A Combination of Exact Algorithms for Inference on Bayesian Belief Networks

H. Jacques Suermondt and Gregory F. Cooper*

Medical Computer Science Group
Knowledge Systems Laboratory
Stanford University, Stanford, California

ABSTRACT

Cutset conditioning and clique-tree propagation are two popular methods for exact probabilistic inference in Bayesian belief networks. Cutset conditioning is based on decomposition of a subset of network nodes, whereas clique-tree propagation depends on aggregation of nodes. We characterize network structures in which the performances of these methods differ. We describe a means to combine cutset conditioning and clique-tree propagation in an approach called aggregation after decomposition (AD), which can perform inference relatively efficiently for certain network structures in which neither cutset conditioning nor clique-tree propagation performs well. We discuss criteria to determine when AD will perform more efficient belief-network inference than will clique-tree propagation.

KEYWORDS: ***probabilistic reasoning; belief networks; artificial intelligence; Bayesian methods; reasoning under uncertainty; expert systems***

INTRODUCTION

Bayesian belief networks are increasingly popular as a means to encode the uncertainty in expert systems (Anderson et al. [1], Chavez and Cooper [2], Heckerman et al. [3], Horvitz et al. [4]). Belief networks are directed acyclic graphs in which the nodes represent variables and the arcs indicate probabilistic dependencies between these variables.[1] Typically, the variables are discrete-valued; the possible values of each variable are mutually exclusive and

*Section of Medical Informatics, University of Pittsburgh, PA 15261.

Address correspondence to H. J. Suermondt, Medical School Office Building X-215, Stanford, CA 94305-5479.

Received February 1, 1990; accepted April 12, 1991.

[1]We shall use the terms *node* and *variable* interchangeably.

exhaustive. The arcs between nodes are implemented as conditional probability distributions. If there exists an arc from node Y to node X, we call Y a parent of X, and X a child of Y. For each node X with parents $Y_1, \ldots, Y_k$, the belief-network representation includes a conditional probability distribution $P(X \mid Y_1, \ldots, Y_k)$ relating each possible value of node X to each possible value of each of $Y_1, \ldots, Y_k$; for each node X that does not have any parents, a prior probability distribution $P(X)$ is specified for the possible values of node X. Belief networks, also known as *probabilistic influence diagrams*, *causal probabilistic networks*, and *Bayesian nets*, are described in more detail by Horvitz et al. [4], Cooper [5], Howard and Matheson [6], Jensen et al. [7], Lauritzen and Spiegelhalter [8], and Pearl [9, 10].

In this paper, we discuss algorithms for exact probabilistic inference (EPI) in belief networks. EPI consists of the following process. For each node for which the value is known with certainty, we fix the value. We shall refer to such nodes as *evidence nodes*, *instantiated nodes*, or *fixed-value nodes*. Once we have fixed the values of the evidence nodes, we propagate the evidence throughout the belief network and calculate posterior marginal probability distributions for all other nodes in the network. The resulting marginal probability distributions are consistent with the conditional probability tables and prior probabilities specified for the network and with the evidence contained in the fixed-value nodes.

Algorithms that perform EPI compute marginal probability distributions for the variables of interest in an exact Bayesian manner, rather than approximating these probabilities. In addition to the EPI algorithms, there exist several methods for approximate Bayesian inference (see Cooper [5], Chavez and Cooper [11], Chin and Cooper [12], Fung and Chang [13], Henrion [14], Pearl [15], and Shachter and Peot [16]). These approximation algorithms yield marginal probabilities that are not necessarily exact. Approximate Bayesian inference is not the focus of this paper.

Exact probabilistic inference for arbitrary belief networks is known to be NP-hard (Cooper [17]). That is, if we do not constrain the type of belief network, and if we allow any subset of the nodes of the network to be instantiated as evidence, then the worst-case time complexity of computing the marginal probability distributions for all nodes in the network is exponential in the size of the network.[2] Thus, exact probabilistic inference can be computationally expensive. However, there are certain classes of belief networks for which some EPI algorithms have time complexities that are less than exponential in the size of the network. When deciding which EPI algorithm to use in a

[2]Although no formal proof exists that the computational time complexity of all NP-hard problems is exponential, most computer scientists believe that this is the case. For a formal discussion of NP-hardness, see Garey and Johnson [18].

given belief network, we must consider the structure of the belief network. In this paper, we identify network structures for which certain methods perform inference more efficiently than others. We discuss a method that combines two EPI algorithms to improve performance in certain belief-network structures.

PROBABILISTIC INFERENCE ALGORITHMS

There are currently several algorithms that perform exact probabilistic inference (Jensen et al. [7, 19], Lauritzen and Spiegelhalter [8], Pearl [9], Lemmer [20], Rousseau [21], and Schachter [22]). Of these, the method of cutset conditioning (CC) of Pearl and the method for inference by clique-tree propagation (CTP) of Lauritzen and Spiegelhalter are among the most well known. There are certain striking similarities between these two algorithms. Both methods perform inference by local propagation of probabilities. To enable such local propagation, both CC and CTP transform the belief network to a structure that is singly connected; that is, the structure is a polytree. However, the manner in which this polytree is constructed differs between CC and CTP: In CC, the underlying principle is decomposition of nodes; in CTP, the idea is aggregation of multiple nodes into cliques. In the remainder of this section we review these two inference methods, as well as the computational complexity of each.

Decomposition of Nodes: Cutset Conditioning

The method of cutset conditioning is based on an algorithm by Pearl for inference in singly connected belief networks, or *polytrees* (Pearl [9, 10, 23]). A belief network is singly connected if, for any two distinct nodes X and Y, there is at most one undirected path between X and Y. An undirected path between nodes X_1 and X_k is a sequency of nodes $[X_1, \ldots, X_k]$ such that there exists an arc from X_i to X_{i+1} or from X_{i+1} to X_i for each $i \in \{1, \ldots, k-1\}$. In graph-theoretic terms, let G be a graph that is formed by conversion of each arc of a belief network B to an undirected edge; then B is singly connected if and only if G is a forest. If a belief network is multiply connected, it contains at least one loop. A loop is an undirected path $[X_1, \ldots, X_k]$ in which (1) the initial node X_1 coincides with the terminal node X_k and (2), apart from the coincidental initial and terminal nodes, every node in $[X_1, \ldots, X_k]$ is distinct. Thus, a loop is an undirected cycle[3]; a loop in a belief network B corresponds to a cycle in the corresponding undirected graph G.

[3]Directed cycles are not allowed in Bayesian belief network. We follow Pearl [10] in using the term *loop*, rather than *cycle*, to refer to an *undirected* cycle in a directed graph.

Within singly connected belief networks, Pearl's *polytree algorithm* performs exact probabilistic inference by propagating update messages from node to node until the marginal probability distribution of each node is consistent with the evidence. This algorithm is efficient for polytrees: The computational time complexity of evidence propagation is linear in the number of nodes of the network, given certain assumptions about the network structure. We discuss these assumptions in the next subsection, in which we determine the number of operations required for a single evidence propagation using the polytree algorithm. The problem with this original polytree algorithm is that it is not applicable to multiply connected belief networks because update messages may cycle through the loops of such networks. Pearl's solution to this limitation was *cutset conditioning* [10, 24].

Cutset conditioning is based on the idea that instantiated nodes block certain update messages, a principle known as *d-separation* (Pearl [10], Geiger et al. [25]). For the purpose of passing update messages, an instantiated node can be seen as a group of nodes among which messages are blocked. The children of the instantiated node are isolated from one another, while the parents of the instantiated node remain connected, as illustrated in Figure 1. Thus, we can think of the instantiated nodes as nodes that are decomposed into several other nodes. By instantiating certain nodes in the network, we can prevent cycling of the polytree algorithm's update messages. We must instantiate sufficient nodes to break all loops through which update messages can cycle (Suermondt and Cooper [26]). We refer to the set of nodes that are instantiated to break loops as the *loop cutset*. Once the loop-cutset nodes have been instantiated, we can perform probabilistic inference using Pearl's polytree algorithm.

Although we treat the nodes of the loop cutset as though they were instantiated (to prevent cycling of update messages), there is no single value to

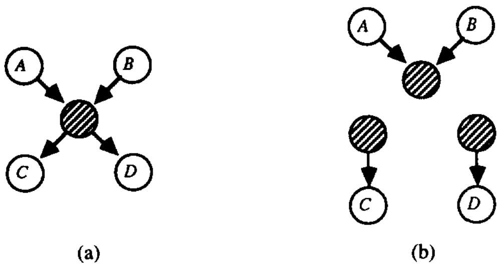

Figure 1. Decomposition of an instantiated node. (a) A single instantiated node, shaded, has two parents, A and B, and two children, C and D. (b) The instantiated node is converted into a set of nodes (shaded) such that the parents A and B remain connected but each child is disconnected from the parents and from the other children of the instantiated node.

which we can instantiate these nodes. Therefore, during inference we must consider all possible combinations of values of the loop-cutset nodes. We call these combinations the *instances* of the loop cutset. The number of loop-cutset instances is equal to the product of the number of possible values of each of the loop-cutset nodes. As we shall see, whether CC provides an efficient vehicle for exact probabilistic inference depends most strongly on the number of loop-cutset instances, which grows exponentially as we add nodes to the loop cutset.

The Computational Complexity of Cutset Conditioning

As mentioned in the previous subsection, the polytree algorithm is linear in the number of nodes, given certain assumptions about the structure of the network. In this section, we shall determine the number of elementary operations (such as arithmetic operations, retrievals, and assignments) required for the propagation of a single piece of evidence throughout the network using the polytree algorithm. We ignore auxiliary operations such as the incrementation of array indexes, and we assume that all elementary operations are of equivalent complexity. We base our derivation on our particular implementation of the equations in Pearl [10].

We introduce the following quantities: V_i, the number of possible values of node i; Λ_i, the number of children of node i; Π_i, the number of parents of node i; $p_j(i)$, the jth parent of node i; Ψ_i, the size of the parent space of node i $(= \Pi_j V_{p_j(i)})$; Ξ_i, the sum of the number of values of the parents of node i $(= \sum_j V_{p_j(i)})$; Γ, the number of loop-cutset instances; and n, the number of nodes.

In our presentation of the complexity of the polytree algorithm, we use specific values for several small constant factors. These values form the part of the derivation that is most likely to vary depending on the specific implementation of Pearl's equations. The specific values of these constants are not central to this analysis; of more interest is the complexity of the total number of operations in terms of various indicators of network size and complexity, such as n and Ψ_i.

For each node, we must perform three steps: (1) combine the evidence coming into the node from its parents and children, to recalculate the marginal probability distribution of the node; (2) update the messages to be sent to the node's parents; and (3) update the messages to be sent to the node's children. Let us consider each in turn.

To update the local measures of belief, we first summarize the "diagnostic" evidence from the node's children, which takes $V_i(2 + 3\Lambda_i)$ operations. Next, we must summarize the "causal" evidence from the node's parents, which takes $V_i(7\Psi_i + \Xi_i + 2)$ operations. Finally, we combine the "causal" and the "diagnostic" evidence to determine the new marginal distribution for node i.

This combination takes $8V_i + 1$ operations. Thus, the total number of operations in this step is

$$V_i(2 + 3\Lambda_i) + V_i(7\Psi_i + \Xi_i + 2) + 8V_i + 1$$

which is equal to

$$V_i(12 + 3\Lambda_i + 7\Psi_i + \Xi_i) + 1 \tag{1}$$

We calculate the messages to be sent to the each parent $p_j(i)$ of node i by combining the evidence from node i's children, the evidence from all i's parents other than $p_j(i)$, and the conditional probability matrix of node i. The number of operations for this rather complex update is

$$\Xi_i + 7\Xi_i V_i + 3\Pi_i \Psi_i V_i \tag{2}$$

Finally, we compute the messages to be sent to each of the children of node i by combining the ''causal'' evidence coming into the node with the evidence from all children except the one to which the message is being sent. Since the ''causal'' evidence has already been computed during step 1, this calculation is efficient: The number of operations is

$$3(\Lambda_i)^2 V_i - \Lambda_i V_i \tag{3}$$

Thus, the total number of operations required to propagate evidence through node i is the sum of expressions (1), (2), and (3), which is equal to

$$V_i(12 + 3\Lambda_i + 7\Psi_i + \Xi_i) + 1 + \Xi_i + 7\Xi_i V_i + 3\Pi_i \Psi_i V_i + 3(\Lambda_i)^2 V_i - \Lambda_i V_i$$

which is equal to

$$V_i[12 + 2\Lambda_i + 3(\Lambda_i)^2 + 8\Xi_i + 7\Psi_i + 3\Pi_i \Psi_i] + \Xi_i + 1 \tag{4}$$

The complexity of expression (4) is dominated by the term $\Pi_i \Psi_i V_i$. Note that $\Psi_i V_i$ is the size of the conditional probability matrix of node i, and Π_i is the number of parents of node i.

If we assume that the network is *bounded* (that is, there is a fixed maximum number of parents and children for each node and the number of possible values of each node is less than some fixed maximum), then each of the terms in expression (4) will be bounded by some fixed constant. In that case, the number of operations required to propagate evidence through any node is bounded, and the complexity of the polytree algorithm is linear in the number of nodes n.

Let us now consider multiply connected networks. We must propagate evidence for *each* loop-cutset instance. As a result, the operation counts in expression (4) must be multiplied by the number of loop-cutset instances Γ.

We see that for bounded networks the complexity of inference using cutset conditioning is proportional to $n\Gamma$. As Γ becomes large, the CC method becomes computationally less tractable.

Aggregation of Nodes: Clique-Tree Propagation

The clique-tree-propagation (CTP) method, developed by Lauritzen and Spiegelhalter [8], provides an alternative mechanism for evidence propagation by local operations. As with cutset conditioning, to use CTP we must transform the original belief network into a tree structure in which probabilities are propagated by local operations. Unlike cutset conditioning, which uses decomposition of nodes to form the tree, the CTP method uses aggregation of nodes into clusters called *cliques* to obtain a tree structure. Within the resulting *clique tree*, we propagate evidence by calculating the joint probability distribution over the nodes in each clique and then making these joint probabilities consistent with those of adjacent cliques in the clique tree (Anderson et al. [1], Jenset et al. [7, 19]). The posterior marginal distribution of each node can be derived directly from the set of clique joint probabilities.

The clique tree is formed by the following process. First, we convert the network into a *moral graph* by adding edges until, for each node i in the network, every parent of i shares an edge with every other parent of node i. Next, the graph is triangulated; that is, edges are added until there exist no loops of four or more nodes without a chord. To triangulate the network for inference using CTP, Lauritzen and Spiegelhalter, in their presentation of CTP [8], recommend a heuristic method called *maximum-cardinality search* (Tarjan and Yannakakis [27]). After triangulation, cliques are formed of all maximal fully connected sets of nodes. A set of nodes $C = \{X_1, \ldots, X_k\}$ is fully connected if and only if, for all distinct i, j in $1, \ldots, k$, X_i shares an edge with X_j. If a set of nodes C is fully connected and there exists no fully connected set of nodes D such that $D \supset C$, then the set C forms a clique. We construct a tree from the resulting cliques by assigning a parent clique to each clique except one, the *top clique* of the tree. The process by which the parent of each clique is chosen is described in detail in Ref. 8.

The Computational Complexity of Clique-Tree Propagation

Let us now determine the number of operations that are required for propagation of evidence using one particular implementation of the CTP method. Our implementation is based on the interpretation of CTP by the HUGIN/MUNIN research group at Aalborg University, Denmark (Andersen et al. [1], Jensen et al. [7, 19]). As in our analysis of the complexity of cutset conditioning, the number of operations is implementation-dependent but gives an upper bound on the complexity of probabilistic inference using this method.

In this section, we make the same simplifying assumptions regarding the determination of complexity as before; that is, we ignore auxiliary operations such as the incrementation of array indexes, and we assume that all elementary operations are of equivalent complexity.

Let us introduce the following additional notation[4]: C, an arbitrary clique; P_C, the parent clique of clique C; R_C, the set of *residual nodes* of C; that is, $\{i: i \in C \setminus P_C\}$; S_C, the set of *separator nodes* of C; that is, $\{i: i \in C \cap P_C\}$; $|S|$, the *cardinality* of a set of nodes S; $|\Omega_S|$, the *state-space size* of a set of nodes S $(= \Pi_{i \in S} V_i)$; and, K, the total state-space of all cliques $(= \sum_C |\Omega_C|)$.

Evidence propagation consists of two sweeps through the clique tree. First, there is a sweep *up* the tree from each clique to its parent clique; this sweep is called *evidence collection*. Evidence collection consists of the *calibration* of the joint probability distribution of each clique's parent to the new information in the (child) clique. We calibrate a clique's probability distribution to the information in another clique by determining the joint probability of the intersection of the two cliques using the information in one and propelling this information into the other. For each clique C, calibration from C, through its separator nodes S_C to its parent clique P_C, takes $3|\Omega_C| + 4|\Omega_{S_C}| + 3|\Omega_{P_C}|$ operations.

After we have completed evidence collection, we normalize the distribution of the top clique of the tree. Normalization of the top clique, T, takes $6|\Omega_T| + 1$ operations.

Subsequently, we perform a sweep through the clique tree in the reverse direction until all evidence has reached all cliques from the top clique. This reverse-direction sweep is called *evidence distribution*. Evidence distribution into each clique C from its parent clique P_C (by calibration) also takes $3|\Omega_C| + 4|\Omega_{S_C}| + 3|\Omega_{P_C}|$ operations.

Thus, we obtain the total number of operations required to update the clique tree by summing the number of operations required for each clique (noting that for the top clique T, $|\Omega_{S_T}| = |\Omega_{P_T}| = 0$). We find that this number of operations is

$$6|\Omega_T| + 1 + 2\sum_{C \neq T} [3|\Omega_C| + 4|\Omega_{S_C}| + 3|\Omega_{P_C}|]$$

which is equal to

$$6K + 1 + \sum_C [8|\Omega_{S_C}| + 6|\Omega_{P_C}|] \tag{5}$$

[4]We follow the notation used by Lauritzen and Spiegelhalter in the original presentation of the method of clique-tree propagation [8].

After evidence distribution, we can determine the posterior marginal probability distribution of any node by selecting any clique containing that node and summing the joint distribution of that clique over all its nodes except the one we are interested in. To determine the number of operations required to compute the marginal distributions for *all* nodes, we use the following two properties of the residual sets R_C:

- For each node i, there is exactly one clique C such that $i \in R_C$;
- $\bigcup_C R_C$ is the set of all nodes.

It is straightforward to derive these properties from the definition and properties of cliques, as described in Ref. 8. Now, we determine the marginal distribution of any node from the *unique* clique in which that node is a member of the residual set. First we compute the joint probability distribution of R_C for each clique C. Next we determine the marginal probability distribution of each node from the joint distribution of the appropriate residual set. The number of operations required to determine the joint distribution of the residual set of clique C is $3|\Omega_C| + |\Omega_{R_C}|$. We must marginalize this set for each member of R_C, which takes an additional number of operations that is equal to $\sum_{i \in R_C}[V_i + 3|\Omega_{R_C}|]$. Thus, the total number of operations for node marginalization is equal to

$$\sum_C \left\{ 3|\Omega_C| + |\Omega_{R_C}| + \sum_{i \in R_C} [V_i + 3|\Omega_{R_C}|] \right\}$$

which is equal to

$$3K + \sum_C [(3|R_C| + 1) \times |\Omega_{R_C}|] + \sum_i V_i \tag{6}$$

and the total number of operations necessary for a single complete evidence propagation through the entire network using CTP is the sum of expressions (5) and (6), which is equal to

$$9K + 1 + \sum_i V_I + \sum_C [8|\Omega_{S_C}| + 6|\Omega_{P_C}| + (3|R_C| + 1)|\Omega_{R_C}|] \tag{7}$$

Expression (7) is dominated by the total state-space size, K; additional influential terms, depending on the structure of the clique tree, may be $\sum_C |\Omega_{P_C}|$ and $\sum_C [|R_C| \, |\Omega_{R_C}|]$. We see that the complexity of the algorithm is determined strongly by the state-space sizes of the individual cliques. For any clique C, $|\Omega_C|$ is exponential in the number of nodes in the clique. If, for a particular belief network, some cliques are very large, then inference using CTP may be computationally intractable.

SELECTION OF AN INFERENCE ALGORITHM

As mentioned in the Introduction, exact probabilistic inference on belief networks is NP-hard. Therefore, it is not surprising that in the worst case the computational time complexities of both CTP and CC are exponential in the size of the network. However, we can identify particular features in the structure of the network that favor one of these methods over the other. Certain structures may lead to triangulations in which there are impractically large cliques, and other structures may force us to generate enormous loop cutsets. Of particular interest are those network structures in which one method is efficient but the other is impractical. For example, there are structures where the loops of the network can be cut with very few loop-cutset nodes but where an efficient triangulation is difficult to find; conversely, there are structures where we can identify a clique tree consisting of very small cliques but many where nodes are required to cut all the loops in the network.

For the CC method, a loop cutset that has a large number of nodes is particularly problematic. Before we can perform inference using Pearl's polytree algorithm in a network, we need to cut all loops. We discuss the conditions to be met by the members of the loop cutset in Ref. 26. Because all loops must be cut, if the network has many loops that do not share nodes that can be used to cut multiple loops, often the CC method will be intractable. In the following subsection, we show an example of such a structure.

The CTP method is not affected strongly by small loops; for example, a loop consisting of three nodes can be converted into a single three-node clique. In practice, cliques of three nodes allow rapid inference. On the other hand, when the clique tree contains large cliques, CTP becomes intractable. Large cliques may arise in networks in which there exist a large number of nodes that all share a common ancestor and a common descendant. Later we show an example of such a structure.

Although it is possible to make generalizations about typical structures in which one method is favored over the other, in practice it is often helpful to construct a loop cutset for a network to determine whether CC may be tractable for the particular network of interest. Similarly, the creation of a clique tree for the network allows us to evaluate whether CTP will be efficient. Fortunately, polynomial-time heuristic algorithms exist both for locating a loop cutset for a belief network (Suermondt and Cooper [26], Stillman [28]) and for determining a clique tree (Lauritzen and Spiegelhalter [8], Tarjan and Yannakakis [27]).

Small Loops in Series

When we have many small loops that cannot be cut with common nodes, CTP provides more efficient inference than CC. An example of such a structure is a *diamond ladder*, as shown in Figure 2.

When we triangulate such a diamond ladder, we find that we have to add only one arc for each loop. After triangulation, the nodes of the loops can be aggregated easily into fairly small cliques, giving us a clique tree in which inference will be efficient. In particular, in the belief network shown in Figure

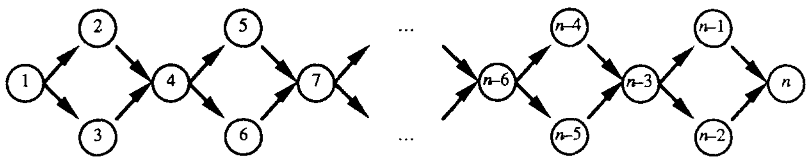

Figure 2. A diamond-ladder belief-network structure. For networks of the diamond-ladder structure, the CTP method provides more efficient inference than does the CC method. Each loop, consisting of four nodes, can be converted into two cliques of three nodes each.

2, there are n nodes and $(n - 1)/3$ loops (for $n \geq 4$); the clique tree for this network will consist of $2(n - 1)/3$ cliques, each consisting of three nodes, as shown in Figure 3. If we assume that all nodes are binary, the total state-space size K is equal to $16(n - 1)/3$. Finally, for half the cliques, the intersection with their parent clique will consist of one node. For the remaining cliques, the intersection with their parent clique will consist of two nodes.

We can determine the number of operations needed for CTP in a diamond ladder by applying expression (7). We separate the sum over all cliques into three parts: (a) those cliques that have one node in common with their parent, (b) those that have two nodes in common with their parent, and (c) the top clique. The number of operations required for inference on the diamond-ladder structure using CTP is

$$\begin{aligned} &9 \times \left(16\frac{n-1}{3}\right) + 1 + 2n \\ &\quad + \left(\frac{n-1}{3} - 1\right)(8 \times 2 + 6 \times 8 + 7 \times 4) && (a) \\ &\quad + \frac{n-1}{3}(8 \times 4 + 6 \times 8 + 4 \times 2) && (b) \\ &\quad + (0 + 0 + 10 \times 8) && (c) \end{aligned}$$

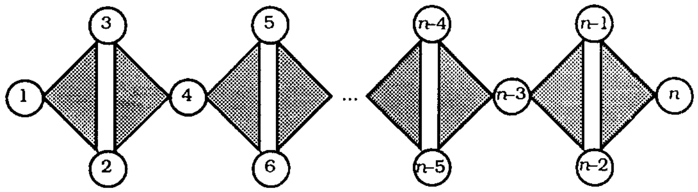

Figure 3. Clique tree for the diamond-ladder belief network. Each clique is represented by a shaded triangle between the nodes that are members of that clique.

which is equal to $110n - 119$ operations. Thus, the number of operations for this type of structure is linear in the number of nodes in the network.[5]

Decomposition (CC) is less suitable for networks fitting this topology. Since the loops are in series, we need to add one node from each of the loops to the loop cutset (Suermondt and Cooper [26]). Thus, the number of loop-cutset nodes in the network of Figure 2 is equal to $(n - 1)/3$, and if we assume again that each node in binary, the number of loop-cutset instances Γ is equal to $2^{(n-1)/3}$. We showed earlier that the number of operations required for CC is proportional to Γ. Thus, even though the diamond-ladder network is bounded, the complexity of CC in such a network is exponential, whereas the complexity of CTP is linear, in the number of nodes.

Large Parallel Loops

When the loops are in parallel, rather than in series, we find that the triangulation for the CTP method is not as straightforward: The resulting clique structure may not lend itself to fast inference. For such networks, the CC method may prove more efficient. Consider, for example, the network in Figure 4. This network consists of s strands of nodes connected by a common ancestor, node 1. Each strand is paired with an adjacent strand by a common descendant to form a loop. These loops are paired with adjacent loops by common descendants, until we reach a single common descendant of all network nodes, node n. Thus, we have a structure of parallel loops. If we use the CC method, the loop cutset for this network need consist on only node 1. If we assume that all nodes are binary, the network will be bounded after decomposition of node 1 because no node will have more than two parents or children. Thus, the time complexity of inference in this network is linear in the number of nodes when we use CC.

In the network of Figure 4, CTP may perform inference less efficiently than CC does. When we apply maximum-cardinality search to triangulate this network, starting with the node labeled 1 in Figure 4,[6] we find that the clique tree contains a number of cliques of $s + 1$ nodes. If we assume that each node in binary, such cliques will have a state-space size of 2^{s+1}, so the total state-space size K will be at least exponential in the number of strands s of the network. Therefore, the worst-case time complexity of inference in the paral-

[5]This analysis is based on the assumption that $n \geq 4$. Note that in the diamond-ladder structure, $(n - 1)/3$ must be an integer because the first diamond in the ladder consists of four nodes, and each additional diamond adds three nodes.

[6]Starting maximum-cardinality search with different nodes can lead to different (and possibly more efficient) triangulations. However, no general efficient algorithm is known that generates a triangulation that minimizes K.

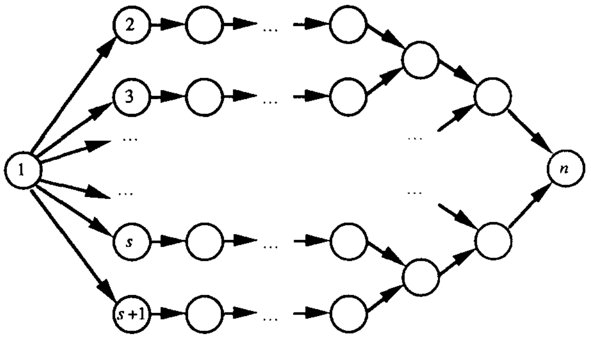

Figure 4. Network structure in which CC generally provides more efficient inference than does CTP.

lel-loop structure using CTP will be exponential in the number of strands s of the network.

COMBINED CUTSET CONDITIONING AND CLIQUE-TREE PROPAGATION

The examples in the previous section illustrate that certain belief-network structures may favor one method of inference over another. However, there are many structures in which neither method dominates. For example, in singly connected networks, both CC and CTP perform inference in time proportional to the number of nodes in the network. Similarly, in networks in which there are small loops in series as well as large loops in parallel, the time complexity of inference using either CC or CTP may be exponential in the number of nodes in the network, depending on the loop cutset or the triangulation that is used. For such problems, an intuitive approach is to use the ideas of cutset conditioning to cut the large parallel loops while using CTP for inference to prevent an exponential number of loop-cutset instances due to the loops in series. In the remainder of this paper, we discuss such a combined approach.

A primary drawback of the CC method is that to use the polytree algorithm for inference we must cut *all* loops in the belief network. In other words, we drive decomposition to an extreme: In every loop in the network, a loop-cutset node must be decomposed (Suermondt and Cooper [26]). If there are many loops, this requirement may lead to a number of loop-cutset instances Γ that is so large that inference becomes intractable. The solution to this problem is to use the CTP method rather than the polytree algorithm for inference in the *partially* decomposed network. After decomposing certain loop-cutset nodes, we aggregate the nodes of the revised network into cliques and form a clique tree, which we use for inference. Since CTP does not require that the network be singly connected, we do not need to cut all loops in the network; rather, we cut only those loops that would force us to create an impractically complex clique tree.

We call this approach *aggregation after decomposition* (AD). As we did in cutset conditioning, we decompose certain loop-cutset nodes (as shown in Figure 1) to disconnect the children of each loop-cutset node from one another and from the loop-cutset node's parents. One objective of this decomposition is to cut large parallel loops, because such loops may lead to large cliques when we aggregate nodes. There is no need to cut small loops that are in series, such as those in the diamond-ladder structure described earlier, because such loops do not lead to large cliques. In summary, we use the principle of decomposition to simplify the structure of the network; after the network has been simplified, we use aggregation to create a tree structure in which we perform exact probabilistic inference.

Inference Using Aggregation After Decomposition

During inference using the AD method, we must consider separately all possible combinations of values, or *instances*, of the loop-cutset nodes. In this section, we discuss how marginal probability distributions can be derived for each node in the belief network, after observation of some new evidence. The process consists of (1) propagating the new evidence using the CTP algorithm for each possible loop-cutset instance; (2) updating the mixing weight of each instance; and (3) combining the results of the various instances. We discussed evidence propagation using CTP in the section on clique-tree propagation. Therefore, we now focus on steps 2 and 3.

For each loop-cutset instance, we maintain a *mixing weight* that is equal to the joint probability of the values to which the loop-cutset nodes have been instantiated in that instance. The mixing weight is used to combine the inference results of the various instances. Let us consider an arbitrary instance of the loop-cutset nodes $X_1, \ldots, X_k$ in which these nodes have been instantiated to values $x_1, \ldots, x_k$, respectively. Initially, when no evidence has been observed for the network, the weight of each instance is equal to the joint prior probability of the loop-cutset nodes' instantiated values, $P(x_1, \ldots, x_k)$. Elsewhere [29], we describe how these joint prior probabilities can be calculated for the CC method; the process is analogous for the AD method.

After new evidence is observed, we can update the weight of each instance to include the evidence in a manner analogous to the CC weight update, as described in detail by Pearl [10, 23] and Suermondt and Cooper [29]. We summarize this process briefly here. Let us denote the set of newly available evidence by E. The new weight, $P(x_1, \ldots, x_k \mid E)$, can be obtained from the prior weight, $P(x_1, \ldots, x_k)$, as follows:

$$P(x_1, \ldots, x_k \mid E) = \alpha P(E \mid x_1, \ldots, x_k) P(x_1, \ldots, x_k)$$

where α is a constant that we obtain by normalizing over all possible loop-cutset instances:

$$\alpha = \frac{1}{P(E)} = \frac{1}{\sum_{x_1, \ldots, x_k} [P(E \mid x_1, \ldots, x_k) P(x_1, \ldots, x_k)]}$$

Thus, to calculate the new weights, we need each instance's joint probability of the evidence in E, $P(E \mid x_1, \ldots, x_k)$. Fortunately, CTP provides us with this joint probability in a straightforward manner. During inference for a particular loop-cutset instance, we collect evidence from each clique of the network into a single top clique (for connected networks) or into a set of independent top cliques (for networks that consist of disconnected portions), as described earlier. In the case where we have a connected network, when we normalize the marginal probabilities of the top clique, the resulting normalization constant is equal to the joint probability of all new evidence. In the case of a network that consists of disconnected portions, the normalization constant of the top clique of each independent portion is equal to the joint probability of all new evidence in that portion; we simply multiply these normalization constants to obtain the joint probability of all new evidence.

After updating the mixing weights, we can determine the posterior marginal probability distribution of each node in the network. For an arbitrary value x of a node X, we obtain $P(x \mid E)$ by calculating the sum over all instances of the product of the instance weight and the probability distribution of the node in that instance:

$$P(x \mid E) = \sum_{x_1, \ldots, x_k} P(x \mid E, x_1, \ldots, x_k) P(x_1, \ldots, x_k \mid E)$$

where $P(x \mid E, x_1, \ldots, x_k)$ is the posterior probability of value x in a particular network instance and $P(x_1, \ldots, x_k \mid E)$ is the updated mixing weight of that instance. This combination of the results of various loop-cutset instances is fully analogous to that performed by Pearl's CC method, which is described in detail in Refs. 10, 23, and 29.

The Computational Complexity of Aggregation After Decomposition

Let us determine the number of operations required for the AD method. The number of operations needed to update a single network instance is given by expression (7). Since we update *all* network instances—each instance corresponding to one of the Γ possible combinations of instantiated values of the loop-cutset nodes—we must multiply expression (7) by a factor Γ. In addition, we calculate the new mixing weights, which takes 8Γ operations. Finally, we compute the posterior marginal probabilities for each node by combining the mixing weights with each instance's inference results. This combination takes $\sum_i V_i + \Gamma + 5\Gamma \sum_i V_i$ operations. Thus, we see that the total number of operations required for AD is equal to

$$\sum_i V_i + \Gamma \Big\{ 9K + 10 + 6 \sum_i V_i + \sum_C \left[8 \,|\Omega_{S_C}| + 6 \,|\Omega_{P_C}| + (3 \,|R_C| + 1) \,|\Omega_{R_C}| \right] \Big\} \tag{8}$$

As with CTP, the total state-space size K plays a dominant role in the complexity of the inference method. In addition, we see that, as with CC, the number of loop-cutset instances Γ is a key factor in the time complexity of the AD method. Let

$$\Phi = 9K + 10 + 6 \sum_i V_i + \sum_C \left[8 \,|\Omega_{S_C}| + 6 \,|\Omega_{P_C}| + (3 \,|R_C| + 1) \,|\Omega_{R_C}| \right]$$

Now we see that the total number of operations required for inference is equal to $\sum_i V_i + \Phi\Gamma$. In this expression, $\sum_i V_i$ is independent of the choice of clique tree or loop cutset; Φ is determined by the structural complexity of the clique tree, and Γ by the number of instances of the loop-cutset nodes.

The AD method does not place any requirements on the nodes of the loop cutset; any subset of the nodes in the network could form a valid loop cutset. For practical purposes, however, we want to add a node to the loop cutset only if addition of that node promises to make inference more efficient. In particular, addition of each new node to the loop cutset leads to an increase in Γ by a factor that is equal to the number of possible values of that node. Thus, addition of a new node to the loop cutset is advisable only if the factor of reduction in Φ due to decomposition of that node is at least as great as the number of possible values of that node.

Of particular interest as loop-cutset candidates are nodes whose decomposition can lead to disconnection of the belief network into multiple distinct portions. Evidence can be propagated in each portion independently, and evidence does not travel from one portion to another. Thus, during inference, we need to update only those network portions in which there is new evidence. This can lead to substantial savings in inference time. In Ref. 30, we and

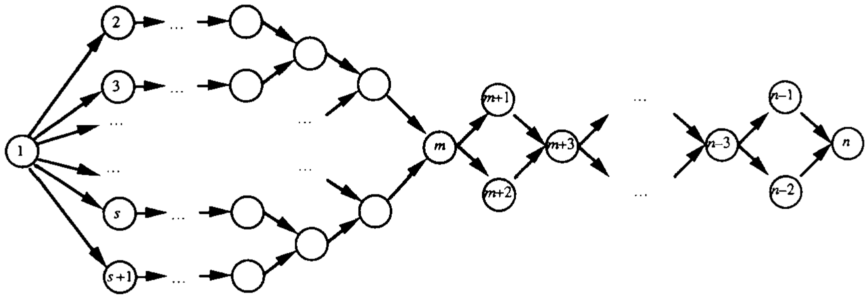

Figure 5. A network that combines the features of the networks in Figures 2 and 4. The subnetwork containing nodes 1–m can be problematic for CTP, and the subnetwork with nodes m–n is problematic for CC.

Heckerman describe our experience with a network that is disconnected due to the instantiation of a single, centrally located node.

Example of Aggregation After Decomposition

The network shown in Figure 5 has features that render it amenable to an inference algorithm that combines the CC and CTP methods. Although in practice it is unlikely that we would encounter a network corresponding exactly to the one shown in Figure 5, this network is prototypical of the networks that may favor combining the CC and CTP methods. In the network shown, nodes 1–m form large parallel loops, which, depending on the triangulation method, would lead to a clique tree that has a value of K that is exponential in the number of strands s. Nodes m–n form small loops in series that, if CC were used, would lead to an exponential number of loop-cutset instances.

In particular, if we were to use CC on this network, we would have to form a loop cutset of node 1 and nodes m, $m + 3, \ldots, n - 3$, leading to a value of Γ that would be exponential in the number of nodes in the network (assuming binary nodes, $\Gamma = 2 \times 2^{(n-m)/3}$ for $m > 1$, $n > m + 3$). If we wish to use CTP on the network of Figure 5, we find that, given the triangulation discussed earlier, several cliques between nodes 1 and m contain $s + 1$ nodes. Therefore, the total state-space size K would be exponential in the number of strands between nodes 1 and m. Thus, under these conditions the worst-case time complexity of inference using either CC or CTP in the example of Figure 5 is exponential in the size of the network.

By decomposing node 1, we cut all parallel loops between nodes 1 and m. Thus, by creating a loop cutset consisting of only node 1, we transform the network of Figure 5 into that of Figure 6, in which the only loops are those between nodes m and n. Now, if we triangulate the resulting network, we find that the clique tree has changed. The portion of the network consisting of nodes 1–m is singly connected; in this portion, we now have $s - 1$ cliques of three nodes (one for each node with two parents), and $m - s$ cliques of two nodes (one for each node with a single parent). The portion between nodes m and n remains unchanged. In this portion, we form $2(n - m)/3$ cliques of three nodes each, in a manner similar to that shown in Figure 3.

Let us assume that all nodes are binary-valued. The number of loop-cutset instances Γ is equal to 2. The new value for K in the decomposed network is $4(m - s) + 8(s - 1) + 16(n - m)/3$. Thus, K is no longer exponential in the number of strands. In the network portion between nodes 1 and m, each clique has a one-node intersection with its parent clique. In the remainder of the network—a diamond-ladder structure—half the cliques have a two-node intersection with their parent clique, and all other cliques except the one containing node n have a one-node intersection with their parent. The three-node clique containing node n functions as the top clique; therefore, this clique does not have a parent clique.

Now we can derive the new total number of operations needed for inference in this network using AD. Recall that this number of operations is given by expression (8). We split the sum over all cliques and consider separately (a) the two-node cliques with a two-node parent clique, (b) two-node cliques with a three-node parent clique, (c) three-node cliques with a one-node intersection with their three-node parent, (d) three-node cliques with a two-node intersection with their parent, and (e) the three-node top clique. We find that the total number of operations required for AD on the network shown in Figure 6 is

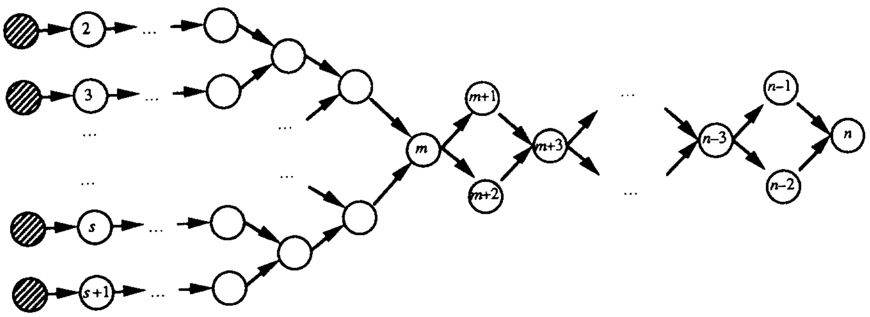

Figure 6. The network of Figure 5 after decomposition of node 1. The shaded nodes represent the group of nodes into which node 1 has been decomposed.

equal to

$$
\begin{aligned}
2n + 2\Big\{9\Big[4(m-s) + 8(s-1) + 16\frac{n-m}{3}\Big] + 10 + (6 \times 2n)& \\
+(m-2s)(8\times 2 + 6\times 4 + 4\times 2)& \qquad (a)\\
+s(8\times 2 + 6\times 8 + 4\times 2)& \qquad (b)\\
+\Big[(s-1) + \Big(\frac{n-m}{3} - 1\Big)\Big](8\times 2 + 6\times 8 + 7\times 4)& \qquad (c)\\
+\frac{n-m}{3}(8\times 4 + 6\times 8 + 4\times 2)& \qquad (d)\\
+(0 + 0 + 10\times 8)\Big\}& \qquad (e)
\end{aligned}
$$

which is equal to $242n - 48m + 208s - 332$ operations. Thus, after decomposition of a single loop-cutset node, the time complexity of inference using AD is linear in the number of nodes and in the number of strands of nodes of the network. Depending on the size and structure of the network, such a hybrid approach can lead to considerably more efficient inference than is realized when CTP or CC is used alone.

DISCUSSION

We have introduced a method, aggregation after decomposition, by which we combine the ideas of two known algorithms for exact probabilistic inference: cutset conditioning and clique tree propagation. The combination method is based on the idea that by conditioning on certain nodes in the network we simplify the network's structure; such a simplification may allow us to find more efficient triangulations than we could find in the untransformed network. Aggregation after decomposition should be applied when the simplification in the clique tree brought about by instantiating certain loop-cutset nodes leads to a reduction in complexity of inference that offsets the increased complexity caused by the fact that we have to perform inference for each of the separate loop-cutset instances. Thus, when we add nodes to the loop cutset, we face a clear trade-off.

The method by which we triangulate a belief network can have a strong effect on the state-space size of the clique tree. For example, in the network shown in Figure 4, when we use maximum-cardinality search starting with node 1, we do not find the most efficient triangulation. However, no polynomial-time algorithms are known that find a triangulation that optimizes the clique tree for inference using CTP. Therefore, in large belief networks, exhaustive search for an optimal clique tree may not be practical. For such networks in particular, instantiation of the nodes of a loop cutset may help us to create efficiently a clique tree that is relatively small.

We have applied the AD method in the Pathfinder system, a medical expert system that offers assistance with diagnosis in hematopathology (Suermondt et al. [30]). The Pathfinder belief network has over 100 nodes representing diseases and diagnostic features. When we tested the AD method on 20 Pathfinder cases, selected in sequence from a library of patient-case referrals, we found that the run times using AD were approximately 0.53 of the run times using CTP. We describe these results in more detail in Ref. 30.

The AD method resulted from research into pragmatic approaches to reasoning under uncertainty. AD combines the underlying ideas of two inference methods for multiply connected belief networks: decomposition of the network by instantiation of loop-cutset nodes and aggregation of nodes into cliques for efficient propagation. Additional hybrid approaches that combine exact probabilistic-inference algorithms are possible. Such approaches may provide the key to tailoring inference algorithms to the structure of the belief-network knowledge base of an expert system.

ACKNOWLEDGMENTS

We thank David Heckerman, Ross Shachter, Eric Horvitz, and Stig K. Andersen for stimulating discussions about this work. Lyn Dupré gave insightful comments on earlier versions of this document. Support for this work was provided by the National Science Foundation under grant IRI-8703710 and by the U.S. Army Research Office under grant P-25514-EL. Computer facilities were provided in part by the SUMEX-AIM resource under grant RR-00785 from the National Institutes of Health.

References

1. Andersen, S. K., Olesen, K. G., Jensen, F. V., and Jensen, F., HUGIN—a shell for building Bayesian belief universes for expert systems, *Proceedings of the 11th International Joint Conference on AI*, Detroit, Mich., 1080-1085, 1989.

2. Chavez, R. M., and Cooper, G. F., KNET: integrating hypermedia and normative Bayesian modeling, *Proceedings of the 4th Workshop on Uncertainty in AI*, Minneapolis, Minn., 49-54, 1988.

3. Heckerman, D. E., Horvitz, E. J., and Nathwani, B. N., Update on the Pathfinder

project, *Proceedings of the 13th Symposium on Computer Applications in Medical Care*, Washington, D.C., 203-207, 1989.

4. Horvitz, E. J., Breese, J. S., and Henrion, M., Decision theory in expert systems and artificial intelligence, *Int. J. Approximate Reasoning* **2**, 247-302, 1988.
5. Cooper, G. F., NESTOR: a computer-based medical diagnostic aid that integrates causal and diagnostic knowledge, Ph.D. Thesis, Stanford University, Stanford, Calif., 1984.
6. Howard, R. A., and Matheson, J. E., *Readings on the Principles and Applications of Decision Analysis*, Strategic Decisions Group, Menlo Park, Calif., 1984.
7. Jensen, F. V., Olesen, K. G., and Andersen, S. K., An algebra of Bayesian belief universes for knowledge-based systems, *Networks* **20**(5), 637-659, 1990.
8. Lauritzen, S. L., and Spiegelhalter, D. J., Local computations with probabilities on graphical structures and their application to expert systems, *J. Roy. Stat. Soc. B* **50**(2), 157-224, 1988.
9. Pearl, J., Fusion, propagation and structuring in belief networks, *AI* **29**, 241-288, 1986.
10. Pearl, J., *Probabilistic Reasoning in Expert Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, Calif., 1988.
11. Chavez, R. M., and Cooper, G. F., A randomized approximation algorithm for probabilistic inference on Bayesian belief networks, *Networks* **20**, 661-685, 1990.
12. Chin, H. L., and Cooper, G. F., Stochastic simulation of Bayesian belief networks, *Proceedings of the 3rd Workshop on Uncertainty in AI*, Seattle, Wash., 106-113, 1987.
13. Fung, R., and Chang, K. C., Weighing and integrating evidence for stochastic simulation in Bayesian networks, *Proceedings of the 5th Workshop on Uncertainty in AI*, Windsor, Ontario, 112-117, 1989.
14. Henrion, M., Propagation of uncertainty by probabilistic logic sampling in Bayes' networks, in *Uncertainty in Artificial Intelligence*, Vol. 2 (J. F. Lemmer and L. N. Kanal, Eds.), Elsevier, New York, 149-164, 1988.
15. Pearl, J., Evidential reasoning using stochastic simulation of causal models, *AI* **32**, 245-257, 1987.
16. Shachter, R. D., and Peot, M., Simulation approaches to general probabilistic inference on belief networks, *Proceedings of the 5th Workshop on Uncertainty in AI*, Windsor, Ontario, 311-318, 1989.
17. Cooper, G. F., The computational complexity of probabilistic inference using Bayesian belief networks, *AI* **42**, 393-405, 1990.
18. Garey, M. R., and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, Calif., 1979.

19. Jensen, F. V., Lauritzen, S. L., and Olesen, K. G., Bayesian updating in recursive graphical models by local computations, Tech. Report R 89-15, Institute for Electronic Systems, Aalborg, Denmark, 1989.

20. Lemmer, J. F., Generalized Bayesian updating of incompletely specified distributions, *Large Scale Syst.*, **5**, 55-68, 1983.

21. Rousseau, W. F., A method for computing probabilities in complex situations, Tech. Report 6252-2, Center for Systems Research, Stanford University, Stanford, Calif., 1968.

22. Shachter, R. D., Evaluating influence diagrams, *Op. Res.* **34**, 871-882, 1986.

23. Pearl, J., Distributed revision of composite beliefs, *AI* **33**, 173-215, 1987.

24. Pearl, J., A constraint-propagation approach to probabilistic reasoning, in *Uncertainty in Artificial Intelligence* (L. N. Kanal and J. F. Lemmer, Eds.), Elsevier, New York, 357-369, 1986.

25. Geiger, D., Verma, T., and Pearl, J., *d*-Separation: from theorems to algorithms, *Proceedings of the 5th Workshop on Uncertainty in AI*, Windsor, Ontario, 118-125, 1989.

26. Suermondt, H. J., and Cooper, G. F., Probabilistic inference in multiply connected belief networks using loop cutsets, *Int. J. Approximate Reasoning* **4**(4), 283-306, 1990.

27. Tarjan, R. E., and Yannakakis, M., Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, *SIAM J. Comput.* **13**(3), 566-579, 1984.

28. Stillman, J., On heuristics for finding loop cutsets in multiply connected belief networks, *Proceedings of the 6th Conference on Uncertainty in AI*, Cambridge, Mass., 265-272, 1990.

29. Suermondt, H. J., and Cooper, G. F., Initialization for the method on conditioning in Bayesian belief networks, Report KSL-89-61, Knowledge Systems Laboratory, Stanford University, Stanford, Calif., 1989. To appear in *AI*.

30. Suermondt, H. J., Cooper, G. F., and Heckerman, D. E., A combination of cutset conditioning with clique-tree propagation in the Pathfinder system, *Proceedings of the 6th Conference on Uncertainty in AI*, Cambridge, Mass., 273-279, 1990.