Junction trees constructions in Bayesian networks

Linda Smail
Department of Mathematics and Statistics, Journal of Physics: Conference Series, IOP Publishing, Zayed University, P.O.Box 19282 Dubai, UAE
E-mail: linda.smail@zu.ac.ae

Abstract. Junction trees are used as graphical structures over which propagation will be carried out through a very important property called the ruining intersection property. This paper examines an alternative method for constructing junction trees that are essential for the efficient computations of probabilities in Bayesian networks. The new proposed method converts a sequence of subsets of a Bayesian network into a junction tree, in other words, into a set of cliques that has the running intersection property. The obtained set of cliques and separators coincide with the junction trees obtained by the moralization and triangulation process, but it has the advantage of adapting to any computational task by adding links to the Bayesian network graph.

1. Introduction
Inference is one of the important tasks in Bayesian networks (BNs). It consists of computing marginal probabilities \( P_A \), the probability of \( X_A = (X_i)_{i \in A} \) where \( A \) is a subset of the BN variables), and conditional probabilities \( P_{A|B} \) (probability of \( X_A \) conditioned on \( X_B \), where \( A \) and \( B \) are two disjoint subsets of the BN variables). It is essentially an optimization calculation problem, as it becomes increasingly heavy with the complexity of the graph relative to the number of variables and their values. The correspondence between the graphical structure and the associated probabilistic structure bring many of the problems with inference to graphs problems. However these problems are relatively complex and give rise to a lot of research: [2], [7], [12]. Therefore, inference in BNs is a complex and a difficult task. [4] showed that this problem is NP-hard, although more research has been undertaken to develop efficient algorithms for such problems ([3], [10]).

There are several other algorithms for exact inference in BNs. They are often based on methods called clustering; for example, we cite the Shenoy-Shafer algorithm ([11]), Hugin ([6]), and the lazy propagation algorithm ([9], [17]) (See also [15], [14], [13], etc). All of these inference algorithms share an exponential complexity relative to the size of the associated junction tree’s largest clique ([8], [16]).

There are two primary methods in literature (and in software) for constructing junction trees. The most commonly used method is by moralization and triangulation, see [5], [6], [8] for details. The other method recursively constructs a proper sequence of cliques that satisfies the running intersection property (see definition in section 3 below), introduced in [1] and [5].

For our part, after presenting the concept of the junction tree (JT), or equivalently, a proper sequence satisfying the running intersection property (RIP), we propose an algorithm that allows
constructing a sequence of subsets of the BN that has the RIP, without restrictive conditions on the set of subsets, as is the case with the junction tree algorithm.

Section 2 provides the details on the above-mentioned inference problem and the motivation behind our work. Section 3 offers complementary information about junctions trees and proper covering sequences that satisfy the running intersection property. Section 4 presents our new technique for constructing a JT on a given family of subsets in the BN.

2. Bayesian Networks and Junction Trees

Consider \( X_I = \{X_i\}_{i \in I} \) a family of discrete random variables. For each \( i \), \( X_i \) has values in the finite domain \( \Omega_i \). For each \( J \subseteq I \), we denote with \( X_J \), the subset \( \{X_i\}_{i \in J} \), which has values in the finite domain \( \Omega_J = \prod_{i \in J} \Omega_i \) and \( P_J \) the joint probability of \( X_J \).

Given \( C \), a collection of subsets that covers \( I \), let us recall that \( C \) and the probability \( P_I \) of the family \( \{X_i\}_{i \in I} \) are compatible if \( P_I \) is written as follows:

\[
\forall \ x_i \in \Omega_i \quad P_I(x_I) = \frac{1}{\alpha} \prod_{C \in \mathcal{C}} f_C(x_C),
\]

where for each \( C \), \( f_C \) is a known function from \( \Omega_C \) in \( \mathbb{R}_+ \) and usually called potential of \( C \); we can write (1) as: \( P_I = \frac{1}{\alpha} \prod_{C \in \mathcal{C}} f_C \), where \( \alpha \) is the normalization constant.

Given a BN on \( I \), \( \mathcal{C} \), in this case, is the set of subsets \( \{i \cup pa(i)\} \), where \( pa(i) \) is the set of parents of \( i \) and \( f_{i \cup pa(i)} \) is the probability of \( X_i \) conditioned on \( X_{pa(i)} = \{X_j\}_{j \in pa(i)} \), \( \alpha \) is known and is 1.

The computation algorithms seek to only sum over subsets \( \Omega_C \) where a smaller \( C \) is desired than \( I \) and well chosen in function of the probability and conditional probability of the subsets of \( X_I \) that we would like to compute.

However, if \( P_I \) is compatible with \( \mathcal{C} \), a fundamental algorithm proposed by Lauritzen [8], the JT algorithm, provides a computing method for all probabilities \( P_C \) of subsets \( X_C(= \{X_i\}_{i \in C}) \) under the condition that \( \mathcal{C} \) is a JT.

Given a set \( \mathcal{K} \) of subsets of \( I \), to compute the probability \( P_K \) for each \( K \in \mathcal{K} \).

We notice that if \( \mathcal{C} \) is a set of subsets of \( I \) such that \( \forall \ K \in \mathcal{K} \ \exists \ C \in \mathcal{C} \text{ s.t. } K \subseteq C \), and we are able to obtain the probabilities \( P_C \), we can deduce each \( P_K \) by choosing a \( C \) such as \( K \subseteq C \) and by eliminating the random variables in \( C - K \); that is, if we decompose \( x_C \) to the pair \( (x_K, x_{C-K}) \), we have

\[
P_K(x_K) = \sum_{x_{C-K} \in \Omega_{C-K}} P_C(x_C).
\]

Given \( P_I \) is compatible with the covering \( \mathcal{J} \), a method of computing the probability of all targets \( P_K \) \( (K \in \mathcal{K}) \) is given as follows:

A. Construct a collection of cliques \( \mathcal{C} \) of \( \mathcal{C} \cup \mathcal{K} \), which is a JT.
B. Use the compatibility of \( P_I \) with \( \mathcal{C} \) to compute \( P_C \).
C. Compute \( P_K \) from \( P_C \).

Step A, which is purely from the graph theory, consists of constructing a JT that is a covering of \( \mathcal{C} \cup \mathcal{K} \). This step can be heavy. We propose in what follows a new method for constructing a proper covering sequence that has the RIP. We think our method will significantly improve the original method and it can be adapted to any computation task.

3. Junction trees and proper covering Sequence that has the RIP

3.1. Junction Tree - Separators

**Definition 1** Given a set \( I \), we call a proper covering sequence of \( I \) any sequence \( (C_1, \ldots, C_k) \)
of subsets of $I$ such that
\[ \bigcup_{i=1}^{k} C_i = I \]
for every pair $(j, j')$ of distinct indices, $C_j - C_{j'} \neq \emptyset$.

In other words, a covering is proper if there is no inclusion among its elements.

Clearly, each non-proper covering can be associated with a proper covering by simply removing the parts that are subsets of others, and keeping just one part in case they are equal parts.

**Definition 2** Given $I$ a finite set, we call a JT on $I$ every tree $(\mathcal{C}, H)$, where $\mathcal{C}$ is a proper covering of $I$ satisfying the following property: for each pair $(C, C')$ of non-adjacent elements of $\mathcal{C}$, each intermediate vertex $C''$ on the path in $H$ joining $C$ and $C'$ satisfies $C \cap C' \subset C''$.

**Remark 1** If $\text{card}(\mathcal{C}) \leq 2$, the property interfering in the definition of junction trees is empty. They are trivially JTs when:
- for $k = 1$, $\mathcal{C} = \{I\}$ (a tree with only one element, so zero arcs),
- for $k = 2$, any pair $\{C, C'\}$ such that $C \cup C' = I$, $C - C' \neq \emptyset$, $C' - C \neq \emptyset$.

**Definition 3** Given $(\mathcal{C}, H)$ a JT on a set $I$, the separators associated with this tree are the intersections $C \cap C'$, where $C$ and $C'$ are two elements of $\mathcal{C}$ that are adjacent for the structure of the tree $H$.

We notice that if the cardinality of $\mathcal{C}$ is $k$, there are $k$ separators (number of arcs in a tree with $k$ vertices).

An equivalent presentation of JTs is given by the following theorem.

**Theorem 1** Consider $\mathcal{C}$ a proper covering of $I$ and $H$ a tree structure on $\mathcal{C}$. $(\mathcal{C}, H)$ is a JT if and only if, for each $i \in I$, the set of subsets of $\mathcal{C}$ to which $i$ belongs, noted as $C_i$, constitutes a subtree of $(\mathcal{C}, H)$

### 3.2. Proper Covering Sequences that have the RIP

**Definition 4** A sequence of subsets of a set $I$, $(C_1, \ldots, C_k)$, is said to have the RIP if
\[ \forall j \in \{2, \ldots, k\} \ \exists \ell \in \{1, \ldots, j - 1\} \text{ s.t. } C_j \cap (\bigcup_{h=1}^{j-1} C_h) \subseteq C_\ell. \]

The RIP can be also written as follows:
\[ \forall j \in \{2, \ldots, k\} \ \exists \ell \in \{1, \ldots, j - 1\} \text{ s.t. } \forall h < j \ C_j \cap C_h \subseteq C_\ell. \]

We notice that this property is always satisfied for $j = 2$, with $\ell = 1$.

We are particularly interested in proper covering sequences that have the RIP. We notice the following two such sequences:
- The sequence has one element $(I)$.
- Any proper covering sequence with 2 elements: $(C_1, C_2)$ such that $C_1 \cup C_2 = I$, $C_1 - C_2 \neq \emptyset$, $C_2 - C_1 \neq \emptyset$.

### 3.3. Link between JTs and RIP

The most important property of junctions trees, for our needs, is given by the following theorem.

**Theorem 2** Any JT on $I$ gives induces to as many proper covering sequences with the RIP as there are topological orderings on this tree. Conversely, any proper covering sequence that satisfies the RIP can be associated with the structure of a JT.

We omit the proof of the theorem due to space limitation.
4. An Alternative Method

4.1. Context

Given a finite covering $F$ of $I$, our goal is to construct step by step a proper covering sequence, $(C_1, \ldots, C_p)$, that has the RIP and such that every $F \in F$ is included in at least one of the $C_j$ $(1 \leq j \leq p)$.

The algorithm requires that there be an order on $I$ denoted $I = \{1, \ldots, n\}$; we consider the smallest element of each $F$ ($F \in F$) for this ordering, and we take the union of subsets $F$ that have same smallest element. Ordering these unions by their smallest elements, we obtain a covering sequence $(Q_1, \ldots, Q_r)$ (where $r \leq n$) such that, if we denote $\alpha_k$ (where $1 \leq k \leq r$) the smallest element of $Q_k$, the sequence $(\alpha_1, \ldots, \alpha_r)$ is strictly increasing.

The imposed condition by our problem—each $F$ is included in one of the $C_j$ is obviously equivalent to the inclusion of each $Q_k$ in one of the $C_j$.

The sequence $(Q_1, \ldots, Q_r)$ is not necessarily proper, which will not change anything to the problem if we clean it first; in other words, removing each subset that is already strictly included in another one. We notice in this regard, by construction, that the inclusion of $Q_j \subseteq Q_{j'}$ cannot hold with $j < j'$, because we will then have $\alpha_j \in Q_j$ but $\alpha_j \notin Q_{j'}$ because all of the elements of $Q_{j'}$ are greater or equal to $\alpha_j$, which is strictly greater than $\alpha_j$.

The cleaning consists of eliminating all $Q_j$ satisfying $\exists j' < j$ s.t. $Q_j \subset Q_{j'}$, but the algorithm we are proposing does not require that the sequence $(Q_1, \ldots, Q_r)$ be proper.

The problem that the algorithm should solve is as follows:

Given a covering sequence $(Q_j)_{1 \leq j \leq r}$ of non-empty subsets of $I = \{1, \ldots, n\}$ such that, if we note $\alpha_j = \inf(Q_j)$, the sequence $(\alpha_1, \ldots, \alpha_r)$ is strictly increasing. We have to construct a proper covering sequence $(C_1, \ldots, C_p)$ that satisfies the RIP such that for each $j$ $(1 \leq j \leq r)$, exists $k$ $(1 \leq k \leq p)$ satisfying $Q_j \subseteq C_k$.

We notice that, necessarily, $\alpha_1 = 1$.

In the case of BNs, where $F$ contains $\{\{i \cup pa(i)\}/i \in I\}$, we achieve the construction of $(Q_1, \ldots, Q_r)$ as follows:

- For $I$, we adopt an anti-hierarchical order relative to the structure of the directed acyclic graph: for each $i$, its parents are superior to it (in other words, given $I$ is $\{1, \ldots, n\}$, the order $(n, n-1, \ldots, 1)$ is hierarchical).
- For each $i$ we define $Q_i$ as the union of $\{i\}$, $pa(i)$ and, if they exist, subsets belonging to $F$, not of the form $\{i \cup pa(i)\}$, where $i$ is the smallest element.
- Optionally, we clean the sequence $(Q_1, \ldots, Q_n)$.

The alternative method will construct a sequence $(C^*_1, \ldots, C^*_r)$ of non-empty subsets of $I$ of a length equal to the given sequence $(Q_1, \ldots, Q_r)$, that has the RIP such that for each $j$, $k$ exists satisfying $Q_j \subseteq C^*_k$. The constructed sequence is not necessarily proper but has no inclusions $C^*_j \subseteq C^*_j$ only if $j' < j$. As a second step, we remove all elements included in others so we obtain the sequence $(C_1, \ldots, C_p)$ that satisfies $\forall j \exists k$ s.t. $Q_j \subseteq C_k$, we notice that making the sequence proper will be completed by respecting and keeping the RIP.

The alternative method constructs successive subsets, noted as $D_1, \ldots, D_r$, such that the reversed sequence $(D_r, \ldots, D_1)$ is the sequence $(C^*_1, \ldots, C^*_r)$ or $(C_1, \ldots, C_p)$ sought. In other words, the constructed sequence satisfies the reversed RIP, which means the following:

$$\forall s \in \{1, \ldots, r-1\} \exists t \in \{s+1, \ldots, r\} s.t. D_s \cap (\cup_{u>s}D_u) \subseteq D_t$$

4.2. Phase 1: Constructing a Non Necessary Proper Sequence

The algorithm has exactly $r$ steps; at each step $j$, it constructs a subset $D_j$, containing $Q_j$ and contained in $\cup_{i<j}Q_i$; consequently, $\cup_{i<j}D_i = \cup_{i<j}Q_i$.

The sequence $(D_1, \ldots, D_r)$ satisfies the reversed RIP but is not necessarily proper; however, by construction, all of its elements are non-empty and distinct, and the inclusions $D_j' \subseteq D_j$
cannot take place if only $j' > j$.

Let us denote, for each $j$, $K_j = \{\alpha_j, \ldots, \alpha_{j+1} - 1\}$ (in this special case where $r = n$ therefore, for each $j \in \{1, \ldots, n\}$, $\alpha_j = j$, on a $K_j = \{j\}$).

Let us recall that the reversed RIP is automatically satisfied for the term before the last one of the sequence. For the sequence $(D_j)_1 \leq j \leq r$ to satisfy the reversed RIP, the algorithm saves in memory, for each $j$ such that $1 \leq j \leq r - 2$, $S_j = D_j \cap (\cup_{\ell>j} Q_{\ell})$. A sufficient condition for a subset $D_k$ (with $k > j$) to satisfy the reversed RIP for $D_j$ (in other words, satisfies $D_j \cap (\cup_{\ell>j} D_{\ell}) \subseteq D_k$) is that $S_j \subseteq D_k$. If $S_j = \emptyset$, which means that $D_j \cap (\cup_{\ell>j} D_{\ell}) = \emptyset$, so the reversed RIP is trivially satisfied for $D_j$. Otherwise, the reversed RIP will be automatically satisfied for $D_j$, including in $D_k$, beside $Q_k$, $S_j$ once $K_k \cap S_j \neq \emptyset$ (i.e., when $\alpha_k \leq \inf(S_j) < \alpha_{k+1}$, with the convention $\alpha_{r+1} = r + 1$), which will not only include in $D_k$ the elements greater or equal to $\alpha_k$ contained in $\cup_{\ell\geq k} Q_{\ell}$.

**Description of the new algorithm**

| Inputs | a covering sequence $(Q_1, \ldots, Q_r)$ of non-empty subsets of $\{1, \ldots, n\}$; $\alpha_1 = \inf(Q_1)$; $\alpha_{r+1} = r + 1$. |
|---------|----------------------------------------------------------------------------------|
| The sequence $(\alpha_1, \ldots, \alpha_r)$ is strictly increasing. |
| Outputs | sequences of subsets of $I$: $(D_1, \ldots, D_r)$ and $(S_1, \ldots, S_{r-2})$; for each $j$ such that $S_j \neq \emptyset$, $s_j = \inf(S_j)$. |
| Initialization | $D_1 = Q_1$; $S_1 = Q_1 - \{1\}$. |
| Current Step $j \geq 2$: |
| $D_j = Q_j \cup [\bigcup_{k<j, \alpha_k \leq \alpha_j} S_k]$ |
| if $D_j = \emptyset, S_j = \emptyset$; if $D_j \neq \emptyset, S_j = D_j \cap (\cup_{\ell>j} Q_{\ell})$. |
| End of the algorithm $j = n$. |

**Example 1** Given the BN in figure 1, we would like to compute $P_{i, \text{pa}(i)}$ (for each $i \in I$).

![Figure 1. Example 1](image)

We define the subsets: $Q_1 = \{1, 2, 3, 5\}$, $Q_2 = \{2, 3\}$, $Q_3 = \{3, 7\}$, $Q_4 = \{4, 5, 6\}$, $Q_5 = \{5, 7\}$, $Q_6 = \{6, 8\}$, $Q_7 = \{7, 9\}$, $Q_8 = \{8, 9\}$, $Q_9 = \{9\}$.

If we do not first clean the sequence, the algorithm will run as shown in table 1.
The obtained sequence is not proper \((D_2 \subset D_1, D_8 \subset D_7, \text{and } D_9 \subset D_8)\).

If we proceed by first making the sequence proper, we will preserve: \(Q_1 = \{1, 2, 3, 5\}, Q_3 = \{3, 7\}, Q_4 = \{4, 5, 6\}, Q_5 = \{5, 7\}, Q_6 = \{6, 8\}, Q_7 = \{7, 9\}, Q_8 = \{8, 9\}\) then the algorithm will run as follows:

\[
\begin{array}{c|c|c}
\text{Step } j & K_j & \text{Table 2. Steps of the algorithm with prior cleaning.} \\
\hline
1 & 1, 2 & D_1 = Q_1 = \{1, 2, 3, 5\} \quad S_1 = \{2, 3, 5\} \\
2 & 3 & D_2 = Q_2 \cup S_1 = \{2, 3, 5\} \quad S_2 = \{3, 5\} \\
3 & 4 & D_3 = Q_3 \cup S_2 = \{3, 5, 7\} \quad S_3 = \{5, 7\} \\
4 & 5 & D_4 = Q_4 = \{4, 5, 6\} \quad S_4 = \{5, 6\} \\
5 & 6 & D_5 = Q_5 \cup S_4 = \{5, 6, 7\} \quad S_5 = \{6, 7\} \\
6 & 7 & D_6 = Q_6 \cup S_5 = \{6, 7, 8\} \quad S_6 = \{7, 8\} \\
7 & 8, 9 & D_7 = Q_7 \cup S_6 = \{7, 8, 9\} \quad S_7 = \{8, 9\} \\
\end{array}
\]

We notice that, in this case as well, the obtained sequence is not proper \((D_7 \subset D_6)\).

\[
\begin{array}{c|c|c|c}
\text{Step } i & D_i & S_i \\
\hline
1 & D_1 = Q_1 = \{1, 2, 3, 5\} & S_1 = \{2, 3, 5\} \\
2 & D_2 = Q_2 \cup S_1 = \{2, 3, 5\} & S_2 = \{3, 5\} \\
3 & D_3 = Q_3 \cup S_2 = \{3, 5, 7\} & S_3 = \{5, 7\} \\
4 & D_4 = Q_4 = \{4, 5, 6\} & S_4 = \{5, 6\} \\
5 & D_5 = Q_5 \cup S_4 = \{5, 6, 7\} & S_5 = \{6, 7\} \\
6 & D_6 = Q_6 \cup S_5 = \{6, 7, 8\} & S_6 = \{7, 8\} \\
7 & D_7 = Q_7 \cup S_6 = \{7, 8, 9\} & S_7 = \{8, 9\} \\
8 & D_8 = Q_8 \cup S_7 = \{8, 9\} & S_8 = \{9\} \\
9 & D_9 = Q_9 \cup S_8 = \{9\} & S_9 = \emptyset \\
\end{array}
\]

4.3. Phase 2: Cleaning the Sequence Obtained in Phase 1

By inverting the \((D_1, \ldots, D_n)\) obtained by the algorithm, we obtain a covering sequence of distance elements \((C_1^*, \ldots, C_n^*)\) that satisfies the RIP, and for each \(i, j\) such that \(1 \leq i < j \leq n\), \(C_i^* \supseteq C_j^*\) is false.

If this sequence is not proper, the possibility of making it, that is removing each \(C_i\) for which it exists \(j > i\), satisfying \(C_i^* \subset C_j^*\) is possible through the following algorithm.
Description of the cleaning phase of the algorithm

| Inputs | a covering sequence \((C_1^*,\ldots,C_n^*)\) that has the RIP and such that \(C_i^* \subset C_j^*\) implies \(i < j\). |
| Current Object | a covering sequence \((C_1,\ldots,C_k)\) satisfying the same properties as the given sequence. |
| Outputs | a proper covering sequence \((C_1,\ldots,C_k)\) at the end of the algorithm. |
| Initialization | \(k = n, (C_1,\ldots,C_k) = (C_1^*,\ldots,C_n^*)\) |
| Current Step | Determine whether \(p = \min\{j : \exists i < j \text{ s.t. } C_i \subset C_j\}\) exists. |
| If \(p\) does not exist, the algorithm ends. |
| If \(p\) exists, consider \(K = \{i < p : C_i \subset C_p\}\) and \(k = \min K\). |
| - If they exist, remove from the sequence the \(C_i\) such that \(i \in K\) and \(i \neq k\) from the sequence. |
| - replace \(C_p\) with \(C_k\). |
| - place \(C_l = C_{l-p}\) for all \(l\) greater than \(p\). |

The algorithm will eventually stop because at each step, the length of the sequence will strictly decrease.

Remark 2 Any other cleaning algorithm on \((C_1^*,\ldots,C_n^*)\) would lead to a set of identical subsets to \(\{C_1,\ldots,C_k\}\), as it is constructed by the above-described algorithm and therefore has the JT structure. The proposed algorithm has the advantage of getting it with an order ensuring the RIP.

Example 2 The sequence \((C_1^*,\ldots,C_9^*)\), found in example 1 was the following: \(C_1^* = \{9\}, C_2^* = \{8,9\}, C_3^* = \{7,8,9\}, C_4^* = \{6,7,8\}, C_5^* = \{5,6,7\}, C_6^* = \{4,5,6\}, C_7^* = \{3,5,7\}, C_8^* = \{2,3,5\}, C_9^* = \{1,2,3,5\}\), we notice that \(C_1^* \subset C_2^* \subset C_3^*\) and \(C_8^* \subset C_9^*\); by application of the above algorithm, we find that the proper sequence satisfying the RIP, \((C_1,\ldots,C_6)\), matches the sequence \((C_1^*,\ldots,C_9^*)\): \(C_1 = \{7,8,9\}, C_2 = \{6,7,8\}, C_3 = \{5,6,7\}, C_4 = \{4,5,6\}, C_5 = \{3,5,7\}, C_6 = \{1,2,3,5\}\). 5. Conclusion

In this paper, we have presented a method for constructing a proper covering sequence of cliques—a junction tree—that satisfies the running intersection property starting from a simple sequence of subsets in a Bayesian network. The proposed method is advantageous in that it provides a proper covering that has the RIP based on the computation task and without restrictions. The obtained set of cliques and separates coincide with the JT’s obtained by the moralization and triangulation process, but it has the advantage of adapting to any computation task by adding links to the BN graph and therefore updates the set of subsets at the start of our algorithm. Future work will compare this method to the moralization and triangulation method on more complex examples of BNs and will also explore the algorithm’s complexity and efficiency.

References

[1] Cecil H and Adnan D 1996 Inference in belief networks: A procedural guide International Journal of Approximate Reasoning, 15 pp 225-263
[2] Cooper F G 1990 The computational complexity of probabilistic inference using Bayesian belief networks Artificial Intelligence 42 pp 393-405
[3] Cooper F G and Herskovits E 1991 Determination of the entropy of a belief network is NP-hard Knowledge Systems Lab.
[4] Cooper F G 1987 Probabilistic inference using belief networks is NP-hard Knowledge Systems Lab. 42 pp 393-405
[5] Cowell R G, Dawid A P, Lauritzen S L and Spiegelhalter D J 1999 Probabilistic Networks and Expert Systems (Berlin: Springer)
[6] Jensen F V 1999 An introduction to Bayesian Networks (London: UCL Press),
[7] Geiger D, Heckerman D, King H and Meek C 2001 Stratified exponential families: Graphical models and model selection The Annals of Statistics 29 pp 505 - 526
[8] Lauritzen S L and Spiegelhalter D J 1988 Local computation with probabilities on graphical structures and their application to expert systems *Proceedings of the Royal Statistical Society* Series B 50 (2)

[9] Madsen A L and Jensen F V 1998 Lazy propagation in junction trees ed Gregory F. Cooper and Serafin Moral *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (Morgan Kaufmann Publishers) pp 362 - 369

[10] Meila M and Heckerman D 2001 *An experimental comparison of several clustering and initialization methods. Machine Learning.* 42 pp 9-29

[11] Shenoy P and Shafer G 1990 Axioms for probability and belief-function propagation ed Shachter, R. et al. *Uncertainty in Artificial Intelligence* 4 (USA: Univ. California Press ) pp 169-198

[12] Smyth P, Heckerman D and Jordan M 1997 Probabilistic independence networks for hidden Markov probability models *J. Neural Computation* 9 pp 227-269

[13] Suermondt H J and Cooper F G 1990 Probabilistic inference in multiply connected brief networks using loop cutsets *Journal of Approximate Reasoning* 4 pp 283 - 306

[14] Suermondt H J and Cooper F G 1991 Initialization for the method of conditioning in Bayesian belief networks *Artificial Intelligence* 50 pp 83-94

[15] Suermondt H J, Cooper F G and Heckerman D 1991 A combination of cutset conditioning with clique-tree propagation in the pathfinder system *Uncertainty in Artificial Intelligence* vol 6 ed P B Bonissone, M Henrion, L N Kanal and J F Lemmer pp 245-253

[16] Uffe K 2000 *Inference in Bayesian networks using nested junction trees. Statistics,* 19 pp 301-312

[17] Xiang Y and Jensen F V 1999 Inference in multiply sectioned Bayesian networks with extended Shafer-Shenoy and lazy propagation *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* 43 pp 680-687