Multiply Sectioned Bayesian Networks and Junction Forests for Large Knowledge-Based Systems

(Published in: Computational Intelligence, Vol.9, No.2, 171-220, 1993)

Yang Xiang

Department of Computing Science
University College of the Cariboo, Kamloops, B.C. Canada, V2C 5N3
David Poole

Department of Computer Science
University of British Columbia, Vancouver, B.C., Canada, V6T 1Z2
Michael P. Beddoes

Department of Electrical Engineering
University of British Columbia, Vancouver, B.C., Canada, V6T 1W5

Abstract

Bayesian networks provide a natural, concise knowledge representation method for building knowledge-based systems under uncertainty. We consider domains representable by general but sparse networks and characterized by incremental evidence where the probabilistic knowledge can be captured once and used for multiple cases. Current Bayesian net representations do not consider structure in the domain and lump all variables into a *homogeneous* network. In practice, one often directs attention to only part of the network within a period of time, i.e., there is "localization" of queries and evidence. In such case, propagating evidence through a homogeneous network is inefficient since the entire network has to be updated each time. This paper derives reasonable constraints, which can often be easily satisfied, that enable a natural (*localization preserving*) partition of a domain and its representation by separate Bayesian subnets. The subnets are transformed into a set of permanent junction trees such that evidential reasoning takes place at only one of them at a time; and marginal probabilities obtained are identical to those that would be obtained from the homogeneous network. We show how to swap in a new junction tree, and absorb previously acquired evidence. Although the overall system can be large, computational requirements are governed by the size of one junction tree.

Key words Knowledge representation, Expert systems, Knowledge-based systems, Bayesian network, Probabilistic reasoning, Reasoning under uncertainty.

1 Introduction

Over the last decade, Bayesian belief networks, combining a graphic representation of a causal domain model and probability theory, have gained increasing popularity as a natural, concise knowledge representation method and a consistent inference formalism for building knowledge-based systems which require reasoning under uncertainty.

Cooper (1990) has shown that probabilistic inference in a general Bayesian net is NP-hard. Several different approaches have been pursued to avoid combinatorial explosion in computation for typical cases, and thus to reduce computational cost. Two classes of approaches can be identified.

One class of approaches explore *approximation*. Stochastic simulation is proposed as one approximate inference scheme (Henrion 1988). Annihilating very small numbers in secondary structures of Bayesian nets is another way to reduce consumption of computer resources (Jensen and Andersen 1990).

Another class of approaches Exploit the structure of the problem to gain efficiency in computing *exact* probabilities. The approach of this paper belongs to this second class. Efficient algorithms have been developed for inference in Bayesian nets with special topologies (Pearl 1986; Heckerman 1990a). Unfortunately many domain models cannot be represented by these special types of Bayesian nets. For general but sparse nets, efficient computation has been achieved by creating a secondary structure with a directed tree topology (Lauritzen and Spiegelhalter 1988) or with an undirected tree topology (Jensen, Lauritzen and Olesen 1990, Shafer and Shenoy 1988). The secondary structures offer also the advantage of trading compile time with running time for systems to be used repeatedly. However, for large applications, the run time overhead (both space and time) is still forbidding. Pruning Bayesian nets with respect to each query instance is yet another exact method with savings in computational cost (Baker and Boult 1990). However, it is hard to know what is relevant a priori where incremental evidence absorption is required. A portion of a Bayesian net may not be relevant given a set of evidence and a set of queries, and therefore can be pruned away before computation. But in light of a piece of new evidence, it may become relevant and could not be restored within the pruning algorithm. Furthermore, the advantage of trading compile time with running time is lost in systems for repeated usage if the network has to be pruned for each set of queries. It is this problem that this paper addresses.

We consider domains representable by general but sparse networks and characterized by incremental evidence. We address *reusable* systems where the probabilistic knowledge can be captured once and be used for multiple cases. Current Bayesian net representations do not consider structure in the domain and lump all variables into a *homogeneous* network. For small applications, this may be appropriate. For a large application domain where evidence arrives incrementally, in practice one often directs attention to only part

of the network within a period of time, i.e., there is "localization" of queries and evidence. More precisely, "localization" means two things. For each phase of a query session, only certain parts of a large network are interesting[1]; and *new* evidence and queries are directed to a small part of a large network repeatedly within a period of time. When this is the case, propagating evidence in the homogeneous network is inefficient since the newly arrived evidence has to be propagated to the entire network before queries can be answered.

A large application domain can often be partitioned naturally in terms of localization. For example (Xiang et al. 1992), a neurologist, assisted by a knowledge-based system, examining a patient with a painful impaired upper limb, may temporarily consider only his findings' implications on a set of possible neuromuscular diseases. He may not start to consider the diagnostic significance of each available laboratory test until he has finished the clinical examination. That is, during the clinical examination, queries and new evidence are repeatedly directed towards a set of clinical symptoms and disease hypotheses. We do not need to consider other parts of the network. After the clinical examination of the patient, the findings highlight certain disease candidates and make others less likely, which may suggest that further nerve conduction studies are of no help at all, and thus no attention will be paid to variables about nerve conduction throughout the diagnosis of this patient. Instead EMG tests form the second stage of the doctor's diagnostic practice. Here evidence and queries for the patient are localized within the clinical and EMG portions. Since EMG tests are usually not comfortable for patients, the neurologist would not perform a test unless it is diagnostically necessary. Thus, he would like to know the updated likelihood of disease hypotheses after each test to see if further tests are necessary and which one can yield the most diagnostic benefit. During this test period, queries and new evidence are localized within a set of EMG tests and disease variables. Therefore, in a knowledge-based system for neuromuscular diagnosis, knowledge about the clinical symptoms and a set of diseases forms a natural subdomain. Knowledge about EMG test results and a subset of diseases forms another subdomain, and knowledge about nerve conduction study results and a different subset of diseases forms yet another subdomain. If this domain is represented in a homogeneous network, each piece of clinical findings has to be propagated to all the EMG and nerve conduction variables which are not *relevant* at the moment. Likewise, after each EMG test, the entire net has to be updated even though the neurologist is only *interested* in planning the next EMG test.

Clearly, the problem is because current Bayesian net representations do not provide means to distinguish variables according to natural subdomains. (Heckerman 1990b) partitions Bayesian nets into small groups of naturally related variables to ease the construction of large networks. But once the construction is finished, the run time representation is still homogeneous.

It can be argued that if groups of naturally related variables in a domain can be identified and represented,

[1] "Interesting" is more restrictive than "relevant". We may not be interested in something even though it is relevant.

the run time computation can be restricted to one group at any given stage of a query session due to the localization. In particular, we may not need to propagate new evidence beyond the current group. Along with the arrival of new evidence, attention can be shifted from one group to another. Chunks of knowledge not required for the current focus of attention remain inactive (but are not thrown away) until the focus of attention shifts and they are activated. This way, the *run time* overhead is governed by the size of the group of naturally related variables, not the size of the application domain. Large computational savings can be achieved when uncertainty about current group needs to be updated *repeatedly*. As demonstrated by Heckerman (1990b), grouping of variables can also help in ease and accuracy in construction of Bayesian networks.

Partitioning a large domain into separate knowledge bases and coordinating them in problem solving have a long history for rule-based expert systems termed *blackboard architectures* (Nii 1986a; 1986b). However, a proper parallel for Bayesian network technology has not appeared yet.

Pearl, in his influential book (1988, page 319), expressed the following ideal:

> "Instead of propagating all the information everywhere, it is possible to assess first the potential impact of every updating operation on the belief of the target node and to limit the updating process so that only relevant information is propagated. Doing so will decrease the amount of data traffic in the network and the amount of computation expended on inference. However, it is important that the information we choose not to propagate be allowed to accumulate at the boundaries and discharge its impact to new areas of knowledge once our current set of belief becomes stagnant."

The technique that we present in this paper provides a computational model to implement this ideal.

This paper derives constraints, which can often be satisfied easily, that enable a natural (localization preserving) partition of a domain and its representation by separate Bayesian subnets. Such a representation is termed *multiply sectioned Bayesian network (MSBN)*. In order to perform efficient evidential reasoning in a general but sparse network, the set of subnets are transformed into a set of junction trees as a secondary representation which is termed a *junction forest*. The junction forest becomes the permanent representation for the reusable system where incremental evidential reasoning takes place. Since the junction trees preserve localization, each of them stands as a computational object which can be used alone during reasoning. Multiple linkages between the junction trees are introduced to allow evidence acquired from previously active junction trees to be absorbed into the newly active junction tree which is of current interest. In this way, the localization naturally existing in the domain can be exploited and the above illustrated idea is realized. The MSBN technique can be viewed as an extension to the d-separation concept (Pearl 1888) and

the junction tree technique (Andersen et al. 1989, Jensen, Lauritzen and Olesen 1990).

Section 2 briefly summarizes the background knowledge and previous research. Section 3 explains why "obvious" solutions to exploit localization do not work, and Section 4 gives an overview of the MSBNs and the junction forests technique. We hope that these two sections will motivate and guide readers into the subsequent sections which present the mathematical theory necessary to the technique. Due to limited space, we have omitted all the proofs in this paper. Readers who are interested in the proofs are referred to Xiang, Poole and Beddoes (1992).

2 Background

2.1 Graphs and hypergraphs

A graph G is a pair (N, E) where $N = \{A_1, \ldots, A_\alpha\}$ is a set of nodes and $E \subseteq \{(A_i, A_j) | A_i, A_j \in N; i \neq j\}$ is a set of links between pairs of nodes in N. A *directed graph* is a graph where links in E are ordered pairs and an *undirected graph* is a graph where links in E are unordered pairs. Links in directed graphs are called *arcs* when their directions are of concern. A *subgraph* of a graph (N, E) is any graph (N^k, E^k) satisfying $N^k \subset N$ and $E^k \subset E$. Given a subset of nodes $N^l \subset N$ of a graph (N, E), the subgraph *induced* by N^l is (N^l, E^l) where $E^l = \{(A_i, A_j) \in E | A_i \in N^l \ \& \ A_j \in N^l\}$. The *union graph* of subgraphs $G^1 = (N^1, E^1)$ and $G^2 = (N^2, E^2)$ is the graph $(N^1 \cup N^2, E^1 \cup E^2)$ denoted $G^1 \sqcup G^2$.

A *path* in graph (N, E) is a sequence of nodes $A_1, A_2, \ldots, A_k$ $(k > 1)$ such that $(A_i, A_{i+1}) \in E$. A path in a directed graph can be directed or undirected (i.e. each arc is considered undirected). A *simple path* is a path with no repeated node except that A_1 is allowed to equal A_k. A *cycle* is a simple path with $A_1 = A_k$. Directed graphs without a directed cycle are called DAGs (directed acyclic graphs). A graph (N, E) is *connected* if for any pair of nodes in N there is an undirected path between them. A graph is *singly connected*, or is a *tree*, if there is a unique undirected path between any pair of nodes. If a graph consists of several unconnected trees, it is called a *forest*. A graph is *multiply connected* if there is a pair of nodes with more than one undirected path between them.

This paper considers only connected DAGs since an unconnected DAG can always be treated as several connected ones. A *subDAG* of a DAG $D = (N, E)$ is defined as any connected subgraph of D. A DAG D is the *union DAG* of subDAG D^1 and D^2 if $D = D^1 \sqcup D^2$.

If there is an arc (A_1, A_2) from node A_1 to A_2, A_1 is called a *parent* of A_2, and A_2 a *child* of A_1. Similarly, if there is a directed path from A_1 to A_k, the two nodes are called, respectively, *ancestor* and *descendent*, relative to each other. The *in-degree* of a node is the number of parents it has.

If for each node in a DAG, links are added between all its parents and the directions on the arcs are dropped, the resultant is the *moral graph* of the DAG. A graph is *triangulated* if every cycle of length > 3

has a chord. A *chord* is a link connecting two nonadjacent nodes. A maximal set of nodes all of which are pairwise linked is called a *clique*.

A *hypergraph* is a pair $(N, \mathbf{C})$ where N is a set and $\mathbf{C} \subseteq 2^N$ (power set of N) is a set of subsets of N. The union of hypergraphs is defined similarly to the union of graphs. The *union hypergraph* of $(N^1, \mathbf{C}^1)$ and $(N^2, \mathbf{C}^2)$ is $(N^1 \cup N^2, \mathbf{C}^1 \cup \mathbf{C}^2)$ denoted $(N^1, \mathbf{C}^1) \sqcup (N^2, \mathbf{C}^2)$. Let (N, E) be a graph, and $\mathbf{C}$ be the set of cliques of (N, E). Then $(N, \mathbf{C})$ is a *clique hypergraph* of graph (N, E). If a clique hypergraph is organized into a tree where the nodes of the tree are labeled with cliques such that for any pair of cliques, their intersection is contained in each of the cliques on the unique path between them then the tree is called a *junction tree* or a *join tree*. The intersection of two adjacent cliques in a junction tree is called the *sepset* of the two cliques.

For formal treatment of the graph theoretical concepts introduced, see Golumbic (1980), Jensen (1988), Lauritzen, Speed and Vijayan (1984).

2.2 Bayesian networks

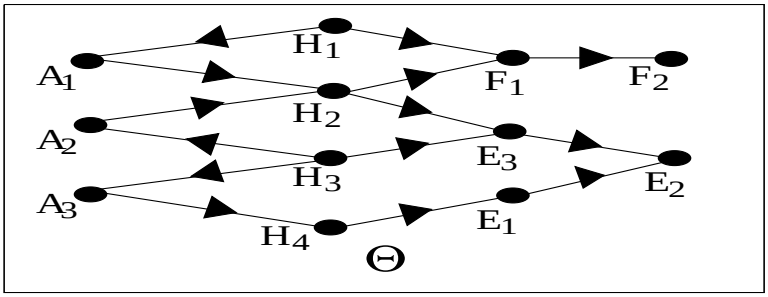

Figure 1: A DAG Θ

A Bayesian network (Pearl 1988) is a triplet (N, E, P).

- N is a set of nodes each of which is labeled with a random variable having a set of *mutually exclusive* and *exhaustive* outcomes. In the context of Bayesian nets, 'node' and 'variable' are used interchangeably.

  Uppercase letters (possibly subscripted) in the beginning of the alphabet are used to denote variables, corresponding script letters to denote their sample spaces, and corresponding lowercase letters with subscripts to denote their outcomes. For example, in binary case, a variable A has its sample space $\mathcal{A} = \{a_1, a_2\}$, and H_i has its sample space $\mathcal{H}_i = \{h_{i1}, h_{i2}\}$. Uppercase letters towards the end of the alphabet are used to denote a set of variables. If $X \subseteq N$ is a set of variables, the *space* $\Psi(X)$ of X is the cross product of sample spaces of the variables $\Psi(X) = \times_{A \in X} \mathcal{A}$. π_i is used to denote the set of parent variables of $A_i \in N$.

For example, Figure 1 shows the DAG of a Bayesian net with a set of nodes

$$\{A_1, A_2, A_3, E_1, E_2, E_3, F_1, F_2, H_1, H_2, H_3, H_4\}$$

. In binary case, the sample space of A_3 is $\mathcal{A}_3 = \{a_{31}, a_{32}\}$.

- E is a set of arcs such that (N, E) is a DAG. The arcs signify the existence of direct causal influences between the linked variables. The basic dependency assumption embedded in Bayesian nets is that a variable is independent of its non-descendants given its parents.

  For example, the arcs in Figure 1 signify the direct causal influences from E_1 and E_3 to E_2. The topology conveys the assumption $p(E_2|E_1E_3H_4) = p(E_2|E_1E_3)$.

- P is a joint probability distribution quantifying the strengths of the causal influences signified by the arcs. P is specified by, for each $A_i \in N$, the distribution of the random variable labeled at A_i conditioned by the values of A_i's parents π_i in the form of a conditional probability table $p(A_i|\pi_i)$. $p(A_i|\pi_i)$ is a normalized function mapping $\Psi(\{A_i\} \cup \pi_i)$ to $[0, 1]$. The joint probability distribution P is

  $$P = p(A_1 \ldots A_\alpha) = \prod_{i=1}^{\alpha} p(A_i|\pi_i)$$

  For example, Table 1 lists the conditional distributions needed to fully specify P for the Bayesian net (Θ, P).

  If we sum a joint distribution over all possible outcomes of the variables in $N \setminus \{A_i\}$, the resultant distribution is the *marginal* distribution over the variable A_i.

$p(h_{11}) =$	.15	$p(a_{21}\|h_{31}) =$	.8	$p(e_{11}\|h_{41}) =$	.8
		$p(a_{21}\|h_{32}) =$	.1	$p(e_{11}\|h_{42}) =$	.15
$p(h_{21}\|a_{21}a_{11}) =$	.8696				
$p(h_{21}\|a_{21}a_{12}) =$	.7	$p(a_{31}\|h_{31}) =$	.3	$p(e_{21}\|e_{31}e_{11}) =$	.9789
$p(h_{21}\|a_{22}a_{11}) =$	.6	$p(a_{31}\|h_{32}) =$	.8	$p(e_{21}\|e_{31}e_{12}) =$	.8
$p(h_{21}\|a_{22}a_{12}) =$	.08			$p(e_{21}\|e_{32}e_{11}) =$	.9
		$p(f_{11}\|h_{11}h_{21}) =$	.7895	$p(e_{21}\|e_{32}e_{12}) =$	.05
$p(h_{31}) =$	.3	$p(f_{11}\|h_{11}h_{22}) =$	.5		
		$p(f_{11}\|h_{12}h_{21}) =$	.6	$p(e_{31}\|h_{21}h_{31}) =$	.7702
$p(h_{41}\|a_{31}) =$	.25	$p(f_{11}\|h_{12}h_{22}) =$	.05	$p(e_{31}\|h_{21}h_{32}) =$	.35
$p(h_{41}\|a_{32}) =$	.4			$p(e_{31}\|h_{22}h_{31}) =$	.65
		$p(f_{21}\|f_{11}) =$	.4	$p(e_{31}\|h_{22}h_{32}) =$	.01
$p(a_{11}\|h_{11}) =$	.8	$p(f_{21}\|f_{12}) =$	.75		
$p(a_{11}\|h_{12}) =$	.1				

Table 1: Probability distribution associated with DAG Θ in Figure 1.

2.3 Operations on belief tables

A *belief table* (Andersen et al. 1989; Jensen, Olesen, and Andersen 1990) or a *potential* (Lauritzen and Spiegelhalter 1988) denoted as $B()$ is a non-normalized probability distribution. It can be viewed as a

function from the space of a set of one of more variables to the reals. For example, the belief table $B(X)$ of a set X of variables maps $\Psi(X)$ to the reals. If $\mathbf{x} \in \Psi(X)$, the belief value of $\mathbf{x}$ is denoted by $B(\mathbf{x})$. Denote a set X of variables and corresponding belief table $B(X)$ with an ordered pair $(X, B(X))$ and call the pair a *world*.

For $Y \subseteq X$, the *projection* $\mathbf{y} \in \Psi(Y)$ of $\mathbf{x} \in \Psi(X)$ to the space $\Psi(Y)$ is denoted as $Prj_{\Psi(Y)}(\mathbf{x})$. Denote the *marginalization* of $B(X)$ to $Y \subseteq X$ by $\sum_{X \setminus Y} B(X)$ which specifies a belief table on Y. The operation is defined as the following: if $B(Y) = \sum_{X \setminus Y} B(X)$ then for all $\mathbf{y} \in \Psi(Y)$,

$$B(\mathbf{y}) = \sum_{Prj_{\Psi(Y)}(\mathbf{x}) = \mathbf{y}} B(\mathbf{x}).$$

Similarly, denote the *multiplication* of $B(X)$ and $B(Y)$ by $B(X) * B(Y)$ which specifies a belief table on $X \cup Y$. If $B(X \cup Y) = B(X) * B(Y)$ then for all $\mathbf{z} \in \Psi(X \cup Y)$, $B(\mathbf{z}) = B(\mathbf{x}) * B(\mathbf{y})$ where $\mathbf{x} = Prj_{\Psi(X)}(\mathbf{z})$ and $\mathbf{y} = Prj_{\Psi(Y)}(\mathbf{z})$. Denote the *division* of $B(X)$ over $B(Y)$ by $B(X)/B(Y)$ which specifies a belief table on $X \cup Y$. If $B(X \cup Y) = B(X)/B(Y)$ then for all $\mathbf{z} \in \Psi(X \cup Y)$, $B(\mathbf{z}) = B(\mathbf{x})/B(\mathbf{y})$ if $B(\mathbf{y}) \neq 0$ where $\mathbf{x} = Prj_{\Psi(X)}(\mathbf{z})$ and $\mathbf{y} = Prj_{\Psi(Y)}(\mathbf{z})$.

2.4 Transform Bayesian nets into junction trees

The MSBN technique is an extension to the junction tree technique (Andersen *et al.* 1989; Jensen, Lauritzen and Olesen 1990) which transforms a Bayesian net into an equivalent secondary structure where inference is conducted (Figure 4). Because of this restructuring, belief propagation in multiply connected Bayesian nets can be performed in a similar manner as in singly connected nets. The following briefly summarizes the junction tree technique.

Moralization Transform the DAG into its moral graph, e.g. Φ in Figure 2 (with respect to Θ in Figure 1).

Triangulation Triangulate the moral graph. Call the resultant graph a *morali-triangulated* graph, e.g. Λ of Figure 2.

Clique hypergraph formation Identify cliques of the morali-triangulated graph, e.g. the nodes in Γ of Figure 2, and obtain a clique hypergraph.

Junction tree construction Organize the clique hypergraph into a junction tree of cliques, e.g. Γ of Figure 2.

Node assignment Assign each node in the DAG to a clique in the junction tree of cliques. For example, H_4 is assigned to either clique 7 or 8, and H_3 is assigned to clique 5, 6, 7 or 8.

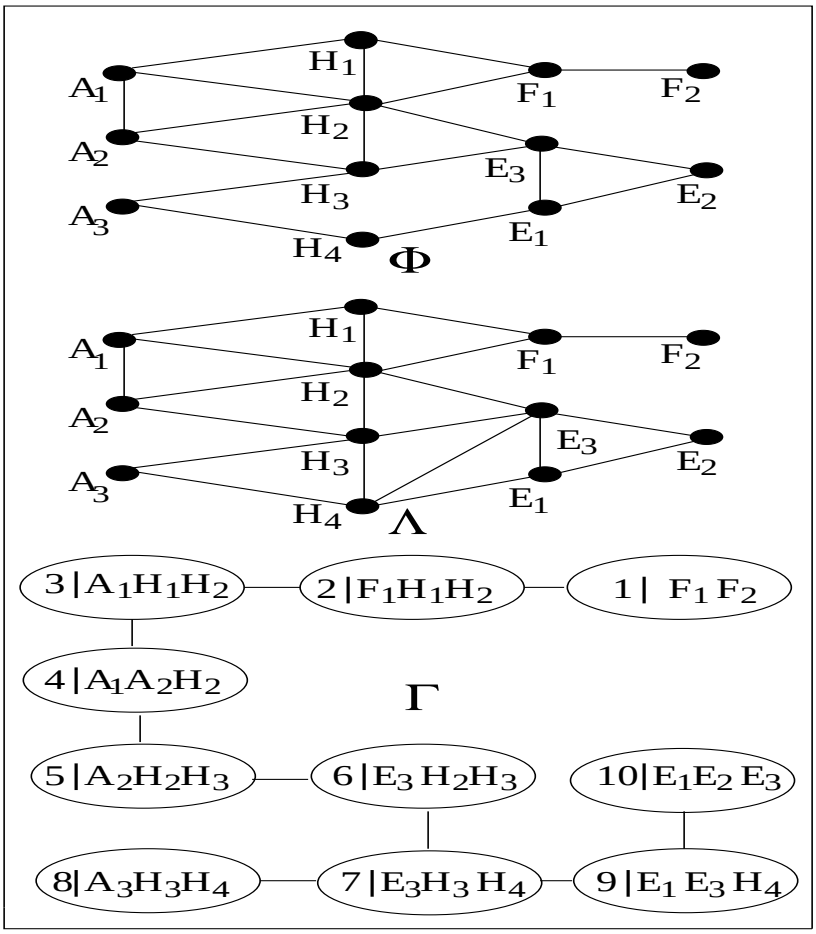

Figure 2: The moral graph Φ of the DAG Θ in Figure 1, one of Φ's triangulated graphs Λ, and the corresponding junction tree Γ. Each clique in Γ is numbered (the number is separated from clique members by a '|').

Belief universes formation For each clique C_i in the junction tree of cliques, obtain its belief table $B(C_i)$ by multiplication of the conditional probability tables of its assigned nodes. Call $(C_i, B(C_i))$ a *belief universe*. When it is clear from the context, no distinction is made between a junction tree of cliques and a junction tree of belief universes.

Inference is conducted through the junction tree representation. An *absorption* operation is defined for local belief propagation. Global belief propagation is achieved by a forward propagation operation *DistributeEvidence* and a backward propagation operation *CollectEvidence*.

Belief initialization Before any evidence can be entered to the junction tree, the belief tables are made *consistent* by *CollectEvidence* and *DistributeEvidence* such that *prior* marginal probability for a variable (of the original Bayesian net) can be obtained by marginalization of a belief table in any universe which contains it.

Evidential reasoning When evidence about a set of variables (of the original Bayesian net) is available, the evidence is entered into universes which contain the variables. Then the belief tables of the junction tree are made consistent again by *CollectEvidence* and *DistributeEvidence* such that *posterior* marginal probability for a variable can be obtained from any universe containing the variable.

The computational complexity of evidential reasoning in junction trees is about the same as the reasoning method by Lauritzen and Spiegelhalter (1988) which can be viewed as performed on a (secondary) directed tree structure (Shachter 1988; Neapolitan 1990). But junction trees are undirected and allow more flexible computation. The junction tree representation is explored in this paper since its flexibility is of crucial importance to the MSBN extension.

2.5 d-separation

The concept of d-separation introduced by Pearl (1988, page 116-118) is fundamental in probabilistic reasoning in Bayesian networks. It permits easy determination, by inspection, of which sets of variables are considered independent of each other given a third set, thus making any DAG an unambiguous representation of dependency and independence. It plays an important role in our partitioning of Bayesian networks.

Definition 2.1 (d-separate (Pearl 1988)) *If X, Y, and Z are three disjoint subsets of nodes in a DAG, then Z is said to* **d-separate** *X from Y, if there is no path between a node in X and a node in Y along which two conditions hold: (1) every node with converging arcs (***head-to-head node***) is in Z or has a descendent in Z and (2) every other node (***non-head-to-head node***) is outside Z.*

A path satisfying the conditions above is said to be active*; otherwise it is said to be* blocked *by Z.*

For example (Figure 1), $\{F_1\}$ d-separates $\{F_2\}$ from $\{H_1, H_2\}$. $\{H_2, H_3, H_4\}$ d-separates $\{E_1, E_2, E_3\}$ from the rest. The path between A_3 and E_1 is blocked by H_4. Detailed illustrations of d-separation can be found in Neapolitan (1990, page 192-207).

The importance of d-separation is that, in a Bayesian network, X and Y are conditionally independent given Z iff Z d-separates X from Y (Geiger, Verma and Pearl 1990).

3 "Obvious" Ways to Explore Localization

"Localization" means the following: (1) For an average query session, only certain parts of a large network are interesting. We would like to concentrate on the part of current interest without the overhead of the uninteresting parts. We don't want to remove those parts a priori as what seemed initially uninteresting may become interesting, and we would like to pay only a small cost to 'swap' those parts in. (2) New evidence

and queries are directed to a small part of a large network repeatedly within a period of time. Making use of this, we only want to incur the swapping cost occasionally.

An obvious way to explore localization in multiply connected networks is to preserve localization within subtrees of a junction tree by clever choice of triangulation and junction tree construction. If this can be done, the junction tree can be split and each subtree can be used as a separate computational object. The following example shows that this is not always workable. Consider the DAG Θ in Figure 2. Suppose variables in the DAG form three groups naturally related which satisfy localization:

$$\begin{aligned} G_1 &= \{A_1, A_2, A_3, H_1, H_2, H_3, H_4\} \\ G_2 &= \{F_1, F_2, H_1, H_2\} \\ G_3 &= \{E_1, E_2, E_3, H_2, H_3, H_4\} \end{aligned}$$

We would like to construct a junction tree which preserves the localization within three subtrees. The graph Φ in Figure 2 is the moral graph of Θ. Only the cycle $A_3 - H_3 - E_3 - E_1 - H_4 - A_3$ needs to be triangulated. There are six distinct ways of triangulation out of which only two do not mix nodes in different groups. The two triangulations have the link (H_3, H_4) in common and which is chosen does not make a significant difference in the following analysis. The Λ in Figure 2 shows one of the two triangulations. The nodes of graph Γ are all the cliques in Λ.

The junction tree Γ does not preserve localization since cliques 3, 4, 5 and 8 correspond to group G_1 but are connected via cliques 6 and 7 which contains E_3 from group G_3. This is unavoidable. When there is evidence towards A_1 or A_2 in Λ, updating the belief in group G_3 requires passing the joint distribution over H_2 and H_3. But updating the belief in A_3 only requires passing the marginal distribution of H_3. That is to say, updating the belief in A_3 needs less information than group G_3. In the junction tree representation, this becomes a path from cliques 3, 4 and 5 to clique 8 via cliques 6 and 7.

In general, let X and Y be two sets of variables in the same natural group, and let Z be a set of variables in a distinct group. Suppose the information exchange between pairs of them requires the exchange of distribution on sets I_{XY}, I_{XZ} and I_{YZ} of variables respectively. Sometime I_{XY} is a subset of both I_{XZ} and I_{YZ}. When this is the case, a junction tree representation will always indirectly connect cliques corresponding to X and Y through cliques corresponding to Z if the method in Andersen et al. (1989), Jensen, Lauritzen and Olesen (1990) is followed.

However, there is a way around the problem with a brute force method. In the above example, when there is evidence towards A_1 or A_2, the brute force method pretends that updating the belief in A_3 needs as much information as G_3. A dummy link (H_2, A_3) is added to the moral graph Φ in Figure 2. Then triangulating the augmented graph gives the graph Λ' in Figure 3. The resultant junction tree Γ' in Figure 3

does have three subtrees which correspond to the three groups desired. However, the largest cliques now have size four instead of three as before. In the binary case, the size of the total state space is 84 instead of 76 as before.

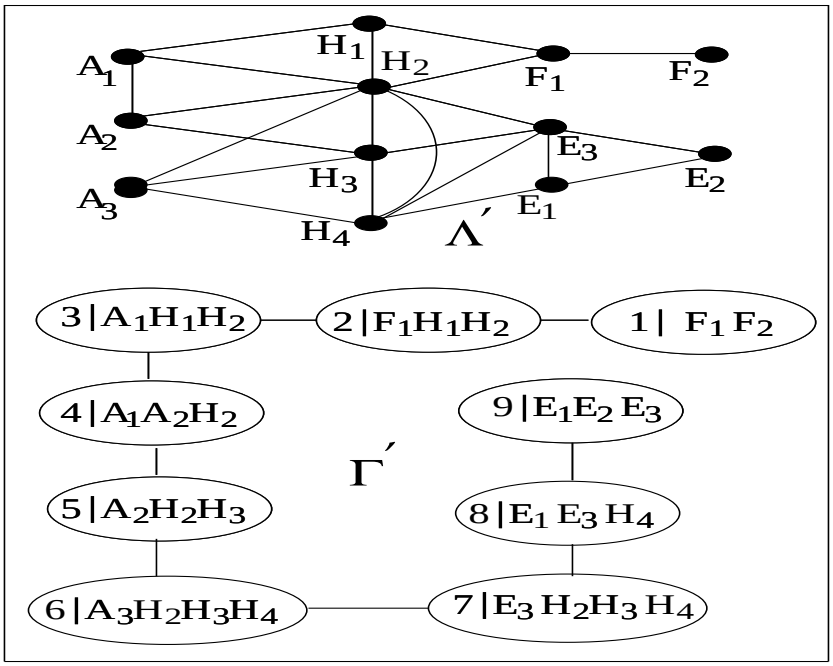

Figure 3: Λ' is a triangulated graph. Γ' is a junction tree of Λ'.

In general, the brute force method preserves natural localization by congregation of the set of interfacing nodes (nodes H_2, H_3, H_4 above) between natural groups. In this way, the joint distribution on interfacing nodes[2] can be passed between groups, and preservation of localization and preservation of tree structure can be compatible. However, in a large application domain with the original network sparse, this will greatly increase the amount of computation in each group due to the exponential enlargement of the clique state space. The computation amount increased could outweigh the savings gained by exploring localization in general.

The trouble illustrated in the above two situations can be traced to the tree structure of a junction tree representation which insists on single path between any two cliques in the tree. The normal triangulation case has small cliques but loses localization. The brute force case preserves localization but does not have small cliques. To summarize, we have shown that the preservation of natural localization and small cliques cannot coexist by the method of Andersen et al. (1989), Jensen, Lauritzen and Olesen (1990). It is claimed here that this is due to a single information channel between local groups of variables. In the following, it

[2]It will be shown later that when the set of interfacing nodes possesses a certain property, the joint distribution on the set is the sufficient information to be exchanged.

is shown that by introducing multiple information channels between groups and by exploring conditional independence, the joint distribution on a set of interfacing variables can be passed between groups by passing only marginal distributions on subsets of the set.

4 Overview of MSBNs and the Junction Forest Technique

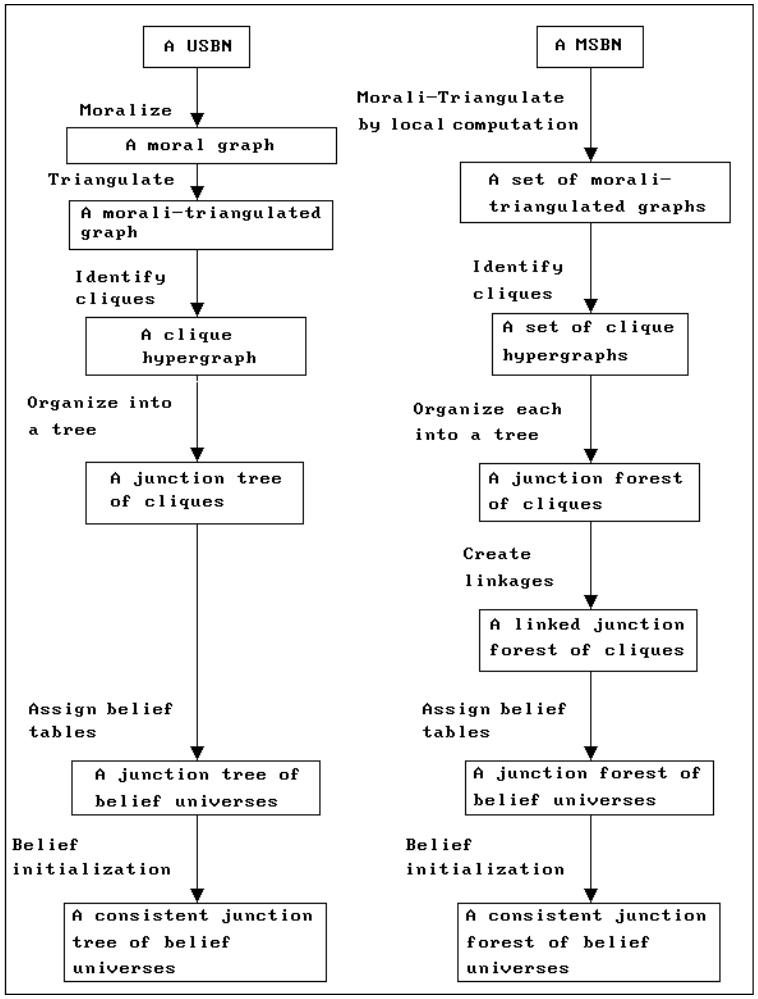

Figure 4: Left: major steps in transformation of a USBN into a junction tree. Right: major steps in transformation of a MSBN into a junction forest.

As demonstrated in Section 3, in order to explore localization, the tree structure and the single channel requirement must be relaxed. Since the computational advantage offered by a tree structure has also been demonstrated repeatedly, it is not desirable to totally abandon the tree structure. Rather, we keep the tree structure within each natural group, but to allow multiple channels between groups. To implement this idea, the MSBN and the junction forest representations extend the d-separation concept and the junction

tree technique. This section outlines the development of these representations. Each major step involved is described in terms of its functionality. The problems possibly encountered and the hints for solutions are discussed. The details are presented in the subsequent sections. Since the technique extends the junction tree technique reviewed in Section 2.4, the parallels and the differences are indicated. Figure 4 illustrates the major steps in the transformation of the original representation into the secondary representation for both techniques.

The d-sepset We want to partition a large domain according to natural localization into subdomains such that each can be represented separately by a Bayesian subnet; and that these subnets can cooperate with each other during inference by exchanging small amount of information between them. We discuss here the technical constraints which have to be followed during the partition in order to solve these goals. This problem can be formulated conceptually in the opposite direction. Suppose the domain has been represented with a homogeneous network. The task is to find the necessary technical constraints to be followed when the net is partitioned into subnets according to natural localization. Section 5 defines *d-sepsets* whose joint distribution is the sufficient information to be exchanged to keep 'adjacent' subnets informed. It is shown that in the junction tree representation of the homogeneous net, the nodes in d-sepsets can serve as information passageways between nodes in different subnets. Thus, the d-sepsets form the interfaces between pairs of subnets.

Sectioning Continuing in the conceptual direction, Section 6 describes how to *section* a homogeneous Bayesian net into subnets called *sect*s. The collection of these sects forms a *MSBN*. It is described how the probability distribution should be assigned to sects relative to the distribution in the homogeneous network. Particularly, it is necessary to assign the original probability table of a d-sepnode to a unique sect which contains the d-sepnode and all its parent nodes, and to assign the same d-sepnode in other sects a uniform table.

In order to perform efficient inference in a general but sparse network, each sect is transformed into a separate junction tree which will stand as an inference entity. When doing so, it is necessary to preserve the intactness of the clique hypergraph resulting from the corresponding homogeneous net. That is, we have to ensure that each clique in the original hypergraph will find at least one host sect. This imposes another constraint, termed *soundness of sectioning*, on the overall organization of the sects. Section 6.2 discusses this constraint.

In addition to a necessary and sufficient condition for soundness of sectioning, two sufficient and natural conditions, that can easily be checked, are provided. The conditions are specified by two rules, the *covering*

subDAG rule and the *hypertree* rule, which, if followed, guarantee sound sectioning. The two rules plus the d-sepset interface impose conditional independence constraints at a macro level (at the level of the sects as opposed to conditional independence at the level of the nodes). This is discussed in Section 6.3. Although there exists MSBNs of sound sectioning which do not follow the two rules, it is shown that computational advantages are obtained in MSBNs sectioned according to the rules. Further discussion will therefore only be directed to MSBNs satisfying the two rules.

Moralization and triangulation To transform a MSBN into a set of junction trees requires moralization and triangulation as reviewed in Section 2.4. In the MSBN context, the transformation can be performed globally or by local computation at the level of the sects. The global computation performs moralization and triangulation in the same way as in the junction tree technique with care not to mix the nodes of distinct sects into one clique. An additional mapping of the resultant morali-triangulated graph into subgraphs corresponding to the sects is needed. But where space saving is concerned, local computation is desired. The pitfalls and procedures involved in moralization and triangulation by local computation are discussed.

Since the number of parents for a d-sepnode may be different for different sects, the moralization in MSBN cannot be achieved by "pure" local computation in each sect. Communication between the sects is required to ensure parent d-sepnodes are moralized identically in different sects.

The criterion of triangulation in the MSBN is to ensure the "intactness" of a resulting hypergraph from the corresponding homogeneous net. Problems arise if we insist on triangulation by local computation at the level of sects. One problem is that an inter-sect cycle will be triangulated in the homogeneous net, but the cycle cannot be identified by examining each of the sects involved individually. Another problem is that d-sepnodes may be triangulated differently in different sects. The solution is to let the sects communicate during triangulation. Since moralization and triangulation both involve adding links and both require communication between sects, the corresponding local operations in each sect can be performed together and messages to other sects can be sent together. Therefore, operationally, moralization and triangulation in MSBN are not separate steps as in the junction tree technique. The corresponding integrated operation is termed *morali-triangulation* to reflect this.

In Section 7.1, the above concept of "intactness" of the hypergraph is formalized in terms of *invertibility* of morali-triangulation. It is shown that if the sectioning of a MSBN is sound then there exists an invertible morali-triangulation such that the "intactness" of the hypergraph is preserved. Section 7.1 provides an algorithm for an invertible morali-triangulation assuming a covering subDAG.

Next steps in the junction tree technique In the junction tree technique, after triangulation, further steps of the transformation are the identification of cliques in the morali-triangulated graph (clique hypergraph formation) and the junction tree construction. In MSBNs, these steps are performed in a similar way for each sect as in the junction tree technique. A MSBN is thus transformed into a set of junction trees called a *junction forest of cliques*. Readers are referred to Andersen *et al.* (1989), Jensen, Lauritzen and Olesen (1990) for technique details involving these steps.

Linkage formation An important extension of MSBNs and junction forests to the junction tree technique is the formation of multiple information channels between junction trees (in a junction forest) such that a joint distribution on a d-sepset can be passed between a pair of junction trees by passing through marginal distributions on subsets of the d-sepset. In this way, the exponential enlargement of the clique state space caused by the brute force method (Section 3) can be avoided. These channels are termed *linkages* (Section 7.2). Each linkage is a set of d-sepnodes which links two cliques. The two cliques are from the pair of junction trees involved, respectively. During inference, if evidence is obtained from previously active junction tree, it can then be propagated to the newly active junction tree through linkages between them.

If we built this naively, multiple linkages could cause redundant information passing or could confuse the information receiver. The problem can be avoided by coordination among linkages during information passing. Since the problem manifests differently during belief initialization and evidential reasoning, the two cases are treated separately. In both cases, information passing is performed one linkage at a time. During initialization, (redundant) information already passed through other linkages is removed from the linkage belief table before the latter is passed over. Operationally, the linkages are ordered. The intersection of a linkage with the union of those linkages ordered before is called the *redundancy set* of the linkage. The redundancy set tells a linkage what portion of the information has to be removed during information passing. During evidential reasoning, the operation DistributeEvidence (Section 2.4) is performed after information passing. The junction forest of cliques with linkages and redundancy sets, forms a *linked junction forest of cliques*.

Formation of joint system belief of junction forest The joint system belief of the junction forest, defined (Section 7.3) in terms of the belief on each of the junction trees, is proportional to the joint probability distribution of the homogeneous net. The junction forest with the joint system belief defined, forms a *junction forest of belief universes*. When it is clear from the context, only "junction forest" is used, without differentiating between its different stages.

Consistency and separability of junction forest As in the case of the junction tree technique, we would like to obtain the marginal probability of a variable by marginalization of the belief in any belief universe of any junction tree which contains the variable. In the case of the junction tree technique, this requires the *consistency* property which can be satisfied by DistributeEvidence and CollectEvidence as reviewed in Section 2.4. In the context of a junction forest, an additional property called *separability* is required (Section 8) due to multiple linkages between junction trees. It imposes a *host composition* constraint on the composition of linkage host cliques. The function of linkages is to pass the joint belief of the corresponding d-sepset. It is shown that if all the junction trees in a junction forest satisfy the host composition condition then separability is guaranteed. Why these conditions usually hold naturally is explained. The remedy when the condition does not hold is also discussed. A junction forest structure satisfying separability, and with a set of operations performed to bring the forest into consistency, can obtain marginal probabilities by local computation.

Belief initialization Belief initialization (Section 9.3) in a junction forest is achieved by first bringing the belief universes in each junction tree into consistency, and then exchanging prior belief between junction trees to bring the junction forest into global consistency. When exchanging beliefs, care is to be taken on two issues. First, non-trivial information (recall that d-sepnodes in some sects are assigned uniform tables during sectioning) could be contained in either side of the two junction trees involved. Second, redundant information could be passed through multiple linkages. Section 9 defines several levels of operations to initialize belief of a junction forest by local computation at the level of junction trees.

Evidential reasoning Only one junction tree in a junction forest needs to be active. Whenever new evidence becomes available to the currently active junction tree, it is entered and the tree is made consistent such that queries can be answered. Thus, the computation complexity of evidential reasoning is governed by the size of one sect. When the user shifts attention, a new junction tree replaces the currently active tree and all previously acquired evidence is absorbed through an operation *ShiftAttention*. The operation requires only a chain of 'intermediate' junction trees to be updated. During the inter-junction tree updating, we need to ensure no confusion results from multi-linkage information passing.

5 The d-sepset and the Junction Tree

5.1 The d-sepset

As discussed in Section 4, the problem of partitioning a Bayesian net by natural localization can be conceptually formulated as though the domain has been represented with a homogeneous network. The task is to

find the technical constraint to partition the net into subnets such that the subnets can be used separately and cooperatively during inference with small amount of information exchange. This section defines the most important concept for partitioning, namely, d-sepset. Then some insights are provided into its implication in the secondary structure of DAGs.

Definition 5.1 (d-sepset) *Let $D = D^1 \sqcup D^2$ be a DAG. The set of nodes $I = N^1 \cap N^2$ is a* **d-sepset** *between subDAG D^1 and D^2 if the following condition holds*[3].

For every $A_i \in I$ with its parents π_i in D, either $\pi_i \subseteq N^1$, or $\pi_i \subseteq N^2$.

Elements of a d-sepset are called **d-sepnodes**. *When the above condition holds, D is said to be* **sectioned** *into $\{D^1, D^2\}$.*

Note that in general a DAG $D = D^1 \sqcup D^2$ does not imply the sectioning of D into $\{D^1, D^2\}$. This is because the intersection of the corresponding two sets of nodes may not be a d-sepset.

Lemma 5.2 *Let a DAG D be sectioned into $\{D^1, D^2\}$ and $I = N^1 \cap N^2$ be a d-sepset. I d-separates $N^1 \setminus I$ from $N^2 \setminus I$.*

The lemma can be generalized into the following theorem which states that, although the d-sepset is pairwise defined, the union of d-sepsets of a subDAG with other subDAGs globally d-separate the subDAG from the rest of the DAG. Note that when a d-sepset is indexed with two superscripts, their order is immaterial.

Theorem 5.3 *Let a DAG D be sectioned into $\{D^1, \ldots, D^\beta\}$ and $I^{ij} = N^i \cap N^j$ be the d-sepset between D^i and D^j. For each i, $\cup_{j \neq i} I^{ij}$ d-separates $N^i \setminus \cup_{j \neq i} I^{ij}$ from $N \setminus N^i$.*

The theorem implies that the joint distribution on d-sepsets is the sufficient information to be exchanged between a Bayesian subnet and the rest of the network.

Corollary 5.4 *Let (D, P) be a Bayesian net, D be sectioned into $\{D^1, \ldots, D^\beta\}$, and $I^{ij} = N^i \cap N^j$ be the d-sepset between D^i and D^j. When evidence is available at variables in N^i, the propagation of the joint distribution on $\cup_{j \neq i} I^{ij}$ from D^i to the rest is sufficient in order to obtain posterior distribution on N.*

Example 5.5 The DAG Θ in Figure 2 is sectioned into $\{\Theta^1, \Theta^2, \Theta^3\}$ in Figure 5. $I^{12} = \{H_1, H_2\}$ is the d-sepset between Θ^1 and Θ^2; $I^{13} = \{H_2, H_3, H_4\}$ is the d-sepset between Θ^1 and Θ^3; and $I^{23} = \{H_2\}$ is the d-sepset between Θ^2 and Θ^3. $I^{12} \cup I^{13} = \{H_1, H_2, H_3, H_4\}$ d-separates the rest of Θ_1 from the rest of Θ_2 and Θ_3.

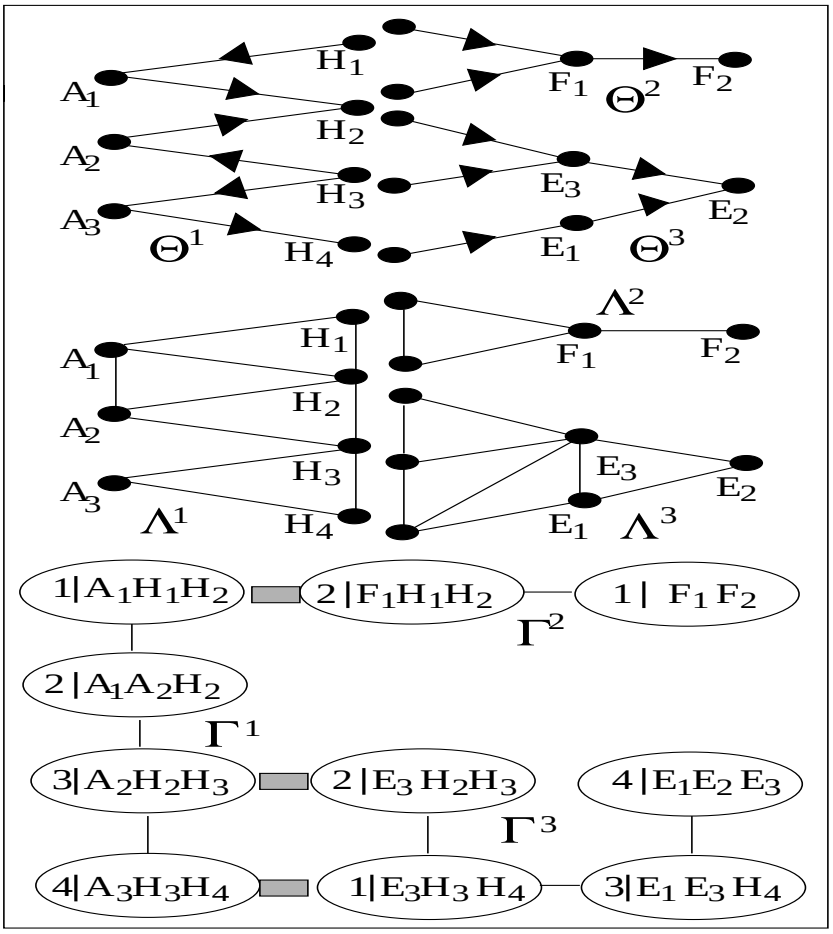

Figure 5: The set $\{\Theta^1, \Theta^2, \Theta^3\}$ of three subDAGs (top) forms a sectioning of Θ in Figure 1. $\{\Lambda^1, \Lambda^2, \Lambda^3\}$ (middle) is the set of morali-triangulated graphs of $\{\Theta^1, \Theta^2, \Theta^3\}$, and $\Xi = \{\Gamma^1, \Gamma^2, \Gamma^3\}$ (bottom) is the corresponding junction forest. The ribbed bands indicate linkages.

There is a close relation between d-sepset and usual graph separator given in proposition 5.6. If Z is the graph separator of X and Y then the removal of the set Z of nodes from the graph (together with their associated links) would render the nodes in X disconnected from those in Y.

Proposition 5.6 *Let a DAG D be sectioned into $\{D^1, D^2\}$. The set of nodes $I = N^1 \cap N^2$ is a d-sepset between D^1 and D^2 iff I is a graph separator in the moral graph of D.*

The properties of d-separation in the DAG representation of Bayesian networks have been studied extensively (Pearl 1988; Geiger, Verma and Pearl 1990). It can be used to derive Pearl's propagation algorithm in singly-connected Bayesian nets (Neapolitan 1990). But to our knowledge, its implication in secondary structure has not been examined. The definition of the d-sepset now allows to do so.

[3]By the definition of $\sqcup$, there can be no arcs from $D^1 \setminus I$ to $D^2 \setminus I$.

5.2 Implication of d-sepset in junction trees

Representing a multiply connected Bayesian network in a secondary structure, namely a junction tree, enables flexible and efficient belief propagation. With the d-sepset concept defined, we would like to know how information is passed in the junction tree between nodes separated by the d-sepset in the original Bayesian network.

Lemma 5.7 *Let a DAG D be sectioned into $\{D^1, D^2\}$ and $I = N^1 \cap N^2$ be the d-sepset. A junction tree T can be constructed from D, such that the following statement is true.*

> *For all pairs of nodes $A_1 \in N^1 \setminus I$ and $A_2 \in N^2 \setminus I$, if A_1 is contained in clique C_1 and A_2 in C_2, then on the unique path between C_1 and C_2 in T, there exists a clique sepset Q containing only d-sepnodes.*

The lemma can be generalized to the case of any finite number of subDAGs. This is the following proposition. Its proof is similar to the lemma.

Proposition 5.8 (belief relay) *Let a DAG D be sectioned into $\{D^1, \ldots, D^\beta\}$ and $I = \cup_{j \neq i} I^{ij}$ be the union of d-sepsets between D^i and other subDAGs. A junction tree T can be constructed from D, such that the following statement is true.*

> *For all pairs of nodes $A_1 \in N^i \setminus I$ and $A_2 \in N \setminus N^i$, if A_1 is contained in clique C_1 and A_2 in C_2, then on the unique path between C_1 and C_2 in T, there exists a clique sepset Q containing only d-sepnodes in I.*

Example 5.9 Recall the DAG Θ in Figure 2 which is sectioned into $\{\Theta^1, \Theta^2, \Theta^3\}$ in Figure 5 with $I = \{H_2, H_3, H_4\}$ being the d-sepset between Θ^1 and Θ^3. Consider the node A_3 in clique $\{H_3, H_4, A_3\}$ and the node E_2 in clique $\{E_4, E_3, E_2\}$ in the junction tree Γ in Figure 2. In the path between the two cliques, the sepset $\{H_3, H_4\}$ between cliques $\{H_3, H_4, A_3\}$ and $\{H_3, H_4, E_3\}$ contains only d-sepnodes.

When new evidence is available, it can be propagated to the junction tree through sepsets between cliques (Jensen, Lauritzen and Olesen 1990). Therefore, the above proposition means that a junction tree can be constructed such that evidence in $N^i \setminus I$ must pass through at least one sepset containing only nodes in I in order to be propagated to nodes in $N \setminus N^i$.

Theorem 5.3 and Proposition 5.8 suggest that the clique hypergraph can be organized such that the cliques corresponding to different subDAGs separated by d-sepsets can be organized into different junction trees. Communication between them can be accomplished through d-sepsets. This idea is formalized below.

6 Multiply Sectioned Bayesian Nets

6.1 Definition of MSBN

Definition 6.1 (MSBN) *Let $S = (N, E, P)$ be a Bayesian network. Suppose $D = (N, E)$ is sectioned into $\{D^1, \ldots, D^\beta\}$ where $D^i = (N^i, E^i)$. Suppose $I^{ij} = N^i \cap N^j$ is the d-sepset between D^i and D^j $(1 \leq i, j \leq \beta; i \neq j)$.*

Each d-sepnode A is assigned to a subDAGs in the following way:

> *Let η^i be the in-degree of A in subDAG D^i. Choose some i such that $\eta^i \geq \eta^j$ $(j = 1, \ldots, \beta)$ (breaking ties arbitrarily). Assign A to subDAG D^i.*

A probability distribution P^i is assigned to each subDAG D^i $(i = 1, \ldots, \beta)$ in the following way.

> *For each node $A \in N^i$, if A is a d-sepnode and A is not assigned to D^i, assign to A a* **uniform** *probability table.*[4] *Otherwise assign to A an identical probability table to that in (N, E, P).*

Call $S^i = (D^i, P^i) = ((N^i, E^i), P^i)$ a **sect** *and call the set of sects $\{S^1, \ldots, S^\beta\}$ a* **Multiply Sectioned Bayesian Network (MSBN)**.

Definition 6.2 (adjacent sects) *Two sects in a MSBN are* **adjacent** *if their d-sepset is non-empty. The two sects are also called* **neighbour** *sects.*

The original Bayesian net S is called as an 'UnSectioned Bayesian Network (USBN)'. Note that the sectioning of a Bayesian network is essentially determined by the sectioning of the corresponding DAG D. It doesn't matter which way ties are broken. There will be no significant difference in further processing.

Example 6.3 Suppose the variables in DAG Θ in Figure 2 are all binary. Associate the probability distribution P given in Table 1 with Θ. (Θ, P) is an USBN.

Given the USBN (Θ, P), and corresponding three subDAGs Θ^1, Θ^2 and Θ^3, a 3-sect MSBN $\{(\Theta^1, P^1)$, (Θ^2, P^2), $(\Theta^3, P^3)\}$ can be constructed. First assign d-sepnodes $H_1, \ldots, H_4$ to the subDAGs. H_2 and H_4 must be assigned to Θ^1. H_1 can be assigned to either Θ^1 or Θ^2, and H_3 can be assigned to either Θ^1 or Θ^3. Here it is chosen to assign all 4 d-sepnodes to Θ^1. Based on this assignment and P given, the probability distribution for each sect can be determined (Table 2). Note the uniform probability tables assigned to d-sepnodes in Θ_2 and Θ_3.

[4] This is necessary for two reasons. If a sect does not contain all a d-sepnode's parents, the size of probability table of the d-sepnode must be decreased. This assignment guarantees that the joint system belief constructed in Section 7.3 is proportional to the joint probability distribution P.

P^1		P^2		P^3	
$p(h_{11}) =$	.15	$p(h_{11}) =$	.5	$p(h_{21}) =$	.5
$p(h_{21}\|a_{21}a_{11}) =$	.8696	$p(h_{21}) =$	.5	$p(h_{31}) =$	.5
$p(h_{21}\|a_{21}a_{12}) =$	.7				
$p(h_{21}\|a_{22}a_{11}) =$	.6	$p(f_{11}\|h_{11}h_{21}) =$	.7895	$p(h_{41}) =$	.5
$p(h_{21}\|a_{22}a_{12}) =$	.08	$p(f_{11}\|h_{11}h_{22}) =$	.5		
		$p(f_{11}\|h_{12}h_{21}) =$	.6	$p(e_{11}\|h_{41}) =$	.8
$p(h_{31}) =$	.3	$p(f_{11}\|h_{12}h_{22}) =$	.05	$p(e_{11}\|h_{42}) =$	.15
$p(h_{41}\|a_{31}) =$	.25	$p(f_{21}\|f_{11}) =$	.4	$p(e_{21}\|e_{31}e_{11}) =$	.9789
$p(h_{41}\|a_{32}) =$	.4	$p(f_{21}\|f_{12}) =$	.75	$p(e_{21}\|e_{31}e_{12}) =$	.8
				$p(e_{21}\|e_{32}e_{11}) =$	.9
$p(a_{11}\|h_{11}) =$	.8			$p(e_{21}\|e_{32}e_{12}) =$	.05
$p(a_{11}\|h_{12}) =$	.1				
				$p(e_{31}\|h_{21}h_{31}) =$	.7702
$p(a_{21}\|h_{31}) =$	.8			$p(e_{31}\|h_{21}h_{32}) =$	.35
$p(a_{21}\|h_{32}) =$	.1			$p(e_{31}\|h_{22}h_{31}) =$	.65
				$p(e_{31}\|h_{22}h_{32}) =$	.01
$p(a_{31}\|h_{31}) =$	.3				
$p(a_{31}\|h_{32}) =$	.8				

Table 2: Probability distribution associated with subDAGs Θ^1, Θ^2 and Θ^3 in Figure 5.

6.2 Soundness of Sectioning

In order to perform efficient inference computation in a multiply connected Bayesian net, the junction tree technique transforms the Bayesian net into a clique hypergraph through moralization and triangulation. The hypergraph is organized into a junction tree within which efficient inference can take place. Because of the computational advantage of junction trees, in the context of a MSBN, we would like to transform each sect into a junction tree. The immediate issue is to define conditions on the transformation that guarantee that correct inference can take place in the resultant set of junction trees. The following reviews the major theoretical results related to this question.

Lauritzen, Speed and Vijayan (1984) showed that the clique hypergraph of a graph is decomposable iff the graph is triangulated. Jensen (1988) proved that a hypergraph has a junction tree iff it is decomposable. Maier (1983) proved the same in the context of relational database. Jensen, Lauritzen, Olesen (1990) and Pearl (1988) showed that a junction tree representation of a Bayesian net is an equivalent representation in the sense that the information about joint probability distribution can be preserved. Finally, a more flexible algorithm (compared to that by Lauritzen and Spiegelhalter (1988)) was devised on the junction tree representation of multiply connected Bayesian nets (Jensen, Lauritzen and Olesen 1990).

The above results highlight the importance of clique hypergraphs resulted from triangulation of the original graphs. Thus, as each sect in a MSBN is transformed into a junction tree, it is necessary to preserve

the intactness of the clique hypergraph resulting from the corresponding USBN. This is possible only if the sectioning of DAG D of the original USBN is sound as defined formally below.

Definition 6.4 (soundness of sectioning) *Let a DAG D be sectioned into $\{D^1, \ldots, D^\beta\}$. If there exists a clique hypergraph from D such that for every clique C_k in the hypergraph there is at least one subDAG D^i satisfying $C_k \subseteq N^i$, then the sectioning is* **sound**. *D^i is said to be a* **host subDAG** *of clique C_k.*

Although the soundness of sectioning is defined in terms of DAGs, the concept is used here in the context of MSBNs. When the sectioning of a DAG is sound, it is said that the sectioning of the corresponding USBN into the MSBN is sound, or the MSBN is said to be sound.

If the sectioning of a DAG D is unsound, there is no host subDAG for at least one clique in all possible hypergraphs from D. If a MSBN is based on such a sectioning, it is impossible to consistently maintain the autonomous status of sects in the secondary representation.

Example 6.5 In Figure 6, $\{D^1, D^2, D^3\}$ is an unsound sectioning of D. The clique hypergraph for D must have clique $\{A, B, C\}$ which finds no host subDAG from D^1, D^2, and D^3.

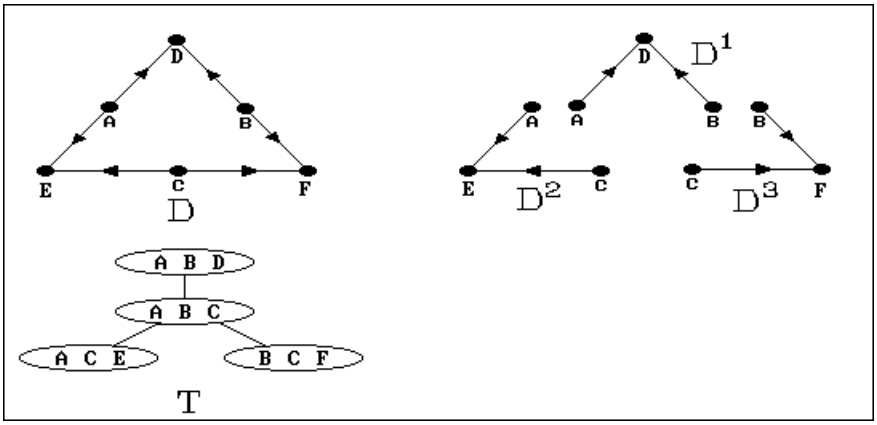

Figure 6: Top left: A DAG D. Top right: The set of subDAGs from an unsound sectioning of D. Bottom left: The junction tree T from D.

The following develops a necessary and sufficient condition for soundness of sectioning.

Lemma 6.6 *Let $A_1 - \ldots - A_{i-1} - B_1 - \ldots - B_{j-1} - C_1 - \ldots - C_{k-1} - \ldots - A_1$ be a cycle consisting of nodes from three or more sets: $X = \{A_1, \ldots, A_{i-1}, B_1\}$, $Y = \{B_1, \ldots, B_{j-1}, C_1\}$, and so on. The nodes from any one set are adjacent in the cycle, and two adjacent sets have a node in common. Then triangulation of this cycle must create a triangle with its three nodes not belonging to any single set.*

If a MSBN has only two sects, the sectioning is *always* sound. Unsoundness can arise only when there are three or more sects. The following shows *exactly* the case where a sectioning is unsound.

Theorem 6.7 (inter-subDAG cycle) *A sectioning of a DAG D to a set of three or more subDAGs is sound iff there exists no (undirected) cycle in D which consists of nodes from three or more distinct subDAGs such that the nodes from each subDAG are adjacent on the cycle.*

6.3 Rules That Guarantee Soundness

Given a DAG and a sectioning, the search for inter-subDAG cycles relative to the sectioning is expensive, especially by local computation when space is of concern. Just that a sectioning is sound does not mean that the resultant MSBN has good computational properties (see the latter part of this Section, Section 7.3, and 9.5). Furthermore, in practice, a large network (MSBN) is constructed one sect at a time. If a sectioning is not sound and it can only be discovered after all sects have been constructed, the overall revision would be disastrous. Thus, we would like to develop a simple guideline for sound sectioning which could be followed during incremental construction of MSBNs. The following covering subDAG rule is one such guideline. This rule, if followed, gives good computational properties, and allows to be checked locally as well.

Theorem 6.8 (covering subDAG) *Let a DAG D be sectioned into $\{D^1, \ldots, D^\beta\}$. Let $I^{jk} = N^j \cap N^k$ be the d-sepset between D^j and D^k. If there is a subDAG D^i such that $N^i \supseteq \cup_{j \neq k} I^{jk}$ then the sectioning is sound. The subDAG D^i is called the* **covering subDAG** *relative to the sectioning.*

In the context of a MSBN, call the sect corresponding to the covering subDAG the *covering sect*. Note that the covering sect rule imposes a conditional independence constraint at a macro level.

Proposition 6.9 *Let S^i and S^j be any two sects in a MSBN with a covering sect S^k ($i \neq k$, $j \neq k$). The two sets of variables N^i and N^j are conditionally independent given N^k.*

Example 6.10 Consider the 3-sect MSBN $\{(\Theta^1, P^1), (\Theta^2, P^2), (\Theta^3, P^3)\}$. (Θ^1, P^1) is the covering sect.

Note, in general, the covering sect of a MSBN may not be unique. As far as soundness is concerned, one is as good as the others. Practically, the one to be consulted most often or the one with the least size is preferred for the sake of computational efficiency which will be clear later.

The covering sect is typically formed naturally. For example (Xiang et al. 1992), in a neuromuscular diagnosis system, the sect containing knowledge about clinical examination contains all the disease hypotheses considered by the system. The EMG sect or nerve conduction sect contains only a subset of the disease hypotheses based on diagnostic importance of these tests to each disease. Thus, the clinical sect is a natural covering sect with all the disease hypothesis as d-sepnodes interfacing the sect with other sects.

The covering subDAG rule can be repeatedly used to create sophisticated MSBNs which are sound. When doing so, a global covering subDAG requirement is replaced by a local covering subDAG requirement.

Definition 6.11 (MSBN of hypertree structure) *A MSBN of hypertree structure is one that is built by the following procedure:*

Start with an empty MSBN. Recursively add a new subDAG D^k to the set of constructed subDAGs $\{D^1, \ldots, D^{k-1}\}$ subject to the constraint:

> *There exists D^i $(i < k)$ such that, for all D^j $(j < k; j \neq i)$, $I^{jk} \subseteq N^i$ where I^{jk} is the d-sepset between D^j and D^k. D^i is called a* **local covering subDAG** *relative to D^k.*

Theorem 6.12 (hypertree) *A MSBN with a hypertree structure is sound.*

The following example illustrate the hypertree rule. It also explains why the sectioning is sound.

Example 6.13 Figure 7 depicts part of a MSBN constructed by the hypertree rule. Each box represents a subDAG with boundaries between boxes representing d-sepsets. The superscripts of subDAGs represent the order of their creation. D^1, D^4, D^5 are local covering subDAGs.

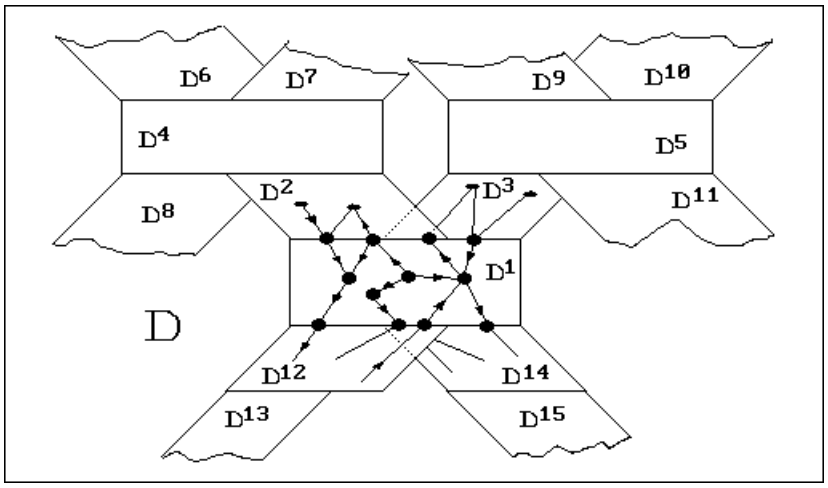

Figure 7: A MSBN with a hypertree structure.

The inter-subDAG cycle as described in theorem 6.7 cannot happen in this MSBN due to its hypertree structure, and hence the sectioning is sound.

Note that the hypertree rule also imposes a conditional independence constraint at a macro level.

Proposition 6.14 *Let S^i and S^j be any two sects with an empty d-sepset in a MSBN sectioned by the hypertree rule. Let S^k be any sect on the unique route mediating S^i and S^j on the hypertree. The two sets of variables N^i and N^j are conditionally independent given N^k.*

It should be indicated that the covering subDAG rule and the hypertree rule do not cover every case where sectioning is sound.

Example 6.15 The 3-sect MSBN $\{D^1, D^2, D^3\}$ in Figure 8 has no covering subDAG. But the sectioning is sound.

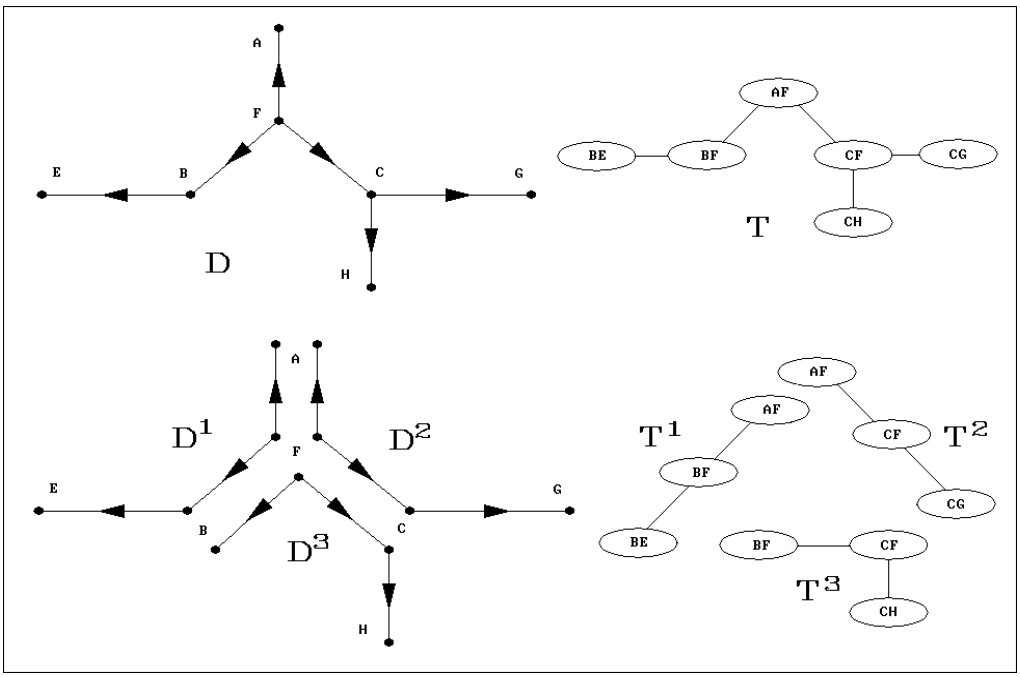

Figure 8: Top left: A DAG D. Top right: A junction tree T from D. Bottom left: $\{D^1, D^2, D^3\}$ forms a sound sectioning of D. Bottom right: The junction trees from the MSBN in Bottom left.

Note that, although the sectioning of the MSBN in Figure 8 is sound, this kind of structure is restricted. For example, arcs can be added between A and B in D^1, between A and C in D^2, but as soon as one more arc is added between B and C in D^3, the theorem 6.7 is violated and the sectioning become unsound. That is, when n subDAGs ($n \geq 3$) are interfaced in this style, there can be *at most* $n-1$ of them being multiply connected. Further computational problems with such structure will be discussed in the appropriate latter sections.

Since MSBNs constructed by the covering subDAG rule or the hypertree rule have sound sectioning, are less restricted, and have extra computational advantages (Section 7.3 and 9.5) over the MSBNs which do not follow these rules, the following study is directed to only the MSBNs that follow these rules.

Conceptually, all MSBNs constructed by the hypertree rule can be viewed as MSBNs with covering subDAGs when attention is directed to local structures. For example, consider D^1 in Figure 7 and its surrounding subDAGs. D^2, D^4, D^6, D^7, D^8 can be considered as one subDAG, $D^3, D^5, D^9, D^{10}, D^{11}$ as another,

D^{12}, D^{13} and D^{14}, D^{15} as two others. Thus, the MSBN is viewed as one with a global covering subDAG D^1. Likewise, when concerned with the relation between D^{14} and D^{15}, the MSBN can be viewed as one satisfying the covering subDAG rule with $\beta = 2$. Therefore, the computation required for a MSBN of a hypertree structure is just the repetition of the computation required for a MSBN with a global covering subDAG. On the other hand, a MSBN with a global covering subDAG is a special case of the hypertree structure. Hence, the following study is often simplified by considering only one of the two cases.

7 Transform MSBN Into Junction Forest

In order to perform efficient inference in a general but sparse network, it is desirable to transform each sect of a MSBN into a junction tree which will stand as an inference entity (Section 4). The transformation takes several steps to be discussed in this section. The set of subDAGs of the MSBN are morali-triangulated into a set of morali-triangulated graphs from which a set of clique hypergraphs are formed. Then the set of clique hypergraphs are organized into a set of junction trees of cliques. Afterwards, the linkages between the junction trees are created. Finally, belief tables are assigned to cliques and linkages and a junction forest of belief universes is constructed.

7.1 Transform subDAGs into junction trees by local computation

The key issue is morali-triangulating subDAGs of a MSBN into a set of morali-triangulated graphs. Once this is done, the formation of the clique hypergraph and the organization of each subDAG into a junction tree are performed the same way as in the case of a USBN and a single junction tree (Andersen *et al.* 1989, Jensen, Lauritzen and Olesen 1990). As mentioned before, the criterion in morali-triangulation of a set of subDAGs of a MSBN into a set of clique hypergraphs is to preserve the 'intactness' of the clique hypergraph resulted from the corresponding USBN. The concept of 'intactness' is formalized below.

Definition 7.1 (invertible morali-triangulation) *Let D be a DAG sectioned into $\{D^1, \ldots, D^\beta\}$. Let N^i be the set of nodes of D^i. If there exists a morali-triangulated graph G of D, with the clique hypergraph H, such that $G = \sqcup_{i=1}^{\beta} G^i$ where G^i is the subgraph of G induced by N^i, or equivalently, $H = \sqcup_{i=1}^{\beta} H^i$ where H^i is the clique hypergraph of G^i, then the set of morali-triangulated graphs $\{G^1, \ldots, G^\beta\}$ is* **invertible**. *Also the transformation of $\{D^1, \ldots, D^\beta\}$ into $\{G^1, \ldots, G^\beta\}$ is said to be an invertible morali-triangulation.*

The invertibility of morali-triangulation depends on the soundness in sectioning. This is given by the following theorem.

Theorem 7.2 (existence of invertible morali-triangulation) *There exists an invertible morali-triangulation for $\{D^1, \ldots, D^\beta\}$ sectioned from a DAG D, iff the sectioning is sound.*

A set of invertible morali-triangulated graphs of a MSBN can be constructed by first performing a global computation (moralization and triangulation) on D to find G, and then determining its subgraphs relative to the sectioning of the MSBN. The moralization and triangulation would be the same as in the junction tree technique with care to be taken not to mix nodes in different subDAGs into one clique. However, when space is of concern, the use of MSBNs offers the possibility of morali-triangulation by local computation at the level of subDAGs of sects. In this method, each subDAG in a MSBN is morali-triangulated separately (message passing may be involved) such that the collection of them is invertible. The following discusses how this can be achieved.

Example 7.3 In the example depicted in Figures 2 and 5, Θ is sectioned into $\{\Theta^1, \Theta^2, \Theta^3\}$ by a sound sectioning and $\{\Lambda^1, \Lambda^2, \Lambda^3\}$ is a set of invertible morali-triangulated graphs relative to the sectioning. We want to find Λ^i $(i = 1, 2, 3)$ from Θ^i $(i = 1, 2, 3)$ by local computation.

Since subDAGs of a MSBN are interfaced through d-sepsets, the focus of finding a set of invertible morali-triangulated graphs by local computation is to decide whether each pair of d-sepnodes is to be linked. Coordination between adjacent subDAGs is necessary to ensure correct decisions. The following considers this systematically.

Call a link between two d-sepnodes a *d-link*. Call a simple path $(A_1, A_2, \ldots, A_k)$ a *d-path* if there is some i, j, $(1 \leq i < j \leq k)$ such that $A_1, \ldots, A_i$ and $A_j, \ldots, A_k$ are all d-sepnodes, while all the other nodes on the path are non-d-sepnodes. A d-link is a trivial d-path. There are six types of d-links:

Arc type inherited from the subDAG. That is, if two d-sepnodes are connected originally in the subDAG, there is a d-link between them in G^i.

ML type created by local moralization. For example, the d-links (H_1, H_2) in Λ^2 and (H_2, H_3) in Λ^3. No communication between subDAGs is required to add these d-links.

ME type created by moralization in neighbour subDAGs. For example, the d-links (H_1, H_2) and (H_2, H_3) in Λ^1. Deciding to add this type of d-link requires communication between neighbour subDAGs.

Cy type created to triangulate inter-subDAG cycles. For example, the d-link (H_3, H_4) in Λ^1 and Λ^3. Deciding to add this type of d-links requires communication between neighbour subDAGs.

TL type created during local triangulation. After the above four types of d-links have been introduced to the moral graph of a subDAG, there may still be un-triangulated cycles within the moral graph

involving four or more d-sepnodes. The example used above is too simple to illustrate this and next type.

TE type created by local triangulation in neighbour subDAGs. The triangulation of a cycle of length > 3 involving only d-sepnodes is not unique. If two neighbour subDAGs triangulate such a cycle by local computation without coordination, they may triangulate in different ways and result in different set of cliques for the nodes in the d-sepset. Therefore communication is required such that a subDAG may adopt the d-links introduced by triangulation in adjacent subDAGs. The argument also applies to the case of triangulating cycles consisting of general d-paths.

An algorithm for morali-triangulation of subDAGs of a MSBN into a set of invertible triangulated graphs under the covering subDAG assumption is given below.

Algorithm 7.4 (morali-triangulation with a covering subDAG) *Let D^1 be the covering subDAG in the MSBN.*

1. *Let ML^i be the set of d-links added in the local moralization of subDAG D^i. Let Cy^i be the set of pairs of nodes that are candidates for becoming Cy type d-links in D^i. For each subDAG D^i, do the following:*
   (a) *Moralize D^i to obtain its moral graph Φ^i. Add new d-links to ML^i.*
   (b) *Search for pairs of d-sepnodes connected by a d-path in Φ^i. Add the pairs found to Cy^i.*
2. *For D^1, do the following:*
   (a) *For each pair of d-sepnodes in D^1 also contained in one of the ML^i $(i > 1)$, connect the pair by a d-link in Φ^1.*
   (b) *For each pair of d-sepnodes contained in both Cy^1 and one of the Cy^j $(j > 1)$, connect the pair by a d-link in Φ^1.*
   (c) *Triangulate Φ^1 to obtain the morali-triangulated graph Λ^1.*
   (d) *Let $DLINK$ be the set of d-links in Λ^1.*
3. *For each D^i $(i = 2, \ldots, \beta)$, do the following:*
   (a) *For each pair of d-sepnodes of Φ^i also contained in $DLINK$, connect the pair by a d-link.*
   (b) *Triangulate Φ^i to obtain the morali-triangulated graph Λ^i.*

Note that Algorithm 7.4 has two passes through all the subDAGs. The following theorem shows the invertibility of the morali-triangulation.

Theorem 7.5 (invertibility of Algorithm 7.4) *The morali-triangulation constructed in Algorithm 7.4 is invertible.*

Example 7.6 The following describe the morali-triangulation of $\sqcup_{i=1}^{3}\Theta^i$ (Figure 5) by Algorithm 7.4.

1. After step 1 of the algorithm, $ML^1 = \phi$, $ML^2 = \{(H_1, H_2)\}$, $ML^3 = \{(H_2, H_3)\}$, $Cy^1 = \{(H_1, H_2), (H_2, H_3), (H_3, H_4)\}$, $Cy^2 = \{(H_1, H_2)\}$, and $Cy^3 = \{(H_2, H_3), (H_2, H_4), (H_3, H_4)\}$.

2. After step 2, the morali-triangulated graph Λ^1 of Θ^1 is completed by adding d-links $(H_1, H_2), (H_2, H_3)$, (H_3, H_4) to Θ^1's moral graph, and then triangulating (nothing is added). $DLINK$ will contain $\{(H_1, H_2), (H_2, H_3), (H_3, H_4)\}$.

3. After step 3, the morali-triangulated graph Λ^2 of Θ^1 is completed without change to its moral graph; the morali-triangulated graph Λ^3 of Θ^3 is completed by adding the d-link (H_3, H_4) to its moral graph, and then triangulating (with the link (E_3, H_4) added).

As mentioned in Section 4, after the morali-triangulation, the other steps in transformation of a MSBN into a set of junction trees of cliques are: identifying cliques of the morali-triangulated graphs to form a set of clique hypergraphs, and organizing each hypergraph into a junction tree. These steps are performed in the same way as in the junction tree technique. Throughout the rest of the chapter, it is assumed that junction trees are obtained through a set of invertible triangulated graphs, and it is said that the junction trees are obtained by an invertible transformation.

Call a set of junction trees of cliques from an invertible transformation of subDAGs of a MSBN a *junction forest of cliques* denoted by $F = \{T^1, \ldots, T^\beta\}$ where T^i is the junction tree from the subDAG D^i.

7.2 Linkages between junction trees

Just as d-sepsets interface subDAGs, linkages interface junction trees transformed from subDAGs and serve as information channels between junction trees during inference. The creation of the linkages is an extension to the junction tree technique. The multiple linkages between pairs of junction trees in a junction forest allow the preservation of localization within junction trees, and allow the avoidance of the exponential explosion of the sizes of clique state spaces associated with the brute force method (Section 3).

Definition 7.7 (linkage set) *Let I be the d-sepset between two subDAGs D^a and D^b. Let T^a and T^b be the junction trees transformed from D^a and D^b respectively. A* **linkage** *of T^a relative to T^b is a set l of nodes such that the following two conditions hold.*

1. *Boundary: there exists a clique $C_x \in T^a$ such that $l = C_x \cap I$. C_x is called a* **host clique** *of l;*

2. *Maximum: there is no subset of l that is also a linkage.*

In general there may be more than one linkage between a pair of junction trees. Define L^{ab} to be the set of all linkages of T^a relative to T^b.

Proposition 7.8 (identity of linkages) *Let T^a and T^b be the junction trees from subDAGs D^a and D^b respectively. If L^{ab} is the set of linkages of T^a relative to T^b and L^{ba} is the set of linkages of T^b relative to T^a then $L^{ab} = L^{ba}$.*

Example 7.9 In Figure 5, linkages between junction trees are indicated with ribbed bands connecting the corresponding host cliques. The two linkages between Γ^1 and Γ^3 are $\{H_3, H_2\}$ and $\{H_3, H_4\}$.

Given a set of linkages between a pair of junction trees, the concept of a redundancy set can be defined. As mentioned in Section 4, redundancy sets provide structures which allow redundant information to be removed during inter-junction tree information passing. The concept will be used for defining joint system belief in Section 7.3 and defining the operation NonRedundancyAbsorption in Section 9.2. To define the redundancy set, we need to index linkages such that the redundancy sets defined based on the indexing possess certain desirable properties described below. We index a set L^{ab} of linkages by the following algorithm.

Algorithm 7.10 (indexing linkages) *Let T^a and T^b be two junction trees with a set L^{ab} of linkages.*

1. *Pick one of the junction trees in the pair, say T^a. Create a tree G with nodes labeled by linkages in L^{ab}. Connect two nodes in G by a link if either the hosts of corresponding linkages are directly connected in T^a, or the hosts of corresponding linkages are (indirectly) connected in T^a by a path on which all intermediate cliques are not linkage hosts. Call this tree a* **linkage tree**.

2. *Index the nodes (linkages in L^{ab}) of G into $L_1, L_2, \ldots$ in any order that is consistent with G, i.e., for every $i > j$ there is a unique predecessor $j(i) < i$ such that $L_{j(i)}$ is adjacent to L_i in G.*

Note, the second step is always possible due to the tree structure of G. With linkages indexed this way, the redundancy set can be defined as the following.

Definition 7.11 (redundancy set) *Let a set of linkages $L^{ab} = \{L_1, \ldots, L_g\}$ be indexed by Algorithm 7.10. Then for this set of indexed linkages, a* **redundancy set** *R_i for index i is defined as*

$$R_i = \begin{cases} \phi & \text{if } i = 1, \\ L_i \cap L_{j(i)} & i > 1;\ j(i) < i;\ L_{j(i)} \text{ is adjacent to } L_i \text{ in the linkage tree } G. \end{cases}$$

Lemma 7.12 *A linkage tree is a junction tree. The redundancy sets are sepsets of the linkage tree.*

Example 7.13 There are two linkages between Γ^1 and Γ^3 in Figure 5. Consider junction tree Γ^3. The linkage tree G has two connected nodes, one labeled by the linkage $\{H_3, H_2\}$ and the other by $\{H_3, H_4\}$. An indexing $L_1 = \{H_3, H_2\}$ and $L_2 = \{H_3, H_4\}$ defines two redundancy sets $R_1 = \phi$ and $R_2 = \{H_3\}$.

With linkages and redundancy sets constructed, we have a *linked junction forest of cliques.*

7.3 Joint system belief of junction forest

The following algorithm associates belief tables with each clique, each clique sepset, and each linkage in a junction forest whose corresponding MSBN has a covering sect. These data structures specify a joint system belief for the junction forest.

Algorithm 7.14 (creating data structures for the joint system belief) *Let (D, P) be a USBN, $S = \{S^1, \ldots, S^\beta\}$ be a corresponding MSBN with a covering sect S^1, and $F = \{T^1, \ldots, T^\beta\}$ be the junction forest from an invertible transformation. Let T^i be the junction tree of D^i with cliques C^i and sepsets Q^i. Let I^i $(i > 1)$ be the d-sepset between S^i and S^1. Let L^i $(i > 1)$ be the set of linkages between T^i and T^1, and R^i $(i > 1)$ be the corresponding set of redundancy sets.*

1. *For each junction tree T^i in F, do the following:*
   - *Assign each node $n_k \in N^i$ to a unique clique $C_x \in C^i$ such that C_x contains n_k and its parents π_k. Break ties arbitrarily.*
   - *Let P_k denote the probability table associated with node n_k. For each clique C_x that has nodes $n_k, \ldots, n_l$ assigned to it, associate C_x with a belief table $B(C_x) = P_k * \ldots * P_l$.*
   - *For each clique sepset $Q_y \in Q^i$, associate it with a constant belief table $B(Q_y)$.*
2. *For each set of linkage L^i, do the following:*
   - *For each linkage $L_z \in L^i$, associate it with a constant belief table $B(L_z)$.*

Definition 7.15 (belief on redundancy set) *Using the notations in Algorithm 7.14, for each redundancy set $R_z \in R^i$, its belief table is defined as*

$$B(R_z) = \sum_{L_z \setminus R_z} B(L_z).$$

Definition 7.16 (belief on d-sepset) *Using the notations in Algorithm 7.14 and Definition 7.15, for each d-sepset I^i, its belief table is defined as*

$$B(I^i) = \frac{\prod_{L_z \in L^i} B(L_z)}{\prod_{R_z \in R^i} B(R_z)}.$$

Definition 7.17 (belief on junction tree) *Using the notations in Algorithm 7.14, for each junction tree T^i, its belief table is defined as*

$$B(T^i) = \frac{\prod_{C_x \in C^i} B(C_x)}{\prod_{Q_y \in Q^i} B(Q_y)}.$$

The above definition uses the notation $B(T^i)$ instead of $B(N^i)$ to emphasize that it is related to the junction tree.

Comparing the form of joint probability distribution for an USBN (Section 2.2) and the assignment of probability tables for nodes in a sect (Definition 6.1), it can be seen that $B(T^i)$ is proportional to the joint probability distribution of S^i relative to that assignment.

Definition 7.18 (belief on junction tree) *Using the notations in Algorithm 7.14, Definitions 7.16 and 7.17, the* joint system belief *for the junction forest F is defined as*

$$B(F) = \frac{\prod_{i=1}^{\beta} B(T^i)}{\prod_{i=2}^{\beta} B(I^i)}.$$

The notation $B(F)$ instead of $B(N)$ is used for the same reason as above. Note that unlike $B(C_x)$, $B(Q_y)$, $B(L_z)$ and $B(R_z)$, $B(I^i)$, $B(T^i)$ and $B(F)$ are mathematical objects which do not have corresponding data structures in the knowledge base.

We have the following lemma.

Lemma 7.19 *The joint belief $B(F)$ of a MSBN is proportional to the joint probability distribution P of the corresponding USBN.*

To see this is true, we indicate that each d-sepnode, appearing in at least two sects, carries its original probability table as in (N, E, P) exactly once by Definition 6.1, and carries an uniform table for the rest.

Example 7.20 Table 3 lists constructed belief tables for belief universes of junction forest $F = \{\Gamma^1, \Gamma^2, \Gamma^3\}$ in Figure 5. The belief tables for sepsets, linkages, and redundancy sets are all constant tables at this stage.

Having constructed belief tables for cliques, sepsets, linkages, redundancy sets and junction forest, using the definition of world of Section 2.3, we talk about *belief universes, sepset worlds, linkage worlds, redundancy worlds*, and junction forest of belief universes. These terms will be used below.

The preceding has an assumption of a covering sect. The joint system belief of a junction forest with a hypertree structure can be defined in a similar way. As the d-sepset/linkages between non-covering sects are not considered in Algorithm 7.14, in the hypertree case, there is no need to consider the d-sepset/linkages between neighbour sects covered by a local covering sect. In practice, these linkages are not created. This is another computational advantage of the covering sect rule and the hypertree rule.

$B(\Gamma^1)$	
$Clique$	$NodeAss.$
$\{H_2, H_1, A_1\}$	H_1, A_1
$Config.$	$B()$
$\{h_{21}, h_{11}, h_{11}\}$	.12
$\{h_{21}, h_{11}, h_{12}\}$	.03
$\{h_{21}, h_{12}, h_{11}\}$	.085
$\{h_{21}, h_{12}, h_{12}\}$	.765
$\{h_{22}, h_{11}, h_{11}\}$	.12
$\{h_{22}, h_{11}, h_{12}\}$	.03
$\{h_{22}, h_{12}, h_{11}\}$	.085
$\{h_{22}, h_{12}, h_{12}\}$	.765
$Clique$	$NodeAss.$
$\{H_2, A_2, A_1\}$	H_2
$Config.$	$B()$
$\{h_{21}, a_{21}, a_{11}\}$	.8696
$\{h_{21}, a_{21}, a_{12}\}$	.7
$\{h_{21}, a_{22}, a_{11}\}$	.6
$\{h_{21}, a_{22}, a_{12}\}$	.08
$\{h_{22}, a_{21}, a_{11}\}$	.1304
$\{h_{22}, a_{21}, a_{12}\}$	.3
$\{h_{22}, a_{22}, a_{11}\}$	.4
$\{h_{22}, a_{22}, a_{12}\}$	.92
$Clique$	$NodeAss.$
$\{H_3, H_2, A_2\}$	H_3, A_2
$Config.$	$B()$
$\{h_{31}, h_{21}, a_{21}\}$	.24
$\{h_{31}, h_{21}, a_{22}\}$	.06
$\{h_{31}, h_{22}, a_{21}\}$	.24
$\{h_{31}, h_{22}, a_{22}\}$	.06
$\{h_{32}, h_{21}, a_{21}\}$	.07
$\{h_{32}, h_{21}, a_{22}\}$	.63
$\{h_{32}, h_{22}, a_{21}\}$	.07
$\{h_{32}, h_{22}, a_{22}\}$	.63
$Clique$	$NodeAss.$
$\{H_3, A_3, H_4\}$	A_3, H_4
$Config.$	$B()$
$\{h_{31}, a_{31}, h_{41}\}$	.075
$\{h_{31}, a_{31}, h_{42}\}$	.225
$\{h_{31}, a_{32}, h_{41}\}$	.28
$\{h_{31}, a_{32}, h_{42}\}$	.42
$\{h_{32}, a_{31}, h_{41}\}$	.2
$\{h_{32}, a_{31}, h_{42}\}$	.6
$\{h_{32}, a_{32}, h_{41}\}$	.08
$\{h_{32}, a_{32}, h_{42}\}$	.12

$B(\Gamma^2)$	
$Clique$	$NodeAss.$
$\{F_2, F_1\}$	F_2
$Config.$	$B()$
$\{f_{21}, f_{11}\}$	.4
$\{f_{21}, f_{12}\}$	.75
$\{f_{22}, f_{11}\}$	.6
$\{f_{22}, f_{12}\}$	.25
$Clique$	$NodeAss.$
$\{H_2, F_1, H_1\}$	H_2, F_1, H_1
$Config.$	$B()$
$\{h_{21}, f_{11}, h_{11}\}$	.7895
$\{h_{21}, f_{11}, h_{12}\}$	.6
$\{h_{21}, f_{12}, h_{11}\}$	.2105
$\{h_{21}, f_{12}, h_{12}\}$	.4
$\{h_{22}, f_{11}, h_{11}\}$	.5
$\{h_{22}, f_{11}, h_{12}\}$	.05
$\{h_{22}, f_{12}, h_{11}\}$	.5
$\{h_{22}, f_{12}, h_{12}\}$	.95

$B(\Gamma^3)$	
$Clique$	$NodeAss.$
$\{H_3, H_2, E_3\}$	H_3, H_2, E_3
$Config.$	$B()$
$\{h_{31}, h_{21}, e_{31}\}$	.7702
$\{h_{31}, h_{21}, e_{32}\}$	.2298
$\{h_{31}, h_{22}, e_{31}\}$	.65
$\{h_{31}, h_{22}, e_{32}\}$	.35
$\{h_{32}, h_{21}, e_{31}\}$	.35
$\{h_{32}, h_{21}, e_{32}\}$	.65
$\{h_{32}, h_{22}, e_{31}\}$	.01
$\{h_{32}, h_{22}, e_{32}\}$	.99
$Clique$	$NodeAss.$
$\{E_2, E_3, E_1\}$	E_2
$Config.$	$B()$
$\{e_{21}, e_{31}, e_{11}\}$	.9789
$\{e_{21}, e_{31}, e_{12}\}$	.8
$\{e_{21}, e_{32}, e_{11}\}$	.9
$\{e_{21}, e_{32}, e_{12}\}$	.05
$\{e_{22}, e_{31}, e_{11}\}$	.0211
$\{e_{22}, e_{31}, e_{12}\}$	.2
$\{e_{22}, e_{32}, e_{11}\}$	.1
$\{e_{22}, e_{32}, e_{12}\}$	.95
$Clique$	$NodeAss.$
$\{E_3, E_1, H_4\}$	E_1, H_4
$Config.$	$B()$
$\{e_{31}, e_{11}, h_{41}\}$	.8
$\{e_{31}, e_{11}, h_{42}\}$	.15
$\{e_{31}, e_{12}, h_{41}\}$	.2
$\{e_{31}, e_{12}, h_{42}\}$	.85
$\{e_{32}, e_{11}, h_{41}\}$	.8
$\{e_{32}, e_{11}, h_{42}\}$	.15
$\{e_{32}, e_{12}, h_{41}\}$	.2
$\{e_{32}, e_{12}, h_{42}\}$	.85
$Clique$	$NodeAss.$
$\{H_3, E_3, H_4\}$	
$Config.$	$B()$
$\{h_{31}, e_{31}, h_{41}\}$	1
$\{h_{31}, e_{31}, h_{42}\}$	1
$\{h_{31}, e_{32}, h_{41}\}$	1
$\{h_{31}, e_{32}, h_{42}\}$	1
$\{h_{32}, e_{31}, h_{41}\}$	1
$\{h_{32}, e_{31}, h_{42}\}$	1
$\{h_{32}, e_{32}, h_{41}\}$	1
$\{h_{32}, e_{32}, h_{42}\}$	1

Table 3: Constructed belief tables for belief universes of junction forest $F = \{\Gamma^1, \Gamma^2, \Gamma^3\}$ in Figure 5. Config: Configuration. Node Ass: Nodes Assigned.

Algorithm 7.21 (linkage creation in a hypertree MSBN) *Let $S = \{S^1, \ldots, S^\beta\}$ be a MSBN constructed according to Definition 6.11. Let $F = \{T^1, \ldots, T^\beta\}$ be the junction forest from S where T^i is the junction tree from S^i.*

For each T^j, if S^i is the local covering sect during the construction of S, then create a set of linkages between T^i and T^j.

Definition 7.22 (Neighbour junction tree) *A pair of junction trees in a junction forest are* **neighbours** *if linkages are created between them.*

8 Consistency and Separability of Junction Forest

In order to perform efficient inference, we need to propagate the information stored in different belief universes in different junction trees of a junction forest to the whole system such that marginal probability of variables can be obtained from any universes containing them with local computation.[5] More precisely, we need to propagate both prior knowledge in the form of products of original probability tables from the corresponding USBN, and evidence entered from a set of universes possibly in different junction trees. The following defines consistency and separability that are properties of junction forests which guarantee the correctness of marginals obtained by local computation.

8.1 Consistency of junction forest

This subsection defines a property of consistency that partially guarantees the correctness of marginal probabilities obtained by local computation. In the context of the junction forest, three levels of consistency can be identified.

The first level concerns the internal consistency of each junction tree.

Definition 8.1 (local consistency) *Neighbor universes $(C_i, B(C_i))$ and $(C_j, B(C_j))$ in a junction tree T^l with sepset world $(Q_k, B(Q_k))$ (Section 7.3) are consistent if*

$$\sum_{C_i \setminus C_j} B(C_i) \propto B(Q_k) \propto \sum_{C_j \setminus C_i} B(C_j)$$

where '$\propto$' reads 'proportional to'. When the relation holds among all neighbour universes, the junction tree T^l is said to be consistent. When all junction trees are consistent, a junction forest F is said to be **locally consistent**.

The second level concerns the consistency between linkage hosts.

[5] Obtaining marginals by local computation is what the junction tree technique is developed for. More is obtained from junction forests, namely, exploiting localization.

Definition 8.2 (boundary consistency) *Host universes* $(C_x^i, B(C_x^i))$ *and* $(C_y^j, B(C_y^j))$ *of linkage world* $(L_k, B(L_k))$ *are consistent if*

$$\sum_{C_x^i \setminus C_y^j} B(C_x^i) \propto B(L_k) \propto \sum_{C_y^j \setminus C_x^i} B(C_y^j)$$

When the relation holds among all linkage host universes, a junction forest is said to have reached **boundary consistency**.

The third level concerns what the name of the following definition suggests.

Definition 8.3 (global consistency) *A junction forest is said to be* **globally consistent** *if for any 2 belief universes (possibly in different junction trees)* $(C_x^i, B(C_x^i))$ *and* $(C_y^j, B(C_y^j))$

$$\sum_{C_x^i \setminus C_y^j} B(C_x^i) \propto \sum_{C_y^j \setminus C_x^i} B(C_y^j)$$

Theorem 8.4 (consistent junction forest) *A junction forest is globally consistent iff it reaches both local consistency and boundary consistency.*

8.2 Separability of junction forests

In the junction tree technique, consistency is all that is required in order to obtain marginals by local computation. In junction forests, this is not sufficient due to the existence of multiple linkages. The function of multiple linkages between a pair of junction trees is to pass the joint distribution on the d-sepset by passing marginal distributions on subsets of the d-sepset. Doing so, avoids the exponential increase in clique state space sizes as outlined in Section 3. When breaking the joint into the marginals, we must ensure that the joint can be reassembled from the marginals, i.e., a correct version of the joint is passed. Otherwise, the correctness of local computation is not guaranteed. Since passing the marginals is achieved by passing the belief tables on linkages and redundancy sets, the structure of linkage hosts is the key factor. The following defines separability of junction forests in terms of the correctness of local computation. Then the structural condition of linkage hosts is given under which the separability holds.

Definition 8.5 (separability) *Let* $F = \{T^i | 1 \leq i \leq \beta\}$ *be a junction forest with nodes* N *and joint system belief* $B(F)$. F *is said to be* **separable** *if, when it is globally consistent, for any* T^i *over subdomain* N^i

$$\sum_{N \setminus N^i} B(F) \propto B(T^i)$$

The following lemma, quoted from Jensen, is used to prove Proposition 8.10.

Lemma 8.6 *(Jensen 1988)*

Let T be a junction tree from clique hypergraph $(N, \mathbf{C})$. Let C_1 and C_2 be two adjacent cliques in T. Let T' be the graph resulting from T by making the union of C_1 and C_2 into one clique, and by keeping the original sepsets. Then T' is a junction tree for clique hypergraph $(N, (\mathbf{C} \setminus \{C_1, C_2\}) \cup \{C_1 \cup C_2\})$.

Definition 8.7 (host tree) *Let a sound MSBN be transformed into a junction forest. Let T^i be a junction tree and L be the set of linkages between T^i and a neighbour junction tree.*

A **host tree** *of T^i relative to L is the clique tree resulting from recursively removing from T^i every leaf clique which is not a linkage host relative to L.*

The following is the structural condition for separability to be proved below.

Definition 8.8 (host composition) *Let a sound MSBN be transformed into a junction forest. Let S^i be a sect in the MSBN, and T^i be the junction tree of S^i. Let I be the d-sepset between S^i and any distinct sect S^j, and L be the set of linkages between T^i and the junction tree for S^j.*

T^i satisfies a **host composition** *condition relative to L if the following are true in the host tree of T^i relative to L.*

1. *No non-d-sepnode is contained in more than one linkage host.*

2. *Two non-d-sepnodes in some non-host clique are not contained in different linkage hosts.*

Example 8.9 The host composition condition is violated in the host trees of Figure 9. The following shows the violation and the resultant problem. Assume both trees are consistent.

First consider the top tree. Let L consist of linkages $L_1 = \{A, D\}$ and $L_2 = \{A, E\}$. Let their hosts be $C_1 = \{A, B, D\}$ and $C_2 = \{A, B, E\}$ which are adjacent in the tree. B is a common non-d-sepnode - a violation of part 1 of the host composition condition. Even if all the belief tables are consistent, in general,

$$\sum_B \frac{B(ABD)B(ABE)}{B(AB)} \not\propto \frac{B(AD)B(AE)}{B(A)}$$

That is, the joint distribution on the d-sepset $\{A, D, E\}$ constructed from belief tables on linkages and redundancy sets is inconsistent in general.

Consider the bottom tree. Let L and C_1 be the same. Let the host $C_2 = \{A, E, G\}$ which is connected to C_1 through a non-host $C_3 = \{A, B, G\}$. $\{B, G\}$ is a set of non-d-sepnodes violating the part 2 of the host composition condition. If C_1 and C_3 are united (forming clique C_{13}) as described in lemma 8.6, the resultant graph is still a junction tree. If let

$$B(C_{13}) = B(C_1)B(C_3)/B(Q_{13})$$

where Q_{13} (containing AB) is the sepset between C_1 and C_3, the joint belief for the new tree is exactly the same as before and the new tree is consistent. Now the common node G in C_{13} and C_2 creates the same problem illustrated above.

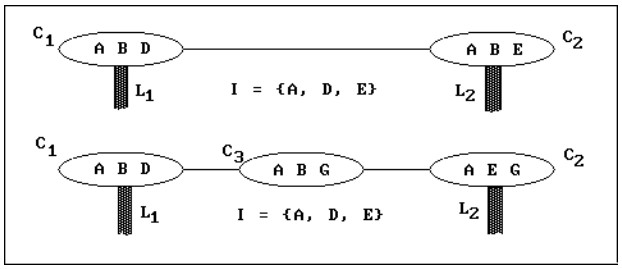

Figure 9: Two partial trees to exemplify violation of the host composition condition. I: the d-sepset. The thick bands indicate linkages.

The following proposition shows that, if a junction forest is globally consistent and the host composition condition holds, then the belief table $B(I)$ defined in Section 7.3 is indeed the joint belief on d-sepset I.

Proposition 8.10 *Let a sound MSBN be transformed into a junction forest F. Let S^x and S^y be sects in the MSBN, and T^x and T^y be the junction trees of S^x and S^y in F, respectively. Let I be the d-sepset between S^x and S^y, and L be the set of linkages between T^x and T^y. Let all belief tables be defined as in Section 7.3.*

When F is globally consistent, $B(I)$ satisfies

$$B(I) \propto \sum_{N^x \setminus I} B(T^x)$$

iff T^x satisfies the host composition condition relative to L.

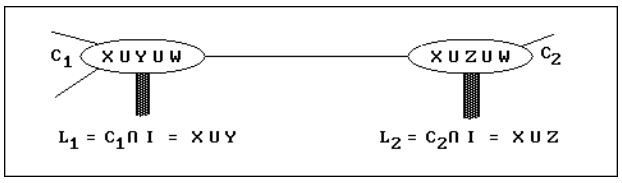

Figure 10: Part of a host tree violating the host composition condition. I: the d-sepset. The thick bands indicate linkages.

Now we are ready for the following result on separability.

Theorem 8.11 (host composition) *Let $S = \{S^1, \ldots, S^\beta\}$ be a MSBN satisfying the hypertree condition. Suppose S has been transformed into a junction forest $F = \{T^i | 1 \leq i \leq \beta\}$, with linkages between the junction*

trees created by Algorithm 7.21. Let $B(F)$ be the joint system belief of F. F is separable iff, for every pair of neighbour junction trees, the host composition condition holds.

The host composition condition can usually be satisfied naturally in an application system. Since d-sepsets are the only media for information exchange between sects, d-sepnodes usually involve many inter-subDAG cycles. The consequence is that they will be heavily connected during morali-triangulation and form several large cliques in the clique hypergraph as well as some small ones. On the other hand, non-d-sepnodes rarely form connections with so many d-sepnodes simultaneously and hence will rarely be the elements of these large cliques. To be an element of more than one such large clique is even more unlikely. Because linkages are defined to be maximal, these large cliques will become linkage hosts.

For example, in the PAINULIM expert system (Xiang et al. 1992), there are three sects and correspondingly three junction trees. The host composition condition is satisfied naturally in all three trees. Figure 11 gives one of them. The four linkage hosts contain no non-d-sepnode at all.

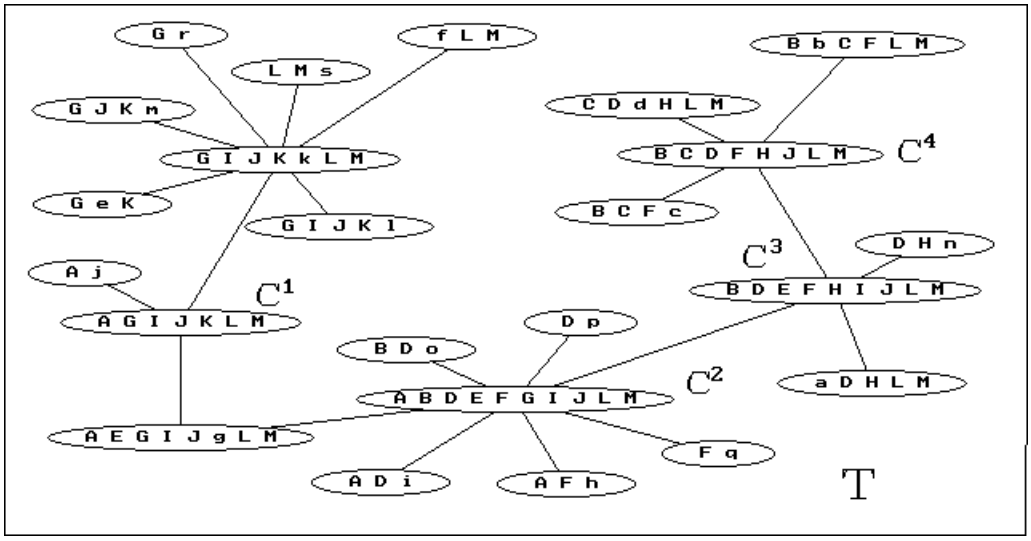

Figure 11: T is a junction tree in a junction forest taken from an application system PAINULIM with variable names revised to simplify. Upper case letters represent d-sepnodes and lower case letters represent non-d-sepnodes. The cliques C_1, C_2, C_3, C_4 are linkage hosts.

When the host composition condition cannot be satisfied naturally, dummy links can be added between d-sepnodes in the moral graph before triangulation such that linkage hosts will be enlarged and the condition is satisfied. Hence, given a MSBN, a separable junction forest can always be realized. The penalty of added links is increased amount of computation during belief propagation due to increased sizes of cliques and linkages. In the worst case, we may have to resort to the brute force method discussed in Section 3 in order to satisfy the host composition condition for certain pairs of junction trees. If the system is large, sectioning

may still yield computational savings on the whole even if cliques are enlarged at a few junction trees.

One of the key results now follows.

Theorem 8.12 (local computation) *Let F be a consistent and separable junction forest with nodes N and joint system belief $B(F)$. Let $(C_x, B(C_x))$ be any universe in F. Then*

$$\sum_{N \setminus C_x} B(F) \propto B(C_x)$$

With the above theorem, the marginal belief of any variable in a consistent and separable junction forest can be computed by marginalization of the belief table of any universe which contains the variable. In this respect, a consistent and separable junction forest behaves the same as a consistent junction tree (Jensen, Lauritzen and Olesen 1990). It will be shown that in the context of junction forests, additional computational advantage is available. That is, the global consistency is not necessary to obtain marginal belief by local computation, which allows the exploitation of localization.

9 Belief Propagation In Junction Forests

Given the importance of consistency of junction forests, we need to introduce a set of operations which bring a junction forest into consistency. First, since our purpose is to exploit localization, only operations for local computation up to the level of junction trees are considered. At any moment, we consider only one junction tree. This junction tree is said to be *active*.

Second, we present the operations in an object-oriented fashion as does Jensen, Lauritzen and Olesen (1990) in describing the junction tree technique. As argued in the above reference, the purpose is to avoid a global control structure, and to exploit parallel processing. Four levels of objects can be identified in the context of junction forests.

1. (Inside a junction tree) The belief universes are objects, and sepsets are communication channels.
2. (Between linkage hosts in neighbour junction trees) The linkage host worlds are objects, and linkages are communication channels.
3. (Between neighbour junction trees) The set of linkage host worlds in a junction tree relative to a neighbour junction tree is an object, and the set of linkages between the two neighbour junction trees is the communication channel.
4. (Top level object) A junction forest is a top level object.

Third, each operation presented in this section is associated with a particular level of objects. Some operations can be initiated by the objects they associate. Other operations can only be invoked (called) by objects at the higher level.

The operations to be presented below are summarized in Figure 12.

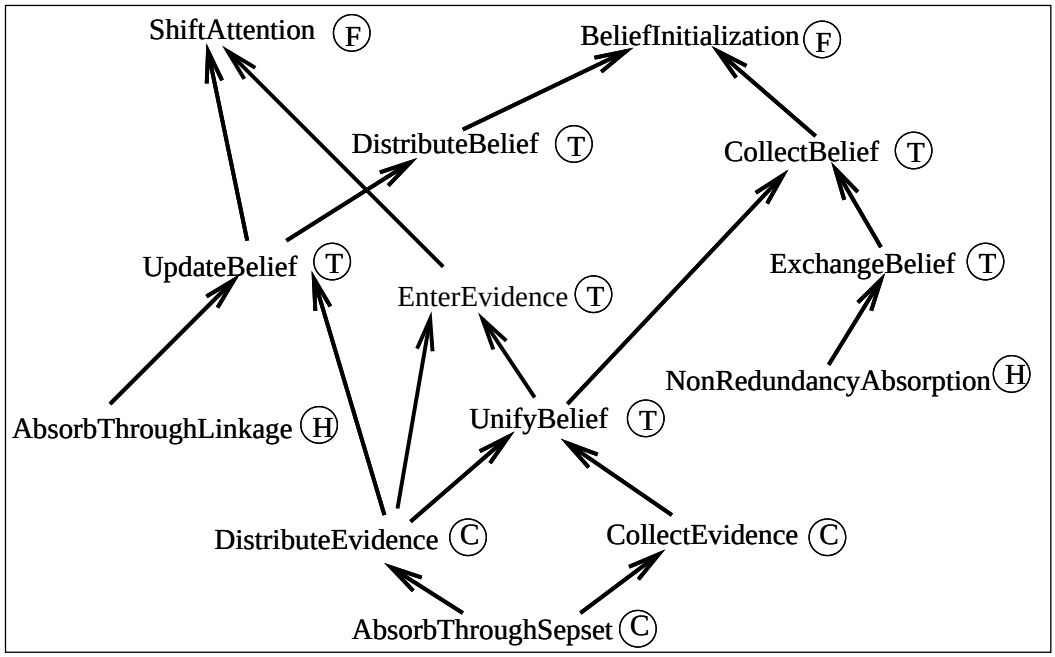

Figure 12: Summary of operations. The node at the right of each operation is the type of objects with which the operation can associate. C: belief universe; H: linkage host; T: junction tree; F: junction forest. Arcs indicate that the lower level operations provide services to the higher level operations.

9.1 Supportiveness

Jensen, Lauritzen and Olesen (1990) introduced the concept of supportiveness. Let $(Z, B(Z))$ be a world. The *support* of $B(Z)$ is defined as

$$\Delta(B(Z)) = \{z \in \Psi(Z) | \text{belief of z} > 0\}.$$

A junction tree is *supportive*, if, for any universe $(C_i, B(C_i))$ and for any neighbouring sepset world $(Q_j, B(Q_j))$, $\Delta(B(Ci)) \subseteq \Delta(B(Q_j))$. The underlying intuition is that, when beliefs are propagated in a supportive junction tree, non-zero belief values will not be turned into zeros.

Here the concept is extended to junction forests. A junction forest is supportive, if (1) all its junction trees are supportive and (2) for any linkage host $(C_i, B(C_i))$ and corresponding linkage world $(L_j, B(L_j))$, $\Delta(B(Ci)) \subseteq \Delta(B(Lj))$.

The construction in Section 7.3 results in a supportive junction forest.

9.2 Basic operations

9.2.1 Operations for consistency within a junction tree

The following operation is used to achieve consistency between a belief universe and its neighbours.

Operation 9.1 (AbsorbThroughSepset) *(Jensen, Lauritzen and Olesen 1990) Let $U_0 = (C_0, B(C_0))$ be a belief universe in a junction tree. Let $U_i = (C_i, B(C_i))$ $(i = 1, \ldots, k)$ be neighbour universes of U_0. Let $(Q_i, B(Q_i))$ $(i = 1, \ldots, k)$ be the sepset world between U_0 and U_i. Suppose $\Delta(B(C_0)) \subseteq \Delta(B(Q_i))$ $(i = 1, \ldots, k)$. When* **AbsorbThroughSepset** *is initiated by U_0 to absorb from $U_1, \ldots, U_k$, the following updates are performed:*

1. *For each i, update sepset belief*

$$B'(Q_i) = \sum_{C_i \setminus C_0} B(C_i).$$

2. *Update belief in C_0*

$$B'(C_0) = B(C_0) \prod_{i=1}^{k} B'(Q_i)/B(Q_i).$$

AbsorbThroughSepset *is associated with belief universes.*

Jensen, Lauritzen and Olesen (1990) showed, in the context of a junction tree, that *AbsorbThroughSepset* changes neither the supportiveness of a junction tree nor the joint system belief. We indicate that, in the context of a junction forest, the supportiveness is also invariant after *AbsorbThroughSepset*. This is because *AbsorbThroughSepset* does not increase the support of any linkage host and does not change linkage beliefs directly. The invariance of the joint system belief for the junction forest is obvious given the definition of the joint system belief and the invariance of beliefs for junction trees.

The following three operations bring a junction tree into consistency.

Operation 9.2 (DistributeEvidence) *(Jensen, Lauritzen and Olesen 1990) Let $U_0 = (C_0, B(C_0))$ be a universe in a junction tree. Let* **caller** *be either the junction tree or a neighbour universe. When* **DistributeEvidence** *is called in U_0, the following are performed:*

1. *If* **caller** *is a neighbour U_i, then U_0 absorbs from U_i by* **AbsorbThroughSepset**.

2. *U_0 calls* **DistributeEvidence** *in all neighbours except* **caller** *if* **caller** *is a neighbour.*

DistributeEvidence *is associated with belief universes.*

Suppose a junction tree T^j is consistent. The multiplication of $B(C_0)$ by another belief table may render T^j inconsistent. Such multiplication is performed in evidence entering to be discussed in Section 9.4. If *DistributeEvidence* is then called in U_0, T^j is again consistent.

Operation 9.3 (CollectEvidence) *(Jensen, Lauritzen and Olesen 1990) Let $U_0 = (C_0, B(C_0))$ be a universe in a junction tree. Let* **caller** *be either the junction tree or a neighbour universe. When* **CollectEvidence** *is called in U_0, the following are performed:*

1. *U_0 calls* **CollectEvidence** *in all neighbours except* **caller** *if* **caller** *is a neighbour.*
2. *After the neighbours being called have finished* **CollectEvidence**, *U_0 absorbs from them by* **AbsorbThroughSepset**.

CollectEvidence *is associated with belief universes.*

Both *DistributeEvidence* and *CollectEvidence* are composed of just *AbsorbThroughSepset*. Thus they do not change the supportiveness and the joint system belief of the junction forest.

The combination of *DistributeEvidence* and *CollectEvidence* yields the following *UnifyBelief*. *UnifyBelief* brings a supportive junction tree into consistency.

Operation 9.4 (UnifyBelief) *Let T^i be a junction tree in a junction forest. When* **UnifyBelief** *is initiated by T^i, the following are performed:*

1. *A belief universe $U_0 = (C_0, B(C_0))$ is arbitrarily selected.*
2. **CollectEvidence** *is called in U_0.*
3. *When U_0 has finished* **CollectEvidence**, **DistributeEvidence** *is called in U_0.*

UnifyBelief *is associated with junction trees.*

9.2.2 Operations for belief exchange in belief initialization

Belief initialization brings a junction forest into global consistency before any evidence is available. One problem arises when there are multiple linkages between junction trees. Care must be taken not to count the same information multiple times by passing through different linkages. The following two operations perform information passing through multiple linkages during belief initialization. They ensure that the prior distribution on d-sepsets is exchanged between junction trees without redundant information passing.

Operation 9.5 (NonRedundancyAbsorption) *Let $U_x^a = (C_x^a, B(C_x^a))$ and $U_x^b = (C_x^b, B(C_x^b))$ be two linkage host universes in junction trees T^a and T^b respectively. Let $(L_x, B(L_x))$ and $(R_x, B(R_x))$ be the worlds for corresponding linkage and redundancy set. Suppose $\Delta(B(C_x^a)) \subseteq \Delta(B(L_x))$. When* **NonRedundancyAbsorption** *is called on U_x^a to absorb from U_x^b through linkage L_x, the following updates are performed:*

1. *Update the linkage belief*

$$B'(L_x) = \sum_{C_x^b \setminus L_x} B(C_x^b).$$

2. *Update the belief on redundancy set*

$$B'(R_x) = \sum_{L_x \setminus R_x} B'(L_x).$$

3. *Update the host belief*

$$B'(C_x^a) = B(C_x^a) * \frac{B'(L_x)/B'(R_x)}{B(L_x)/B(R_x)}.$$

The factor $1/B'(R_x)$ *above has the function of* **redundancy removal**. **NonRedundancyAbsorption** *is associated with linkage hosts.*

At initialization, the belief tables for linkages and redundancy sets are in the state of construction and so constant (see Algorithm 7.14 and Definition 7.15). Thus, $B(L_x)$ and $B(R_x)$ above are constant tables. There are three possible consequences of *NonRedundancyAbsorption* depending on the states of the two linkage hosts involved.

1. If $B'(L_x)$ is constant, which is possible because constant probability tables are assigned to d-sepnodes in some sects in Definition 6.1, then after *NonRedundancyAbsorption*

$$\sum_{C_x^a \setminus L_x} B'(C_x^a) \propto \sum_{C_x^a \setminus L_x} B(C_x^a).$$

   That is, if C_x^b has no information to offer, then C_x^a will not change its belief.

2. If $\sum_{C_x^a \setminus L_x} B(C_x^a)$ is constant, then after *NonRedundancyAbsorption*

$$\sum_{C_x^a \setminus L_x} B'(C_x^a) \propto \left(\sum_{C_x^b \setminus L_x} B(C_x^b) \right) \Big/ \left(\sum_{C_x^b \setminus R_x} B(C_x^b) \right).$$

   That is, if C_x^b has new information and C_x^a contains no non-trivial information, then the belief of C_x^b will be copied with redundancy removed.

3. If none of $B'(L_x)$ and $\sum_{C_x^a \setminus L_x} B(C_x^a)$ is constant, then after *NonRedundancyAbsorption*

$$\sum_{C_x^a \setminus L_x} B'(C_x^a) \propto \sum_{C_x^a \setminus L_x} \left(B(C_x^a) * \left(\sum_{C_x^b \setminus L_x} B(C_x^b) \right) \Big/ \left(\sum_{C_x^b \setminus R_x} B(C_x^b) \right) \right).$$

   That is, if none of the above two cases is true, the belief from both sides will be combined with redundancy removed.

The supportiveness of a junction forest is invariant under *NonRedundancyAbsorption*, since

$$\Delta(B(C_x^b)) \subseteq \Delta(\sum_{C_x^b \setminus L_x} B(C_x^b)) = \Delta(B'(L_x)).$$

The joint system belief is invariant under *NonRedundancyAbsorption* since

$$\frac{B'(C_x^a)}{B'(L_x)/B'(R_x)} = \frac{B(C_x^a)}{B(L_x)/B(R_x)}.$$

NonRedundancyAbsorption is associated with linkage hosts.

Operation 9.6 (ExchangeBelief) *Let L be the set of linkages between junction trees T^a and T^b. When* **ExchangeBelief** *is initiated in T^a to exchange belief with T^b, the following is performed:*

> *For each linkage $L_x \in L$ with corresponding hosts U_x^a and U_x^b,* **NonRedundancyAbsorption** *is called in U_x^a to absorb from U_x^b.*

ExchangeBelief *is associated with junction trees.*

Since *ExchangeBelief* is composed of just *NonRedundancyAbsorption*, the supportiveness of the junction tree and its joint system belief are invariant under *ExchangeBelief*. After *ExchangeBelief*, the non-trivial content of joint distribution on d-sepset at T^b is passed onto T^a without redundancy.

9.2.3 Operations for belief update in evidential reasoning

During evidential reasoning, it may be needed to propagate evidence obtained in a junction tree T^b to the rest of the junction forest. A junction tree T^a receiving, from a neighbour junction tree T^b, the updated belief on their d-sepset may be confused due to multiple linkage evidence passing. The following two operations handle the evidence propagation between junction trees. *AbsorbThroughLinkage* propagates evidence from T^b to T^a through one linkage. *UpdateBelief* propagates evidence from T^b to T^a through the set of linkages between the two. *UpdateBelief* is used during evidential reasoning when both T^a and T^b are internally consistent but may not reach boundary consistency between them.

Operation 9.7 (AbsorbThroughLinkage)
Let $U_x^a = (C_x^a, B(C_x^a))$ and $U_x^b = (C_x^b, B(C_x^b))$ be two linkage host universes in junction trees T^a and T^b respectively. Let $(L_x, B(L_x))$ be the corresponding linkage world. Suppose $\Delta(B(C_x^a)) \subseteq \Delta(B(L_x))$. When **AbsorbThroughLinkage** *is called on U_x^a to absorb from U_x^b, the following updates are performed:*

1. *Update the linkage belief*

$$B'(L_x) = \sum_{C_x^b \setminus L_x} B(C_x^b).$$

2. *Update the host belief*

$$B'(C_x^a) = B(C_x^a) * B'(L_x)/B(L_x).$$

AbsorbThroughLinkage *is associated with linkage hosts.*

After *AbsorbThroughLinkage*,

$$\sum_{C_x^a \setminus L_x} B'(C_x^a) = \sum_{C_x^b \setminus L_x} B(C_x^b).$$

The supportiveness of a junction forest is invariant after *AbsorbThroughLinkage*, since

$$\Delta(B(C_x^b)) \subseteq \Delta(\sum_{C_x^b \setminus L_x} B(C_x^b)) = \Delta(B'(L_x)).$$

AbsorbThroughLinkage makes the belief of C_x^a up-to-date with respect to the belief of C_x^b on their common variables.

Operation 9.8 (UpdateBelief) *Let $L = \{L_1, \ldots, L_k\}$ be the set of linkages between junction trees T^a and T^b with U_i^a being the linkage host universe in T^a and U_i^b being the linkage host universe in T^b. When* **UpdateBelief** *is initiated by T^a or is called in T^a to update its belief relative to T^b, the following is performed:*

> *AbsorbThroughLinkage is called in each U_i^a to absorb from U_i^b through L_i. After each* **AbsorbThroughLinkage**, **DistributeEvidence** *is called in U_i^a.*

UpdateBelief *is associated with junction trees.*

Since *UpdateBelief* is composed of *AbsorbThroughLinkage* and *DistributeEvidence*, the supportiveness of the junction forest is invariant.

After *UpdateBelief*, T^a is consistent and

$$\sum_{C_x^a \setminus L_x} B'(C_x^a) = \sum_{C_x^b \setminus L_x} B(C_x^b) \quad x = 1, \ldots, k.$$

Thus the effect of the operation is

$$B'(T^a) = B(T^a) * B'(I)/B(I)$$

where I is the d-sepset between S^a and S^b. Equivalently

$$B'(T^a)/B'(I) = B(T^a)/B(I),$$

which implies the joint system belief is invariant.

Note that, in *UpdateBelief*, *DistributeEvidence* needs to be performed after each *AbsorbThroughLinkage*. Recall that *DistributeEvidence* will restore the consistency in a junction tree if changes on belief are made on

exactly one belief universe. If changes on belief are made on more than one universe, *DistributeEvidence* will not be able to restore the consistency. The following example shows what can happen if *DistributeEvidence* is not performed after each *AbsorbThroughLinkage*.

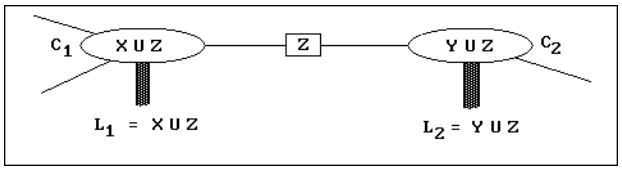

Figure 13: An example illustrating the operation UpdateBelief.

Example 9.9 Let junction tree T^i (Figure 13) have two linkage host $C_1 = L_1 = X \cup Z$ and $C_2 = L_2 = Y \cup Z$ where X, Y, Z are three disjoint sets of nodes. Let $B(C_1)$, $B(C_2)$ and $B(Z)$ be the belief tables of the two hosts and their sepset, respectively. Suppose new information is passed over to T^i through the two linkages from its neighbour junction tree. If *AbsorbThroughLinkage* is performed at C_1 and then C_2 without *DistributeEvidence* being carried out between the two operations, the belief on the two host cliques will be updated to $B'(C_1)$, $B'(C_2)$, while $B(Z)$ is unchanged. If *AbsorbThroughSepset* is called on C_1 to absorb from C_2 in the process of propagating the new information to the rest of T^i, the belief on C_1 will become

$$B''(C_1) = B'(C_1)(\sum_Y B'(C_2)/B(Z)) \not\propto B'(C_1)$$

which is incorrect because $\sum_Y B'(C_2) \not\propto B(Z)$.

When *DistributeEvidence* is performed after each *AbsorbThroughLinkage*, $B'(Z) = \sum_Y B'(C_2)$. The result of *AbsorbThroughSepset* is

$$B''(C_1) = B'(C_1)(\sum_Y B'(C_2)/B'(Z)) \propto B'(C_1)$$

which is correct.

9.3 Belief initialization

Before any evidence is available, an internal representation of beliefs is to be established. The establishment of this representation is termed *initialization* by Lauritzen and Spiegelhalter (1988) for their method. The functionality of initialization in the context of junction forests is to propagate the prior knowledge stored in different belief universes of different junction trees to the rest of the forest such that (1) prior marginal probability distribution for any variable can be obtained in any universe containing the variable, and (2) subsequent evidential reasoning can be performed.

We define operations *DistributeBelief* and *CollectBelief* which are analogous to *DistributeEvidence* and *CollectEvidence* but are associated with junction trees. The operation *BeliefInitialization* relates to *DistributeBelief* and *CollectBelief* just as *UnifyBelief* relates to *DistributeEvidence* and *CollectEvidence*.

Operation 9.10 (DistributeBelief) *Let T^i be a junction tree in a junction forest. Let* **caller** *be either the junction forest or a neighbour junction tree. When* **DistributeBelief** *is called in T^i, the following are performed:*

1. *If* **caller** *is a neighbour T^j, then T^i updates its belief relative to T^j by* **UpdateBelief**.
2. *T^i calls* **DistributeBelief** *in all neighbours except* **caller** *if* **caller** *is a neighbour.*

DistributeBelief *is associated with junction trees.*

Operation 9.11 (CollectBelief) *Let T^i be a junction tree in a junction forest. Let* **caller** *be either the junction forest or a neighbour junction tree. When* **CollectBelief** *is called in T^i, the following are performed:*

1. *T^i calls* **CollectBelief** *in all neighbours except* **caller** *if* **caller** *is a neighbour.*
2. *After each neighbour being called has finished* **CollectBelief**, *T^i exchanges belief respect to the neighbour by* **ExchangeBelief**, *followed by* **UnifyBelief** *in T^i.*

CollectBelief *is associated with junction trees.*

Operation 9.12 (BeliefInitialization) *When* **BeliefInitialization** *is initiated at a junction forest F, the following are performed:*

1. *A junction tree T^i in F is arbitrarily selected.*
2. **CollectBelief** *is called in T^i.*
3. *when T^i has finished* **CollectBelief**. **DistributeBelief** *is called in T^i.*

BeliefInitialization *is associated with the junction forest.*

All three operations do not change the supportiveness and joint system belief. This gives us the following theorem.

Theorem 9.13 (belief initialization with hypertree) *Let $\{S^1, \ldots, S^\beta\}$ be a MSBN with a hypertree structure (Definition 6.11). Let $F = \{T^1, \ldots, T^\beta\}$ be a junction forest with T^i being the junction tree of S^i. Let $B(F)$ be the joint system belief constructed as in Section 7.3. After BeliefInitialization, the junction forest is globally consistent.*

Example 9.14 After *BeliefInitialization* is initiated at the junction forest in Figure 5, Γ^1 is selected and *CollectBelief* is called in it. It then calls *CollectBelief* in Γ^2 and Γ^3. Since Γ^2 and Γ^3 do not have neighbours other than Γ^1, only *UnifyBelief* is performed in Γ^2 and Γ^3. Table 4 lists the belief tables for belief universes in junction trees Γ^2 and Γ^3 after their *UnifyBeliefs*.

Table 5 gives the belief tables for belief universes of the junction forest. Table 6 gives the (prior) marginal probabilities for all variables of junction forest after the completion of *BeliefInitialization*. The marginal probabilities are identical to what would be derived from the USBN (Γ, P) where Γ is given in Figure 2 and P is given in Table 1.

$B(\Gamma^2)$

$Clique$	$NodeAss.$
$\{F_2, F_1\}$	F_2
$Config.$	$B()$
$\{f_{21}, f_{11}\}$	.7758
$\{f_{21}, f_{12}\}$	1.545
$\{f_{22}, f_{11}\}$	1.164
$\{f_{22}, f_{12}\}$	.5151
$Clique$	$NodeAss.$
$\{H_2, F_1, H_1\}$	H_2, F_1, H_1
$Config.$	$B()$
$\{h_{21}, f_{11}, h_{11}\}$	.7895
$\{h_{21}, f_{11}, h_{12}\}$	.6
$\{h_{21}, f_{12}, h_{11}\}$	.2105
$\{h_{21}, f_{12}, h_{12}\}$	.4
$\{h_{22}, f_{11}, h_{11}\}$	.5
$\{h_{22}, f_{11}, h_{12}\}$	.05
$\{h_{22}, f_{12}, h_{11}\}$	.5
$\{h_{22}, f_{12}, h_{12}\}$	.95

$B(\Gamma^3)$

$Clique$	$NodeAss.$
$\{H_3, H_2, E_3\}$	H_3, H_2, E_3
$Config.$	$B()$
$\{h_{31}, h_{21}, e_{31}\}$	1.54
$\{h_{31}, h_{21}, e_{32}\}$	.4596
$\{h_{31}, h_{22}, e_{31}\}$	1.3
$\{h_{31}, h_{22}, e_{32}\}$	.7
$\{h_{32}, h_{21}, e_{31}\}$	.7
$\{h_{32}, h_{21}, e_{32}\}$	1.3
$\{h_{32}, h_{22}, e_{31}\}$	.02
$\{h_{32}, h_{22}, e_{32}\}$	1.98
$Clique$	$NodeAss.$
$\{E_2, E_3, E_1\}$	E_2
$Config.$	$B()$
$\{e_{21}, e_{31}, e_{11}\}$	1.656
$\{e_{21}, e_{31}, e_{12}\}$	1.495
$\{e_{21}, e_{32}, e_{11}\}$	1.898
$\{e_{21}, e_{32}, e_{12}\}$	.1165
$\{e_{22}, e_{31}, e_{11}\}$	.0356
$\{e_{22}, e_{31}, e_{12}\}$	.3738
$\{e_{22}, e_{32}, e_{11}\}$	.2109
$\{e_{22}, e_{32}, e_{12}\}$	2.214
$Clique$	$NodeAss.$
$\{E_3, E_1, H_4\}$	E_1, H_4
$Config.$	$B()$
$\{e_{31}, e_{11}, h_{41}\}$	1.424
$\{e_{31}, e_{11}, h_{42}\}$	.2670
$\{e_{31}, e_{12}, h_{41}\}$	.3560
$\{e_{31}, e_{12}, h_{42}\}$	1.513
$\{e_{32}, e_{11}, h_{41}\}$	1.776
$\{e_{32}, e_{11}, h_{42}\}$	.3330
$\{e_{32}, e_{12}, h_{41}\}$	.4440
$\{e_{32}, e_{12}, h_{42}\}$	1.887
$Clique$	$NodeAss.$
$\{H_3, E_3, H_4\}$	
$Config.$	$B()$
$\{h_{31}, e_{31}, h_{41}\}$	1.42
$\{h_{31}, e_{31}, h_{42}\}$	1.42
$\{h_{31}, e_{32}, h_{41}\}$	.5798
$\{h_{31}, e_{32}, h_{42}\}$	.5798
$\{h_{32}, e_{31}, h_{41}\}$	.36
$\{h_{32}, e_{31}, h_{42}\}$	.36
$\{h_{32}, e_{32}, h_{41}\}$	1.64
$\{h_{32}, e_{32}, h_{42}\}$	1.64

Table 4: Belief tables for belief universes in junction trees Γ^2 and Γ^3 in Figure 5 after CollectBelief during BeliefInitialization.

Once belief initialization is completed, the junction forest becomes the permanent representation which will be reused for each query session.

9.4 Evidential reasoning

The joint system belief defined in Section 7.3 is proportional to the *prior* joint distribution representing the background domain knowledge. Initialization allows us to obtain prior marginal probabilities with efficient local computation. When evidence about a particular case becomes available, we want the prior distribution to change into the posterior distribution. Call the overall process of entering evidence and propagating evidence *evidential reasoning*.

$B(\Gamma^1)$

$Clique$	$NodeAss.$
$\{H_2, H_1, A_1\}$	H_1, A_1
$Config.$	$B()$
$\{h_{21}, h_{11}, h_{11}\}$	.8203
$\{h_{21}, h_{11}, h_{12}\}$	.08166
$\{h_{21}, h_{12}, h_{11}\}$	.5810
$\{h_{21}, h_{12}, h_{12}\}$	2.082
$\{h_{22}, h_{11}, h_{11}\}$	.3797
$\{h_{22}, h_{11}, h_{12}\}$	.2183
$\{h_{22}, h_{12}, h_{11}\}$	.2690
$\{h_{22}, h_{12}, h_{12}\}$	5.568
$Clique$	$NodeAss.$
$\{H_2, A_2, A_1\}$	H_2
$Config.$	$B()$
$\{h_{21}, a_{21}, a_{11}\}$	.5526
$\{h_{21}, a_{21}, a_{12}\}$	1.725
$\{h_{21}, a_{22}, a_{11}\}$	.8487
$\{h_{21}, a_{22}, a_{12}\}$	.4388
$\{h_{22}, a_{21}, a_{11}\}$	.08289
$\{h_{22}, a_{21}, a_{12}\}$	.7394
$\{h_{22}, a_{22}, a_{11}\}$	.5658
$\{h_{22}, a_{22}, a_{12}\}$	5.047
$Clique$	$NodeAss.$
$\{H_3, H_2, A_2\}$	H_3, A_2
$Config.$	$B()$
$\{h_{31}, h_{21}, a_{21}\}$	1.763
$\{h_{31}, h_{21}, a_{22}\}$	.1120
$\{h_{31}, h_{22}, a_{21}\}$	.6366
$\{h_{31}, h_{22}, a_{22}\}$	.4880
$\{h_{32}, h_{21}, a_{21}\}$	.5143
$\{h_{32}, h_{21}, a_{22}\}$	1.176
$\{h_{32}, h_{22}, a_{21}\}$	.1857
$\{h_{32}, h_{22}, a_{22}\}$	5.124
$Clique$	$NodeAss.$
$\{H_3, A_3, H_4\}$	A_3, H_4
$Config.$	$B()$
$\{h_{31}, a_{31}, h_{41}\}$	.225
$\{h_{31}, a_{31}, h_{42}\}$	.675
$\{h_{31}, a_{32}, h_{41}\}$	.84
$\{h_{31}, a_{32}, h_{42}\}$	1.26
$\{h_{32}, a_{31}, h_{41}\}$	1.4
$\{h_{32}, a_{31}, h_{42}\}$	4.2
$\{h_{32}, a_{32}, h_{41}\}$	.56
$\{h_{32}, a_{32}, h_{42}\}$	.84

$B(\Gamma^2)$

$Clique$	$NodeAss.$
$\{F_2, F_1\}$	F_2
$Config.$	$B()$
$\{f_{21}, f_{11}\}$	1.160
$\{f_{21}, f_{12}\}$	5.324
$\{f_{22}, f_{11}\}$	1.741
$\{f_{22}, f_{12}\}$	1.775
$Clique$	$NodeAss.$
$\{H_2, F_1, H_1\}$	H_2, F_1, H_1
$Config.$	$B()$
$\{h_{21}, f_{11}, h_{11}\}$	.7121
$\{h_{21}, f_{11}, h_{12}\}$	1.598
$\{h_{21}, f_{12}, h_{11}\}$	.1899
$\{h_{21}, f_{12}, h_{12}\}$	1.065
$\{h_{22}, f_{11}, h_{11}\}$	.2990
$\{h_{22}, f_{11}, h_{12}\}$	.2918
$\{h_{22}, f_{12}, h_{11}\}$	.2990
$\{h_{22}, f_{12}, h_{12}\}$	5.545

$B(\Gamma^3)$

$Clique$	$NodeAss.$
$\{H_3, H_2, E_3\}$	H_3, H_2, E_3
$Config.$	$B()$
$\{h_{31}, h_{21}, e_{31}\}$	1.444
$\{h_{31}, h_{21}, e_{32}\}$	.4310
$\{h_{31}, h_{22}, e_{31}\}$	.731
$\{h_{31}, h_{22}, e_{32}\}$	.3936
$\{h_{32}, h_{21}, e_{31}\}$	.5915
$\{h_{32}, h_{21}, e_{32}\}$	1.098
$\{h_{32}, h_{22}, e_{31}\}$	.0531
$\{h_{32}, h_{22}, e_{32}\}$	5.257
$Clique$	$NodeAss.$
$\{E_2, E_3, E_1\}$	E_2
$Config.$	$B()$
$\{e_{21}, e_{31}, e_{11}\}$	1.020
$\{e_{21}, e_{31}, e_{12}\}$	1.422
$\{e_{21}, e_{32}, e_{11}\}$	2.182
$\{e_{21}, e_{32}, e_{12}\}$	.2378
$\{e_{22}, e_{31}, e_{11}\}$	.02194
$\{e_{22}, e_{31}, e_{12}\}$	.3555
$\{e_{22}, e_{32}, e_{11}\}$	.2424
$\{e_{22}, e_{32}, e_{12}\}$	4.518
$Clique$	$NodeAss.$
$\{E_3, E_1, H_4\}$	E_1, H_4
$Config.$	$B()$
$\{e_{31}, e_{41}, h_{11}\}$	.7622
$\{e_{31}, e_{41}, h_{12}\}$	.2801
$\{e_{31}, e_{42}, h_{11}\}$	.1906
$\{e_{31}, e_{42}, h_{12}\}$	1.587
$\{e_{32}, e_{41}, h_{11}\}$	1.658
$\{e_{32}, e_{41}, h_{12}\}$	.7662
$\{e_{32}, e_{42}, h_{11}\}$	.4145
$\{e_{32}, e_{42}, h_{12}\}$	4.342
$Clique$	$NodeAss.$
$\{H_3, E_3, H_4\}$	
$Config.$	$B()$
$\{h_{31}, e_{31}, h_{41}\}$	.7723
$\{h_{31}, e_{31}, h_{42}\}$	1.403
$\{h_{31}, e_{32}, h_{41}\}$	.2927
$\{h_{31}, e_{32}, h_{42}\}$	.5319
$\{h_{32}, e_{31}, h_{41}\}$	.1805
$\{h_{32}, e_{31}, h_{42}\}$	.4641
$\{h_{32}, e_{32}, h_{41}\}$	1.780
$\{h_{32}, e_{32}, h_{42}\}$	4.576

Table 5: Belief tables for belief universes of junction forest $F = \{\Gamma^1, \Gamma^2, \Gamma^3\}$ in Figure 5 obtained after the completion of BeliefInitialization.

A piece of *evidence* is a conjunction of values of variables such that the variables are contained in the *same* sect (localization), and the values are obtained at one time. As the value of a variable, we allow evidence to specify a disjunction of outcomes (e.g. only one outcome, or ruling out one outcome). There may be multiple pieces of evidence, in a query session, that may involve different sects and may be obtained at different time (incremental evidence). We assume that, after each piece of evidence is available, the posterior distribution

$p(h_{11}) =$	.15	$p(a_{11}) =$	.205	$p(f_{11}) =$	.2901	$p(e_{11}) =$	.3466
$p(h_{21}) =$	.3565	$p(a_{21}) =$	.31	$p(f_{21}) =$	.6485	$p(e_{21}) =$	.4862
$p(h_{31}) =$	.3	$p(a_{31}) =$	.65			$p(e_{31}) =$	.282
$p(h_{41}) =$	.3025						

Table 6: Prior marginal probabilities from junction forest $F = \{\Gamma^1, \Gamma^2, \Gamma^3\}$ in Figure 5 obtained after the completion of BeliefInitialization.

on the variables in the current sect is to be computed.

Evidence is represented in terms of evidence functions in the same manner as Jensen, Olesen and Andersen (1990). A evidence function maps the outcomes of one variable to $\{0, 1\}$. '0' stands for the fact that the corresponding outcome is impossible and '1' stands for the fact that the corresponding outcome is still possible. An evidence function can be entered to a junction forest by multiplying the prior distribution with the evidence function.

Operation 9.15 (EnterEvidence) *(Jensen, Lauritzen and Olesen 1990) Let T be a junction tree in a junction forest. When* **EnterEvidence** *is initiated at T to enter a piece of evidence E, the following are performed:*

1. *For each variable A_i involved in E a belief universe $U_j = (C_j, B(C_j))$ such that $A_i \in C_j$ is arbitrarily selected, and $B(C_j)$ is multiplied by the evidence function for A_i.*

2. *If U_j is the only universe that is affected by the above step,* **DistributeEvidence** *is called in U_j, otherwise* **UnifyBelief** *is called in any universe U_k.*

EnterEvidence *is associated with junction trees.*

After *EnterEvidence*, the junction T is updated with respect to the evidence and is consistent internally. As far as a single junction tree is concerned, this computation is the same as the junction tree technique where the consistency within a junction tree constitutes the global consistency. However, in order to obtain correct posterior marginal distributions on variables in the currently active junction tree, the global consistency of the junction forest is *not* necessary. Before a formal treatment, several concepts are defined below.

Here only junction forests transformed from MSBNs with hypertree structures are considered. When a user wants to obtain marginal distributions or add evidence on variables not contained in the currently active junction tree, it is said that there is an *attention shift*. The junction tree which contains the desired variables is called the *destination tree*.

Definition 9.16 (intermediate tree) *Let S^i, S^j, S^k be three different sects in a MSBN with a hypertree structure, and T^i, T^j, T^k be their junction trees in the corresponding junction forest F, respectively. T^j is the* intermediate tree *between T^i and T^k if the removal of T^j would disconnect T^i from T^k in F.*

Due to the hypertree structure, we have the following lemma.

Lemma 9.17 *Suppose a junction forest has been transformed from a MSBN with a hypertree structure. Let T^i and T^j be two different junction trees in the forest. The set of intermediate junction trees between T^i and T^j is unique.*

The following defines an operation *ShiftAttention* at the junction forest level. It is performed when the user's attention shifts.

Operation 9.18 (ShiftAttention) *Let F be a junction forest whose corresponding MSBN has a hypertree structure. Let T^{j_0} be the currently active tree and $T^{j_{m+1}}$ be the destination tree in F. Let $\{T^{j_1}, \ldots, T^{j_m}\}$ be the set of m intermediate trees between T^{j_0} and $T^{j_{m+1}}$ such that $T^{j_0}, T^{j_1}, \ldots, T^{j_m}, T^{j_{m+1}}$ form a chain of neighbours. When* **ShiftAttention** *is initiated at F to shift attention from T^{j_0} to $T^{j_{m+1}}$, the following are performed:*

For $i = 1$ to $m+1$, **UpdateBelief** *is called in T^{j_i} to update its belief with respect to $T^{j_{i-1}}$.*

ShiftAttention *is associated with the junction forest.*

Before each attention shift, several pieces of evidence can be entered to the currently active tree. When an attention shift happens, ShiftAttention swaps in and out only the intermediate trees between the currently active tree and destination tree without the participation of the rest of the forest. The following theorem shows that this is sufficient in order to obtain the correct marginal distributions in the destination tree.

Theorem 9.19 (multiple attention shifts) *Let F be a consistent junction forest whose corresponding MSBN has a hypertree structure. Start with any active junction tree. Repeat the following cycle a finite number of times:*

1. *Use* **EnterEvidence** *to enter evidence to the currently active tree a finite number of times.*
2. *Use* **ShiftAttention** *to shift attention to any destination tree.*

The marginal distributions obtained in the final active tree are identical as would be obtained when the forest is globally consistent.

Example 9.20 Let us continue the example with the junction forest (Figure 5) $F = \{\Gamma^1, \Gamma^2, \Gamma^3\}$ by demonstration of evidential reasoning. Suppose the outcome of variable E_3 in Γ^3 is observed to be e_{31}. Table 7 lists $B'(\Gamma^3)$, which is obtained after *EnterEvidence*, and also $B'(\Gamma^1)$ and $B'(\Gamma^2)$, which are obtained after *ShiftAttention* with destination Γ^2. Table 8 lists the posterior marginal probability of each variable after *ShiftAttention*.

$B'(\Gamma^3)$		$B'(\Gamma^1)$		$B'(\Gamma^2)$	
$Clique$	$NodeAss.$	$Clique$	$NodeAss.$	$Clique$	$NodeAss.$
$\{H_3, H_2, E_3\}$	H_3, H_2, E_3	$\{H_2, H_1, A_1\}$	H_1, A_1	$\{F_2, F_1\}$	F_2
$Config.$	$B()$	$Config.$	$B()$	$Config.$	$B()$
$\{h_{31}, h_{21}, e_{31}\}$	14.44	$\{h_{21}, h_{11}, h_{11}\}$	4.105	$\{f_{21}, f_{11}\}$	5.523
$\{h_{31}, h_{21}, e_{32}\}$	0	$\{h_{21}, h_{11}, h_{12}\}$	.5036	$\{f_{21}, f_{12}\}$	10.79
$\{h_{31}, h_{22}, e_{31}\}$	7.31	$\{h_{21}, h_{12}, h_{11}\}$	2.908	$\{f_{22}, f_{11}\}$	8.285
$\{h_{31}, h_{22}, e_{32}\}$	0	$\{h_{21}, h_{12}, h_{12}\}$	12.84	$\{f_{22}, f_{12}\}$	3.598
$\{h_{32}, h_{21}, e_{31}\}$	5.915	$\{h_{22}, h_{11}, h_{11}\}$	.4627	$Clique$	$NodeAss.$
$\{h_{32}, h_{21}, e_{32}\}$	0	$\{h_{22}, h_{11}, h_{12}\}$	.2661	$\{H_2, F_1, H_1\}$	H_2, F_1, H_1
$\{h_{32}, h_{22}, e_{31}\}$	.5310	$\{h_{22}, h_{12}, h_{11}\}$	.3278	$Config.$	$B()$
$\{h_{32}, h_{22}, e_{32}\}$	0	$\{h_{22}, h_{12}, h_{12}\}$	6.785	$\{h_{21}, f_{11}, h_{11}\}$	3.638
$Clique$	$NodeAss.$	$Clique$	$NodeAss.$	$\{h_{21}, f_{11}, h_{12}\}$	9.450
$\{E_2, E_3, E_1\}$	E_2	$\{H_2, A_2, A_1\}$	H_2	$\{h_{21}, f_{12}, h_{11}\}$	.9702
$Config.$	$B()$	$Config.$	$B()$	$\{h_{21}, f_{12}, h_{12}\}$	6.300
$\{e_{21}, e_{31}, e_{11}\}$	1.020	$\{h_{21}, a_{21}, a_{11}\}$	3.732	$\{h_{22}, f_{11}, h_{11}\}$	.3644
$\{e_{21}, e_{31}, e_{12}\}$	1.422	$\{h_{21}, a_{21}, a_{12}\}$	11.65	$\{h_{22}, f_{11}, h_{12}\}$	.3556
$\{e_{21}, e_{32}, e_{11}\}$	0	$\{h_{21}, a_{22}, a_{11}\}$	3.281	$\{h_{22}, f_{12}, h_{11}\}$	.3644
$\{e_{21}, e_{32}, e_{12}\}$	0	$\{h_{21}, a_{22}, a_{12}\}$	1.696	$\{h_{22}, f_{12}, h_{12}\}$	6.757
$\{e_{22}, e_{31}, e_{11}\}$	.02194	$\{h_{22}, a_{21}, a_{11}\}$	.4190		
$\{e_{22}, e_{31}, e_{12}\}$	.3555	$\{h_{22}, a_{21}, a_{12}\}$	3.737		
$\{e_{22}, e_{32}, e_{11}\}$	0	$\{h_{22}, a_{22}, a_{11}\}$	.3715		
$\{e_{22}, e_{32}, e_{12}\}$	0	$\{h_{22}, a_{22}, a_{12}\}$	3.313		
$Clique$	$NodeAss.$	$Clique$	$NodeAss.$		
$\{E_3, E_1, H_4\}$	E_1, H_4	$\{H_3, H_2, A_2\}$	H_3, A_2		
$Config.$	$B()$	$Config.$	$B()$		
$\{e_{31}, e_{11}, h_{41}\}$	7.622	$\{h_{31}, h_{21}, a_{21}\}$	13.58		
$\{e_{31}, e_{11}, h_{42}\}$	2.801	$\{h_{31}, h_{21}, a_{22}\}$	.8623		
$\{e_{31}, e_{12}, h_{41}\}$	1.906	$\{h_{31}, h_{22}, a_{21}\}$	4.138		
$\{e_{31}, e_{12}, h_{42}\}$	15.87	$\{h_{31}, h_{22}, a_{22}\}$	3.172		
$\{e_{32}, e_{11}, h_{41}\}$	0	$\{h_{32}, h_{21}, a_{21}\}$	1.800		
$\{e_{32}, e_{11}, h_{42}\}$	0	$\{h_{32}, h_{21}, a_{22}\}$	4.115		
$\{e_{32}, e_{12}, h_{41}\}$	0	$\{h_{32}, h_{22}, a_{21}\}$	.01857		
$\{e_{32}, e_{12}, h_{42}\}$	0	$\{h_{32}, h_{22}, a_{22}\}$	.5124		
$Clique$	$NodeAss.$	$Clique$	$NodeAss.$		
$\{H_3, E_3, H_4\}$		$\{H_3, A_3, H_4\}$	A_3, H_4		
$Config.$	$B()$	$Config.$	$B()$		
$\{h_{31}, e_{31}, h_{41}\}$	7.723	$\{h_{31}, a_{31}, h_{41}\}$	1.632		
$\{h_{31}, e_{31}, h_{42}\}$	14.03	$\{h_{31}, a_{31}, h_{42}\}$	4.895		
$\{h_{31}, e_{32}, h_{41}\}$	0	$\{h_{31}, a_{32}, h_{41}\}$	6.091		
$\{h_{31}, e_{32}, h_{42}\}$	0	$\{h_{31}, a_{32}, h_{42}\}$	9.137		
$\{h_{32}, e_{31}, h_{41}\}$	1.805	$\{h_{32}, a_{31}, h_{41}\}$	1.289		
$\{h_{32}, e_{31}, h_{42}\}$	4.641	$\{h_{32}, a_{31}, h_{42}\}$	3.867		
$\{h_{32}, e_{32}, h_{41}\}$	0	$\{h_{32}, a_{32}, h_{41}\}$	.5157		
$\{h_{32}, e_{32}, h_{42}\}$	0	$\{h_{32}, a_{32}, h_{42}\}$	.7735		

Table 7: Belief tables updated after entering evidence for $F = \{\Gamma^1, \Gamma^2, \Gamma^3\}$ where Γ^is are as in Figure 5. First, $B'(\Gamma^3)$ is obtained after $E_3 = e_{31}$ is entered to Γ^3 by *EnterEvidence* in Γ^3. $B'(\Gamma^1)$ and $B'(\Gamma^2)$ are obtained afterwards by *ShiftAttention*.

$p(h_{11}) =$	.1893	$p(a_{11}) =$	.2767	$p(f_{11}) =$	.4897	$p(e_{11}) =$	.3696
$p(h_{21}) =$	.7219	$p(a_{21}) =$	.6928	$p(f_{21}) =$	.5786	$p(e_{21}) =$	.8661
$p(h_{31}) =$	.7714	$p(a_{31}) =$	.4143			$p(e_{31}) =$	1
$p(h_{41}) =$	.3379						

Table 8: Posterior probabilities from junction forest $F = \{\Gamma^1, \Gamma^2, \Gamma^3\}$ in Figure 5 after evidence $E_3 = e_{31}$ is propagated by ShiftAttention.

Before closing this subsection, the following example demonstrates the computational advantage, during attention shift, provided by the covering sects.

Example 9.21 In Figure 8, D is sectioned into $\{D^1, D^2, D^3\}$ by sound sectioning without a covering sect. The MSBN is transformed into the junction forest $\{T^1, T^2, T^3\}$ by an invertible transformation. If evidence about E and then about G comes, the first piece of evidence will be entered to T^1 and then T^1 will send message to T^2. After entering the second piece of evidence, T^2 will be the only one up-to-date. Now if we are interested in the belief on H, the belief tables on $\{B, F\}$ and $\{C, F\}$ in T^3 have to be updated. However, T^2 cannot provide distribution on $\{B, F\}$. Thus, T^2 has to send message to T^3 about $\{C, F\}$ and then send message to T^1. T^1 can then become up-to-date and send distribution on $\{B, F\}$ to T^3. Three instances of message passing are necessary, and linkages between each pair of junction trees have to be created and maintained. More message passing and more linkages are needed when there are more sects organized in this structure. When n sects are interconnected and there is a covering sect, only n-1 sets of linkages need to be created; and a maximum of two message passings are needed to update the belief in any destination tree.

9.5 Computational complexity

Theorem 9.19 shows the most important characterization of the MSBN and junction forests, namely, the capability of exploiting localization to reduce the computational complexity.

Under the assumption of localization, the user interest and new evidence remain in the sphere of one junction tree for a period of time. Thus the time and space requirement, while reasoning within a junction tree, is bounded above by what is required by the largest junction tree. The judgments obtained, however, are at the knowledge level of the overall junction forest. Compared to the USBN and the single junction tree representation where the evidence has to be propagated to the entire system, this leads to savings when localization is valid.

When the user shifts interest to another set of variables contained in a different destination tree, only the intermediate trees need to be updated. The time required is linear to the number of intermediate trees and to the number of linkages between each pair of neighbours. No matter how large the entire junction forest, the time requirement for attention shift is fixed once the destination tree and mediating trees are fixed. For example, in a MSBN with a covering sect, no matter how many sects are in the MSBN, the attention shift updates a maximum of two sects. The space requirement is bounded above by what is needed by the largest junction tree on the path between the starting and destination trees. Under the localization assumption, the computational cost for attention shift is incurred only occasionally.

Given the above analysis, the computational complexity of evidential reasoning in a MSBN with β sects

of equal size is about $1/\beta$ of the corresponding USBN system when localization is valid. The actual time requirement is a little more than $1/\beta$ due to the computation required for attention shift. The actual space requirement is a little more than $1/\beta$ due to the repetition of d-sepnodes and the set of linkages required for attention shift.

The MSBN and the junction forest technique has been implemented in PAINULIM (Xiang et al. 1992) - a system for diagnosis of neuromuscular diseases characterized by a painful impaired upper limb. The system has three sects: the clinical, the EMG, and the nerve conduction with the clinical sect being the covering sect. According to the statistics in Xiang et al. (1992), about 27% of patients in this category need nerve conduction studies only, and about 60% of patients need EMG tests only. Thus, for about 87% of the patients, about one third of the junction forests will not be computed at all, and there is only one attention shift (from the clinical sect to either the nerve conduction or the EMG sect). There are five clinical findings on an average patient. The number of tests performed on an average patient is about four for nerve conduction and about six for EMG. After each clinical finding and each test, users would like to know posterior probabilities for diseases and outcomes of possible examinations or tests not yet performed. Thus, the localization assumption works well in the PAINULIM domain. The overall computational savings in PAINULIM by applying the MSBN and the junction forest technique is about half.

10 Summary

This paper presents MSBNs and junction forests as a flexible knowledge representation and as an efficient inference formalisms to exploit localization naturally existing in large knowledge-based systems. The systems which can benefit from the technique are those that are reusable, representable by general but sparse networks, and characterized by incremental evidential reasoning and where localization is valid.

The MSBNs allow the partition of a large application domain into smaller natural subdomains such that each of them can be represented as a Bayesian subnetwork (a sect), and can be tested and refined individually. This makes the representation of a complex domain easier for knowledge engineers and potentially makes the resultant system more natural and more understandable to system users. The modularity facilitates implementation of large systems in an incremental fashion. When partitioning, a knowledge engineer has to take into account the technical constraints imposed by the MSBN, namely that the interfaces must be d-sepsets and the sectioning must be sound. These constraints are not very restrictive.

Two important guidelines, the covering subDAG rule and the hypertree rule, for sound sectioning are derived. MSBNs that follow the rules can have multiply connected sects, do not require expensive computation to verify soundness of sectioning, and have additional computational advantage during attention shift in evidential reasoning.

Each sect in the MSBN is transformed into a junction tree such that the MSBN is transformed into a junction forest representation where evidential reasoning takes place. The constraints on transformation are the invertibility of morali-triangulation and separability.

Each sect/junction tree in the MSBN/junction forest stands as a separate computational object. Since the technique allows transformation of sects into junction trees through local computation at the sect level, and allows reasoning to be conducted with junction trees as units, the space requirement is governed by the size of one sect/junction tree. Hence large applications can be built and run on relatively small computers wherever hardware resources are of concern. This was, in fact, our original motivation to develop the MSBN technique.

For large application domains, an average case may involve only a portion of the total knowledge encoded in a system, and one portion may be used repeatedly over a period of time. A MSBN and a junction forest representation allows the 'interesting' or 'relevant' sect/junction tree to be loaded while the rest of the junction forest remains inactive and uses no computational resources. The judgments made on variables in the active junction tree are consistent with all the knowledge available, including both prior knowledge and all the evidence contained in the entire junction forest. When the user's attention shifts, inactive junction trees can be made active and previous accumulation of evidence is preserved. This is achieved by passing the joint beliefs on d-sepsets. The overall computational resource required is governed by the size of the largest sect, and not by the size of the application domain.

The technique of the MSBN and the junction forest has been applied to an application knowledge-based system PAINULIM capable of diagnosing neuromuscular diseases characterized by a painful impaired upper limb. Our experience with PAINULIM supports the significance of the technique (Xiang et al. 1992).

Acknowledgements

This work is supported by Operating Grants A3290, OGPOO44121 from NSERC and a UBC Graduate Fellowship to Y. Xiang. Revision of the final draft is partially supported by the University College of the Cariboo. The authors would like to thank Andrew Eisen and Bhanu Pant, whose cooperation in the PAINULIM project has inspired many of the ideas included in this paper. Our thanks are also directed to Finn V. Jensen who was kind enough to send us his papers. We are grateful to Lianwen Zhang for his helpful comments on the earlier draft of this paper.

References

Andersen, S.K., Olesen, K.G., Jensen, F.V. and Jensen, F. 1989. HUGIN - a shell for building Bayesian belief universes for expert systems. *Proceedings of the Eleventh International Joint Con-*

ference on Artificial Intelligence, Detroit, Michigan, Vol. 2, pp. 1080-1085.

Baker, M. and Boult, T.E. 1990. Pruning Bayesian networks for efficient computation. *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, Mass., pp. 257-264.

Cooper G.F. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42, pp. 393-405.

Geiger, D., Verma, T. and Pearl, J. 1990. d-separation: from theorems to algorithms. In *Uncertainty in Artificial Intelligence 5.* Edited by M. Henrion, R.D. Shachter, L.N. Kanal and J.F. Lemmer. Elsevier Science Publishers, pp. 139-148.

Golumbic, M.C. 1980. *Algorithmic graph theory and perfect graphs.* Academic Press, NY.

Heckerman, D. 1990a. A tractable inference algorithm for diagnosing multiple diseases. In *Uncertainty in Artificial Intelligence 5.* Edited by M. Henrion, R.D. Shachter, L.N. Kanal and J.F. Lemmer. Elsevier Science Publishers, pp. 163-171.

Heckerman, D. 1990b. *Probabilistic Similarity Networks*, Ph.D. Thesis, Stanford University.

Henrion, M. 1988. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. *Uncertainty in Artificial Intelligence 2.* Edited by J.F. Lemmer and L.N. Kanal, Elsevier Science Publishers, pp. 149-163.

Jensen, F.V. 1988. Junction tree and decomposable hypergraphs. *JUDEX Research Report*, Aalborg, Denmark.

Jensen, F.V., Lauritzen, S.L. and Olesen, K.G. 1990. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly.* 4, pp. 269-282.

Jensen, F. and Andersen, S.K. 1990. Approximations in Bayesian belief universes for knowledge-based systems. *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, Cambridge, Mass., pp. 162-169.

Jensen, F.V., Olesen, K.G. and Andersen, S.K. 1990. An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, Vol. 20, pp. 637-659.

Lauritzen, S.L. and Spiegelhalter, D.J. 1988. Local computation with probabilities on graphical structures, and their application to expert systems. *Journal of the Royal Statistical Society*, Series B, 50: 157-244.

Lauritzen, S.L., Speed, T.P. and Vijayan, K. 1984. Decomposable graphs and hypergraphs. *Journal of Australian Mathematical Society*, Series A, 36: 12-29.

Maier, D. 1983. *The Theory of Relational Databases.* Computer Science Press.

Neapolitan, R.E. 1990. *Probabilistic Reasoning in Expert Systems.* John Wiley & Sons.

Nii, H.P. 1986a. Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, Vol. 7, No. 2, pp. 38-53.

Nii, H.P. 1986b. Blackboard systems: blackboard application systems, blackboard systems from a knowledge engineering perspective. *AI Magazine*, Vol. 7, No. 3, pp. 82-106.

Pearl, J. 1986. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29: 241-288.

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

Shafer, G., and Shenoy, P.P. 1988. Local computation in hypertrees. School of Business Working Paper No. 201, University of Kansas, U.S.A.

Shachter, R.D 1988. Discussion published in (Lauritzen and Spiegelhalter 1988).

Xiang, Poole, D., and Beddoes, M.P. 1992. Multiply sectioned Bayesian networks and junction forests for large knowledge-based systems. Tech. Report CSS-IS TR 92-04, Simon Fraser University.

Xiang, Y. Pant, B., Eisen, A., Beddoes, M.P., and Poole, D. 1992. Multiply sectioned Bayesian networks for neuromuscular diagnosis. To appear in *Artificial Intelligence in Medicine.*

Appendix

Term	Definition
AbsorbThroughLinkage	Section 9.2.3
AbsorbThroughSepset	Section 9.2.1
belief universe	Section 2.4
BeliefInitialization	Section 9.3
clique	Section 2
CollectBelief	Section 9.3
CollectEvidence	Section 9.2.1
consistency	Section 8.1
covering subDAG	Theorem 6.8
d-sepnode	Definition 5.1
d-sepset	Definition 5.1
DistributeBelief	Section 9.3
DistributeEvidence	Section 9.2.1
ExchangeBelief	Section 9.2.2
host tree	Definition 8.7
hypertree	Definition 6.11
invertible morali-triangulation	Definition 7.1
junction forest	Section 7
junction tree	Section 2
linkage	Definition 7.7
linkage tree	Algorithm 7.10
marginal probability	Section 2.2
marginalization	Section 2.3
moral graph	Section 2
morali-triangulation	Section 4
MSBN	Definition 6.1
neighbour sect	Definition 6.2
neighbour junction tree	Definition 7.22
NonRedundancyAbsorption	Section 9.2.2
redundancy set	Definition 7.11
separability	Definition 8.5
ShiftAttention	Section 9.4
soundness of sectioning	Definition 6.4
supportiveness	Section 9.1
triangulated graph	Section 2
UnifyBelief	Section 9.2.1
UpdateBelief	Section 9.2.3
world	Section 2.3