INFERENCE IN BAYESIAN NETWORKS USING NESTED JUNCTION TREES

UFFE KJÆRULFF

Department of Computer Science, Aalborg University
Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø, Denmark

Abstract. The efficiency of inference in both the Hugin and the Shafer-Shenoy architectures can be improved by exploiting the independence relations induced by the incoming messages of a clique. That is, the message to be sent from a clique can be computed via a factorization of the clique potential in the form of a junction tree. In this paper we show that by exploiting such nested junction trees in the computation of messages both space *and* time costs of the conventional propagation methods may be reduced. The paper presents a structured way of exploiting the nested junction trees technique to achieve such reductions. The usefulness of the method is emphasized through a thorough empirical evaluation involving ten large real-world Bayesian networks and both the Hugin and the Shafer-Shenoy inference algorithms.

1. Introduction

Kim and Pearl (1983) first formulated inference in a Bayesian network through message passing, but for singly connected networks only. Later this was extended to multiply connected networks, where the messages are passed in a junction tree (or join tree) corresponding to the network (Lauritzen and Spiegelhalter, 1988; Jensen *et al.*, 1990; Shafer and Shenoy, 1990). More precisely, a posterior probability distribution for a particular variable can be computed by sending messages inward from the leaf nodes of the tree toward the root node containing the variable of interest. If a subsequent outward propagation of messages from the root toward the leaves is performed, all nodes (or cliques) will then contain the correct posterior distributions (at least up to a normalizing constant).

The Hugin and the Shafer-Shenoy propagation methods will be reviewed briefly in the following; for more in-depth presentations, consult above references. We shall assume that all variables of a Bayesian network are discrete.

2. Bayesian Networks and Junction Trees

A Bayesian network can be defined as a pair $(\mathcal{G}, p)$, where $\mathcal{G} = (V, E)$ is an acyclic, directed graph or, more generally, a chain graph (Frydenberg, 1989) with vertex set V and edge set E, and p is a discrete probability function with index set V. To each vertex $v \in V$ corresponds a (discrete) random variable X_v with domain $\mathcal{X}_v$. Similarly, a subset $A \subseteq V$ corresponds to a set of variables X_A with domain $\mathcal{X}_A = \times_{v \in A} \mathcal{X}_v$. Elements of $\mathcal{X}_A$ are denoted $x_A = (x_v)$, $v \in A$.

A probability function $\kappa_v(x_v, x_{\mathrm{pa}(v)})$ is specified for each variable X_v, where $\mathrm{pa}(v)$ denotes the parents of v (i.e., the set of vertices of $\mathcal{G}$ from which there are directed links to v). If $\mathrm{pa}(v)$ is non-empty, κ_v is a conditional probability distribution for X_v given $X_{\mathrm{pa}(v)}$; otherwise, κ_v is a marginal probability distribution for X_v.

The joint probability function $p_V = p$ for X_V is Markov with respect to the acyclic, directed graph $\mathcal{G}$. That is, $\mathcal{G}$ is a map of the independence relations represented by p: each pair of vertices not connected by a directed edge represents an independence relation. Thus, $\mathcal{G}$ is often referred to as the independence graph of p. The probability function p being Markov with respect to $\mathcal{G}$ implies that p factorizes according to $\mathcal{G}$:

$$p = \prod_{v \in V} \kappa_v.$$

For a more detailed account of Markov fields over directed graphs, consult e.g. the work of Lauritzen *et al.* (1990.

Exact inference in Bayesian networks involves the computation of marginal probabilities for subsets $C \subseteq V$, where each C induces a fully connected (i.e., complete) subgraph of an undirected graph derived from $\mathcal{G}$. (Formally, a set $A \subseteq V$ is said to *induce* a subgraph $\mathcal{G}_A = (A, E \cap A \times A)$ of $\mathcal{G} = (V, E)$.) A fixed set $\mathcal{C}$ of such complete subsets can be used no matter which variables have been observed and no matter for which variables we want posterior marginals. This is the idea behind the junction tree approach to inference.

The derived undirected graph is created through the processes of moralization and triangulation. In moralization an undirected edge is added between each pair of disconnected vertices with common children and, when this has been completed, all directed edges are replaced by undirected ones. In the triangulation process we keep adding undirected edges (*fill-ins*) to the moral graph until there are no cycles of length greater than three with-out a chord (i.e., an edge connecting two non-consecutive vertices of the cycle).

The maximal complete subsets of the triangulated graph are called *cliques*. It can easily be proved that an undirected graph is triangulated if and only if the cliques can be arranged in a tree structure such that the intersection between any pair of cliques is contained in each of the cliques on the path between the two. A tree of cliques with this property is referred to as a *junction tree*. Now, inference can be performed by passing messages between neighbouring cliques in the junction tree.

3. Inference in Junction Trees

In the following we will briefly review the junction tree approach to inference. For a more in-depth treatment of the subject consult e.g. the book of Jensen (1996).

In the Hugin architecture, a potential table ϕ_C is associated with each clique, C. The link between any pair of neighbouring cliques, say C and D, shall be denoted a *separator*, and it is labelled by $S = C \cap D$. Further, with each separator, S, is associated one or two potential tables ('mailboxes'). The Hugin algorithm uses one mailbox per separator, referred to as ϕ_S, and the Shafer-Shenoy algorithm uses two mailboxes, referred to as ϕ_S^{in} and ϕ_S^{out}.

In addition, we assign each function κ_v, $v \in V$, to a clique, C, such that $\{v\} \cup \mathrm{pa}(V) \in C$. That is, for each clique, C, is associated a subset of the conditional probabilities specified for the Bayesian network. Let $\mathcal{K}_C$ denote this subset, and define

$$\psi_C = \prod_{\kappa \in \mathcal{K}_C} \kappa.$$

As mentioned above, inference in a junction tree is based on message passing. The scheduling of message passes is controlled by the following rule: a clique, C, is allowed to send a message to a neighbour, D, if C has received messages from all of its neighbours, except possibly from D, and C has not previously sent a message to D. Thus propagation of messages is initiated in leaf cliques and proceeds inwards until a 'root' clique, R, has received messages from all of its neighbours. (Note that any clique may be root.) The root clique is now fully informed in the sense that it has got information (via its neighbours) from all cliques of the junction tree, and thus $p_R \propto \phi_R$, with the proportionality constant being the probability of the evidence (i.e., observations), if any. An outward propagation of messages from R will result in $p_C \propto \phi_C$ for all $C \in \mathcal{C}$, where $\mathcal{C}$ denotes the set of cliques.

A message is passed from clique C to clique D via separator $S = C \cap D$, as follows:

1. C generates the message $\phi^*_S = \sum_{C \setminus S} \phi_C$ and sends it to D.
2. D absorbs the message such that $\sum_{D \setminus S} \phi_D = \phi^*_S$, where ϕ_D is the clique potential for D after the absorption.

Now, assume that we are performing the inward pass, and that clique C is absorbing messages $\phi^*_{S_1}, \ldots, \phi^*_{S_n}$ from neighbouring cliques $C_1, \ldots, C_n$ and sending a message $\phi^*_{S_0}$ to clique C_0. This is done as follows.

1. Absorb messages $\phi^*_{S_1}, \ldots, \phi^*_{S_n}$: $\phi^*_C = \psi_C \prod_{i=1}^{n} \phi^*_{S_i}$
2. Generate message $\phi^*_{S_0}$: $\phi^*_{S_0} = \sum_{C \setminus S} \phi^*_C$
3. Store received messages:
   if algorithm = Hugin then
   $\phi_{S_i} \leftarrow \phi^*_{S_i}$, $i = 1, \ldots, n$
   else
   $\phi^{\text{in}}_{S_i} \leftarrow \phi^*_{S_i}$, $i = 1, \ldots, n$
4. If algorithm = Hugin, then store clique potential: $\phi_C \leftarrow \phi^*_C$
5. Discard $\phi^*_C, \phi^*_{S_1}, \ldots, \phi^*_{S_n}$
6. Send $\phi^*_{S_0}$ to C_0

(Note that if C is the root clique, we skip steps 2 and 6.) Thus, considering the inward pass, the only difference between the Hugin and the Shafer-Shenoy algorithms is that the Hugin algorithm stores the clique potential. Storing the clique potential increases the space cost, but (most often) reduces the time cost of the outward pass, as we shall see shortly.

Next, assume that we are performing the outward pass, and that clique C receives a message, $\phi^*_{S_0}$, from its 'parent clique' C_0 (i.e., the neighbouring clique on the path between C and the root clique), and sends messages $\phi^*_{S_1}, \ldots, \phi^*_{S_n}$ to its remaining neighbours, $C_1, \ldots, C_n$. (The ϕ^*'s should not be confused by the similar potentials of the inward pass.) This is done as follows.

1. Absorb message $\phi^*_{S_0}$:
   if algorithm = Hugin then
   $\phi^*_C = \phi_C \dfrac{\phi^*_{S_0}}{\phi_{S_0}}$
   for $i = 1$ to n do

$$\phi^*_{S_i} = \sum_{C \setminus S_i} \phi^*_C$$

else

for $i = 1$ to n do

$$\phi^*_C = \psi_C \phi^*_{S_0} \prod_{j=1,\ldots,i-1,i+1,\ldots,n} \phi^{\text{in}}_{S_j}$$

$$\phi^*_{S_i} = \sum_{C \setminus S_i} \phi^*_C$$

2. Store $\phi^*_{S_0}$:

if algorithm = Hugin then

$\phi_{S_0} \leftarrow \phi^*_{S_0}$

else

$\phi^{\text{out}}_{S_0} \leftarrow \phi^*_{S_0}$

3. If algorithm = Hugin, then store clique potential: $\phi_C \leftarrow \phi^*_C$
4. Discard ϕ^*_C and $\phi^*_{S_0}$
5. Send $\phi^*_{S_1}, \ldots, \phi^*_{S_n}$

Inference in junction trees may be very expensive due to the generation of clique potentials, ϕ^*_C. Thus, if the generation of these can be relaxed, both space and time savings can be obtained. This is what is exploited in the nested junction trees approach.

4. Nested Junction Trees

In a Bayesian network $(\mathcal{G} = (V, E), p)$ each potential κ_v, $v \in V$, induces a complete subgraph $(\text{fam}(v), \text{fam}(v) \times \text{fam}(v))$ of the moral graph of $\mathcal{G}$, where $\text{fam}(v) = \{v\} \cup \text{pa}(v)$. In general, a set of potentials, $\xi_{U_1}, \ldots, \xi_{U_m}$, with index sets $U_1, \ldots, U_m \subseteq V$ induce an undirected graph, $\mathcal{H} = (U, \bigcup_i U_i \times U_i)$ $(U = \bigcup_i U_i)$, where each ξ_{U_i} induces a complete subgraph $(U_i, U_i \times U_i)$ of $\mathcal{H}$ and the potential

$$\xi_U = \prod_i \xi_{U_i}$$

is Markov with respect to $\mathcal{H}$ (i.e., $\mathcal{H}$ is an independence graph of ξ_U). Thus, if triangulating $\mathcal{H}$ does not result in a complete graph, we can build a junction tree corresponding to the triangulated graph, and then exploit the message passing principle for computing marginals of ξ_U. Assume that $\xi_{U_1}, \ldots, \xi_{U_m}$ are the potentials involved in computing a message (i.e., a marginal), say $\phi^*_{S_0}$, in a clique C. That is, $\xi_{U_1}, \ldots, \xi_{U_m}$ are the incoming messages and the κ_v's associated with C. Then, instead of computing the clique potential

$$\phi^*_C = \prod_i \xi_{U_i}$$

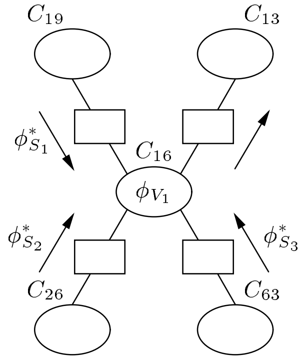
(cf. Step 1 of the above inward algorithm), inducing a complete graph, and marginalizing from ϕ^*_C to compute $\phi^*_{S_0}$ (cf. Step 2 of the above inward algorithm), we might be able to exploit conditional independence relationships between variables of S_0 given the remaining variables of C in computing $\phi^*_{S_0}$ through message passing in a junction tree residing inside C (which, remember, is a clique of another, larger junction tree). This, in essence, is the idea pursued in the present paper. In principle, this nesting of junction trees can extend to arbitrary depth. In practice, however, we seldomly encounter instances with depth greater than three to four.

As a little sidetrack, note the following property of triangulations. Let C be a clique of a junction tree corresponding to a triangulation of a moral graph $\mathcal{G}$, let $\mathcal{K}_C = \{\phi_{V_1}, \ldots, \phi_{V_m}\}$, and let $C_1, \ldots, C_n$ be the cliques connected to C via separators $S_1, \ldots, S_n$. Triangulating the graph induced by $V_1 \cup \cdots \cup V_m \cup S_1 \cup \cdots \cup S_n$ results in a complete graph, unless the triangulation of $\mathcal{G}$ contains a superfluous fill-in: a fill-in between vertices u and v is superfluous if $\{u, v\} \not\subseteq S_i$ for each $i = 1, \ldots, n$, because then C can be split into two smaller neighbouring cliques $C' = C \setminus \{u\}$ and $C'' = C \setminus \{v\}$ with C_i, $i = 1, \ldots, n$, connected to C' if $v \in C_i$, to C'' if $u \in C_i$, or, otherwise, to either C' or C''.

Therefore, assuming the triangulation to be minimal in the sense of not containing any superfluous fill-ins, the nested junction tree principle cannot be applied when a clique, C, receives messages from all of its neighbours, or when a clique potential ϕ_C is involved in the computation (cf. the outward pass of the Hugin algorithm). However, in the inward pass and in the outward pass of the Shafer-Shenoy algorithm, any non-root clique receives messages from all but one, say C_0, of its neighbours, making it possible to exploit the junction tree algorithm for generating the message to C_0.

To illustrate the process of constructing nested junction trees, we shall consider the situation where clique C_{16} is going to send a message to clique C_{13} in the junction tree of a subnet, here called Munin1, of the Munin network (Andreassen *et al.*, 1989). Clique C_{16} and its neighbours are shown in Figure 1. The variables of $C_{16} = \{22, 26, 83, 84, 94, 95, 97, 164, 168\}$ (named corresponding to their node identifiers in the network) have 4, 5, 5, 5, 5, 5, 5, 7, and 6 possible states, respectively.

The undirected graph induced by the potentials $\phi^*_{S_1}$, $\phi^*_{S_2}$, $\phi^*_{S_3}$, and ϕ_{V_1} may be depicted as in Figure 2. At first sight this graph looks quite messy, and it might be hard to believe that its triangulated graph will be anything but complete. However, a closer examination reveals that the graph



$S_1 = \{22, 26, 83, 84, 94, 95, 168\}$
$S_2 = \{83, 84, 97, 164, 168\}$
$S_3 = \{94, 95, 97\}$
$V_1 = \{22, 26, 97\}$

Figure 1. Clique C_{16} receives messages $\phi^*_{S_1}$, $\phi^*_{S_2}$, and $\phi^*_{S_3}$ from cliques C_{19}, C_{26}, and C_{63}, respectively. Based on these messages and the probability potential, $\phi_{V_1} = P(97 \,|\, 22, 26)$, a message must be generated and sent to clique C_{13}.

is already triangulated and that its cliques are $\{83, 84, 97, 164, 168\}$ and $\{22, 26, 83, 84, 94, 95, 97, 168\}$.

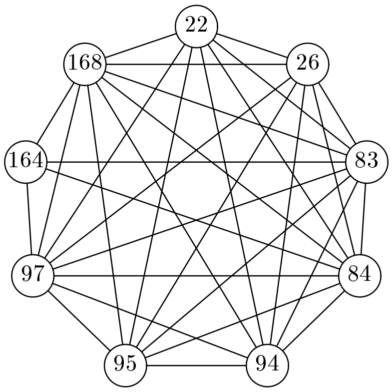


Figure 2. The undirected graph induced by potentials $\phi^*_{S_1}$, $\phi^*_{S_2}$, $\phi^*_{S_3}$, and ϕ_{V_1}.

So, the original 9-clique (i.e., clique containing nine variables) with a table of size $2,625,000$ has been reduced to a junction tree with a 5-clique and an 8-clique with tables of total size $381,000$ (including a separator table of size 750).

Thus encouraged we shall try to continue our clique break-down. In the two-clique junction tree, the 5-clique has associated with it only potential $\phi^*_{S_2}$, so it cannot be further broken down. The 8-clique, on the other hand, has got the remaining three potentials associated with it. These potentials (i.e., $\phi^*_{S_1}$, $\phi^*_{S_3}$, and ϕ_{V_1}) induce the graph shown in Figure 3.

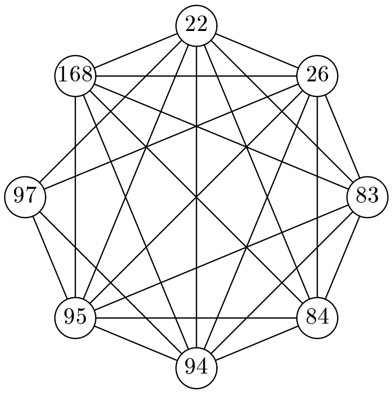


Figure 3. The undirected graph induced by potentials $\phi^*_{S_1}$, $\phi^*_{S_3}$, and ϕ_{V_1}.

This graph also appears to be triangulated and contains the 5-clique $\{22, 26, 94, 95, 97\}$ and the 7-clique $\{22, 26, 83, 84, 94, 95, 168\}$ with tables of total size $78,000$ (including a separator table of size 500). Thus, the space cost is reduced by a factor of $375,000/78,000 \approx 5$.

In this junction tree, the 7-clique cannot be further broken down since it contains only one potential. The 5-clique, however, contains two potentials, $\phi^*_{S_3}$ and ϕ_{V_1}, and can therefore possibly be further broken down. The two potentials induce the graph shown in Figure 4, hence a further break-down is possible as the graph is triangulated and contains two cliques.

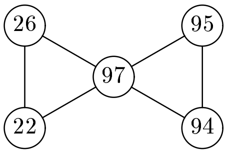


Figure 4. The undirected graph induced by potentials $\phi^*_{S_3}$ and ϕ_{V_1}.

Now, no further break-down is possible. The resulting nested junction tree, shown in Figure 5, has a total space cost of $81,730$, which is significantly less than the original $2,625,000$. Carrying out the nesting to this depth, however, has a big time cost. To see this, let us describe how the computation of $\phi^*_{S_0}$ (the message to C_{13}) proceeds. To compute $\phi^*_{S_0}$ we need to perform an inward propagation of messages in the (C_a, C_b) junction tree. However, since neither C_a nor C_b contain all variables of S_0, we need to perform multiple message passes from C_b to C_a (or vice versa). For example, letting C_a be the root clique of the inward pass, we need to send a message from C_b for each instantiation of the variables in

$(C_b \cap S_0) \setminus C_a = \{22, 26, 94, 95\}$; that is, $4 \times 5 \times 5 \times 5 = 500$ messages must be sent via separator $\{83, 84, 97, 168\}$ in order to generate $\phi^*_{S_0}$. Sending a message from C_b to C_a involves inward propagation of messages in the (C_c, C_d) junction tree, but, again, since neither C_c nor C_d contain all variables of the (C_a, C_b) separator, we need to send multiple message from C_d to C_c (or vice versa). For example, letting C_c be the root clique of the inward pass at this level, we need to send 5 messages from C_d (i.e., one for each instantiation of the variables in $(C_d \cap \{83, 84, 97, 168\}) \setminus C_c = \{97\})$ for each of the 500 messages to be sent from C_b (actually from C_c) to C_a. Similarly, for each message sent from C_d to C_c, 20 messages must be sent from C_e to C_f (or 25 messages from C_f to C_e). Clearly, it becomes a very time consuming job to generate $\phi^*_{S_0}$, exploiting nesting to this extent.

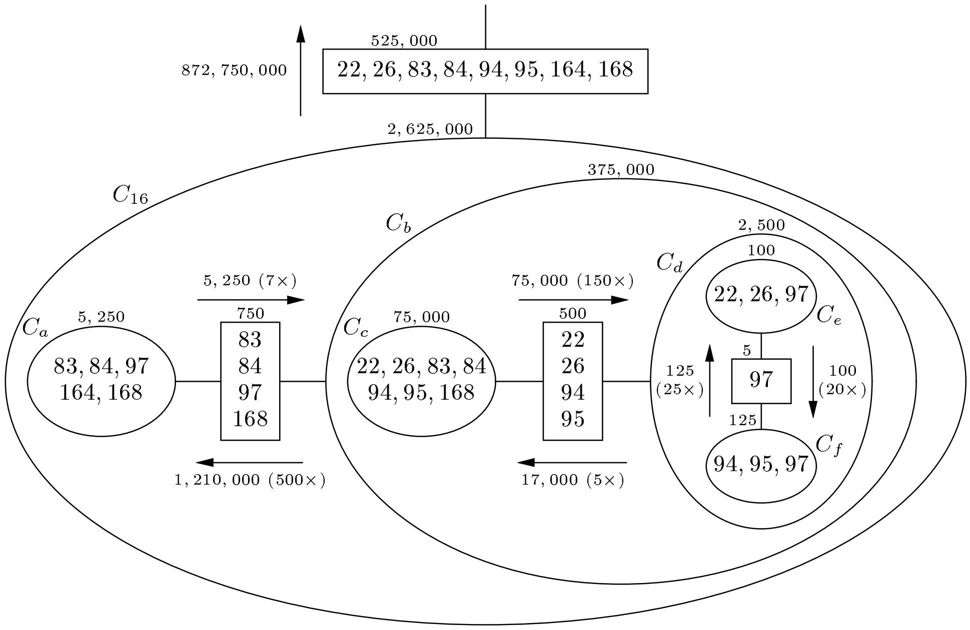


Figure 5. The nested junction tree for clique C_{16} in Munin1. Only the connection to neighbour C_{13} is shown. The small figures on top of the cliques and separators indicate table sizes, assuming no nesting. The labels attached to the arrows indicate (1) the time cost of sending a single message, and (2) the number of messages required to compute the separator marginal one nesting level up.

A proper balance between space and time costs will most often be of interest. We shall address that issue in the next section.

Finally, however, let us briefly analyze the case where C_b is chosen as root instead of C_a (see Figure 6). First, note that, since C_b contains the three potentials $\phi^*_{S_1}$, $\phi^*_{S_3}$ and ϕ_{V_1}, and receives message potentials, $\phi^1_S, \ldots, \phi^7_S$, from C_a via $S = \{83, 84, 97, 168\}$, C_b collapses to a complete subgraph. The time cost of computing the first clique potential, $\phi^*_{C_b} =$

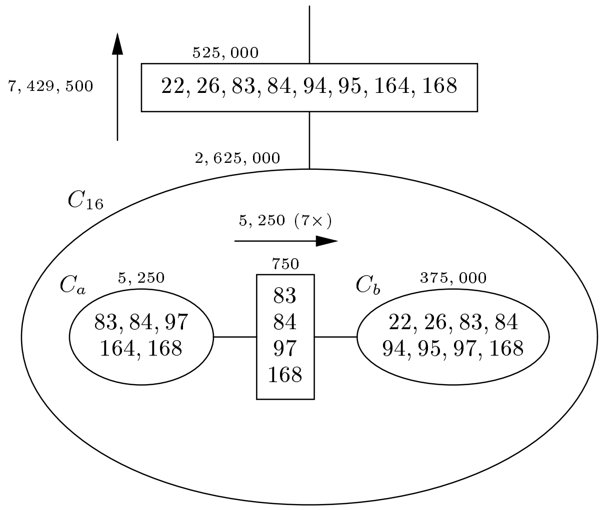


Figure 6. The nested junction tree for clique C_{16} in Munin1, with C_b being the root clique (i.e., the message to be generated by C_{16} is computed via inward message propagation with C_b as root). The labels attached to the arrows indicate (1) the time cost of sending a single message, and (2) the number of messages required to compute the separator marginal one nesting level up.

$\phi^*_{S_1} * \phi^*_{S_3} * \phi_{V_1} * \phi^1_S$, has a time cost of $4 \times 375,000$. Using the Hugin algorithm, the time cost of computing the remaining six clique potentials is $6 \times (750 + 375,000)$. Each marginalization of the clique potentials has a time cost of $525,000$ (this is a conservative measure based on the larger of the $\phi^*_{C_b}$ and the $\phi^*_{S_0}$ tables). Thus, the total time cost of generating $\phi^*_{S_0}$, using C_b as root, is

$$4 \times 375,000 + 6 \times (750 + 375,000) + 7 \times 525,000 = 7,427,500.$$

The space and time costs of conventional (i.e., non-nested) message generation is $2,625,000$ and $5 \times 2,625,000 = 13,125,000$, respectively. Thus, in this case, using the nested junction trees approach provides savings of 85% and 43%, respectively, of space and time costs.

5. Space and Time Costs

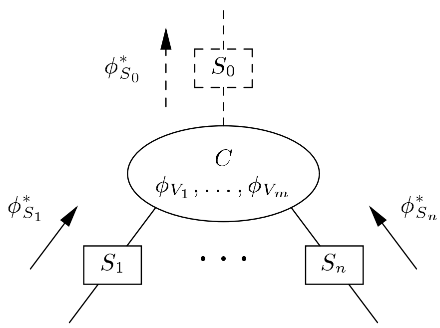
In order to evaluate the applicability of the nested junction tree approach, we must compare its space and time costs with those of the conventional junction tree approaches.

5.1. CONVENTIONAL APPROACH

The space and time costs (denoted c_s and c_t, respectively) of conventional Hugin and Shafer-Shenoy message passing for a single clique are shown in Figures 7 and 8 for, respectively, inward and outward message passing. In Figure 7, clique C receives messages from outward neighbours $C_1, \ldots, C_n$ via separators $S_1, \ldots, S_n$, and, if it is not the root clique, sends a message to the inward neighbour C_0 via S_0. In Figure 8, having received a message from the inward neighbour C_0, unless it is the root clique, clique C sends messages to the outward neighbours.

5.1.1. *Inward*

The cost of message processing in clique C during the inward pass in shown in Figure 7. Note that, using the Shafer-Shenoy algorithm with $|\mathcal{K}_C| > 1$



$\mathrm{ConvCost}^1_{\mathrm{in}}(C)$
  if (algorithm = Shafer-Shenoy) $\wedge$ $(m > 1)$
    $c_t \leftarrow m|\mathcal{X}_{V_1 \cup \cdots \cup V_m}|$ {computing ψ_C}
    $c_s \leftarrow |\mathcal{X}_{V_1 \cup \cdots \cup V_m}|$ {storing ψ_C}
  if (algorithm = Hugin) $\vee$ $(m + n > 1)$
    $c_t \leftarrow c_t + (m+n)|\mathcal{X}_C|$ {computing ϕ^*_C}
  if algorithm = Hugin
    $c_s \leftarrow |\mathcal{X}_C|$ {storing ϕ^*_C in ϕ_C}
  if $S_0 \neq \emptyset$
    $c_t \leftarrow c_t + |\mathcal{X}_C|$ {computing $\phi^*_{S_0}$}
    $c_s \leftarrow c_s + |\mathcal{X}_{S_0}|$ {storing $\phi^*_{S_0}$}

Figure 7. Space and time costs of receiving messages from outward neighbours and sending to the inward neighbour.

(or, equivalently, $m > 1$), computing and storing $\psi_C = \prod_i \phi_{V_i}$ will save time during the outward pass, since ψ_C is then going to be reused. However, since m is typically small (less than three to four), we might want to save the space required, $|\mathcal{X}_{V_1 \cup \cdots \cup V_m}|$, and use a little extra time during the outward pass, recomputing ψ_C. Anyway, we shall assume that ψ_C is stored in the inward pass.

In general, with $m+n$ tables contributing to the clique potential, ϕ_C^*, the time cost of computing ϕ_C^* is $(m+n)|\mathcal{X}_C|$. Note that if information on the domain of the contributing potentials are utilized, this cost may be reduced (Shenoy, 1996). For simplicity, we shall, however, refrain from using such refined methods of combination.

The Hugin algorithm will always compute ϕ_C^* and store it in ϕ_C, whereas the Shafer-Shenoy algorithm will only compute ϕ_C^* when more than one table contribute to it. If C is not the root clique (i.e., $S_0 \neq \emptyset$), the message, $\phi_{S_0}^*$, to clique C_0 is generated.

5.1.2. *Outward*

The cost of message processing in clique C during the outward pass is shown in Figure 8. Note that using the Hugin algorithm, there will be a clique table, ϕ_C, if message processing in C in the inward pass was based on the conventional method; otherwise, ϕ_C $(= \phi_C^*)$ must be generated. Note that we shall always assume an outward pass to be preceded by an inward pass.

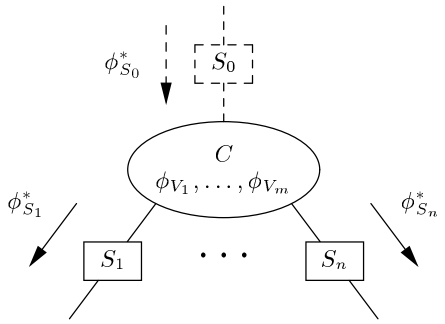
Using the Shafer-Shenoy algorithm with C being a leaf clique, we do not have to do anything, since $\phi_C^* = \phi_{S_0}^*$, unless $\mathcal{K}_C \neq \emptyset$ in which case $\phi_C^* = \psi_C \phi_{S_0}^*$. Note the difference between the two algorithms: the Hugin algorithm computes ϕ_C^* only once, whereas the Shafer-Shenoy algorithm does it n times, unless only one table contributes to ϕ_C^* in which case we marginalize directly from that table. The number of tables contributing to ϕ_C^* when generating the message, $\phi_{S_i}^*$, to C_i equals $n+1$ (i.e., tables $\psi_C, \phi_{S_0}^*, \phi_{S_1}^{\text{in}}, \ldots, \phi_{S_{i-1}}^{\text{in}}, \phi_{S_{i+1}}^{\text{in}}, \ldots, \phi_{S_n}^{\text{in}}$) if $S_0 \neq \emptyset$ and $\mathcal{K}_C \neq \emptyset$, n if either $S_0 = \emptyset$ or $\mathcal{K}_C = \emptyset$ (but not both), and $n-1$ if both $S_0 = \emptyset$ and $\mathcal{K}_C = \emptyset$.

5.2. NESTED APPROACH

In describing the costs associated with the nested junction tree approach, we shall distinguish between message processing at level 1 (i.e., the outermost level) and at deeper levels.

5.2.1. *Level 1*

The processing of messages in a non-root clique C (i.e., receiving messages from neighbours $C_1, \ldots, C_n$ via separators $S_1, \ldots, S_n$, and sending to clique C_0) may involve inference in a junction tree induced by $V_1 \cup \cdots \cup V_m \cup S_1 \cup \cdots \cup S_n$ (see Figure 9). Note that, in this setup, C may either be involved in inward message passing using either algorithm, or it may be involved in outward Shafer-Shenoy message passing. In the inward case, $C_1, \ldots, C_n$ are

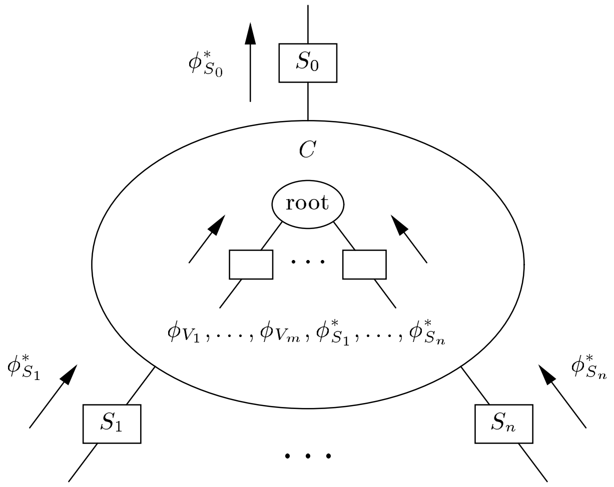


$\mathrm{ConvCost}^1_{\mathrm{out}}(C)$
  if algorithm = Hugin
    if $S_0 \neq \emptyset$
      if ϕ_C exists {conventional inward message processing in C}
        $c_t \leftarrow |\mathcal{X}_{S_0}| + |\mathcal{X}_C|$ {computing $\phi^*_C = \phi_C \dfrac{\phi^*_{S_0}}{\phi_{S_0}}$}
      else
        $c_t \leftarrow (m+n+1)|\mathcal{X}_C|$ {computing $\phi^*_C = \phi^*_{S_0} \prod_{i=1}^n \phi_{S_i} \prod_{j=1}^m \phi_{V_j}$}
    for $i = 1$ to n
      $c_t \leftarrow c_t + |\mathcal{X}_C|$ {computing $\phi^*_{S_i} = \sum_{C \setminus S_i} \phi^*_C$}
  else
    if $(S_0 \neq \emptyset) \wedge (n = 0) \wedge (m > 0)$ {C is a leaf clique with $\mathcal{K}_C \neq \emptyset$}
      $c_t \leftarrow c_t + 2|\mathcal{X}_C|$ {computing $\phi^*_C = \psi_C \phi^*_{S_0}$}
    for $i = 1$ to n
      if $r > 1$ {r is the number of tables contributing to ϕ^*_C}
        $c_t \leftarrow c_t + r|\mathcal{X}_C|$ {computing $\phi^*_C = \psi_C \phi^*_{S_0} \prod_{j=1,\ldots,i-1,i+1,\ldots,n} \phi^{\mathrm{in}}_{S_j}$}
      $c_t \leftarrow c_t + |\mathcal{X}_C|$ {computing $\phi^*_{S_i} = \sum_{C \setminus S_i} \phi^*_C$}
      $c_s \leftarrow c_s + |\mathcal{X}_{S_i}|$ {storing $\phi^*_{S_i}$ in ϕ_{S_i}}

Figure 8. Space and time costs of receiving a message from the inward neighbour and sending to the outward neighbours. Note that $r = n+1$ if $m > 0 \wedge S_0 \neq \emptyset$, $r = n$ if $m > 0 \vee S_0 \neq \emptyset$, and $r = n-1$ if $m = 0 \wedge S_0 = \emptyset$.

the outward neighbours and C_0 the inward. In the outward case, C_0 is the outward neighbour to which C is going to send, and $C_1, \ldots, C_n$ includes the inward neighbour plus the remaining outward neighbours. (Recall that, assuming a minimal triangulation, computing a marginal in a root clique cannot be done through inference in an induced junction tree.)

The message $\phi^*_{S_0}$ is generated through inference in the level 2 junction tree. Thus, the cost of message processing in clique C equals the cost of performing inward propagation towards a root clique in the level 2 junction



NestedCost$^1_{\text{in/out}}(C)$

$c_s \leftarrow c_s^{\text{root}}$	{space cost of inward propagation towards clique 'root'}
$c_t \leftarrow c_t^{\text{root}}$	{time cost of inward propagation towards clique 'root'}
$c_s \leftarrow c_s + \lvert\mathcal{X}_{S_0}\rvert$	{storing $\phi^*_{S_0}$}

Figure 9. Space and time costs of receiving n messages and sending 0 or 1, where the relevant marginal is computed in a junction tree induced by $V_1 \cup \cdots \cup V_m \cup S_1 \cup \cdots \cup S_n$.

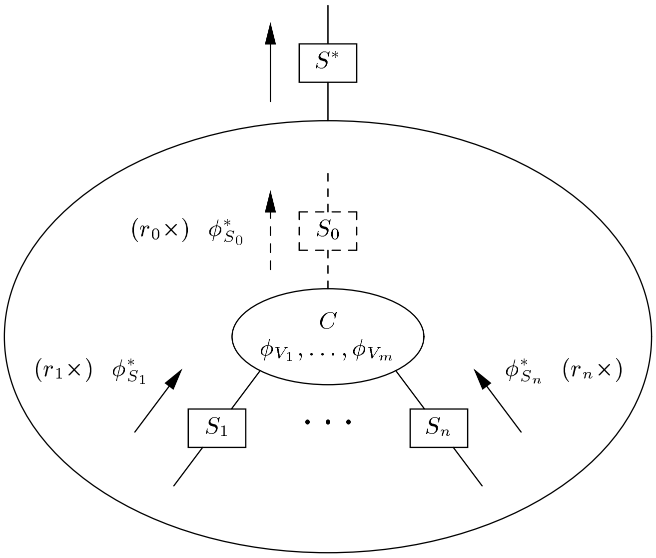
tree (see below) plus the space cost of storing $\phi^*_{S_0}$. In Section 6, we shall describe how the cost of performing inward propagation towards a root clique is calculated.

5.2.2. *Level 2 or deeper — conventional*

We shall now analyze the space and time costs of conventional message passing in junction trees at level 2 or deeper. Since the purpose of performing inference in a junction tree at level $l > 1$ is to generate a message from the clique at level $l - 1$ containing the junction tree, we should only perform inward message passing towards a root clique to compute a marginal or (more typically) a set of marginals.

Consider the situation in Figure 10 where clique C receives messages from its outward neighbours in a junction tree at level $l > 1$ which is contained in a clique, say C^*, at level $l - 1$, wanting to send a message to a neighbour via separator S^*. Note that C^* can be engaged in either inward or outward message processing, and that the potentials involved in that operation are those defining the junction tree containing clique C.

Now, since an S^*-marginal is going to be generated and C might share



$\mathrm{ConvCost}^{>1}_{\mathrm{in}}(C)$
  if algorithm $=$ Hugin
    if $m+n>1$
      $c_t \leftarrow (m+n)|\mathcal{X}_C|$ {computing the first ϕ^*_C}
      $c_s \leftarrow |\mathcal{X}_C|$ {storing ϕ^*_C}
    $c_t \leftarrow c_t + \sum_{i=1}^n r^i(|\mathcal{X}_{S_i}| + |\mathcal{X}_C|)$ {comp. rem. $\sum_{i=1}^n (r_i-1)\prod_{j=1}^{i-1} r_j\ \phi^*_C$'s}
  else
    if $(a>1)\wedge(m>1)$ {a = number of absorptions}
      $c_t \leftarrow m|\mathcal{X}_{V_1\cup\cdots\cup V_m}|$ {computing $\psi_C = \prod_i \phi_{V_i}$}
      $c_s \leftarrow |\mathcal{X}_{V_1\cup\cdots\cup V_m}|$ {storing ψ_C}
      $c_t \leftarrow c_t + (n+1)\prod_i r_i|\mathcal{X}_C|$ {computing $\prod_i r_i\ \phi^*_C$'s}
    else
      if $(a>0)\vee(m>1)$
        $c_t \leftarrow c_t + (m+n)\prod_i r_i|\mathcal{X}_C|$ {computing $\prod_i r_i\ \phi^*_C$'s}
  $c_s \leftarrow c_s + \sum_{i=2}^n (r_i-1)|\mathcal{X}_{S_i}|$ {storing multiple messages for each $i>1$}
  if $S_0 \neq \emptyset$
    $c_s \leftarrow c_s + |\mathcal{X}_{S_0}|$ {storing $\phi^*_{S_0}$'s}
    $c_t \leftarrow c_t + r_0|\mathcal{X}_C|$ {comp. $r_0 = |\mathcal{X}_{(C\cap S^*)\setminus C_0}|\prod_{i=1}^n r_i\ \phi^*_{S_0}$'s}
  else
    $c_t \leftarrow c_t + \prod_i r_i \max\{|\mathcal{X}_{S^*}|, |\mathcal{X}_C|\}$ {computing $\prod_i r_i$ marginals}

Figure 10. Space and time costs of receiving messages from outward neighbours and sending to the inward neighbour in a junction tree at nesting level greater than 1.

variables with S^* which it does not share with clique C_0 to which it is going to send (assuming $S_0 \neq \emptyset$), C will have to send $|\mathcal{X}_{(C\cap S^*)\setminus C_0}|$ messages to C_0. That is, a message must be sent for each configuration $x_{(C\cap S^*)\setminus C_0}$ for C_0 to be able to generate appropriate messages to its inward neighbour, or to generate the S^*-marginal if C_0 is the root clique. Furthermore, if, for the same reason, C's outward neighbours are sending multiple messages to C, the number of messages needed to be sent from C will be multiplied by the product of the number of messages received from the outward neighbours. This way of computing arbitrary marginals in a junction tree is referred to as *variable firing* by Jensen (1996); an alternative called *variable propagation* (Xu, 1994) might be worth considering, but will shall refrain from that in the present paper.

Denote by r_i the number of messages that outward neighbour C_i have to send to C, $i = 1, \ldots, n$. Assuming that the messages are scheduled such that $C_2, \ldots, C_n$ send all of their messages in one batch, and that C_1 synchrononizes its activity with C such that, for each message received from C_1, C processes all combinations of messages from $C_2, \ldots, C_n$, we need extra storage of size $(r_i - 1)|\mathcal{X}_{S_i}|$ for each $i > 1$. Thus, for each combination of messages from the outward neighbours, C either sends $|\mathcal{X}_{(C\cap S^*)\setminus C_0}|$ messages to C_0, or, if $S_0 = \emptyset$, generates the part of the S^*-marginal corresponding to the actual combination of messages. (Note that, typically, $|C| < |S^*|$.)

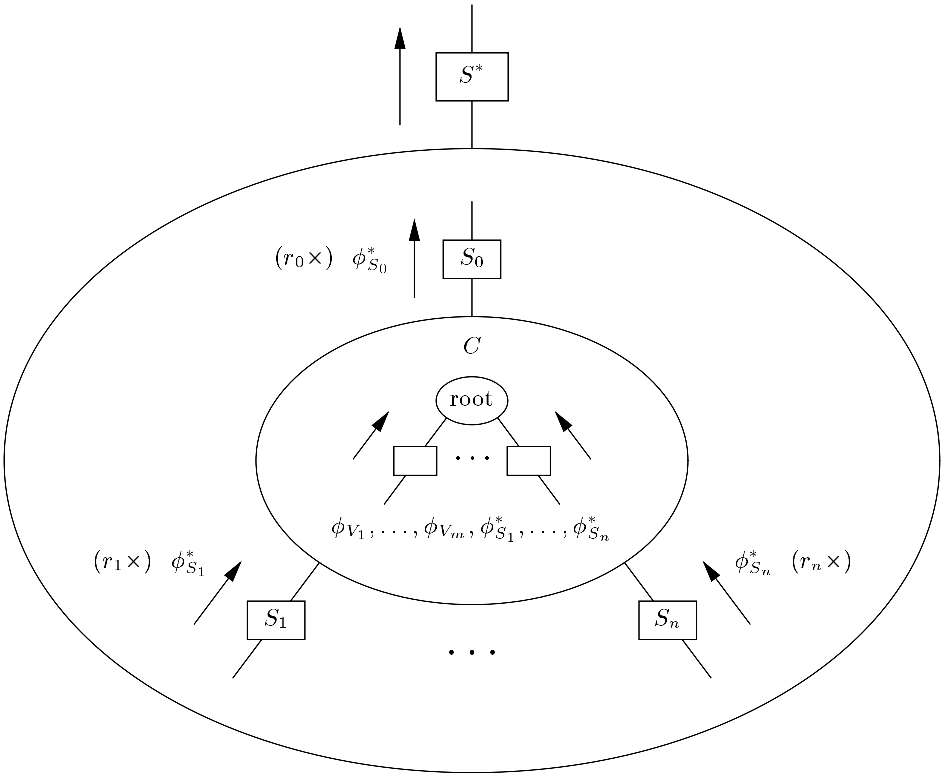
Using the Hugin algorithm, the first step taken by C is to compute (the first) $\phi^*_C = \prod_{i=1}^m \phi_{V_i} \prod_{j=1}^n \phi^*_{S_j}$, which have time cost $(m+n)|\mathcal{X}_C|$. The remaining $\prod_i r_i - 1 = \sum_i r^i$ combinations of messages are processed by replacing one message at a time such that r^i replacements are performed for messages originating from C_i, where $r^i = (r_i - 1)\prod_{j=1}^{i-1} r_j$. Note that replacing a message from C_i involves $|\mathcal{X}_{S_i}|$ divisions and $|\mathcal{X}_C|$ multiplications.

Using the Shafer-Shenoy algorithm, we distinguish two cases: (i) the number of combinations of messages (i.e., the number of absorptions, a) is greater than 1 *and* $m > 1$, and (ii) $a > 0$ *or* $m > 1$. In case (i), it pays (wrt. time) to compute $\psi_C = \prod_i \phi_{V_i}$ before computing the multiple ϕ^*_C's, the time cost of which therefore is $(n+1)\prod_i r_i |\mathcal{X}_C|$. Case (ii) is similar except that ψ_C is not computed, since $\prod_i \phi_{V_i}$ is only going to be computed once.

If C is the root clique, the S^*-marginal is going to be composed. For each of the $\prod_i r_i$ ϕ^*_C's computed we must compute the corresponding S^*-marginal. The time cost of generating an S^*-marginal equals the larger of $|\mathcal{X}_{S_0}|$ and $|\mathcal{X}_C|$.

5.2.3. *Level 2 or deeper — nested*

The processing of messages in a non-root clique C at level $l > 1$, where the message, $\phi^*_{S_0}$, to be sent is generated through inference in an induced junction tree at level $l+1$, is shown in Figure 11. This situation resembles



NestedCost$^{>1}_{\text{in}}(C)$

$c_s \leftarrow c_s^{\text{root}}$	{space cost of inward prop. towards 'root'}
$c_t \leftarrow c_t^{\text{root}}$	{time cost of inward prop. towards 'root'}
$c_s \leftarrow c_s + \lvert\mathcal{X}_{S_0}\rvert$	{storing $\phi^*_{S_0}$}
$c_t \leftarrow c_t \lvert\mathcal{X}_{(C\cap S^*)\setminus C_0}\rvert \prod_i r_i$	{inward prop. $\lvert\mathcal{X}_{(C\cap S^*)\setminus C_0}\rvert \prod_i r_i$ times}
$c_s \leftarrow c_s + \sum_{i=2}^{n}(r_i - 1)\lvert\mathcal{X}_{S_i}\rvert$	{storing multiple messages for each $i > 1$}

Figure 11. Space and time costs of receiving messages from outward neighbours and sending to the inward neighbour in a junction tree at nesting level greater than 1.

the situation shown in Figure 9, the only difference being that C may receive multiple messages from each outward neighbour, and that it may have to send multiple messages to clique C_0. Since C needs to perform $\prod_i r_i$ absorptions, with each absorption corresponding to an inward pass in the junction tree at level $l+1$, and $|\mathcal{X}_{(C\cap S^*)\setminus C_0}|$ marginalizations for each combination of messages, a total of $|\mathcal{X}_{(C\cap S^*)\setminus C_0}|\prod_i r_i$ inward passes must be performed in the level $l+1$ tree.

5.3. SELECTING COST FUNCTION

Now, depending on the level, directionality of propagation, and algorithm used, we should be able to select which of the five cost functions given in Figures 7–11 to use. In addition, we need a function for comparing two pairs of associated space and time costs to select the smaller of the two. To determine which of two costs, say $c = (c_s, c_t)$ and $c' = (c'_s, c'_t)$, is the smaller, we compare the linear combinations $c_s + \gamma c_t$ and $c'_s + \gamma c'_t$, where the time factor γ is chosen according to the importance of time cost. The algorithm Cost(C) for selecting the minimum cost is shown in Figure 12, where '$\prec$' refers to the cost comparison mentioned above.

```
Cost(C)
    if level = 1
        if (direction = inward) ∨ (algorithm = Shafer-Shenoy)
            if ConvCost^1_in(C) ≺ NestedCost^1_in/out(C)
                c = ConvCost^1_in(C)
            else
                c = NestedCost^1_in/out(C)
        else
            c = ConvCost^1_out(C)
    else
        if ConvCost^>1_in(C) ≺ NestedCost^>1_in(C)
            c = ConvCost^>1_in(C)
        else
            c = NestedCost^>1_in(C)
```

Figure 12. Selecting cost functions and the minimum cost associated with message processing in clique C.

5.4. SUM OF COSTS

Under a given time factor, γ, the overall minimum space and time costs of inward or outward propagation of messages towards/from a given root clique, R, can now be computed as

$$c^R = \sum_{C \in \mathcal{C}} \mathrm{Cost}(C) + c_s^{\mathrm{temp}}, \tag{1}$$

where

$$c_s^{\mathrm{temp}} = \begin{cases} \max_{S \in \mathcal{S}} |\mathcal{X}_S| + \max_{\{C \,|\, \phi_C \text{ not stored}\}} |\mathcal{X}_C| & \text{if outward Hugin prop.} \\ \max_{C \in \mathcal{C}} |\mathcal{X}_C| & \text{if Shafer-Shenoy prop.} \\ 0 & \text{otherwise.} \end{cases}$$

During outward Hugin propagation we need auxiliary space when generating messages; thus, a space of size $\max_{S \in \mathcal{S}} |\mathcal{X}_S|$ suffices. Further, for each clique, C, for which ϕ_C was not stored during the inward pass (since a nested junction tree was applied to generate the message), we need temporary storage for the clique potential ϕ_C^*. Thus we add the maximal table size for such cliques.

Using the Shafer-Shenoy algorithm, we need auxiliary space for temporary clique potentials both for the inward and the outward pass. (In computing the sum of the c^R's for inward and outward propagation, c_s^{temp} should only be added once.)

Note that $c_s^{\mathrm{temp}} = \max_{C \in \mathcal{C}} |\mathcal{X}_C|$ may be an approximation when one or more $\mathrm{Cost}(C)$'s originate from $\mathrm{NestedCost}^1_{\mathrm{in/out}}(C)$; thus, $\max_{C \in \mathcal{C}} |\mathcal{X}_C|$ is an upper limit.

6. Propagating Costs

All of the cost functions mentioned above are relative to a given root clique, R, and we are therefore only able to compute the cost of probability propagation (inward or outward) with R as root. However, we want to select the root clique such that the associated cost is minimal, and, therefore, we must be able to compute the 'root cost' c^C for each clique $C \in \mathcal{C}$.

Assuming that root clique R has neighbours $C_1, \ldots, C_n$, Equation 1 can be re-expressed as

$$c^R = \mathrm{Cost}(R) + \sum_{i=1}^{n} c^{C_i \setminus R} + c_s^{\mathrm{temp}}, \tag{2}$$

where $c^{C_i \setminus R}$ denotes the root cost in the subtree rooted at C_i and with the R-branch cut off. Note that, since

$$c^{C_i \setminus R} = \mathrm{Cost}(C_i) + \sum_{C \in \mathrm{neighbours}(C_i) \setminus \{R\}} c^{C \setminus C_i},$$

the root costs can be computed through inward propagation of costs. This is illustrated in Figure 13, where each clique C sends the cost message $\mathrm{Cost}(C) + \sum_{i=1}^{n} c^{C_i \setminus C}$ to its inward neighbour C_0.

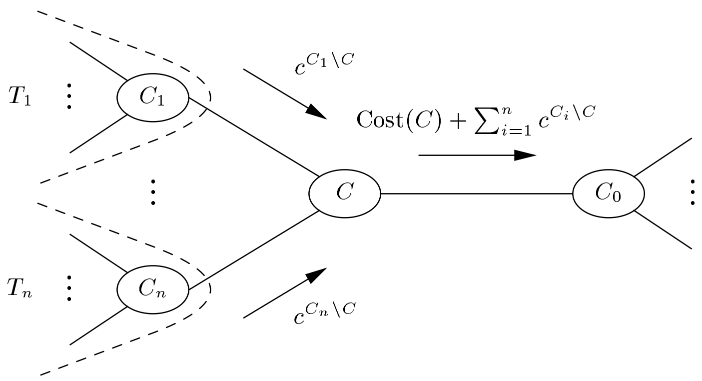


Figure 13. Propagating costs of probability propagation.

Thus, what lacks to compute the root cost for C_i, $i = 1, \ldots, n$, is $c^{R \setminus C_i}$ (i.e., the root cost for the subtree rooted at R and with the C_i-branch cut off). However, this is nothing but the cost message sent from R to C_i if we perform outward propagation of costs from R. That is, after a full propagation of costs (i.e., inward and outward) we can easily compute the root cost of any clique.

Note that Cost() depends on the directionality (i.e., inward or outward). So, to compute the root costs for both inward and outward probability propagation we need to perform two full cost propagations.

7. Experiments

To investigate the practical relevance of nested junction trees, the cost propagation scheme described above has been implemented and run on a variety of large real-world networks.

The following networks were selected. The KK network (50 variables) is an early prototype model for growing barley. The Link network (724 variables) is a version of the LQT pedigree by Professor Brian Suarez extended for linkage analysis (Jensen and Kong, 1996). The Pathfinder network (109 variables) is a tool for diagnosing lymph node diseases (Heckerman *et al.*, 1992). The Pignet network (441 variables) is a small subnet of a pedigree of breeding pigs. The Diabetes network (413 variables) is a time-sliced network for determining optimal insulin dose adjustments (Andreassen *et al.*, 1991). The Munin1–4 networks (189, 1003, 1044, and 1041 variables, respectively) are different subnets of the Munin system (Andreassen *et al.*, 1989).

The Water network (32 variables) is a time-sliced model of the biological processes of a water treatment plant (Jensen *et al.*, 1989).

The average space and time costs of performing a probability propagation is measured for each of these ten networks. Tables 1–4 summarize the results obtained for inward Hugin propagation, inward Shafer-Shenoy propagation, full Hugin propagation (i.e., inward and outward), and full Shafer-Shenoy propagation, respectively. All space/time figures should be read as millions of floating point numbers/arithmetic operations. The first pair of space/time columns lists the costs associated with conventional junction tree propagation. The remaining three pairs of space/time columns show, respectively, the least possible space cost with its associated time cost, the costs corresponding to the highest average relative saving, and the least possible time cost with its associated space cost. The largest average relative savings were found by running the algorithm with various γ-values for each network. The optimal values, γ^*, are shown in the rightmost columns.

TABLE 1. Space and time costs for inward Hugin propagation using (i) the conventional approach, and the nested junction trees approach with (ii) maximum nesting (minimum space cost), (iii) maximum average relative saving of space and time costs, and (iv) minimum time cost.

	Conventional		Nested						
			$\gamma = 0$		$\gamma = \gamma^*$		$\gamma = 100$		
Network	Space	Time	Space	Time	Space	Time	Space	Time	γ^*
KK	15.1	50.3	0.7	1119.1	6.7	40.9	8.3	37.5	0.30
Link	28.1	83.3	0.3	29692.3	11.5	62.5	12.3	61.7	0.30
Pathfinder	0.2	0.7	0.1	1.1	0.2	0.6	0.2	0.6	0.25
Pignet	0.8	2.2	0.04	61.8	0.3	2.0	0.6	1.8	0.25
Diabetes	11.0	33.1	0.5	90.0	0.9	34.6	5.3	29.1	0.15
Munin1	207.7	728.7	19.9	95818.2	68.4	315.9	68.8	315.3	0.30
Munin2	3.2	9.7	0.5	203.7	1.5	9.5	2.4	8.4	0.30
Munin3	3.7	12.1	0.5	85.8	1.5	9.9	2.4	9.4	0.25
Munin4	18.3	64.2	4.9	1128.1	6.3	63.0	12.9	50.0	0.30
Water	8.6	28.3	0.6	3532.7	2.7	26.1	3.9	22.3	0.15

As anticipated, for all networks but Pathfinder, the time costs associated with minimum space costs ($\gamma = 0$) are much larger than the time costs of conventional propagation. Thus, although maximum nesting yields minimum space cost, it is not recommended in general, since the associated time cost may be unacceptably large. The Pathfinder network behaves differently because that nesting is only possible to a very limited degree.

TABLE 2. Space and time costs for inward Shafer-Shenoy propagation using (i) the conventional approach, and the nested junction trees approach with (ii) maximum nesting (minimum space cost), (iii) maximum average relative saving of space and time costs, and (iv) minimum time cost.

	Conventional		Nested						
			$\gamma = 0$		$\gamma = \gamma^*$		$\gamma = 100$		
Network	Space	Time	Space	Time	Space	Time	Space	Time	γ^*
KK	7.9	52.2	6.0	787.9	6.8	42.7	7.0	42.4	0.15
Link	4.5	83.3	2.1	587.4	3.6	76.1	4.0	73.8	0.05
Pathfinder	0.1	0.7	0.1	1.3	0.1	0.7	0.1	0.7	0.10
Pignet	0.3	2.2	0.2	4.0	0.2	2.4	0.3	2.1	0.15
Diabetes	1.2	33.9	0.5	51.0	0.7	36.2	1.2	33.3	0.05
Munin1	97.7	728.8	79.2	31260.7	85.3	379.3	88.0	368.6	0.10
Munin2	0.7	9.9	0.2	50.7	0.4	11.1	0.7	9.1	0.05
Munin3	0.7	12.2	0.2	324.0	0.5	12.0	0.7	10.5	0.05
Munin4	3.0	64.3	1.1	235.7	2.1	52.6	2.4	51.4	0.05
Water	6.1	29.2	5.3	115.8	5.9	27.1	6.1	26.9	0.15

However, as the $\gamma = \gamma^*$ columns show, a moderate increase in the space costs tremendously reduces the time costs. (The example in Figure 5 demonstrates the dramatic effect on the time cost as the degree of nesting is varied.) In fact, for $\gamma = \gamma^*$, the time costs of nested computation are either roughly identical or smaller than those of conventional computation, while space costs are still significantly reduced for most of the networks.

Interestingly, for all networks the minimum time costs ($\gamma = 100$) are less than the time costs of conventional propagation, and, of course, the associated space costs are also less than in the conventional case, since the saving on the time side is due to nesting which inevitably reduces the space cost.

Comparing Tables 3 and 4 with $\gamma = \gamma^*$, we note, somewhat surprisingly, that the time costs of a full Hugin propagation are consistently smaller than those obtained using the Shafer-Shenoy algorithm, while the space costs are either comparable or smaller for the Hugin algorithm. Note, however, that the γ^*'s are significantly smaller in the Shafer-Shenoy case, indicating an attempt to keep the space costs under control.

ACKNOWLEDGEMENTS

I wish to thank Steffen L. Lauritzen for suggesting the cost propagation scheme, Claus S. Jensen for providing the Link and Pignet networks, David

TABLE 3. Space and time costs for a full Hugin propagation using (i) the conventional approach, and the nested junction trees approach with (ii) maximum nesting (minimum space cost), (iii) maximum average relative saving of space and time costs, and (iv) minimum time cost.

	Conventional		Nested						
			$\gamma = 0$		$\gamma = \gamma^*$		$\gamma = 100$		
Network	Space	Time	Space	Time	Space	Time	Space	Time	γ^*
KK	15.8	93.3	1.4	1162.2	2.7	104.1	9.0	80.5	0.20
Link	28.4	164.1	0.5	29773.2	7.2	164.6	12.6	142.6	0.20
Pathfinder	0.2	1.2	0.1	1.6	0.2	1.1	0.2	1.1	0.15
Pignet	0.9	4.5	0.1	64.1	0.4	4.3	0.6	4.1	0.20
Diabetes	11.0	59.5	0.6	116.5	1.0	61.0	5.3	55.5	0.15
Munin1	213.3	1362.3	25.5	96451.8	74.0	949.6	74.4	948.8	0.15
Munin2	3.2	18.7	0.6	212.6	1.4	19.0	2.4	17.3	0.20
Munin3	3.8	23.6	0.5	97.3	1.4	21.6	2.5	20.8	0.15
Munin4	18.4	119.4	5.0	1183.2	5.5	122.1	13.0	105.2	0.20
Water	9.0	46.3	1.0	3550.7	3.2	44.0	4.4	40.3	0.15

TABLE 4. Space and time costs for a full Shafer-Shenoy propagation using (i) the conventional approach, and the nested junction trees approach with (ii) maximum nesting (minimum space cost), (iii) maximum average relative saving of space and time costs, and (iv) minimum time cost.

	Conventional		Nested						
			$\gamma = 0$		$\gamma = \gamma^*$		$\gamma = 100$		
Network	Space	Time	Space	Time	Space	Time	Space	Time	γ^*
KK	9.0	153.6	7.1	889.3	7.9	114.9	8.1	114.4	0.05
Link	6.8	263.1	4.5	767.1	6.0	222.5	6.4	220.8	0.05
Pathfinder	0.1	2.1	0.1	2.6	0.1	2.1	0.2	2.0	0.05
Pignet	0.5	7.0	0.3	8.9	0.4	7.1	0.5	6.8	0.05
Diabetes	1.8	80.9	1.1	97.9	1.3	82.0	1.8	79.0	0.05
Munin1	117.0	2411.9	98.4	32943.8	105.8	1309.1	110.9	1263.4	0.05
Munin2	1.1	31.6	0.6	72.4	0.8	31.2	1.1	29.2	0.05
Munin3	1.2	44.8	0.7	356.6	1.0	44.1	1.2	42.5	0.05
Munin4	4.9	209.9	3.0	381.3	3.1	199.4	5.7	154.9	0.01
Water	6.7	60.6	5.9	147.3	6.5	55.5	6.6	55.3	0.15

Heckerman for providing the Pathfinder network, Kristian G. Olesen for providing the Munin networks, and Steen Andreassen for providing the Diabetes network.

References

Andreassen, S., Hovorka, R., Benn, J., Olesen, K. G. and Carson, E. R. (1991) A model-based approach to insulin adjustment, *in* M. Stefanelli, A. Hasman, M. Fieschi and J. Talmon (eds), *Proceedings of the Third Conference on Artificial Intelligence in Medicine*, Springer-Verlag, pp. 239–248.

Andreassen, S., Jensen, F. V., Andersen, S. K., Falck, B., Kjærulff, U., Woldbye, M., Sørensen, A. R., Rosenfalck, A. and Jensen, F. (1989) MUNIN — an expert EMG assistant, *in* J. E. Desmedt (ed.), *Computer-Aided Electromyography and Expert Systems*, Elsevier Science Publishers B. V. (North-Holland), Amsterdam, Chapter 21.

Frydenberg, M. (1989) The chain graph Markov property, *Scandinavian Journal of Statistics*, **17**, pp. 333–353.

Jensen, F. V. (1996) *An Introduction to Bayesian Networks.* UCL Press, London.

Jensen, F. V., Kjærulff, U., Olesen, K. G. and Pedersen, J. (1989) Et forprojekt til et ekspertsystem for drift af spildevandsrensning (an expert system for control of waste water treatment — a pilot project), *Technical report*, Judex Datasystemer A/S, Aalborg, Denmark. In Danish.

Jensen, C. S. and Kong, A. (1996) Blocking Gibbs sampling for linkage analysis in large pedigrees with many loops, *Research Report R-96-2048*, Department of Computer Science, Aalborg University, Denmark, Fredrik Bajers Vej 7, DK-9220 Aalborg Ø.

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.

Heckerman, D., Horvitz, E. and Nathwani, B. (1992) Toward normative expert systems: Part I. The Pathfinder project, *Methods of Information in Medicine*, **31**, pp. 90–105.

Kim, J. H. and Pearl, J. (1983) A computational model for causal and diagnostic reasoning in inference systems. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 190–193.

Lauritzen, S. L., Dawid, A. P., Larsen, B. N. and Leimer, H.-G. (1990) Independence properties of directed Markov fields, *Networks*, **20**, pp. 491–505.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988) Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, **50**, pp. 157–224.

Shafer, G. and Shenoy, P. P. (1990) Probability propagation, *Annals of Mathematics and Artificial Intelligence*, **2**, pp. 327–352.

Shenoy, P. P. (1996) Binary Join Trees, *in* D. Geiger and P. Shenoy (eds.), *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers, San Francisco, California, pp. 492–499.

Xu, H. (1994) Computing marginals from the marginal representation in Markov trees, in *Proceedings of the Fifth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, Cite Internationale Universitaire, Paris, France, pp. 275–280.