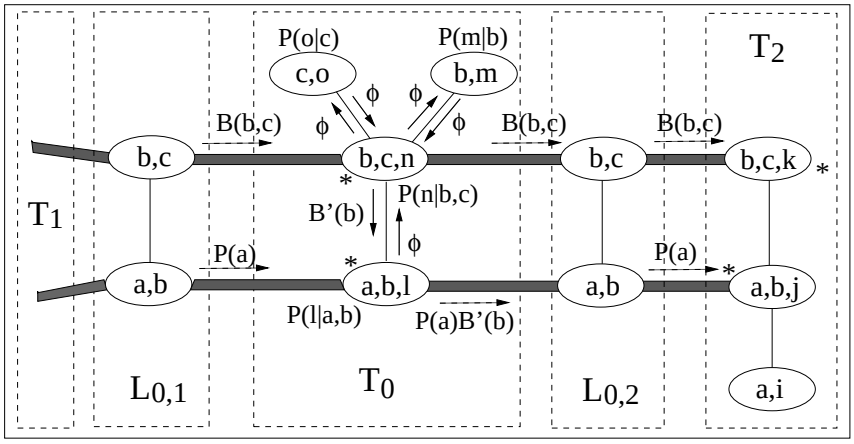

Fig. 1.4. Outward propagation from T_0 to T_2.

to A_2 is $B(b,c)$. The message through linkage $\{a,b\}$ to A_2 is $P(a)B'(b)$. Again, information on variable b is duplicated in the two linkage messages. After duplication $B'(b)$ is deleted from the message to linkage $\{a,b\}$, the resultant messages from A_0 to A_2 are $B(b,c)$ through linkage $\{b,c\}$ and $P(a)$ through $\{a,b\}$. Lazy communication is now complete.

After communication, each agent performs inference in its JT, which allows the prior probability of each variable x to be obtained from any cluster containing x in any subnet. The local inference extends UnifyPotential by including messages from linkages. For instance, to perform UnifyPotential in T_2, cluster $\{b,c,k\}$ includes linkage message $B(b,c)$ in computing the message to cluster $\{a,b,j\}$. To answer a query on $P(b)$, A_2 picks a cluster that contains b, say, $\{b,c,k\}$, and marginalizes the product of local potential $P(k|b,c)$, message from cluster $\{a,b,j\}$ (empty in this case) and linkage message $B(b,c)$. Below, we present inference algorithms of which the above example is a trace.

1.4.3 Local Lazy Propagation

The most primitive operation is SendPotential. To take into account message passing over linkages, we extend SendPotential (Algorithm 1) by extending the notion of *adjacency*: Two clusters are *adjacent* if

(1) they are directly connected in a JT, or

(2) they are hosts of a linkage between two JTs.

We refer to the extended Algorithm 1 as SendPotential*.

We redefine CollectPotential (Algorithm 2) and DistributePotential (Algorithm 3) to process messages over linkages. They use extended adjacency. In the algorithms, C is a cluster in a JT and `caller` is the local agent or an adjacent cluster not connected through a linkage.

Algorithm 6 (CollectPotential*) *When caller calls CollectPotential* in cluster C, C does the following:*

(1) If caller is the only adjacent cluster, perform SendPotential relative to caller.*

(2) Otherwise, for each adjacent cluster Q not connected through a linkage except caller, call CollectPotential in Q. After all calls are completed, perform SendPotential* relative to caller if it is an adjacent cluster.*

Note that CollectPotential* only receives messages from linkage in-buffers and does not send to linkage out-buffers because calling CollectPotential* across linkages is disallowed. Under the multiagent paradigm, CollectPotential* is a local operation of an agent, while sending messages across linkages involves a remote agent. CollectPotential* can be executed autonomously to answer local queries, while message passing across linkages requires coordination and incurs communication cost. Next, we redefine DistributePotential.

Algorithm 7 (DistributePotential*) *When the caller calls operation DistributePotential* in cluster C, for each adjacent cluster Q not connected through a linkage except caller, C performs SendPotential* relative to Q followed by a call of DistributePotential* in Q.*

Local lazy propagation uses Algorithm 4, with CollectPotential* and DistributePotential*, which we refer to as UnifyPotential*.

1.4.4 Lazy Communication

During communication, messages are sent from one agent with JT T to an adjacent agent with JT T' through their linkage tree. The messages are originated from linkage hosts in T. To ensure that each linkage host has the necessary information, UnifyPotential* must be performed before these messages are computed. This renders T locally consistent. As a result, for every two linkages adjacent in the linkage tree, the same information on their shared variables will be sent by their hosts. If such messages are directly passed to T', the new belief in T' will be incorrect due to information duplication. We consider below how to compute cross-linkage messages without information duplication.

To compute messages going from a source JT T to a destination JT T', the linkage tree L can be directed. For each linkage Q in L, the following message buffers are then allocated.

in-buffer$_1$ in-buffer from the host cluster in T.
in-buffer$_2$ in-buffer from the parent linkage in L. If Q has no parent linkage, its in-buffer$_2$ is null.
out-buffer$_1$ out-buffer to the host cluster in T'.
out-buffer$_2$, out-buffer$_3$, ... out-buffers to child linkages in L.

The message from Q to T' is computed as follows:

Algorithm 8 (SendLinkageMsg)
For each linkage Q, Q requests its linkage host to fill in-buffer$_1$ by SendPotential* relative to Q. After both in-buffers are filled, Q does the following:

(1) For each child linkage Q', marginalize out variables $Q \setminus Q'$ from potentials in in-buffer$_1$, and send resultant potentials to the out-buffer to Q'.

(2) Divide the set α of potentials in in-buffer$_1$ by the set γ of potentials in in-buffer$_2$ as follows and sends the resultant α to out-buffer$_1$:

(2.1) If a potential appears in both α and γ, delete it from both.

(2.2) For each potential f in γ, delete f from γ, multiply the set θ of potentials in α whose domains overlop with that of f, and divide the product by f. Replace θ in α by the result of the division.

Note that sending to out-buffer$_1$ involves inter-agent message transmission. Using SendLinkageMsg, algorithms below perform lazy communication in LJFs. In the algorithms, A is an agent and `caller` is the MSBN or an adjacent agent of A. CollectBeliefLLJF defines inward lazy communication along hypertree.

Algorithm 9 (CollectBeliefLLJF) *When caller calls CollectBeliefLLJF in agent A, A does the following:*

(1) If caller is not the only adjacent agent, call CollectBeliefLLJF in each adjacent agent except caller. After all calls are completed, receive linkage messages from each adjacent agent except caller.

(2) If caller is an adjacent agent, do UnifyPotential* using linkage messages from each adjacent agent except caller, followed by SendLinkageMsg relative to caller.

The inward propagation described in Section 1.4.2 is a trace of a call of CollectBeliefLLJF in A_0. A_0 then calls in A_1 and A_2. DistributeBeliefLLJF below defines outward lazy communication along hypertree.

Algorithm 10 (DistributeBeliefLLJF) *When caller calls DistributeBeliefLLJF in A, for each adjacent agent A' except caller, A does UnifyPotential* using linkage messages from each adjacent agent except A', followed by SendLinkageMsg relative to A' and a call of DistributeBeliefLLJF in A'.*

The outward propagation described in Section 1.4.2 is a trace of a call of DistributeBeliefLLJF in A_0. A_0 then calls it in A_1 and A_2. Since A_1 and A_2 have no adjacent agents except A_0, recursive calls terminate. CommunicateBeliefLLJF below combines above algorithms to accomplish lazy inference in a LJF.

Algorithm 11 (CommunicateBeliefLLJF) *Select an agent A arbitrarily. Call CollectBeliefLLJF in A. Call DistributeBeliefLLJF in A. Each agent performs UnifyPotential* using linkage messages from all adjacent agents.*

An agent A calls UnifyPotential* before sending messages to each adjacent agent. If A has k adjacent agents, then one call is made during CollectBeliefLLJF, $k-1$ calls are made during DistributeBeliefLLJF, and a final call is made at the end of CommunicateBeliefLLJF. Hence, a total of $k+1$ rounds of local lazy propagations are needed to complete CommunicateBeliefLLJF.

1.5 Soundness

In the following, we use *const* to denote a positive constant. Proposition 6 says that messages sent over a linkage tree define the marginal potential over the d-sepset.

Proposition 6 *Let T over N be a local JT, T' be a local JT adjacent of T, I be their d-sepset, and L be the linkage tree over I. Let UnifyPotential* be performed in T followed by SendLinkageMsg relative to T'. Let $B(N)$ be the potential*

$$B(N) = \prod_{C \in T} \beta(C) \prod_{Q' \notin L} \beta(Q'),$$

where $\beta(C)$ is the product of potentials assigned to a cluster C, $\beta(Q')$ is the product of potentials received from a linkage Q', and only linkages other than those in L are included. For each linkage $Q \in L$, let $\alpha(Q)$ be the product of potentials that Q sends to T' by SendLinkageMsg. Then

$$\prod_{Q \in L} \alpha(Q) = const \sum_{N \setminus I} B(N).$$

Proof:

First, we consider the effect of UnifyPotential* by applying Proposition 5. To do so, for each cluster C in T, we define the equivalent cluster potential of C as

$$\beta'(C) = \beta(C) \prod_{Q' \to C} \beta(Q'),$$

where $Q' \to C$ means that Q' is a linkage that feeds a message to C. We can then disregard each Q' in the remaining proof and Proposition 5 is now directly applicable.

Next, for any linkage $Q \in L$, consider its linkage host X. From Proposition 5, after UnifyPotential*, the set of potentials (including those from its in-buffers) associated with X defines the marginal of $B(N)$ onto X. This set, marginalized onto Q, is sent to in-buffer$_1$ of Q. Denote the product of potentials in in-buffer$_1$ by $\alpha'(Q)$ and the product of potentials in in-buffer$_2$ by $\theta'(Z)$, where Z is the separator between Q and its parent linkage. By Proposition 7.5 of reference [8], a linkage tree is a JT. Hence,

$$\prod_{Q \in L} \alpha'(Q) / \prod_{Q \in L} \theta'(Z) = const \sum_{N \setminus I} B(N).$$

The proposition follows since the message that Q sends to out-buffer$_1$ is

$$\alpha(Q) = \alpha'(Q)/\theta'(Z).$$

□

The following theorem says that the local potential of an agent and linkage tree messages it receives define the marginal of the joint system potential:

Theorem 7 *Let F over $\mathcal{N}$ be the LJF of an MSBN with the joint system potential $B_F(\mathcal{N})$ and let CommunicateBeliefLLJF be performed in F. Let T be any local JT over N and $B(N)$ be the potential*

$$B(N) = \prod_{C \in T} \beta(C) \prod_{Q \to T} \beta(Q),$$

where $\beta(C)$ is the product of potentials assigned to a cluster C, $\beta(Q)$ is the product of potentials received from a linkage Q into T (denoted by $Q \to T$). Then,

$$B(N) = const \sum_{\mathcal{N} \setminus N} B_F(\mathcal{N}).$$

Proof:

Denote the agent in charge of T as A. Given T, F can be viewed as a directed hypertree with A at the root. During CommunicateBeliefLLJF, only inter-agent messages directed towards A has an impact on $B(N)$. These messages are sent in semi-parallel order from leaves to the root. We analyze the impact of these messages by letting agents send one by one starting from any leaf agent A'.

Since A' (with subdomain N') is a leaf, it is adjacent to only one agent A'' (with subdomain N''). By Proposition 6, messages A' sent to A'' define the marginal of $B(N')$ onto their d-sepset. Since these messages are the only impact that A' has on $B(N)$ and they are received by A'', agent A' is effectively removed from the system. The new joint system potential defined by the local potentials in the remaining agents and the messages A'' received is

$$const \sum_{N' \setminus N''} B_F(\mathcal{N}).$$

By applying the above argument recursively to each leaf agent, eventually, all other agents in F will be removed except A. The result follows.

□

The following corollary states that the local potentials in a cluster and its in-buffer messages define the marginal of the joint system potential. In the corollary, in-buffers include both those from adjacent clusters in the same local JT and those from linkages.

Corollary 8 *Let F over $\mathcal{N}$ be the LJF of an MSBN with the joint system potential $B_F(\mathcal{N})$ and let CommunicateBeliefLLJF be performed in F. Let C be any cluster in any local JT and $B(C)$ be the potential*

$$B(C) = \beta(C) \prod_{R \to C} \beta(R) \prod_{Q \to C} \beta(Q),$$

where $\beta(C)$ is the product of potentials assigned to C, $\beta(R)$ is the product of potentials received from the in-buffer associated with a separator R with an adjacent cluster of C, and $\beta(Q)$ is the product of potentials received from a linkage Q into C. Then,

$$B(C) = const \sum_{\mathcal{N} \setminus C} B_F(\mathcal{N}).$$

Proof:

It follows from Theorem 7 and Proposition 5. Theorem 7 ensures the marginal of $B_F(\mathcal{N})$ onto the subdomain of the local JT and Proposition 5 ensures further marginalization onto C.

□

1.6 Enter Observations

When observation is obtained on a private variable, it can be entered using EnterObservation (Algorithm 5). The effect is that the new joint system potential corresponds to the posterior distribution given the observation. Consider a variable x with its value x_0 observed. If x is a root variable, EnterObservation does two things: (a) It removes $P(x)$. (b) For each child variable y of x, it replaces $P(y|x, \pi(y) \setminus \{x\})$ by $P(y|x = x_0, \pi(y) \setminus \{x\})$. Hence, the new joint system potential corresponds to $P(\mathcal{N} \setminus \{x\}|x = x_0)$.

If x is not a root variable, (a) is not applicable. EnterObservation will, in addition to (b), replace $P(x|\pi(x))$ by $P(x = x_0|\pi(x))$. This is equivalent to an operation

$$\sum_x P(\mathcal{N} \setminus \{x\}, x = x_0) = P(\mathcal{N} \setminus \{x\}|x = x_0).$$

When observation is obtained by an agent A on a public variable, however, the above is not sufficient. By performing EnterObservation in A, the local belief of A is updated. However, x may have parents or children in other agents.[1] Unless they take corresponding actions, the joint system potential has not been updated correctly. We do not require other agents to do so immediately following A's observation as agent communication is costly. Instead, it's desirable that the coordinated observation entering is delayed until the next communication. We therefore modify CollectBeliefLLJF into CollectBeliefLLJF* below.

[1] See [8] for reasons why x may not be observed by all relevant agents.

- For (1), receive observations on d-sepnodes as well as linkage messages.
- For (2), EnterObservation before UnifyPotential*, and send observations on d-sepnodes to caller before SendLinkageMsg.

Similarly, DistributeBeliefLLJF is modified into DistributeBeliefLLJF* by performing EnterObservation before UnifyPotential*.

We refer to Algorithm 11, modified with CollectBeliefLLJF* and DistributeBeliefLLJF*, as CommunicateBeliefLLJF*. The following theorem establishes its effect whose proof is straightforward given Corollary 8 and the above discussion.

Theorem 9 *Let F over $\mathcal{N}$ be the LJF of an MSBN with jpd $P(\mathcal{N})$. Let Obs be the set of variables observed at value obs. Let CommunicateBeliefLLJF* be performed in F after observations on Obs have been entered by the corresponding agents through EnterObservation. Let C be any cluster in any local JT and B_C be the potential*

$$B_C = \beta(C) \prod_{R \to C} \beta(R) \prod_{Q \to C} \beta(Q),$$

where $\beta(C)$ is the product of potentials associated with C, $\beta(R)$ is the product of those received from the in-buffer associated with a separator R with an adjacent cluster of C, and $\beta(Q)$ is the product of potentials received from a linkage Q into C. Then,

$$B_C = const \sum_{C \cap Obs} \sum_{\mathcal{N} \setminus C} P(\mathcal{N}|obs).$$

Note that we have used word 'associated' instead of 'assigned' regarding potentials in C to emphasize the possible change of these potentials due to EnterObservation. We have also used notation B_C instead of $B(C)$ to emphasize that the product does not include observed variables in its domain. The inner summation above marginalizes $P(\mathcal{N}|obs)$ to variables in C and the outer summation marginalizes out any variable in C that has been observed.

The following theorem establishes inference autonomy for each agent. Its proof is trivial given Theorem 9 and Proposition 5.

Theorem 10 *Let observations $Obs' = obs'$ be obtained by agent A after global observations $Obs = obs$ followed by CommunicateBeliefLLJF*. Let A perform EnterObservation relative to obs' followed by UnifyPotential*. Then, for each cluster C in A's local JT,*

$$B_C = const \sum_{C \cap Obs \cap Obs'} \sum_{\mathcal{N} \setminus C} P(\mathcal{N}|obs, obs').$$

1.7 Remarks

We presented an alternative exact method for multiagent inference in MSBNs with a simpler run-time structure than a previously proposed method. In the worst case, the complexity of lazy inference in MSBNs is upper-bounded by that of HUGIN-like inference. However, in average cases, it is expected to be much reduced due to factorized representation of cluster potentials. Further experimental investigation will provide empirical evidence on the actual complexity and comparison among the three inference methods: HUGIN-like inference, that of [6], and the method presented.

Acknowledgements

The funding support from Natural Sciences and Engineering Research Council (NSERC) of Canada to the first author is acknowledged.

References

1. P. Haddawy. 1999. An overview of some recent developments in Bayesian problem-solving techniques. *AI Magazine*, 20(2):11–19.
2. F.V. Jensen. 1996. *An Introduction To Bayesian Networks*. UCL Press.
3. A.L. Madsen and F.V. Jensen. 1998. Lazy propagation in junction trees. In *Proc. 14th Conf. on Uncertainty in Artificial Intelligence*.
4. J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
5. G. Shafer. 1996. *Probabilistic Expert Systems*. Society for Industrial and Applied Mathematics, Philadelphia.
6. Y. Xiang and F.V. Jensen. 1999. Inference in multiply sectioned Bayesian networks with extended Shafer-Shenoy and lazy propagation. In *Proc. 15th Conf. on Uncertainty in Artificial Intelligence*, pages 680–687, Stockholm.
7. Y. Xiang and V. Lesser. 2003. On the role of multiply sectioned Bayesian networks to cooperative multiagent systems. *IEEE Trans. Systems, Man, and Cybernetics-Part A*, 33(4):489–501.
8. Y. Xiang. 2002. *Probabilistic Reasoning in Multi-Agent Systems: A Graphical Models Approach*. Cambridge University Press.