Symbolic Probabilistic Inference in Belief Networks

Ross D. Shachter, Engineering-Economic Systems Dept. Stanford University Stanford, CA 94305-4025 shachter@sumex-aim.stanford.edu

Bruce D'Ambrosio, and Department of Computer Science Oregon State University Corvallis, OR 97331-3902 dambrosio@cs.orst.edu

Brendan A. Del Favero Engineering-Economic Systems Dept. Stanford University Stanford, CA 94305-4025 a.apollo@macbeth.stanford.edu

Abstract

The Symbolic Probabilistic Inference (SPI) Algorithm [D'Ambrosio, 1989] provides an efficient framework for resolving general queries on a belief network. It applies the concept of dependency-directed backward search to probabilistic inference, and is incremental with respect to both queries and observations. Unlike most belief network algorithms, SPI is goal directed, performing only those calculations that are required to respond to queries. The directed graph of the underlying belief network is used to develop a tree structure for recursive query processing. This allows effective caching of intermediate results and significant opportunities for parallel computation. A simple preprocessing step ensures that, given the search tree, the algorithm will include no unnecessary distributions. The preprocessing step eliminates dimensions from the intermediate results and prunes the search path.

1. Introduction

Belief networks, directed graphical structures representing the probabilistic dependency among a set of variables, are an increasingly popular knowledge representation for uncertain reasoning. Much of their success is due to a growing body of methods for evaluating queries and performing probabilistic inference. The most popular methods [Jensen et al., 1990; Kim and Pearl, 1983; Lauritzen and Spiegelhalter, 1988; Pearl, 1986] gain much of their performance by efficient precomputation of simple queries in response to new observations.

The Symbolic Probabilistic Inference Algorithm (SPI) [D'Ambrosio, 1989; D'Ambrosio and Shachter, 1990], on the other hand, is a goal-driven method, which can respond to arbitrary conditional or conjunctive queries. SPI is incremental with respect to both queries and observations. It uses the structural information in the belief network graph to construct a search tree of efficiently evaluable factored symbolic expressions, which allows parallel computation and caching of intermediate results. By incorporating an efficient preprocessing step and recognizing when we do not need a full joint distribution, it can achieve additional savings in search and computation.

Section 2 presents an overview of SPI and the key concepts which underlie it, while Section 3 is a formal presentation of the framework and proofs of the correctness of SPI. The details of the algorithm are presented in Section 4, and conclusions and extensions are in Section 5.

2. Overview

SPI reorganizes the nodes in a belief network into a tree structure for query processing by a procedure in which every node is visited at most twice. Queries are directed to the root of the tree, which in turn generates queries for its subtrees and so forth until the response to a particular query can be determined and returned to the next higher level. Once a node has responses from all of its subtrees it can compute its own response. This process continues until the root of the tree returns a response for the original query. The performance of SPI depends critically on the organization of this search tree.

There is a condition on the construction of the search tree: if there is an arc between two nodes in the original belief network, then one node must be the root of a subtree containing the other. In other words, there can be no arcs in the belief net between nodes that are in parallel subtrees. Conversely, whenever two nodes are "separated" by higher root nodes, they can and should be placed in different subtrees. (Two nodes are said to be <u>separated</u> by a set if every undirected path between the nodes contains an element from the set.)

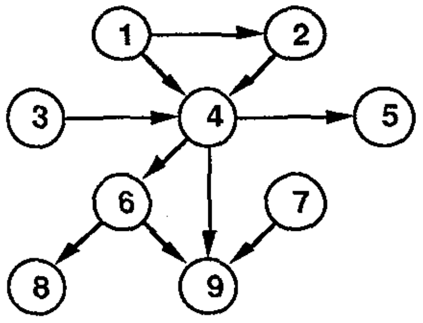
Figure 1. Example belief network.

For example, consider the belief network shown in Figure 1, containing nodes numbered 1 through 9. Any node can be chosen to be the root of a search tree, but node 4 seems like a promising choice, since it separates the other nodes into four subtrees, {1, 2}, {3}, {5}, and {6, 7, 8, 9}. A possible search tree for this network is shown in Figure 2a. Some other search trees for the network are drawn in Figure 2b and 2c. There are many different configurations for the subtree {6, 7, 8, 9} in a tree rooted by node 4. Several of these are shown in Figures 2d, 2e, 2f, and 2g. Note that in the tree drawn in Figure 2e, node 7 roots one of the subtrees for node 6. There is no requirement that a subtree root, such as node 7, have an arc in the belef network to one of its higher level roots. The only requirement is that, since there is an arc (7, 9) in the belief net, node 9 must either be higher or lower than node 7 in the tree.

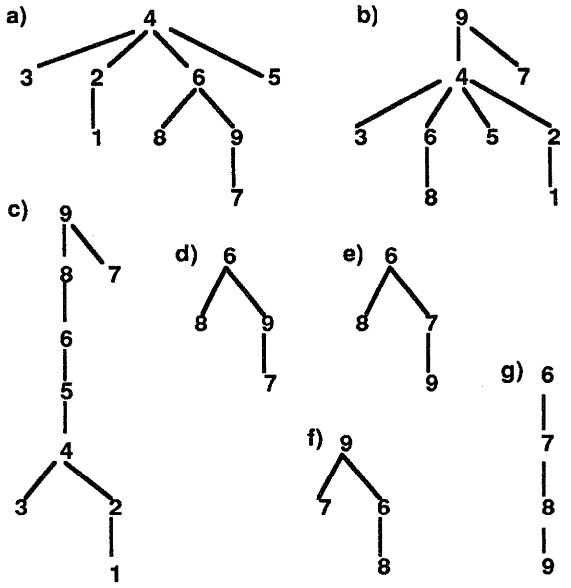

Figure 2. Possible search trees for the network in Fig. 1.

In general, there are many alternate tree structures possible for a given belief network. We would like to choose a structure that facilitates optimal performance for the anticipated queries. Since the processing in parallel subtrees is independent, it can be performed in parallel and using independent information. This suggests that a good heuristic might be to branch wherever possible and as high as possible in the tree. One rule of thumb might be to select that root node for which the largest subtree is as small as possible. In this respect the tree drawn in Figure 2a is better than the ones in 2b and 2c.

We can illustrate the basic concepts underlying SPI with a few simple examples. Corresponding to each node j in the belief network is a conditional probability distribution π_j for the variable X_j, conditioned on j's parents. The joint probability distribution for all of the variables is obtained in factored form by multiplying their conditional distributions,

$$P\{X_1=x_1, \dots, X_9=x_9\}$$
$$= P\{X_1=x_1\}\, P\{X_2=x_2 \mid X_1=x_1\} \dots P\{X_9=x_9 \mid X_{4,6,7}=x_{4,6,7}\}$$
$$= \pi_1(x_1)\, \pi_2(x_{1,2}) \dots \pi_9(x_{4,6,7,9}),$$

where $x_{4,6,7,9}$ is a vector whose components are symbolically matched to the proper dimensions.

Suppose that we want to find $P\{X_1\}$. Of course, this is simply stored in π_1, but we can find it by using the more general method of summing over the joint distribution,

$$P\{X_1=x_1\} = \Sigma_{x_{2,\dots,9}} P\{X_1=x_1, \dots, X_9=x_9\}$$
$$= \Sigma_{x_{2,\dots,9}} \pi_1(x_1)\, \pi_2(x_{1,2}) \dots \pi_9(x_{4,6,7,9}).$$

Now we can recognize that π_1 does not vary in the summation, and thus it can be brought out of the sum. Using the property that conditional distributions sum to 1, we can eliminate each of the distributions in turn to obtain

$$P\{X_1=x_1\}$$
$$= \pi_1(x_1)\, \Sigma_{x_{2,\dots,9}} \pi_2(x_{1,2}) \dots \pi_9(x_{4,6,7,9})$$
$$= \pi_1(x_1).$$

In a similar way we can find $P\{X_1 \mid X_2\}$, which can easily be computed[1] from $P\{X_1, X_2\}$,

$$P\{X_1=x_1, X_2=x_2\}$$
$$= \Sigma_{x_{3,\dots,9}} P\{X_1=x_1, \dots, X_9=x_9\}$$
$$= \pi_1(x_1)\, \pi_2(x_{1,2}).$$

This approach can also be applied to more complex problems, such as $P\{X_5 \mid X_2\}$,

$$P\{X_{5,2}=x_{5,2}\}$$
$$= \Sigma_{x_{1,3,4,6,7,8,9}} P\{X_1=x_1, \dots, X_9=x_9\}$$
$$= \Sigma_{x_{1,3,4,6,7,8,9}} \pi_1(x_1)\, \pi_2(x_{1,2})\, \pi_9(x_{4,6,7,9})$$
$$= \Sigma_{x_{1,3,4}} \pi_1(x_1)\, \pi_2(x_{1,2})\, \pi_3(x_3)\, \pi_4(x_{1,2,3,4})\, \pi_5(x_{4,5}).$$

We are unable to simplify this final expression any more by distributing terms. To compute any further, we must use the actual numbers. Nonetheless, it does still pay to order the terms so as to minimize the number of calculations and the size of intermediate results [D'Ambrosio and Shachter, 1990].

Suppose that X_3 has been observed with value x^*_3. The solution for $P\{X_5, X_2 \mid X_3 = x^*_3\}$ is similar to the one above, except that the specific value for x_3 can be substituted into all of the distributions and it should not be summed,

$$P\{X_{5,2}=x_{5,2} \mid X_3=x^*_3\}$$
$$= \Sigma_{x_{1,4}} \pi_1(x_1)\, \pi_2(x_{1,2})\, \pi_3(x^*_3)\, \pi_4(x_{1,2,4}, x^*_3)\, \pi_5(x_{4,5}).$$

Finally, suppose that we want to find $P\{X_5 \mid X_4\}$. We could easily obtain this from $P\{X_5, X_4\}$, but we can save work if we recognize the difference between the two

[1]A conditional probability is defined as $P\{X_1=x_1 \mid X_2=x_2\} = P\{X_1=x_1, X_2=x_2\} \,/\, \Sigma_{y_1} P\{X_1=y_1, X_2=x_2\}$.

queries. The former is simply the distribution, $P\{ X_5=x_5 \mid X_4=x_4 \} = \pi_5(x_{4,5})$, while the latter requires multiplication and summation. In general, we can recognize cases in which we can obtain the conditional more easily that the joint distribution.

The operations shown in these examples correspond exactly to those in SPI. There are three basic operators: product of distributions, summation of a distribution over a set of variables, and substitution of an observed value into a distribution. We can apply these operators both symbolically and numerically. Symbolic operations help us recognize when we can apply the distributive law to pull a factor out of a summation. On the other hand, we must sooner or later evaluate to numbers. If we maintain a cache of numerical results, we can avoid repeating their calculation.

3. Framework and Notation

In this section, we provide a formal description of the framework for the algorithm, and prove the fundamental results underlying it.

We assume that we are given a fully specified belief network, which contains a directed acyclic graph on a set of nodes N. Each node j corresponds to a random variable X_j, which can take on a finite number of possible values $x_j \in \Omega_j$ with conditional probability distribution π_j. As a convention, a lower case letter represents a single node while an upper case letter represents a set of nodes, so that X_J denotes the vector of variables indexed by the set J. Therefore, the conditional distribution π_j for node j can be expressed in terms of its parents or conditional predecessors C(j),

$$\pi_j (x_{j \cup C(j)}) = P\{ X_j = x_j \mid X_{C(j)} = x_{C(j)} \}.$$

If the node j has no parents, $C(j) = \emptyset$ and π_j is an unconditional probability distribution. Distribution π_j has dimensions D(j),

$$D(j) = j \cup C(j),$$

and we can think of the distribution as a nonnegative function, π_j: $\Omega_{D(j)} \rightarrow \boldsymbol{R}$. We can extend the definition of node conditional distributions and their dimensions to apply to sets so that

$$D(J) = \cup_{j \in J} D(j),$$

and

$$\pi_J (x_{D(J)}) = \Pi_{j \in J} \ \pi_j(x_{D(j)}).$$

We say that there is an undirected path or chain between nodes i and j if we can get from node i to node j in the network along arcs ignoring their direction. When there is a chain between two nodes, they are said to be connected; otherwise they are disconnected. Clearly, if node i is connected to node j, then node i will be connected to node k if and only if j is connected to k. We can therefore identify the maximal connected sets of nodes, called components. If there is some set S such that every chain between nodes i and j contains a node from S, S is said to separate i and j. Any set of nodes disjoint from S can be partitioned into maximal sets which are the components separated by S.

A structure on a set of nodes is said to be a tree if exactly one of the nodes is identified as its root and the others are partitioned into subtrees, which are themselves trees. In SPI, all of the nodes are organized into a search tree. For each node i, let T(i) be the nodes in the subtree rooted at node i and let S(i) be the roots of all subtrees containing i. The components of $T(i) \setminus \{i\}$ [2] separated by S(i) form the subtrees of T(i), and the roots of those subtrees are given by R(i). For example, in the tree drawn in Figure 2a, T(6) = {6, 8, 9, 7 }, S(6) = { 4, 6 }, and R(6) = { 8, 9 }. If there are separate components in the original belief network, one can create a "fictitious node" 0 with no arcs, so $D(0) = \emptyset$ and $\pi_0 = 1$. Each of the components is then a subtree in T(0).

As stated earlier, the search tree has one restriction on its organization. If there is an arc between nodes i and j in the belief network then either $i \in S(j)$ or $j \in S(i)$. This is enforced automatically by including all (conditionally) connected nodes in the same subtree. There is still tremendous flexibility in the construction of the search tree, since the choice of root node for any tree is arbitrary.

Generalized Distributions

A nonnegative function Q: $\Omega_{J \cup K} \rightarrow \boldsymbol{R}$ will be called a (generalized) distribution for J given K if it can be normalized to compute $P\{ X_J \mid X_K \}$, namely,

$$P\{ X_J = x_J \mid X_K = x_K \} = Q(x_J, x_K) / \Sigma_{y_J \in \Omega_J} Q(y_J, x_K) .$$

Although the generalized distribution contains sufficient information to compute the conditional distribution, it might not have enough to compute the joint distribution $P\{ X_{J \cup K} \}$. We define three operators on these distributions: product, summation, and substitution. Given distributions Q_1 and Q_2 on sets J_1 and J_2, the conformal product of the distributions is given by

$Q = \langle *, Q_1, Q_2 \rangle$

where $Q\{ x_{J_1 \cup J_2} \} = Q_1\{ x_{J_1} \} Q_2\{ x_{J_2} \}$.

The summation over dimensions K is $Q = \langle \Sigma, K, Q_1 \rangle$ where $Q\{ x_{J \setminus K} \} = \Sigma_{x_K \in \Omega_K} Q_1(x_J)$.

Lastly, the substitution of observation x^*_i in dimension i is $Q = \langle \downarrow, i, x^*_i, Q_1 \rangle$

where $Q(x_{J_1 \setminus \{i\}}) = Q_1\{ X_i = x^*_i, X_{J_1 \setminus \{i\}} = x_{J_1 \setminus \{i\}} \}$.

Note that the product increases the dimension of the resulting distribution, while the summation and substitution decrease it. Processing time is proportional to the size of the distributions, which are exponential in the number of dimensions, so it is advantageous to postpone performing the product operation as long as possible, and instead to perform summation and substitution as soon as possible.

We can represent a distribution either by the actual

[2]The symbol "\" is set subtraction, $A \setminus B = \{ j \in A : j \notin B \}$.

function, or by a symbolic expression of operators applied to distributions. Symbolic manipulation to the expression allows us to reorder terms to reduce computation time. Of course, at any time, the expression can be evaluated to obtain the numerical distribution. The key to reordering operations is given by the following lemma, which just states the distributive law for addition and multiplication of real numbers in terms of the notation for SPI.

Lemma 1. Distributive Law for Distributions

Given distributions Q_1 and Q_2 on sets J_1 and J_2

$$\langle \Sigma, K, \langle *, Q_1, Q_2 \rangle \rangle = \langle \Sigma, K \cap J_1, \langle *, Q_1, \langle \Sigma, K \cap (J_2 \setminus J_1), Q_2 \rangle \rangle.$$

The lemma states that instead of summing after multiplying two distributions, we can sum one distribution over dimensions for which the other distribution does not vary before multiplying the distributions. Since this reduces the number of dimensions processed in the outer sum, it can lead to substantial savings. We will try to exploit this property whenever possible in evaluating a query. Consider the search tree shown in Figure 3, in which A, B, and C are now sets of nodes. This represents the general case in SPI, in which B is a subtree, C is the union of B's sibling subtrees, if any, and A is the remainder of the nodes N in the network. If we want to find $P\{ X_M \}$ for some subset M of N, then the distributive law can be applied to this search tree, by the following theorem.

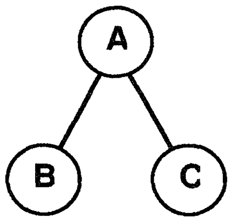

Figure 3. General Network for the Proof of SPI.

Theorem 1.

Given an arbitrary subset M of N, and a partition of N into sets A, B, and C, such that B and C are separated by A, then the generalized distribution $Q(x_M) = P\{X_M = x_M\}$ is

$$Q = \langle \Sigma, M' \setminus M, \langle *, \pi_A, \langle *, \langle \Sigma, B \setminus M', \pi_B \rangle, \langle \Sigma, C \setminus M', \pi_C \rangle \rangle \rangle \rangle,$$

where $M' = M \cup D(A)$.

Proof:

Q can be obtained by summing all variables except M from the joint distribution, which is just the product of the node distributions,

$$Q = \langle \Sigma, N \setminus M, P\{ X_N \} \rangle = \langle \Sigma, N \setminus M, \langle *, \pi_A, \langle *, \pi_B, \pi_C \rangle \rangle \rangle.$$

Since dimensions D(A) appear in distributions π_A we cannot pull π_A out of the sum for D(A). Nonetheless, the construction of the search tree ensures that $D(B) \cap D(C) \subseteq D(A)$, so three applications of Lemma 1 yield

$$Q = \langle \Sigma, M' \setminus M, \langle *, \pi_A, \langle \Sigma, \emptyset, \langle *, \langle \Sigma, B \setminus M', \pi_B \rangle, \langle \Sigma, C \setminus M', \pi_C \rangle \rangle \rangle \rangle \rangle.$$

In general, the response to a query is a generalized distribution and not necessarily a full joint distribution. Suppose that the desired distribution has dimensions M and is obtained by multiplying the distributions in nodes L, where $D(L) = M \cup L$. The resulting distribution, $P\{ X_{M \cap L} \mid X_{M \setminus L} \}$, is then obtained by summing over $L \setminus M$. For example, Theorem 1 is the special case for which L = N. The search tree in Figure 3 can now be applied to the conditional case.

Theorem 2.

Given arbitrary subsets M and L of N such that $D(L) = M \cup L$, and a partition of N into sets A, B, and C, such that B and C are separated by A, then the generalized distribution $Q(x_M) = P\{ X_{M \cap L} = x_{M \cap L} \mid X_{M \setminus L} = x_{M \setminus L} \}$ is

$$Q = \langle \Sigma, M' \setminus M, \langle *, \pi_{A \cap L}, \langle *, \langle \Sigma, B \cap L \setminus M', \pi_{B \cap L} \rangle, \langle \Sigma, C \cap L \setminus M', \pi_{C \cap L} \rangle \rangle \rangle \rangle,$$

where $M' = L \cap [M \cup D(A \cap L)]$.

Proof:

Because $D(L) = M \cup L$, summation and product yield

$$Q = \langle \Sigma, L \setminus M, \langle *, \pi_{A \cap L}, \langle *, \pi_{B \cap L}, \pi_{C \cap L} \rangle \rangle \rangle$$
$$= \langle \Sigma, M' \setminus M, \langle *, \pi_{A \cap L}, \Sigma, \emptyset, \langle *, \langle \Sigma, B \cap L \setminus M', \pi_{B \cap L} \rangle, \langle \Sigma, C \cap L \setminus M', \pi_{C \cap L} \rangle \rangle \rangle \rangle,$$

by the same logic as Theorem 1.

Incorporating Evidence

Evidence is entered in the system in the form of exact observations of the values of variables. The set of variables which have been observed is denoted by E. Suppose that X_i has been observed with the value x^*_i. The substitution operator $\langle \downarrow, i, x^*_i, Q \rangle$ can now be applied to any distribution Q to incorporate that evidence into the distribution. There is no longer any need to sum out dimension i and, in fact, it would be an error to do so. Because dimension i can no longer be summed over and the substitution operator eliminates all instances of it, it can be freely distributed within all products, down to the node distributions themselves. This substitution at each node distribution can be performed at every computation, but it is much simpler to substitute for dimension i in every distribution in which it appears whenever its observation is reported. Those distributions in which dimension i appears belong to node i and its children. Therefore, the node distributions π_j for those nodes should be modified by

$$\pi_j(x_{D(j) \setminus E}) \leftarrow \pi_j(x_{D(j) \setminus E}, x^*_i) = \langle \downarrow, i, x^*_i, \pi_j \rangle .$$

Once the evidence has been incorporated into the node distributions, all of our earlier results can be applied, using the new dimensions of the distributions. For example, suppose that a desired distribution has dimensions M and is obtained by multiplying the distributions in nodes L, where $D(L) = M \cup L \cup E$ and $M \cap E = \emptyset$. This is a straightforward extension of Theorem 2.

Corollary 1.

Given arbitrary subsets M and L of N such that $D(L) = M \cup L \cup E$ and $M \cap E = \emptyset$, and a partition of N into sets A, B, and C, such that B and C are separated by A, then the generalized distribution

$$\begin{aligned} &Q(x_M) \\ &= P\{ X_{M\cap L} = x_{M\cap L}, X_{E\cap L} = x^*_{E\cap L} \mid X_{M\setminus L} = x_{M\setminus L}, X_{E\setminus L} = x^*_{E\setminus L} \} \text{ is} \\ &Q = \langle \Sigma, M' \setminus M, \langle *, \pi_{A\cap L}, \langle *, \langle \Sigma, B \cap L \setminus (E \cup M'), \pi_{B\cap L} \rangle, \langle \Sigma, C \cap L \setminus (E \cup M'), \pi_{C\cap L} \rangle \rangle \rangle \rangle, \end{aligned}$$

where $M' = (L \setminus E) \cap [M \cup D(A\cap L)]$.

4. The Symbolic Probabilistic Inference Algorithm

We can now present a complete description of SPI in terms of the framework developed in the previous section, given a belief network and a search tree.

The general form of query received by SPI is of the form $P\{ X_J \mid X_K, X_E = x^*_E \}$, where x^*_E are the recorded observations. This query is transformed and sent to the highest root node in the search tree. When that node has obtained a response to all of its queries from its subtrees, it returns a generalized distribution for J given K, which can then be normalized to the desired result.

The query actually sent to the highest root node consists of the set of node distributions L needed to respond to the query and the dimensions M of the desired response. The sets L and M can be computed by an algorithm which runs in time linear in the number of nodes and arcs in the belief network graph [Geiger et al., 1989; Geiger et al., 1990; Shachter, 1988; Shachter, 1990]. This algorithm operates on a copy of the graph: after deleting all outgoing arcs from $K \cup E$ and deleting all nodes which are neither in nor ancestors of $J \cup K \cup E$, the set L is those nodes connected to J and $M = (J \cup K) \cap D(L)$.[3] These sets satisfy $D(L) = M \cup L \cup E$ and $M \cap E = \emptyset$. The formulae are efficient and simple to implement. We refer the reader to the above-cited literature for more explanation.

The response from the search tree will be a generalized distribution Q for J given K satisfying

$$P\{ X_J = x_J \mid X_K = x_K, X_E = x^*_E \} = Q(x_M) / \Sigma_{y_J \in \Omega_J} Q(y_J, x_{M\setminus J}).$$

The heart of the SPI algorithm can now be described. At any node i, a request arrives for a distribution to be obtained by multiplying the distributions in nodes L and summing over dimensions $L \setminus (E \cup M)$, which must therefore have dimensions $W = M \cup [D(L) \setminus (L \cup E)]$. The node's response is a generalized distribution Q for $W \cap L$ given $W \setminus L$,

$$Q(x_W) = P\{ X_{W\cap L} = x_{W\cap L}, X_{E\cap L} = x^*_{E\cap L} \mid X_{W\setminus L} = x_{W\setminus L}, X_{E\setminus L} = x^*_{E\setminus L} \}.$$

Of course, if such a distribution had been computed earlier and cached, it could be returned immediately. Usually, however, it will be necessary to send requests to the subtrees below this node in order to compute the response.

The recursive step of the SPI request is shown in Figure 4. The first three **if** statements check whether there is a cached result and whether the main processing block can be avoided by recognizing two important special cases. The main processing block in general has been verfied by Corollary 1.

```
algorithm Request ( i, L, M )
begin
    if ( L, M ) is cached at this node
    then $Q_i \leftarrow$ cached result
    else if $L = \emptyset$
         then $Q_i \leftarrow 1$
    else if $L = \{i\}$
         then $Q_i \leftarrow \pi_i$
    else begin main processing block
       if $i \in L$
       then $M' \leftarrow (L \setminus E) \cap [M \cup D(i)]$
       else $M' \leftarrow M$ ;
       $Q_i \leftarrow 1$ ;
       for $r \in R(i)$
           $Q_i \leftarrow \langle *, Q_i,$ Request $(r, L\cap T(r), M'\cap T(r)) \rangle$ ;
       if $i \in L$
       then $Q_i \leftarrow \langle \Sigma, M' \setminus M, \langle *, \pi_i, Q_i \rangle \rangle$ ;
       if caching of result is desired
       then cache result $Q_i$ for (L, M) ;
    end main processing block ;
    return $Q_i$ ;
end ;
```

*Figure 4. The recursive **request** procedure for each node in the SPI algorithm.*

The query nature of SPI is designed to allow caching of responses at each node. Whenever an observation is received for node i, and the substitution operation performed on the distribution of node i and its children, all caches above node i and its children in the search tree become suspect, and should be removed from the caches. Alternatively, a check similar to the algorithm for determining the original M set could be performed to recognize for which cached expressions the new evidence is relevant, and only those expressions need be removed from the cache [Shachter, 1990].

The cache management scheme can exploit the generalized distributions being returned. Suppose that the

[3]We assume without loss of generality that E, K, and J are disjoint. If they are not, then the response can be computed using E, $K\setminus E$, and $J\setminus(K\cup E)$, with an additional check for consistency among the overlapping values.

distribution Q_c, the response to a request (L_c, M_c), has been cached. Q_c can serve as the response to a subsequent request (L, M) with the same dimensions, provided that $L \subseteq L_c$, since in that case Q_c has "too much" information. On the other hand, if $L \supset L_c$, then a new response must be computed, and it might as well replace Q_c in the cache.

Examples

Suppose that SPI were applied to the query P{ $X_5 \mid X_2$ } for the belief network in Figure 1 using the search tree in Figure 2a. The following results would be obtained:

$J \leftarrow \{5\}, K \leftarrow \{2\}, E \leftarrow \emptyset$;

$Q \leftarrow$ Request(4, {1,2,3,4,5}, {2,5}) ;

$Q_4 \leftarrow \langle \Sigma, \{1,3,4\}, \langle *, \pi_4,$ Request(2, {1,2}, {1,2}),
Request(3, {3}, {3}),
Request(5, {5}, {5}),
Request(6, $\emptyset$, $\emptyset$) $\rangle\rangle$;

$Q_2 \leftarrow \langle \Sigma, \emptyset, \langle *, \pi_2,$ Request(1, {1}, {1}) $\rangle\rangle$;

$Q_1 \leftarrow \pi_1$; $Q_3 \leftarrow \pi_3$; $Q_5 \leftarrow \pi_5$; and $Q_6 \leftarrow 1$.

Now, suppose that observations were made of $X_6 = x^*_6$ and $X_9 = x^*_9$. The caches for nodes 8, 9, 6, and 4 would be invalidated and $E \leftarrow \{6,9\}$;

$\pi_6 \leftarrow \langle \downarrow, 6, x^*_6, \pi_6 \rangle$; $\pi_8 \leftarrow \langle \downarrow, 6, x^*_6, \pi_8 \rangle$; and

$\pi_9 \leftarrow \langle \downarrow, 9, x^*_9, \pi_9 \rangle$.

In response to query P{ $X_2 \mid X_6 = x^*_6, X_9 = x^*_9$ }:

$J \leftarrow \{2\}, K \leftarrow \emptyset$;

$Q \leftarrow$ Request(4, {1,2,3,4,6,7,9}, {1}) ;

$Q_4 \leftarrow \langle \Sigma, \{1,3,4\}, \langle *, \pi_4,$ Request(2, {1,2}, {1,2}),
Request(3, {3}, {3}),
Request(5, $\emptyset$, $\emptyset$),
Request(6, {6,7,9}, $\emptyset$) $\rangle\rangle$;

$Q_2 \leftarrow$ cached result from above ;

$Q_6 \leftarrow \langle \Sigma, \emptyset, \langle *, \pi_6,$ Request(8, $\emptyset$, $\emptyset$),
Request(9, {7,9}, {$\emptyset$}) $\rangle\rangle$;

$Q_9 \leftarrow \langle \Sigma, \{7\}, \langle *, \pi_9,$ Request(7, {7}, {7}) $\rangle\rangle$;

$Q_3 \leftarrow \pi_3$; $Q_5 \leftarrow 1$, $Q_7 \leftarrow \pi_7$; and $Q_8 \leftarrow 1$;

5. Conclusions and Extensions

In this paper, we have presented the Symbolic Probabilistic Inference Algorithm along with a proof of its correctness. SPI is a goal-driven rather than data-driven algorithm, which performs a variation of dependency-directed backward search in response to arbitrary conditional queries.

There are many ways that SPI can be refined for practical implementation. The most significant issue is the construction of the search tree. A promising approach appears to be recursive decomposition, in which the search tree is made as balanced and shallow as possible [Cooper, 1990; Fiduccia and Mattheyses, 1982]. This not only allows for parallel processing along the independent branches, but means that observations will invalidate the fewest possible caches.

Given a search tree, there are still difficult (NP-hard) decisions to made, since there can be significant benefit to postponing evaluation of an expression, maintaining it in symbolic rather than numeric form. For example, if one factor in a distribution product shares none of the dimensions being summed over, then that factor can be pulled out of the sum as in Lemma 1. It might also be worthwhile to reorder the factors in the product before performing the summation [D'Ambrosio, 1989; D'Ambrosio and Shachter, 1990] .

There are additional interesting tradeoffs between symbolic and numeric computations of intermediate results. By evaluating expressions completely at nodes, caches can be created to prevent repeated computations. Sometimes, however, postponing those evaluations can lead to improved factoring opportunities. Finally, the search tree can be constructed dynamically in response to queries, yielding the most efficient search structure, but eliminating caching opportunities. Much needs to be learned about the relative benefits of such customized query-processing versus maintaining the accumulated information.

6. Acknowledgements

We benefited greatly from the comments of Greg Cooper, Mike Fehling, Eric Horvitz, Perry McCarty, Ramesh Patil, and Mike Wellman.

7. References

Cooper, G. G. (1990). Bayesian belief-network inference using recursive decomposition (KSL 90-05). Knowledge Systems Laboratory, Stanford University.

D'Ambrosio, B. (1989). Symbolic probabilistic inference in belief nets . Department of Computer Science, Oregon State University.

D'Ambrosio, B. and Shachter, R. D. (1990). Factoring Heuristics for Generalized SPI. Conference on Uncertainty in Artificial Intelligence, Boston, submitted.

Fiduccia, C. M. and Mattheyses, R. M. (1982). A linear-time heuristic for improving network reliability. Nineteenth Design Automation Conference, Las Vegas.

Geiger, D., Verma, T., and Pearl, J. (1989). d-separation: from theorems to algorithms. Fifth Workshop on Uncertainty in Artificial Intelligence, University of Windsor, Ontario, 118-125.

Geiger, D., Verma, T., and Pearl., J. (1990). Identifying independence in Bayesian networks. *Networks*, to appear.

Jensen, F. V., Olesen, K. G., and Andersen, S. K. (1990). An algebra of Bayesian belief universes for knowledge based systems. *Networks*, to appear.

Kim, J. H. and Pearl, J. (1983). A computational model for causal and diagnostic reasoning in inference engines. 8th International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statist. Soc. B*, 50(2), 157-224.

Pearl, J. (1986). Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29(3), 241-288.

Shachter, R. D. (1988). Probabilistic Inference and Influence Diagrams. *Operations Research*, 36(July-August), 589-605.

Shachter, R. D. (1990). An Ordered Examination of Influence Diagrams. *Networks*, to appear.