

Research Note

Fusion and propagation with multiple observations in belief networks

Mark A. Peot

Department of Engineering–Economic Systems, Stanford University, Stanford, CA 94305, USA; and Rockwell International Science Center, Palo Alto Laboratory, 444 High Street, Suite 400, Palo Alto, CA 94301, USA

Ross D. Shachter

Department of Engineering–Economic Systems, Terman Engineering Center, Stanford University, Stanford, CA 94305, USA

Received October 1989

Revised July 1990

Abstract

Peot, M.A. and R.D. Shachter, Fusion and propagation with multiple observations in belief networks (Research Note), *Artificial Intelligence* 48 (1991) 299–318.

The Polytree Algorithm of Kim and Pearl [16, 19] is a fundamental method for evidential reasoning in belief networks. Not only does it provide exact solutions to singly connected networks using efficient, local computations, but variations of it can be applied to more general networks as well. When a belief network is singly connected, the Polytree Algorithm can compute a posterior marginal distribution for each variable by visiting each node at most once for each piece of evidence. By contrast, the related algorithms based on undirected graphs [1, 14–16, 28] only need to visit each node at most twice no matter how much evidence is observed. In this paper, a Revised Polytree Algorithm is developed with the same complexity as the undirected methods, but within the directed framework of the Polytree Algorithm. When this new algorithm is applied via “cutset conditioning” to general networks it obtains not just the corresponding significant improvement in speed, but also a much simpler form for combination. Furthermore, the revised algorithm requires only minor modifications to existing implementations of the Polytree Algorithm.

1. Introduction

Belief networks are directed acyclic graphs in which the nodes represent random variables and the arcs signify their conditional dependence. In addition to representing the domain knowledge of an expert, this kind of network provides a computational architecture for the propagation of evidence. One of the main reasons for its increasing popularity as a representation for uncertain knowledge is the existence of an efficient algorithm for processing singly connected belief networks with local computations. The Polytree Algorithm of Kim and Pearl as described in the literature [16, 19] propagates each new observation by visiting each node exactly once and sending exactly one message along every arc. Although in the worst case this process must be repeated for each new observation, we will consider a control strategy for multiple observations which can be much more efficient.

The Polytree Algorithm has also been applied successfully to networks which are not singly connected, even though such problems are NP-hard in general [8]. The most common techniques for generalizing it involve either aggregating nodes [17, 22] or “cutset conditioning” [20], in which the network is rendered singly connected by “observing” all possible values for a set of variables.

There are also a number of related propagation algorithms that operate on an undirected graph analog to the original directed graph. Starting with the algorithm developed by Lauritzen and Spiegelhalter [17], these have been further refined into the “HUGIN concept” [14, 15] and similar methods [28]. These new approaches claim to be the fastest methods available for exact solution of general belief networks [1]. By aggregating nodes into sets or “cliques”, they achieve a singly connected structure even when the original problem is not singly connected. This structure allows efficient updating of the network as evidence is observed, exploiting the conditional independence captured in the network. Rather than propagate each observation throughout the network, multiple observations can be incorporated at the same time while visiting each clique at most twice and sending at most one message in each direction along every arc.

In this paper, we develop a Revised Polytree Algorithm, which requires only slight changes to the original Polytree Algorithm. Nonetheless these modifications allow it to propagate multiple observations on the original directed graph in a manner similar to the undirected algorithms. The original Polytree Algorithm is a purely distributed algorithm and it is the control strategy which must be changed for efficient propagation of multiple observations. (A similar modification had also been developed by the HUGIN team [13].) Although this can be applied with comparable savings to the node aggregation problems, it is particularly useful with cutset conditioning. As originally conceived [20], cutset conditioning requires an initialization and multiple observations per case, plus a complicated method for weighting and combining cases. With the Revised

Polytree Algorithm, the network is only initialized once, multiple observations are quickly propagated, and the weighting computations are replaced by a simple sum.

In Section 2, we present the Decomposition Theorem and Decomposition Algorithm which underlie the Revised Polytree Algorithm for singly connected networks, presented in Section 3. Section 4 applies these techniques to the method of cutset conditioning, while Section 5 shows how to coordinate these methods with other solution approaches. Finally, Section 6 discusses conclusions and extensions.

2. Decomposition Theorem

When an arc or a node separates a network into two pieces, it will be said to “decompose” the network. The Decomposition Theorem proves that only a limited amount of information needs to be passed between those pieces, and leads directly to the Decomposition Algorithm, with properties similar to the undirected HUGIN concept [14, 15].

A belief network is based on a directed acyclic graph in which each node in the graph represents an uncertain quantity with a finite number of possible values. Capital letters, such as “ X ”, denote individual nodes or sets of nodes and also the variables corresponding to them, while small letters, such as “ x ”, denote specific values which those variables might take on. The arcs into a node X indicate its parents or conditional predecessors in the graph, U_X , and its children, Y_X . Each node in the network contains some representation for the conditional probability distribution for its corresponding variable, conditioned on the values of its parents, $P\{X|U_X\}$. Let e be the evidence which has been obtained about the variables in the network. This evidence consists of observations of the values of some of the variables: the observations about X are denoted by e_X .

A (simple) *chain* is an undirected path formed from arcs in the network, ignoring their directions, in which any node is included at most once. A network is said to be *connected* if there is at least one chain between any two nodes. (In this paper, it will be assumed that the network is connected. If not, then every connected component can be processed independently.) It is *singly connected* if there is at most one chain between any two nodes, and it will be called a *knot* if there are multiple chains (sharing no arcs) between *every* pair of nodes. A network will be called a *block* if it is either a node or a knot, and every network can be thought of as a singly connected set of blocks [7]. At one extreme, if every block is a node, then the network is singly connected. At the other extreme, there are no nodes which have degree one and the whole network is a single knot.

Suppose that the nodes in the network can be partitioned into two sets, A

and B , which are only connected by a single arc, from a node X in A to a node Y in B , as shown in the diagram in Fig. 1. Such an arc will be said to *decompose* the network. If the network is singly connected, then every arc decomposes the network. The evidence can also be partitioned into e_A and e_B , corresponding to the two sets. Define two messages to be transmitted along the arc. The message from X to Y , $\pi_Y(x)$, is defined as

$$\pi_Y(x) := P\{X = x, e_A\}$$

for all possible values x of X . (Note that this is slightly different from the definition of Pearl [19] which is a conditional rather than an unconditional probability. This change leads to a simpler form for cutset conditioning, shown in Section 4, which can exploit the multiple observation updates in the Revised Polytree Algorithm.) Similarly, the message from Y to X , $\lambda_Y(x)$ is defined as

$$\lambda_Y(x) := P\{e_B | X = x\} .$$

The structure of the network shown in Fig. 1 implies that B is conditionally independent of A given X , or X *d-separates* B from A . Therefore, it can be shown [18, 19, 31] that e_B is conditionally independent of A and e_A given X :

$$\lambda_Y(x) = P\{e_B | A, e_A, X = x\} = P\{e_B | X = x\} .$$

Using these definitions, one can compute various probabilities. For example, by conditioning,

$$\begin{aligned} P\{X = x, e\} &= P\{X = x, e_A, e_B\} \\ &= P\{e_B | X = x, e_A\} P\{X = x, e_A\} \\ &= \lambda_Y(x) \pi_Y(x) . \end{aligned}$$

The probability of the evidence is computed by summing the distribution over x ,

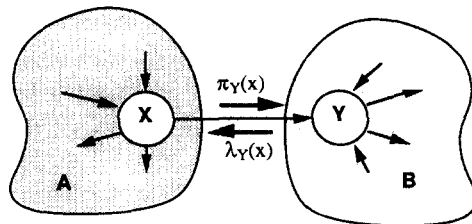


Fig. 1. Conditions for the Decomposition Theorem.

$$P\{\mathbf{e}\} = \sum_x P\{X = x, \mathbf{e}\} = \sum_x \lambda_Y(x) \pi_Y(x). \quad (1)$$

The distribution for A and the evidence can be computed by,

$$\begin{aligned} P\{A, \mathbf{e}\} &= P\{A, \mathbf{e}_A, \mathbf{e}_B\} \\ &= P\{\mathbf{e}_B | A, \mathbf{e}_A\} P\{A, \mathbf{e}_A\} \\ &= \lambda_Y(X) P\{A, \mathbf{e}_A\}, \end{aligned} \quad (2)$$

while the distribution for B and the evidence is computed by summing and then conditioning,

$$\begin{aligned} P\{B, \mathbf{e}\} &= P\{B, \mathbf{e}_A, \mathbf{e}_B\} \\ &= \sum_x P\{B, \mathbf{e}_A, \mathbf{e}_B, X = x\} \\ &= \sum_x P\{B | \mathbf{e}_A, \mathbf{e}_B, X = x\} P\{\mathbf{e}_A, \mathbf{e}_B, X = x\} \\ &= \sum_x P\{B | \mathbf{e}_B, X = x\} \lambda_Y(x) \pi_Y(x). \end{aligned} \quad (3)$$

When an arc emanating from node X decomposes the network, these properties provide intuitive insight as well as computational power. The message from X to Y , $\pi_Y(x)$, is a distribution for the probability of the evidence “upstream” of the arc and each possible value of X in the light of that evidence. (Although that distribution can theoretically be defined over an uncountable sample space for X , our algorithm must assume that it is finite.) On the other hand, the message from Y to X , $\lambda_Y(x)$, is the likelihood for the remaining evidence as a function of the possible values for X . It shows how much that evidence confirms or challenges our original beliefs about the distribution for X . Likelihood functions, by themselves, are not natural to think about, but the posterior distribution, given by the product $\lambda_Y(x) \pi_Y(x)$, is a more intuitive distribution. It describes our beliefs about the observed evidence and each possible value of X given that evidence. Because the arc from X to Y decomposes the network, the two components which comprise this posterior distribution can be organized independently.

These results lead to the following decomposition theorem similar to Pearl’s [19], corresponding to the network shown in Fig. 1. It is the basis for all of the results in this paper.

Theorem 1 (Belief Network Decomposition). *Given an arbitrary belief network that can be partitioned into two sets of nodes, A and B , connected by a single arc from a node X in A to a node Y in B . If the evidence in the sets is \mathbf{e}_A and \mathbf{e}_B , then the posterior probabilities $P\{A | \mathbf{e}_A, \mathbf{e}_B\}$ and $P\{B | \mathbf{e}_A, \mathbf{e}_B\}$ can be com-*

puted by passing a single message (with one number for each possible value of X) from each set to the other. Furthermore, the two messages are independent.

Proof. The message from A to B is $\pi_Y(x)$ and the message from B to A is $\lambda_Y(x)$. The messages are independent because all of the relevant information required to compute $\pi_Y(x)$ is in the ancestral set of $\{x\} \cup e_A$ which is contained completely within A , and all of the information to compute $\lambda_Y(x)$ is contained completely within B [9, 10, 23, 25].

Finally the posterior probabilities are simply proportional to the joint distributions given in (2) and (3) with the normalizing constant equal to the probability of evidence as in (1):

$$\begin{aligned} P\{A|e\} &= P\{A|e_A, e_B\} = \alpha P\{A, e_A, e_B\} \\ &= \alpha \lambda_Y(X) P\{A, e_A\} \end{aligned}$$

and

$$\begin{aligned} P\{B|e\} &= P\{B|e_A, e_B\} = \alpha P\{B, e_A, e_B\} \\ &= \alpha \sum_x P\{B|e_B, x\} \lambda_Y(x) \pi_Y(x), \end{aligned}$$

where

$$\alpha = [P\{e\}]^{-1} = [P\{e_A, e_B\}]^{-1} = \left[\sum_x \lambda_Y(x) \pi_Y(x) \right]^{-1}$$

In each expression, all of the information from the other set is contained in the messages, $\lambda_Y(x)$ and $\pi_Y(x)$, already shown to be independent. \square

The messages sent between A and B contain all of the relevant information about each set. As long as the possible values for X do not change, *any* other changes to nodes, arcs, probability distributions, or evidence within either block will be reflected in its message. Conversely, if there is no change within a set, then its message will stay the same and there is no need to revise it.

Suppose that every chain from the parents of node X to its children all contain X . Such a node is said to *decompose* the network. (If the network is singly connected, then every node decomposes the network.) The results above can be applied by dividing such a node X into two nodes so that the network can be partitioned into the sets A and B connected by the arc between the two nodes. Let e_X^+ denote the evidence ‘‘above’’ X , that is, all the evidence separated from X if its incoming arcs were deleted. By construction e_X^+ will correspond to e_A . Similarly, define e_X^- as the remaining evidence, corresponding to e_B . Let the *causal support* for X , $\pi(x)$, be

$$\pi(x) := P\{X = x, e_X^+\},$$

and the *diagnostic support* for X , $\lambda(x)$, be

$$\lambda(x) := P\{e_X^- | X = x\}.$$

Then, by the results above,

$$P\{X = x, e_X^+, e_X^-\} = \lambda(x)\pi(x)$$

and

$$P\{e_X^+, e_X^-\} = \sum_x \lambda(x)\pi(x).$$

For general networks, recursive application of the Decomposition Theorem leads to an algorithm in which each arc between blocks transmits one message in each direction in order to compute the posterior distributions for all blocks. First, choose an arbitrary block in the network to serve as the *pivot* block. Its posterior distribution can be computed once every other block in the network sends a message in its direction beginning with the most distal blocks and proceeding inward. At that point, it can send return messages to all of its neighbors. They in turn will send messages to all of their neighbors distal to the pivot block, and this process continues with messages passing away from the pivot block. Since every arc between blocks decomposes the network, the Decomposition Theorem ensures that every block has the information it needs to compute its posterior distribution. Furthermore, only one message in each direction need be sent on any arc, and every node must be visited at most twice. Since the number of arcs in a singly connected network is less than the number of nodes, this leads to the following corollary.

Corollary 1. *Regardless of the amount of new evidence (and other changes) in a belief network, the number of messages which must be sent between blocks in order to compute a posterior distribution for each block is less than twice the number of blocks.*

Suppose that some changes have been made to some of the blocks in the network. Let S be the smallest connected set of blocks which contains all of the modified blocks. Such a set can be found in time linear in the number of blocks using local computations. The Decomposition Theorem ensures that there is no need to receive a new message within S from any block outside S . Thus the algorithm described above can be modified to obtain the Decomposition Algorithm:

Decomposition Algorithm. First select an arbitrary block within S to be the pivot block.

First Pass. Each block in S sends a message toward the pivot block after it has received messages from all but one of its neighboring blocks within S .

Second Pass. Starting with the pivot block, messages are sent away from the pivot block to all of the blocks in the network.

Each block in S except the pivot block will be visited twice. The pivot block and each block outside of S will be visited once.

Consider, for example, the network shown in Fig. 2(a), with blocks A through L . Some new information has been obtained for blocks C , J , and K . The smallest connected set containing these blocks is $S = \{J, E, B, F, C, G, K\}$. Any block could serve as a pivot block but suppose that F were chosen. The diagram is now drawn in Fig. 2(b) as a tree rooted at the pivot block F . Messages are passed toward the pivot block as shown in Fig. 2(c), but only from the blocks in S . Some of these messages are from child to parent, such as the message from K to G , and some are from parent to child, such as the message from C to F . Once the pivot block has received messages from both B and C , messages are passed to the rest of the network as shown in Fig. 2(d). Every arc in the network will be used to send exactly one message, some of which are from child to parent and the others will be from parent to child.

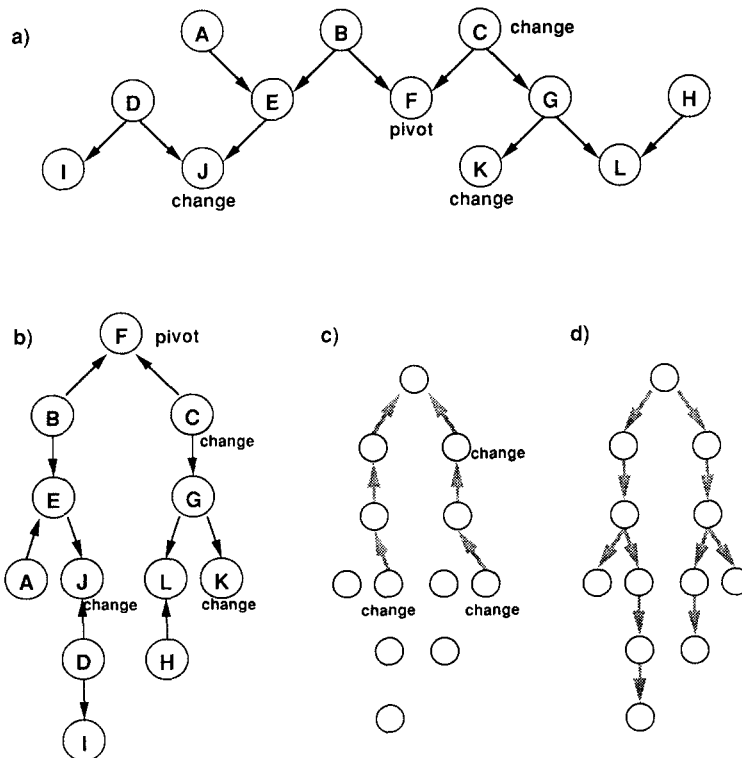


Fig. 2. Example of the Revised Polytree Algorithm with multiple observations.

Additional changes to one or more blocks can be incorporated at any time, with the pivot block chosen independently each time, and the set S determined only by the new changes. The nature of those changes can include revisions to the conditional distributions or observations, or even to forget that there was an observation.

There are several strategies to the choice of the pivot block. If changes have only been made to one block, then it should be the pivot block, each node will be visited exactly once, and the Decomposition Algorithm passes messages just as in the Polytree Algorithm [19]. In general, there will be multiple blocks with changes and S would contain more than one block. If one block in S is particularly difficult to process, then it is a good candidate to be the pivot block since it would only need to be visited once instead of twice. When there are multiple processors, the pivot block can be “centered” in the set S , so that all of its neighbors can compute their messages in parallel. If the pivot block were fixed in advance in order to “hardwire” the first and second passes as in [13], then S is effectively the smallest connected set of blocks which contains all of the modified blocks *and* the pivot block. Alternatively, if there is a single block at which updated information is required, then it can serve as the pivot block and there is no need for a second pass.

All of these operations can be managed with local computations. In a “Zero Pass”, the pivot block sends a request to each of its neighbors for updated messages, and they pass it along to their neighbors in turn. If all but one of a block’s neighbors return messages indicating that there has been no change, and there has been no change within the block, then it can send a message to its remaining neighbor that there has been no change. If there has been some change, then this block must be in the set S , and it must compute a new message for its remaining neighbor. Once the pivot block receives messages from all of its neighbors, it should return messages to all of them to inform them of any changes.

3. Fusion and propagation in singly connected networks

In this section, the Decomposition Algorithm is applied to a singly connected network. The resulting Revised Polytree Algorithm is a modified version of the Polytree Algorithm of Kim and Pearl [16, 19]. Regardless of the number of observations and other changes, in order to compute the posterior distribution for each node, the number of messages which must be sent is less than twice the number of nodes. (An earlier modification to the Polytree Algorithm with similar complexity was developed for the MUNIN belief network [13].) The Revised Polytree Algorithm is only slightly different from the Polytree Algorithm, and these differences will be highlighted in this section. Although these differences are necessary to obtain the “cutset conditioning” algorithm

described in Section 4, all of the properties of the algorithm presented here also apply to the algorithm as presented by Pearl [19].

There are only two attributes which are truly local to each node X . These are its conditional probability distribution, $P\{X|U_X\}$, and its *local evidence*,

$$\lambda_X(x) := P\{e_X|X=x\}.$$

These attributes can be set independently for each node at any time, although there are global restrictions on what constitutes a consistent or coherent set of attributes for the entire network. The test for global coherence is simply whether $P\{e\} > 0$. The local coherence requirements are that the conditional probability distribution must be nonnegative and sum to one for each conditioning case, and the local evidence must be nonnegative and nonzero. Typical values for $\lambda_X(x)$ would be an elementary vector $I_j = (0, \dots, 1, 0, \dots)$ for a perfect observation of the j th value for x , or $\mathbf{1} = (1, \dots, 1)$ when there is no local evidence. Any node can have local evidence, regardless whether it has children. (In some cases, without full knowledge of the experimental design, the probability of the local evidence is indeterminate. However, in those cases, it is sufficient that

$$\lambda_X(x) = \alpha P\{e_X|X=x\} \quad \text{for some positive } \alpha.$$

Within cutset conditioning, described in Section 4, there is an additional requirement that α be the same for all instantiations of X .)

Since each node X decomposes the network, the causal support $\pi(x)$ and diagnostic support $\lambda(x)$ are always defined. They serve to reconcile and synthesize the local attributes for all of the nodes in the network. Each local attribute is incorporated into either $\pi(x)$ or $\lambda(x)$, but not both. All local information in nodes connected to X by its incoming arcs will be assimilated in $\pi(x)$, and $\lambda(x)$ contains local information from all nodes connected to X by its outgoing arcs. For X , itself, its conditional probability distribution, $P\{X|U_X\}$, is integrated into its causal support, $\pi(x)$, and its local evidence, $\lambda_X(x)$, is included in its diagnostic support, $\lambda(x)$.

The formulae for the Revised Polytree Algorithm are almost identical to those of the Polytree Algorithm except for the inclusion of $\lambda_X(x)$ and the absence of most of the normalizing constants, which might otherwise destroy some valuable information. Consider any node X , with parents $U = \{U_1, \dots, U_m\}$, and children $Y = \{Y_1, \dots, Y_n\}$. The initialization of the network should set $\pi(x)$ and $\pi_{Y_j}(x)$ to the prior probability $P\{X=x\}$, and $\lambda(x)$, $\lambda_X(x)$ and $\lambda_{Y_j}(x)$ to $\mathbf{1}$. Since the prior probabilities in a singly connected network can be computed by visiting all of the nodes in graph order,

$$P\{X=x\} = \sum_u P\{X=x|U=(u_1, \dots, u_m)\} \prod_k P\{U_k=u_k\},$$

the initialization operations can be performed by visiting every node and arc in the network exactly once. This initialization can be performed when the network is first created or at the time of the first update by setting each local evidence to **1** and marking every node as revised. To update the network, one can compute the probability for any node X and the evidence by

$$P\{X = x, \mathbf{e}\} = \lambda(x)\pi(x),$$

so

$$\text{BEL}\{x\} := P\{X = x | \mathbf{e}\} = \alpha P\{X = x, \mathbf{e}\} = \alpha \lambda(x)\pi(x),$$

where α is a normalizing constant equal to $[P\{\mathbf{e}\}]^{-1}$. The causal support at the node is

$$\begin{aligned} \pi(x) &= P\{X = x, \mathbf{e}_X^+\} \\ &= \sum_u P\{X = x | U = u\} \prod_k \pi_X(u_k), \end{aligned}$$

and its diagnostic support is given by

$$\lambda(x) = P\{\mathbf{e}_X^- | X = x\} = \lambda_X(x) \prod_k \lambda_{Y_k}(x).$$

The message X passes to its parent U_i is

$$\lambda_X(u_i) = \sum_x \lambda(x) \sum_{u_k: k \neq i} P\{x | u\} \prod_{k \neq i} \pi_X(u_k),$$

and it sends its child Y_j the message

$$\pi_{Y_j}(x) = \pi(x) \lambda_X(x) \prod_{k \neq j} \lambda_{Y_k}(x). \quad (4)$$

If $\lambda_{Y_j}(x) > \mathbf{0}$ then

$$\pi_{Y_j}(x) = \pi(x) \lambda(x) / \lambda_{Y_j}(x).$$

Otherwise, (4) should be used to compute $\pi_{Y_j}(x)$, since it is *not* indeterminate as defined.

These formulae are only slightly modified from those in Pearl [19], which also contains a proof of their correctness. The only differences are

- (1) the change in the definition of $\pi(x)$ means the normalizing constants are left off all equations except to compute BEL;
- (2) local evidence $\lambda_X(x)$ is stored within each node instead of creating “dummy” or “evidence” nodes, and must be included in the formulae for each node; and

- (3) local attributes can be freely revised at any time, and therefore messages $\pi(x)$ and $\lambda(x)$ are never set directly from local beliefs or observations.

These updating formulae can now be applied using the Decomposition Algorithm. Whenever information, either the local evidence or the conditional probability distribution, changes at one or more nodes, the new posterior distributions at all nodes can be computed by passing at most two messages per arc.

4. Updating knots using cutset conditioning

When the network is not singly connected there must be one or more blocks which are knots. One technique for resolving these knots, commonly called “cutset conditioning”, applies the singly connected algorithm presented in the previous section, by “observing” all possible values for a set of variables [20, 29, 30]. If this set is chosen properly, then observing values for the variables will make the network singly connected, and the Revised Polytree Algorithm can be applied to compute posterior distributions and messages to be sent from the knot. These computations must be performed again for every possible combination of values for the variables in the set. Once that has been done, the results must then be combined into summary distributions and messages.

The Revised Polytree Algorithm simplifies cutset conditioning in two key respects. First, by efficiently propagating multiple observations, it is more efficient than the Polytree Algorithm at setting the observed values for the set of variables. If there are s variables in the set, then this is a savings of a factor of at least $\frac{1}{2}s$, clearly a reduction of order $O(s)$, although the method is still NP-hard. Second, the complicated and sequential process used to compute weights for combining results [20] is replaced by a simple sum. Although this saves computation, it does not affect the dominating terms in the complexity analysis. Nonetheless, it avoids an unnecessary and confusing calculation. This is accomplished by the change in the definition of the π -message described in Section 2.

To understand how cutset conditioning works, consider the belief network shown in Fig. 3(a). This is a knot, since there are multiple chains, sharing no arcs, between every pair of nodes. If, however, we were able to observe that variable E takes on value e , then its outgoing arcs can be *absorbed* [21]. Once there is no longer any uncertainty about a node, it becomes independent of its children and its value can be instantiated in their distributions. If D had conditional distribution $P\{D|E\}$ before E was observed, it now has an unconditional distribution given by $P\{D\} = P\{D|E=e\}$. Once the outgoing arcs from E have been observed, the network is no longer a knot: E would become disconnected from the other nodes; D , H , and I become single-node

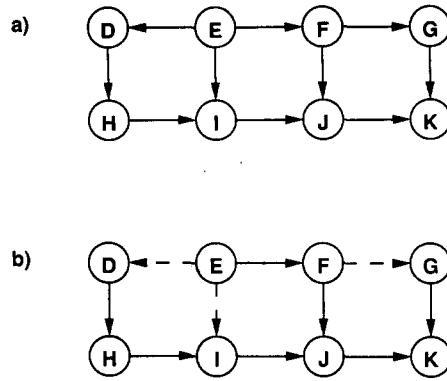


Fig. 3. Absorbing arcs for cutset conditioning.

blocks; and $\{F, G, J, K\}$ continue to form a knot. If the node F were also observed, and its outgoing arcs absorbed, that knot would also be broken, and there would be no multiple chains left in the network. Therefore, observing nodes E and F , is sufficient to break up the knot. This is not a unique combination since observing E and G , for example, will also break it up.

A set of nodes that will break up all of the knots in a network when their values are observed is called a (*loop*) *cutset*. (From some heuristics for choosing a cutset for conditioning see [29, 30].) Suppose that a particular cutset C has been selected and that the possible combinations of values for the cutset variables are given by c_1, \dots, c_s . These are called the *cutset instantiations*. The number of instantiations grows exponentially in the size of the cutset, since it is the product of the number of possible values for each variable in the cutset. Thus there can be substantial savings in recognizing and separating two adjacent knots rather than solving them both together. If they have instantiation sizes s_1 and s_2 , respectively, then the saving is by a factor of at least $\frac{1}{2}s_1s_2/(s_1 + s_2)$, a reduction exponential in the number of nodes in the smaller cutset.

Although outgoing arcs must be absorbed from cutset nodes to break up the knots, it can be desirable to leave some of those arcs unabsorbed. The propagation mechanism requires that the network be connected, so that information about a single block can be incorporated into the messages throughout the network. A simple check allows us to absorb enough arcs to break up the knot without destroying its connectedness. For each cutset node X , examine each of its outgoing arcs (X, Y) in turn, and only absorb the arc if there is a chain, not containing that arc, from X to Y . Afterwards, the knot will be connected but only singly connected. Consider the diagram drawn in Fig. 3(a) with cutset $C = \{E, F\}$. Examining first the outgoing arcs from E , since there is a chain from E to D , (E, D) should be absorbed, and there is still a chain from E to I so (E, I) should also be absorbed. (The absorbed arcs are

drawn as dashed lines in Fig. 3(b).) Now, however, the only chain from E to F is the arc, so (E, F) should not be absorbed or it would disconnect the network. By a similar argument, (F, G) should be absorbed but (F, J) should not. Notice that the choices of which arcs to absorb are not unique since they depend on the order arcs are examined. Different choices would lead to different singly connected networks.

We have seen how the knot becomes singly connected by observing the cutset variables. In fact, however, we have not really observed those variables, so we must consider all possible values or instantiations for them. Whenever a new value is assigned to a cutset node X , its children must be informed. If an arc (X, Y) were not absorbed, then the Revised Polytree Algorithm would propagate this change automatically. On the other hand, if arc (X, Y) were absorbed, we need some other way for Y to learn about the change to X . A phantom message $\pi_Y(x)$ is sent along the absorbed arc to indicate that the "constant" value for X has changed. This arc is used exclusively to inform Y about changes to the instantiation for X and should never be used for messages during the Revised Polytree Algorithm.

Once the arcs have been absorbed, the singly connected knot must be initialized. First, a phantom message must be generated for each absorbed arc. This first message is arbitrary, provided it is a nonnegative vector summing to 1, since it will be revised at the first instantiation. Next, the knot should be initialized either before or while the messages from its neighbors plus any evidence within it should be propagated using a single application of the Revised Polytree Algorithm. Let e be the combined evidence for the neighbors' messages and any evidence within the knot. Since this information will not change from one instantiation to another, these messages will never have to be sent again.

Let $\pi(x, c_i)$ and $\lambda(x|c_i)$ denote the causal and diagnostic supports for the instantiation c_i . They can be computed by setting the local evidence for each cutset node and the phantom message for any of its absorbed arcs to the same $I_j = (0, \dots, 1, 0, \dots)$ corresponding to the j th value for the cutset variable. Once the changes to the cutset nodes and their "phantom children" have been made, the Revised Polytree Algorithm should be used to restore consistency throughout the knot. Of course, if the value of a particular node in the cutset has not changed from its previous instantiation, there is no need to change its local evidence or to send the corresponding phantom messages. Therefore, some additional savings can be realized by ordering instantiations so that only one or two cutset node values change between instantiations.

It is now easy to compute the unconditional values for the distribution for any node X . Since

$$P\{x, e, c_i\} = \lambda(x|c_i)\pi(x, c_i),$$

simply sum this over the cutset instantiations to obtain

$$P\{x, \mathbf{e}\} = \sum_{c_i} P\{x, \mathbf{e}, c_i\} = \sum_{c_i} \lambda(x | c_i) \pi(x, c_i). \quad (5)$$

Thus, $P\{x, \mathbf{e}\}$ can be calculated by keeping a running total as $\lambda(x | c_i) \pi(x, c_i)$ is determined for each instantiation of the cutset. When an arc decomposes a belief network, its messages can be calculated similarly, since

$$\begin{aligned} \pi(x) &= P\{X = x, \mathbf{e}_X^+\} \\ &= \sum_{c_i} P\{X = x, C = c_i, \mathbf{e}_X^+\} \\ &= \sum_{c_i} \pi(x, c_i) \end{aligned} \quad (6)$$

and

$$\begin{aligned} \lambda(x) &= P\{\mathbf{e}_X^- | X = x\} \\ &= \sum_{c_i} P\{C = c_i, \mathbf{e}_X^- | X = x\} \\ &= \sum_{c_i} \lambda(x | c_i). \end{aligned} \quad (7)$$

If node X is in a knot, note that in general,

$$P\{x, \mathbf{e}\} \neq \lambda(x) \pi(x),$$

because X does not decompose the belief network. In this case, $P\{x, \mathbf{e}\}$ must be computed using (5) instead. Nonetheless, the outgoing messages to the knot's neighbors can be computed as in (6) and (7), since their arcs do decompose the network. Note that the operations in (5), (6), and (7) are simple sums over the different cutset instantiations. Unlike earlier versions of cutset conditioning [20, 29, 30], we do not have to compute weights for the terms in these sums because of the change in the definition of the π -messages introduced in Section 2.

Finally, the nature of the computations depends on when this knot is being processed in the Decomposition Algorithm for the entire network. If this knot is the pivot block then all distributions and messages should be computed. Otherwise, if this is the first pass, then only compute the message to send toward the pivot block. Within the knot, this can be accomplished by sending all messages toward that arc. During the second pass, of course, a message will be returned on that arc (with no other changes in the knot), so distributions can be computed for the rest of the knot, including messages to all of the knot's other neighbors, incorporating all of the information in the entire network.

5. Other methods for solving knots

There are other exact methods for resolving knots in current use [1, 12, 14, 15, 17, 22, 23], and there are likely to be others in the future. To be used in conjunction with the Revised Polytree Algorithm, these methods must be willing and able to accept and deliver π - and λ -messages from and to their neighboring blocks. The objective of this section is to show how an arbitrary belief network algorithm can generate these messages. For algorithms of the Lauritzen–Spiegelhalter variety, this is no problem, due to their message-passing nature, and their underlying similarities to the Polytree Algorithm. For example, in the terminology of [17], the ϕ_B -function corresponds to a proportional change in the λ -message, and clique marginal $p(S_i)$ corresponds to the π -message in the original Polytree Algorithm.

On the other hand, approximation algorithms such as the simulation methods [2, 4, 5, 8, 11, 21, 26, 27], produce estimates of the posterior marginal distributions at each node rather than messages between (sets of) nodes. Such methods can be adapted to utilize message passing through some simple transformations [27], but some information is necessarily lost in the process.

Consider an arc that goes into the knot from outside. A node X should be placed at the tail of the arc. Let Y be the node in the knot at the head of the arc and $\pi_Y(x)$ be the message entering the knot. This message should be normalized and used as the prior distribution $P\{X\}$ for the simulation. Afterwards, there will be a posterior distribution, $P\{X|\mathbf{e}\}$, computed for the node. Set the outgoing message $\lambda_Y(x)$ to

$$\lambda_Y(x) = P\{X|\mathbf{e}\} / \pi_Y(x).$$

If any of the elements of the incoming message $\pi_Y(x)$ are zero, then the corresponding element of $\lambda_Y(x)$ is indeterminate. When this arises, the two messages in opposite directions on the arc are no longer orthogonal, a violation of the Decomposition Theorem. This is unavoidable when computing from a posterior distribution, so caution should be exercised whenever an incoming message to the knot contains zeros. If that incoming message should change to one which does not have zeros for those elements, then the return message must be recomputed.

Consider the analogous case of an arc that goes from the knot to outside. Let X be the node at the head of the arc. Let U be the node in the knot at the tail of the arc and $\lambda_X(u)$ be the message entering the knot. Node X should be treated as an observed evidence node and this message should be used as its likelihood function for the simulation. Afterwards, its parent U will have a posterior distribution, $P\{U|\mathbf{e}\}$, and the outgoing message $\pi_X(u)$ can be set to

$$\pi_X(u) = P\{U|\mathbf{e}\} / \lambda_X(u).$$

As in the earlier case, this quotient is indeterminate when components of the incoming message are zero, leading to a violation of the Decomposition Theorem. If the incoming message changes to one without zeros for those elements, then the simulation must be recomputed.

There is one other problem with the outgoing messages. The message passing in the Revised Polytree Algorithm maintains information about the evidence throughout the network, in that

$$P\{\mathbf{e}\} = \sum_x \lambda(x)\pi(x)$$

for any node X in the entire network. Some of this information can be lost when messages are passed into the simulation as described above. Although it is not difficult to modify some of the methods such as [27] to maintain this information, it would only be worthwhile when a simulation is being used to resolve a knot as a subproblem under cutset conditioning. As was shown in the last section, the probability of evidence, in particular the probability for the cutset instantiation, plays a critical role in cutset conditioning.

As was also noted in the case of cutset conditioning, if the knot being solved with simulation is to be visited more than once then only a message toward the pivot will be required during the first pass. It is only when the knot is being visited during the second pass in the Decomposition Algorithm that posterior probabilities should be computed for all of the nodes, along with any required messages away from the pivot.

6. Conclusions and extensions

In this paper, we have developed a Revised Polytree Algorithm for singly connected belief networks in Section 3, based on the insight of the Decomposition Theorem in Section 2. Unlike the Polytree Algorithm [19] which requires time proportional to the number of observations under its implied control strategy, the Revised Polytree Algorithm never needs to visit any node more than twice in order to propagate all of the evidence and thus achieves a savings proportional to the number of observations. Since the Revised Polytree Algorithm consists of only a few changes to the original Polytree Algorithm, it is easy to modify an existing implementation of the Polytree Algorithm. The power and elegance of the new method are inherited directly from the original one. Most of the features of the revised method, such as a simple mechanism to allow unlimited revision and retraction of local observations and beliefs, are also possible under the original framework.

Just as the Polytree Algorithm has been applied to more general problems by clustering [17, 22] and cutset conditioning [20], the Revised Polytree Algorithm can also be adapted to solve those problems. The benefits of

propagating multiple observations at once result in savings proportional to the number of observations for clustering, and to the number of nodes in a cutset for cutset conditioning. There are other advantages to using the new method for cutset conditioning as described in the paper. The algorithm becomes much simpler, because it is unnecessary to compute weights for the different cutset instantiations. It would not be surprising if there were similar benefits to using new heuristics [3] to perform clustering with the Revised Polytree Algorithm corresponding to the undirected tree structure the Lauritzen–Spiegelhalter Algorithm [17] uses for its cliques.

Comparisons of the Revised Polytree Algorithm with undirected clique methods [14, 15, 17, 28] is a promising area for future research. It is unclear for which problems these different approaches are better suited, but we can speculate on their basic similarities and differences by considering a singly connected network (or one in which the cliques correspond to clustering [22] thus reducing the original problem to a singly connected network). In that case, there is a correspondence between those nodes with parents and cliques. If X is a node with parents, then its corresponding clique contains X and its parents U . The clique (posterior) marginal distribution is given by

$$P\{X = x, U = u | e\} \propto P\{X = x | U = u\} \lambda(x) \prod_i \pi_X(u_i).$$

The main difference between the methods appears to be that the undirected methods, such as the HUGIN concept, precompute and maintain this clique marginal, while the Revised Polytree Method keeps it in factored form. Although the complexity of the two approaches should be the same, HUGIN seems to be more efficient [1], in part because it precomputes the clique marginal, which a factored method might have to recompute many times, and also because it can efficiently exploit the sparsity in that clique marginal. Nonetheless, the Revised Polytree Algorithm could also be implemented to exploit that sparsity. Moreover, if we are hypothesizing and retracting changes, as in cutset conditioning or probabilistic sensitivity analysis, the factored form might prove to be a more efficient representation, since only the changing local attributes need to be revised before propagation. Although we still have much to learn about the relative merits of these approaches on many problems, the most compelling argument for a directed algorithm such as the one developed here is to maintain as close a correspondence as possible between the representation for analysis and the representation originally used to formulate, elicit, and revise the belief network knowledge base.

Finally, a general framework of block structuring for belief networks has been introduced, which easily allows large networks to be decomposed into simpler models. In general, these smaller subproblems should be significantly easier to analyze, and they can be solved by a variety of methods. Generalizations of the block structures to recognize additional opportunities for decompo-

sition will become increasingly important as the belief network representation is applied to develop larger and more sophisticated knowledge bases.

Acknowledgement

We benefited greatly from the comments of Stig Andersen, Jack Breese, David Heckerman, Judea Pearl, Sampath Srinivas, Jaap Suermondt, and the anonymous referees. The algorithm was developed using the IDEAL system developed by Jack Breese and Sampath Srinivas at Rockwell International. This research was partially supported by the National Science Foundation through a Graduate Fellowship.

References

- [1] S.K. Andersen, K.G. Olesen, F.V. Jensen and F. Jensen, HUGIN—a shell for building belief universes for expert systems, in: *Proceedings IJCAI-89*, Detroit, MI (1989).
- [2] C. Berzuini, R. Bellazzi and S. Quaglino, Temporal reasoning with probabilities, in: *Proceedings Fifth Workshop on Uncertainty in AI*, Windsor, Ont. (1989).
- [3] K.C. Chang and R. Fung, Node aggregation for distributed inference in bayesian networks, in: *Proceedings IJCAI-89*, Detroit, MI (1989).
- [4] R.M. Chavez, Hypermedia and randomized algorithms for probabilistic expert systems, Ph.D. Dissertation, Computer Science Department, Stanford University, Stanford, CA (1990).
- [5] R.M. Chavez and G.F. Cooper, A randomized approximation algorithm for probabilistic inference, *Networks* (to appear).
- [6] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, *Artif. Intell.* **42** (1990) 393–405.
- [7] S. Even, *Graph Algorithms* (Computer Science Press, Potomac, MD, 1979).
- [8] R. Fung and K.C. Chang, Weighing and integrating evidence for stochastic simulation in bayesian networks, in: *Proceedings Fifth Workshop on Uncertainty in AI*, Windsor, Ont. (1989).
- [9] D. Geiger, T. Verma and J. Pearl, *d*-separation: from theorems to algorithms, in: *Proceedings Fifth Workshop on Uncertainty in AI*, Windsor, Ont. (1989).
- [10] D. Geiger, T. Verma, and J. Pearl, Identifying independence in Bayesian networks, *Networks* (1990).
- [11] M. Henrion, Propagating uncertainty in bayesian networks by probabilistic logic sampling, in: J.F. Lemmer and L.N. Kanal, eds., *Uncertainty in Artificial Intelligence 2* (North-Holland, Amsterdam, 1988).
- [12] E.J. Horvitz, J.S. Breese and M. Henrion, Decision theory in expert systems and artificial intelligence, *Int. J. Approx. Reasoning* **2** (1988) 247–302.
- [13] F.V. Jensen, S.K. Andersen, U. Kjaerulff and S. Andreassen, A causal network prototype in the domain of electromyography—an implementation of coherent probabilistic methods, Research Rept. R-87-8, Institute of Electronic Systems, Aalborg University, Aalborg, Denmark (1987).
- [14] F.V. Jensen, S.L. Lauritzen and K.G. Olesen, Bayesian updating in recursive graphical models by local computations, *Comput. Stat. Q.* (1990).
- [15] F.V. Jensen, K.G. Olesen and S.K. Andersen, An algebra of Bayesian belief universes for knowledge based systems, *Networks* (1990).
- [16] J.H. Kim and J. Pearl, A computational model for causal and diagnostic reasoning in inference engines, in: *Proceedings IJCAI-83*, Karlsruhe, FRG (1983).

- [17] S.L. Lauritzen, A.P. Dawid, B.N. Larsen and H.-G. Leimer, Independence properties of directed markov fields, *Networks* (1990).
- [18] S.L. Lauritzen and D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, *J.R. Statist. Soc. B* **50** (2) (1988) 157–224.
- [19] J. Pearl, Fusion, propagation and structuring in belief networks, *Artif. Intell.* **29** (1986) 241–288.
- [20] J. Pearl, A constraint-propagation approach to probabilistic reasoning, in: L.N. Kanal and J.F. Lemmer, eds., *Uncertainty in Artificial Intelligence* (North-Holland, Amsterdam, 1986).
- [21] J. Pearl, Evidential reasoning using stochastic simulation of causal models, *Artif. Intell.* **32** (1987) 245–257.
- [22] J. Pearl, *Probabilistic Reasoning in Intelligent Systems* (Morgan Kaufmann, San Mateo, CA, 1988).
- [23] R.D. Shachter, Probabilistic inference and influence diagrams, *Oper. Res.* **36** (1988) 589–604.
- [24] R.D. Shachter, Evidence absorption and propagation through evidence reversals, in: *Proceedings Fifth Workshop on Uncertainty in AI*, Windsor, Ont. (1989).
- [25] R.D. Shachter, An ordered examination of influence diagrams, *Networks* (1990).
- [26] R.D. Shachter and M.A. Peot, Simulation approaches to general probabilistic inference on belief networks, in: *Proceedings Fifth Workshop on Uncertainty in AI*, Windsor, Ont. (1989).
- [27] R.D. Shachter and M.A. Peot, Evidential reasoning using likelihood weighting, *Artif. Intell.* (submitted).
- [28] G. Shafer and P.P. Shenoy, Probability propagation, *Ann. Math. Artif. Intell.* (1990).
- [29] H.J. Suermondt and G.F. Cooper, Updating probabilities in multiply connected belief networks, in: *Proceedings Fourth Workshop on Uncertainty in AI*, Minneapolis, MN (1988).
- [30] H.J. Suermondt and G.F. Cooper, Probabilistic inference in multiply connected belief networks using loop cutsets, Knowledge Systems Laboratory Rept. No. KSL-89-17, Stanford University, Stanford, CA (1989).
- [31] T. Verma and J. Pearl, Causal networks: semantics and expressiveness, in: *Proceedings Fourth Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, MN (1988) 352–359.