A simple approach to Bayesian network computations

Nevin Lianwen Zhang
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

David Poole
Department of Computer Science
University of British Columbia
Vancouver, B.C., V6T 1Z2, Canada

Abstract

The general problem of computing posterior probabilities in Bayesian networks is NP-hard (Cooper 1990). However efficient algorithms are often possible for particular applications by exploiting problem structures. It is well understood that the key to the materialization of such a possibility is to make use of conditional independence and work with factorizations of joint probabilities rather than joint probabilities themselves. Different exact approaches can be characterized in terms of their choices of factorizations. We propose a new approach which adopts a straightforward way for factorizing joint probabilities. In comparison with the clique tree propagation approach, our approach is very simple. It allows the pruning of irrelevant variables, it accommodates changes to the knowledge base more easily. it is easier to implement. More importantly, it can be adapted to utilize both intercausal independence and conditional independence in one uniform framework. On the other hand, clique tree propagation is better in terms of facilitating precomputations.

Keywords: reasoning under uncertainty, Bayesian networks, algorithm

1 Introduction

Several exact approaches to the computing of posterior probabilities in Bayesian networks have been proposed, studied, and some of them implemented. The one that is most well known is clique tree propagation, which has been developed over the past few years by Pearl (1988), Lauritzen and Spiegehalter (1988), Shafer and Shenoy (1988), and Jensen *et al* (1990). Other approaches include Shachter's arc reversal node reduction approach (Shachter 1988), symbolic probabilistic inference first proposed by D'Ambrosio (Shachter *et al* 1990), recursive decomposition by Cooper (1992), and component tree propagation by Zhang and Poole (1992).

This paper grew out of an attempt to understand those approaches and the relationships among them. We asked ourselves: are there any common principles that underlie all those approaches? If yes, what are the choices that render them different from one another? What are the advantages and disadvantages of these choices? Are there any better choices and/or any better combinations of choices?

Shachter *et al* (1992) has demonstrated the similarities among the various approaches. In this paper, we are more interested in the differences among them.

Cooper (1990) has proved that the general problem of computing posterior probabilities in Bayesian networks is NP-hard. In particular applications, however, it is often possible to compute them efficiently by exploiting the problem structures. The key technique that enables the materialization of such a possibility, as pointed out by Shafer and Shenoy (1988), is to work with factorizations of joint probabilities rather than the joint probabilities themselves. What all the exact approaches have in common is that they all adopt this technique, while they differ in their own choices of factorizations.

These understandings lead to a new approach that chooses a straightforward factorization for joint probabilities. Though very simple, the new approach has several advantages over clique tree propagation

in terms of pruning irrelevant variables, accommodating changes to the knowledge base and easiness of implementation. It also leads to a uniform framework for utilizing both conditional and intercausal independence. The only disadvantage we can think of is that it does not facilitate precomputation.

The organization of the paper is as follows. Preliminary definitions are given in section 2. Section 3 reviews results concerning the irrelevance of variables to a query. After the removal of irrelevant variables, queries about posterior probabilities can be transformed into a standard form, i.e queries about marginal probabilities. For technical convenience, further exposition will be carried out in terms of potentials rather the probabilities (section 4). In section 5, we illustrate the technique of working with factorizations of joint potentials. The subproblem of data management is identified in section 6. Clique tree propagation is one solution to this subproblem. A new and very simple solution is proposed in section 7, which is based on a simple way for factorizing joint potentials. In section 8, we compare the solution to clique tree propagation. Some conclusions are provided in section 9.

2 Preliminaries

We begin by giving a definition of Bayesian networks.

A *Bayesian network* $\mathcal{N}$ is a triplet $(V, A, \mathcal{P})$, where

1. V is a set of variables.
2. A is a set of arcs, which together with V constitutes a directed acyclic graph $G = (V, A)$.
3. $\mathcal{P} = \{P(v|\pi_v) : v \in V\}$, where π_v stands for the set of parents of v. In words, $\mathcal{P}$ is the set the conditional probabilities of the all variables given their respective parents [1].

Figure 1 show a simple Bayesian network **net1** with seven variables a, b, c, d, e, f, and g. The network contains the following prior and conditional probabilities: $P(a)$, $P(f|a)$, $P(b|a)$, $P(c|b)$, $P(d|b)$, $P(e|c,d)$, and $P(g|f,e)$.

Note that variables in a Bayesian network will be referred as nodes when they are viewed as members of the underlying graph. Also note that the graphical structure of a Bayesian network can be read from the set of the prior and conditional probabilities. So, we can use the symbol $\mathcal{N}$ to refer to the set of prior and conditional probabilities $\mathcal{P}$ without causing any confusion.

The *prior joint probability* $P_{\mathcal{N}}$ of a Bayesian network $\mathcal{N}$ is defined by

[1] Note that when v is a root, π_v is empty. In such a case, the expression $P(v|\pi_v)$ simply stands for the prior probability of v.

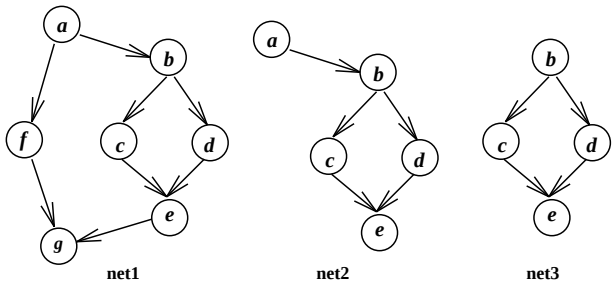

Figure 1: Bayesian network and irrelevant variables.

$$P_{\mathcal{N}}(V) = \prod_{v \in V} P(v|\pi_v). \quad (1)$$

For example, the prior joint probability P_{net1} of **net1** is given by

$$P_{net1}(a,b,c,d,e,f,g) = P(a)P(f|a)P(b|a)P(c|b)P(d|b)P(e|c,d)P(g|f,e).$$

For any subset X of V, the *marginal probability* $P_{\mathcal{N}}(X)$ is defined by

$$P_{\mathcal{N}}(X) = \sum_{V-X} P_{\mathcal{N}}(V).$$

Some variables may be observed to have specific values. For example, the variable b in **net1** may be observed to be a specific value b_0. Let $Y \subseteq V$ be the set of variables observed and Y_0 be the corresponding set of values. Let $X \subseteq V$ be the set of variables of interest. The *posterior probability* $P_{\mathcal{N}}(X|Y = Y_0)$ of X is defined by

$$P_{\mathcal{N}}(X|Y = Y_0) = \frac{P_{\mathcal{N}}(X, Y = Y_0)}{P_{\mathcal{N}}(Y = Y_0)}. \quad (2)$$

The problem of concern to this paper is how to compute $P_{\mathcal{N}}(X|Y = Y_0)$?

3 Irrelevant variables and standard queries

Given a query to a Bayesian network $\mathcal{N}$, it is often possible to graphically identify certain variables being irrelevant to the query. This issue is addressed in Geiger *et al* (1990), Lauritzen *et al* (1990), and Baker and Boult (1990). The materials in this section are extracted from those papers.

To *remove* a node v from a Bayesian network $\mathcal{N} = (V, A, \mathcal{P})$ is to: (1) remove v from V, (2) remove from A all the arcs that contain v, (3) remove from $\mathcal{P}$ all the items that involve v, and (4) set the prior probabilities

for all the nodes, if any, that become roots[2] because of the removal to be the uniform distribution.

A node in a Bayesian network $\mathcal{N}$ is *leaf* if it has no children. A node is *barren* w.r.t a query $P_{\mathcal{N}}(X|Y = Y_0)$, if it is a leaf and it is not in $X \cup Y$. In **net1**, g is barren w.r.t $P_{net1}(e|b = b_0)$.

Theorem 1 *Suppose $\mathcal{N}$ is a Bayesian network, and v is a leaf node. Let $\mathcal{N}'$ be the Bayesian network obtained from $\mathcal{N}$ by removing v. If v is barren w.r.t to the query $P_{\mathcal{N}}(X|Y = Y_0)$, then*

$$P_{\mathcal{N}}(X|Y = Y_0) = P_{\mathcal{N}'}(X|Y = Y_0). \quad (3)$$

Consider computing $P_{net1}(e|b = b_0)$. The node g is barren w.r.t the query and hence irrelevant. According to Theorem 1, g can be harmlessly removed. This creates a new barren node f. After the removal of g and f, **net1** becomes **net2**. Thus the query $P_{net1}(e|b = b_0)$ is reduced to the query $P_{net2}(e|b = b_0)$.

Let $An(X \cup Y)$ be the *ancestral set* of $X \cup Y$, i.e the set of nodes in $X \cup Y$ and of the ancestors of those nodes. By repeatedly applying Theorem 1, one can easily show that

Corollary 1 *All the nodes outside $An(X \cup Y)$ are irrelevant to the query $P(X|Y = Y_0)$.*

The *moral* graph $m(G)$ (Lauritzen and Spiegehalter 1988) of the an directed graph $G = (V, A)$ is the undirected graph obtained from G be marrying the parents of each node (i.e adding an edge between each pair of parents), and then dropping all directions. If two nodes x and y are separated by a set B in $m(G)$, we say that x and y are *m-separated* by B in G. The term m-separation is new, but the concept itself was used Lauritzen *et al* (1990).

Theorem 2 *Suppose $\mathcal{N} = (V, A, \mathcal{P})$ is a Bayesian network. Given a query $P_{\mathcal{N}}(X|Y = Y_0)$, let $\mathcal{N}'$ be the Bayesian network obtained from $\mathcal{N}$ by removing all the nodes that are m-separated from X by Y. Then*

$$P_{\mathcal{N}}(X|Y = Y_0) = P_{\mathcal{N}'}(X|Y = Y_0). \quad (4)$$

In our example, since a is m-separated from e by b in **net2**, the query can be further reduced to $P_{net3}(e|b = b_0)$ Note that a is not m-separated from e by b in **net1**.

It can be proved (Lauritzen *et al* 1990 and Geiger *et al* 1990) that, given a query, all the irrelevant nodes that are graphically recognizable can be recognized by applying those two theorems.

From equation (2), we see that $P_{\mathcal{N}}(X|Y = Y_0)$ can be obtained from $P_{\mathcal{N}}(X, Y = Y_0)$ by multiplying a renormalization constant. However, the two queries are different in terms of irrelevant variables. For example, a is irrelevant to the query $P_{net2}(e|b = b_0)$, but relevant to the query $P_{net2}(e, b = b_0)$.

From now on, we will assume that all the irrelevant variables have been removed unless otherwise indicated. Under this assumption, we can replace the query $P_{\mathcal{N}}(X|Y = Y_0)$ with the query $P_{\mathcal{N}}(X, Y = Y_0)$. We call the latter a *standard query*. The rest of the paper will only be dealing with standard queries.

4 Potentials

A *potential* is a non-negative function which takes a positive value at at least one point. Here are some example potentials. The probability $P(X)$ is a potential of X, the conditional probability $P(X|Y)$ is a potential of X and Y, and $P(X, Y = Y_0)$ is a potential of X.

Let S be a set of potentials over a set of variables V. The *marginal potential* $P_S(X)$ is defined as follows: Multiply all the potentials in S together to get the *joint potential* $P_S(V)$, and $P_S(X)$ is obtained from $P_S(V)$ by summing out all the variables outside X. It is obvious that marginal probability is a special case of marginal potentials. For technical convenience, we shall be talking about marginal potential $P_S(X, Y = Y_0)$ instead of marginal probability $P_{\mathcal{N}}(X, Y = Y_0)$ from now on.

5 The key technique

Let S be a set of potentials over the set V of variables. A naive way to compute $P_S(X, Y = Y_0)$ is first to explicitly compute and store the joint potential $P_S(V)$, and then compute $P_S(X, Y = Y_0)$ from $P_S(V)$. This method is not efficient.

Even though the general problem of computing posterior probabilities in Bayesian networks is NP-hard, efficient algorithms often exist for particular applications due to the underlying structures. The purpose of this section is to describe a key technique that allows us to make use one aspect of problem structure, namely conditional independencies. The technique is to work with factorizations of joint potentials (probabilities) rather than joint potentials (probabilities) themselves.

We say that a set S_1 of potentials is a *factorization of the joint potential* $P_S(V)$ if $P_S(V)$ is the result of multiplying the potentials in S_1. The set S itself is certainly a factorization of $P_S(V)$, and it is the most straightforward one because it is what one has to begin with. We call S the *primary factorization* of $P_S(V)$.

We will see later that clique tree propagation does not directly adopts the primary factorization. Rather it first performs some pre-organizations and precomputations on S and then proceeds with the resulting more organized factorization.

[2] Nodes that do not have parents.

Exponential explosion can be in terms of both storage space and time. It is quite easy to see why factorization is able to help us to save space. For the sake of illustration, consider the Bayesian network **net1** in Figure 1. If all the variables are binary, to store the set of potentials of **net1**, i.e the prior and conditional probabilities, one needs to store $2 + 4 * 4 + 2 * 8 = 34$ numbers. On the other hand, to explicitly store the joint potential (probability) $P_{net1}(a, b, c, d, e, f, g)$ itself, one needs to store $2^7 = 128$ numbers.

To see how factorizations of joint potentials enable us to save time, we assume that the summing-out-variables-one-by-one strategy is adopted for computing marginal potentials.[3] We also assume that an ordering has been given for this purpose. This ordering will be referred as the *elimination ordering*.

Since we choose to work with the a factorization, which is a set of potentials, we need to define how to sum out one variable from a set of potentials. *To sum out a variable v from a set of potentials S* is to: (1) remove from S all the potentials that contain v, (2) multiply all those potentials together, (3) sum out v from the product, and (4) add the resulting potential to S.

For example, to sum a out of **net1**, we first remove $P(a)$, $P(b|a)$ and $P(f|a)$ from **net1**, then compute

$$\psi_a(b, f) = \sum_a P(a)P(f|a)P(b|a), \tag{5}$$

and finally add $\psi_a(b, f)$ to **net1**. After all these operations, **net1** becomes $\{P(c|b),\ P(d|b),\ P(e|c, d),\ P(g|f, e),\ \psi_a(b, f)\}$.

Usually, it takes much less arithmetic calculations to sum out one variable from a factorization of a joint potential than from the joint potential itself. For example, equation (5) denote all the arithmetic calculations needed to sum out a from **net1**. It involves only three variables: a, b, and f. On the other hand, to sum out a explicitly from $P_{net1}(a, b, c, d, e, f, g)$ itself, one needs to perform the following calculations,

$$\sum_a P_{net1}(a, b, c, d, e, f, g),$$

which involves all the seven variables in the network. This is the exactly why working the factorizations of joint potentials enables us to reduce time complexity.

6 Three components

In implementing Bayesian networks, if one adopts the key technique outlined in the previous section, then the resulting system can be divided into three components.

[3] It must be noted that they are exact approaches that do not adopt this strategy. See Poole and Neufeld (1991) and Poole (1992) for examples.

The first component finds an elimination ordering. We call it the *ordering determination* component. Roughly speaking, an elimination ordering is good if the arithmetic calculations needed to sum out each variable, from the primary factorization, involve only a small number of other variables. Even with a clear and crisp definition, the ordering determination problem proves to be a difficult one. See Kjærulff (1990) and Klein *et al* (1990) for research progresses on the problem. In this paper, we shall not discuss it any further.

The third component is the *arithmetic calculation* component. It takes a bunch of potentials, multiply them together, sum out a certain variable from the product, and return the result. A major goal in designing Bayesian network inference algorithms is to minimize the total number of arithmetic calculations.

In between the first and the third components lies the *data management* component. It determines, from the elimination ordering produced by the first component, what arithmetic calculations the third component is going to perform, and in which order. The component also hides the design decisions as to how to store the potentials, how to retrieve a potential when it is needed, and how to update the set of potentials after a variable has been summed out. We call a design of the data management component a *data management scheme*.

Among the existing exact approaches to Bayesian network computations, clique tree propagation (Jensen *et al* 1990), the arc reversal node reduction approach (Shachter 1988), and symbolic probabilistic inference (Shachter *et al* 1990) are data management schemes; while recursive decomposition (Cooper 1990) and component tree propagation (Zhang and Poole 1992) are mixtured of data management schemes and ordering determination methods.

In the remainder of the paper, we shall first propose a very simple data management scheme (section 7), and we shall compare this scheme with clique tree propagation and the arc reversal and node reduction approach (sections 8 and 9).

7 A simple data management scheme

The following algorithms describes our design of the data management component. It takes, as input, a set of potential S, a standard query $(X,\ Y = Y_0)$ and an elimination ordering *Ordering*. The output is $P_S(X, Y = Y_0)$.

PROCEDURE **P**(S, $(X,\ Y = Y_0)$, *Ordering*)

1. Set Y to Y_0 in the potentials in S, resulting in S_1.
2. Associate each potential ψ of S_1 with the variable that appears earliest in

Ordering among all the variables of ψ,

3. **Repeat** till *Ordering* becomes empty,
   - Remove the first variable on *Ordering*. Denote this variable by v. Call subroutine **Arithemetic-Calculation** to multiply all the potentials associated v together and to sum out v from the product, resulting in ψ_v,
   - Associate ψ_v with the variable that appear earliest in *Ordering* among all the variables of ψ_v, and
4. Return the potential produced from the removal of the last variable in *Ordering*, which is $P_S(X, Y = Y_0)$.

As an example, let us consider computing $P_{net1}(g, a = a_0)$. Suppose the elimination ordering is (b, c, d, e, f). Then, the initial variable-potential association scheme is as follows:

$b:$	$c:$	$d:$	$e:$	$f:$
$P(b\|a_0)$	$P(e\|c,d)$		$P(g\|e,f)$	$P(f\|a_0)$
$P(c\|b)$				
$P(d\|b)$				

Let $\psi_b(c,d) = \sum_b P(b|a_0)P(c|b)P(d|b)$. Then after the removal of b, the association scheme becomes:

$c:$	$d:$	$e:$	$f:$
$P(e\|c,d)$		$P(g\|e,f)$	$P(f\|a_0)$
$\psi_b(c,d)$			

Let $\psi_c(d,e) = \sum_c P(e|c,d)\psi_b(c,d)$. After the removal of c, we get

$d:$	$e:$	$f:$
$\psi_c(d,e)$	$P(g\|e,f)$	$P(f\|a_0)$

Let $\psi_d(e) = \sum_d \psi_c(d,e)$. After the removal of d, we get

$e:$	$f:$
$P(g\|e,f)$	$P(f\|a_0)$
$\psi_d(e)$	

Let $\psi_e(f,g) = \sum_e P(g|e,f)\psi_d(e)$. After the removal of d, we get

$f:$
$P(f\|a_0)$
$\psi_e(f,g)$

Finally, let $\psi_f(g) = \sum_f P(g|e,f)\psi_e(f,g)$. The potential $\psi_f(g)$ is $P_{net1}(g, a = a_0)$.

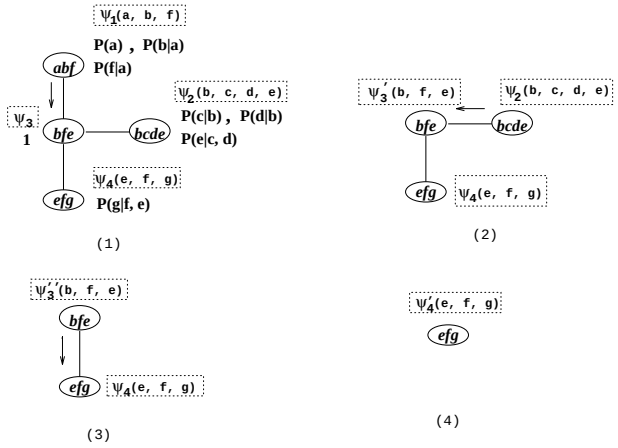

Figure 2: Clique tree propagation.

8 Comparing with clique tree propagation

We begin this section with a brief review of clique tree propagation. For this paper, a *clique* can be simply understood as a subset of variables. A *clique tree* is a tree whose nodes are cliques such that if a variable appear on two different nodes, then it also appears in all the nodes in the path between the two nodes. A *clique tree for a set of potentials* is a clique tree such that for each potential in the set, there is at least one clique in the tree that contains all its variables.

Like our approach, clique tree propagation reduces time and space complexities by working with factorizations of joint potentials. Unlike in our approach, which begins with the primary factorization, clique tree propagation associates the potentials in the primary factorization with the cliques in a clique tree, such that each potential is associated with one and only one clique that contains all its variables. All the potentials associated with one clique are multiplied together, resulting one single potential. If there is no potential associated with a clique, the constant potential 1 is stuck in. Thus the initial factorization for clique tree propagation consists of one and only one potential for each clique.

Figure 2 (1) shows a clique tree for the Bayesian network `net1` in Figure 1, together with a grouping of its prior and conditional probabilities (potentials). The initial factorization is $\{\psi_1(a,b,f),\ \psi_2(b,c,d,e),\ \psi_3(b,f,e),\ \psi_4(e,f,g)\}$, where $\psi_1(a,b,f) =_{def} P(a)P(b|a)P(f|a)$, $\psi_2(b,c,d,e) =_{def} P(c|b)P(d|b)P(e|c,d)$, $\psi_3(b,f,e) =_{def} 1$, and $\psi_d(e,f,g) =_{def} P(g|f,e)$.

Clique tree propagation computes a marginal potential by message passing in the clique tree. To pass a message from one node C (a clique) to one of its neighbors D, one sums out, from the potential associated with C, the variables in $C - D$, send the resulting potential to

D, and update the potential currently associated with D by multiplying it with the potential from C. Figure 2 show the message passing process for computing $P_{net1}(g, a = a_0)$.

In Jensen *et al* (1990), the prior marginal probability of each clique (node) in the clique tree is precomputed and stored at the node. To compute $P_{net1}(g|a = a_0)$ in this scheme, one only needs to pass proper messages from node (abf) to (bfe), and then to (efg). No message from node $bcde$ to node (bfe) is necessary. See the cited paper for details.

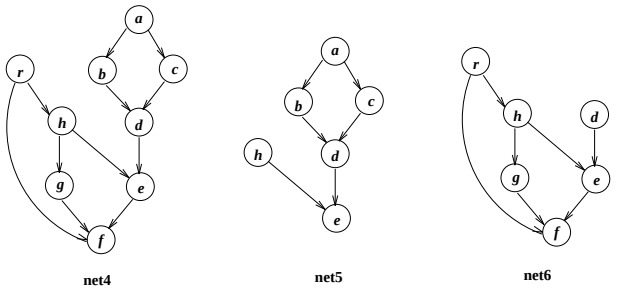

Figure 3: Pruning vs. precomputing.

8.1 Comparisons

Before commencing the comparisons, let us point out the both our approach and clique tree propagation have the same starting point. Our approach begin with an elimination ordering, and clique tree propagation begins with a clique tree. The availability of a clique tree is equivalent to the availability of an elimination ordering for the empty query $P_S(\emptyset)$. There are linear time algorithms to obtain an elimination ordering from a clique tree and to get back the clique tree from the ordering (see, for example, Zhang 1991).

To compare our data management scheme with clique tree propagation, we notice that our approach handles changes to the knowledge base more easily. Clique tree is a secondary structure. Any topology changes to the original network, like adding or deleting variable, or adding or deleting an arc, require recomputing the clique tree and the potential associated with each clique. In the Jensen *et al* (1990) scheme, one has to recompute the marginal probabilities for all the cliques even when there are only numerical adjustments to the conditional probabilities.

Secondly, if in a query $P_{\mathcal{N}}(X|Y = Y_0)$, X is not contained in any clique of the clique tree, then the secondary structure has to be modified, at least temporarily. This is even more cumbersome in the Jensen *et al* (1990) Scheme.

Two major issues in comparing our approach with clique tree propagation are pruning and precomputing, to which we devote the next subsection.

8.2 Pruning vs. precomputing

Pruning irrelevant variables and precomputing the prior probabilities of some variables are two techniques to cut down computations. In this section, we shall illustrate those two techniques through an example and discuss some related issues.

Consider the query about posterior probability $P_{net4}(e|h = h_0)$, where `net4` is given in Figure 3. One can first prune f because it is barren. Thereafter, g and r can also be pruned because g becomes barren after the removal of f and r becomes m-separated from e by h. Thus, pruning enables one to transform the original query into $P_{net5}(e|h = h_0)$, a query to a Bayesian network with three variables less.

One the other hand, it is easy to see that summing out variables a, b, and c in `net4` results in `net6`, where $P(d) = \sum_{a,b,c} P(a)P(b|a)P(c|a)P(d|b,c)$. So, if $P(d)$ is precomputed, the query $P_{net4}(e|h = h_0)$ can be immediately transformed into $P_{net6}(e|h = h_0)$, a query to a Bayesian network with three variables less.

So, both pruning and precomputing enable us to cut down computations. However, they both have prices to pay as well.

Pruning irrelevant variables implies that we will be working with a potentially different sub-network for each query. This usually means that an elimination ordering is to be found for each particular query from scratch, instead of deriving it from an ordering for the empty query.

We argue that this is not a serious drawback for two reasons. First, if a query only involve a few variables, pruning may give us a sub-network which is only a small portion of the original network. After all, only those variables who are ancestors of variables in the query could be relevant to the query. Thus pruning makes finding a good elimination ordering much easier. Second, some existing hueristics for finding elimination, like maximal cardinality search and lexicographic search (Rose 1970, Kjæruff 1990), are quite simple. It does not take much more time to find an ordering from scratch.

As far as precomputing is concerned, it is difficult to decide what to precompute. For example, precompute $P(g)$ is not very helpful in the previous example. One solution is to compute the prior probabilities for all the combinations of variables. But this implies an exponential number of precomputations.

Clique tree propagation works on the secondary structure of clique tree, which is kept static. This makes precomputing possible (Jensen *at al* 1990). As we pointed out early, however, there is a price to pay. The approach does not prune irrelevant variables, and it is hard for it to accommodate changes to the knowledge base.

8.3 Intercausal independence

A major reason for us to come up with a new data management scheme is that it leads to a uniform framework for utilizing conditional independence as well as intercausal independence.

In Zhang and Poole (1994), we give a constructive definition of intercausal independence. Noisy OR-gates and noisy adders satisfy our definition. A nice property of our definition is that it relates intercausal independence with factorization of conditional probabilities, in a way very similar to that conditional independence is related factorization of joint probabilities. The only difference lies in the way the factors are combined. While conditional independence implies that a joint can be factorized as a *multiplication* of several factors, intercausal independence implies that a conditional probability can be factorized as a *certain combination* of several factors, where combination is usually not multiplication.

The concept of heterogeneous factrization is proposed. A heterogeneous factorization is one where different factors can be combined in different ways. We have adapted the data management scheme proposed in this paper to handle heterogeneous factorizations.

9 Conclusions

A key technique to reduce time and space complexities in Bayesian networks is to work factorizations of joint potentials rather than joint potentials themselves. Different exact approaches can be characterized by their choices of factorizations. We have proposed a new approach which begins with the primary factorization. Our approach is simpler than clique tree propagation. Yet it is advantageous in terms of pruning irrelevant variables and accommodating changes to the knowledge base. More importantly, our approach leads to a uniform framework for dealing with both conditional and intercausal independence. However, our approach does not support precomputation as clique tree propagation does.

Acknowledegment:

We wish to thank Mike Horsch for his valuable comments on a draft of this paper and Runping Qi for useful discussions. Research is supported by NSERC Grant OGPOO44121 and travel grants from Hong Kong University of Science and Technology.

References:

M. Baker and T. E. Boult (1990), Pruning Bayesian networks for efficient computation, in *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, July, Cambridge, Mass. , pp. 257 - 264.

G. F. Cooper (1990) The computational complexity of probabilistic inference using Bayesian belief networks, *Artificial Intelligence*, **42**, pp. 393-405.

G. F. Cooper (1990), Bayesian belief-network inference using recursive decomposition, Report No. KSL 90-05, Knowledge Systems Laboratory, Medical Computer Science, Standford University.

D. Geiger, T. Verma, and J. Pearl (1990), *d*-separation: From theorems to algorithms, in *Uncertainty in Artificial Intelligence* **5**, pp. 139-148.

F. V. Jensen, K. G. Olesen, and K. Anderson (1990), An algebra of Bayesian belief universes for knowledge-based systems, *Networks*, 20, pp. 637 - 659.

U. Kjærulff (1990), Triangulation of Graphs - Algorithms giving small total state space, R 90-09, Institute for Electronic Systems, Department of Mathematics and Computer Science, Strandvejen, DK 9000 Aalborg, Denmark.

P. Klein, A. Agrawal, A. Ravi, and S. Rao (1990), Approximation through multicommodity flow, in Proceedings of 31st Symposium on Foundations of Computer Science, pp. 726-737.

S. L. Lauritzen and D. J. Spiegehalter (1988), Local computations with probabilities on graphical structures and their applications to expert systems, *Journal of Royal Statistical Society B*, 50: 2, pp. 157 - 224.

S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H. G. Leimer (1990), Independence Properties of Directed Markov Fields, *Networks*, 20, pp. 491-506.

J. Pearl (1988), *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, Los Altos, CA.

D. Poole and E. Neufeld (1991), Sound probabilistic inference in Prolog: An executable specification of Bayesian networks, Department of Computer Science, University of British Columbia, Vancouver, B. C., V6T 1Z2, Canada.

D. Poole (1992), Search for Computing posterior probabilities in Bayesian networks, Technical Report 92-24, Department of Computer Science, University of British Columbia, Vancouver, Canada.

D. J. Rose (1970), Triangulated graphs and the elimination process, *Journal of Mathematical Analysis and Applications*, **32**, pp 597-609.

R. Shachter (1986), Evaluating Influence Diagrams, *Operations Research*, 34, pp. 871-882.

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

R. D Shachter, S. K. Andersen, and P. Szolovits (1992), The equivalence of exact methods for probabilistic inference in belief networks, Department of

Engineering-Economic Systems, Stanford University.

R. D. Shachter, B. D'Ambrosio, and B. A. Del Favero (1990), Symbolic Probabilistic Inference in Belief Networks, in *AAAI-90*, pp. 126-131.

G. Shafer and P. Shenoy (1988), Local computation in hypertrees, Working Paper No. 201, Business School, University of Kansas.

L. Zhang (1991), Studies on hypergraphs (I): Hyperforests, accepted for publication on *Discrete Applied Mathematics*.

L. Zhang and D. Poole (1992) Sidestepping the triangulation problem in Bayesian net computations, in *Proc. of 8th Conference on Uncertainty in Artificial Intelligence*, July 17-19, Standford University, pp. 360-367.

L. Zhang and D. Poole (1994) Intercausal independence and heterogeneous factorizations, submitted to *The Tenth Conference on Uncertainty in Artificial Intelligence*