Journal of Machine Learning Research 6 (2005) 1517–1550 Submitted 6/05; Revised 8/05; Published 9/05

Local Propagation in Conditional Gaussian Bayesian Networks

Robert G. Cowell RGC@CITY.AC.UK
Faculty of Actuarial Science and Statistics
Sir John Cass Business School
106 Bunhill Row
London EC1Y 8TZ, U.K.

Editor: Craig Boutilier

Abstract

This paper describes a scheme for local computation in conditional Gaussian Bayesian networks that combines the approach of Lauritzen and Jensen (2001) with some elements of Shachter and Kenley (1989). Message passing takes place on an elimination tree structure rather than the more compact (and usual) junction tree of cliques. This yields a local computation scheme in which all calculations involving the continuous variables are performed by manipulating univariate regressions, and hence matrix operations are avoided.

Keywords: Bayesian networks, conditional Gaussian distributions, propagation algorithm, elimination tree

1. Introduction

Bayesian networks were developed within the field of artificial intelligence as a tool for representing and managing uncertainty (Pearl, 1988; Cowell et al., 1999), but are now finding many applications beyond that field. When a Bayesian network represents the joint distribution of a set of finite discrete random variables, exact and efficient local computations schemes may be used to evaluate marginal distributions of interest. Shachter and Kenley (1989) introduced Gaussian influence diagrams to represent multivariate Gaussian distributions and performed inference on them using standard influence diagram operations such as arc-reversals and barren node removal. Raphael (2003) presented an alternative computational scheme for degenerate multivariate Gaussian distributions and has applied it to problems of rhythmic parsing of music.

Lauritzen (1992) introduced a method of exact local computation of means and variances for Bayesian networks with conditional Gaussian distributions (Lauritzen and Wermuth, 1984, 1989), but it was later discovered that the method was numerically unstable. More recently Lauritzen and Jensen (2001) have developed an alternative and stable local computation scheme in junction trees for these conditional Gaussian networks. Apart from the improved numerical stability compared to the previous algorithm, their method is able to calculate full mixture marginals of continuous variables, and is also able to include deterministic linear relationships between continuous variables. However their method is quite complicated, requiring evaluations of matrix generalized inverses, and recursive combinations of potentials. This paper presents an alternative scheme in which the local computation is performed on an elimination tree, rather than using a junction tree. As will be shown this means that matrix manipulations are avoided because all message passing opera-

tions involving the densities of the continuous variables are performed by manipulating univariate regressions, and complex operations such as recursive combination of potentials are avoided.

The plan of the paper is as follows. The following section presents notation for conditional Gaussian regressions. Then elimination trees are defined and compared to junction trees. A simple network is used to illustrate the various phases of the message passing scheme. Then the general procedure is presented: deriving an elimination tree derived from a network; initializing the elimination tree is given (this is perhaps the most complicated part of the scheme); entering evidence and evaluating posterior marginal densities. A discussion relating the current scheme to those of Shachter and Kenley (1989) and Lauritzen and Jensen (2001) is presented, followed by an algorithm for sampling from the posterior and two maximization operations.

2. CG Regressions

Following the notation of Lauritzen and Jensen (2001), a mixed Bayesian network consists of a set of nodes V partitioned into a set of *discrete* nodes, Δ, and a set of continuous nodes, Γ. Each node represents a random variable. The Bayesian network is directed acyclic graph, with the restriction that discrete nodes are not allowed to have continuous parents. Associated with each $Y \in \Gamma$ of the continuous nodes are conditional Gaussian (CG) regressions, one for each configuration in the state space of the discrete parents I of Y, given by

$$\mathcal{L}(Y \mid I = i, Z = z) = \mathcal{N}(\alpha(i) + \beta(i)^T z, \sigma^2(i)),$$

where $\alpha(i)$ is a real number, Z is a vector of the continuous parents of Y, $\beta(i)$ is a vector of real numbers of the same size of Z, and $\sigma^2(i)$ is a non-negative real number. If the variance $\sigma^2(i) = 0$, then the regression represents a deterministic linear relationship between Y and the Z. Associated with each discrete random variable $X \in \Delta$ is a conditional probability distribution of the variable given its parents in the graph.

The product of the densities associated with the continuous random variables gives the (multivariate normal) density of the continuous variables Γ conditional on the discrete variables Δ. On multiplying this by the product of the conditional probability distributions of each of the discrete variables, the joint density of both discrete and continuous variables is obtained.

Lauritzen and Jensen (2001) introduce as their basic computational object a *CG potential*, represented by the tuple $\phi = [p, A, B, C](H \mid T)$, where: H is a set of r continuous variables, called the *head*; T is a set of s continuous variables, called the *tail*; $H \cap T = \emptyset$; $p = \{p(i)\}$ is a table of nonnegative numbers; $A = \{A(i)\}$ is a table of $r \times 1$ vectors; $B = \{B(i)\}$ is a table of $r \times s$ matrices; and $C = \{C(i)\}$ is a table of $r \times r$ positive semidefinite symmetric matrices. They introduce various operations on such potentials: multiplication, extension, marginalization, direct combination, complementation, and recursive combination. These operations are required for their message passing algorithm on the junction tree structure, and in the main correspond to operations on probability distributions. However there are restrictions that must be observed for these operations to be permissible; for example, it is not possible to directly combine two CG potentials together if the intersection of their heads is non-empty. Such constraints are obeyed in their propagation algorithm.

In comparison to the propagation scheme presented in this paper, much of the complexity of their algorithm arises because their local computational structure is a strong junction tree of cliques. The cliques with continuous variables essentially contain, after a basic initialization, multivariate CG regressions. Sending a sum-marginal message between two cliques could require marginaliza-

tion over both discrete and continuous variables, in which case the latter are marginalized first. In contrast, we work with univariate regressions on an elimination tree, avoiding matrix operations. (Indeed one could implement the current scheme using the CG potentials and their operations, restricting them to heads that contain only one variable, so that $r = 1$.)

3. Junction Trees and Elimination Trees

In this section we review the notions of a junction tree of cliques and of an elimination tree, and of their relative advantages and disadvantages. More details about these structures may be found in Cowell et al. (1999).

3.1 Making a Junction Tree

Since the work of Lauritzen and Spiegelhalter (1988) the most common graphical structure on which to perform exact inference on Bayesian networks by local message passing has been a junction tree of cliques. This is a tree structure, with the node set being the set of cliques $\mathcal{C}$ of a chordal graph, such that the intersection $C_1 \cap C_2$ of any two cliques $C_1 \in \mathcal{C}$ and $C_2 \in \mathcal{C}$ is contained in every clique on the path in the junction tree between C_1 and C_2. The intersection of two cliques adjacent in a junction tree is called a *separator*. The basic algorithmic steps in constructing the junction tree, starting from a directed acyclic graph G, are as follows:

1. Add moral edges to G, and then drop directionality on the edges to form the moral graph G^m.
2. Add sufficient fill-in edges to G^m to make a chordal graph G^c.
3. Identify the cliques of G^c, and join them up to form a tree structure which has the running-intersection property.

Step 1 is straightforward, and Step 3 may be done efficiently using the *maximum cardinality search* algorithm (Tarjan and Yannakakis, 1984). It is Step 2, also commonly known as *triangulation*, that presents the main obstacle to efficient message passing. There are many ways to triangulate the moral graph G^m, what is desirable is that the cliques that arise are kept small, or more specifically the sum total state space size over the cliques is minimized. Finding optimal triangulations is NP-hard (Yannakakis, 1981), and so early work focused on heuristic algorithms, typically of a one-step-look-ahead type (Kjærulff, 1990), but other methods, for example genetic algorithms (Larrañaga et al., 1997) have also been used. More recent work has focussed on divide-and-conquer approaches that can yield close to optimal or even optimal triangulations (Becker and Geiger, 2001; Olesen and Madsen, 2002), and an optimal triangulation algorithm is implemented in the commercial expert system HUGIN.[1]

3.2 Making an Elimination Tree

Elimination trees were introduced by Cowell (1994) for analysing decision problems, and are described on pages 58–60 of Cowell et al. (1999). An elimination tree is similar to a junction tree, in that it is a tree structure, but with the node set being a subset of the complete subgraphs of a chordal graph (rather than the set of cliques) such that the intersection $C_1 \cap C_2$ of any two nodes

1. The company web site is at *http://www.hugin.com*.

in the elimination tree is contained in every node on the path in the tree between C_1 and C_2. The subset of complete subgraphs is determined by an elimination sequence, this being an ordering of the nodes of the chordal graph. The basic steps to make an elimination tree starting with the directed acyclic graph G are as follows:

1. Add moral edges to G, and then drop directionality on the edges to form the moral graph G^m.

2. Take an elimination sequence $v_k, v_{k-1}, \ldots, v_1$ of the k nodes (suitably re-numbered) of the moral graph G^m, and use this to form a triangulated graph G^c.

3. For each node v_i associate a so called *cluster set* of nodes consisting of v_i and its neighbours later in the elimination sequence (and hence of lower number), call this set C_i.

4. Join the sets $C_1, C_2, \ldots, C_k$, which has the running-intersection property, together to form a tree.

Step 1 is the same step as for making a junction tree. For Step 2, we first start with the node v_k, and add edges so that it and its neighbours form a complete subgraph (this will make the set C_k in Step 3). Then move onto node v_{k-1}, add edges so that it and its neighbours that occur later in the elimination sequence form a complete subgraph (this will make the set C_{k-1} in Step 3). Repeat this last step for the other nodes in the elimination sequence. There will be k cluster sets, one for each of the k nodes in the graph G, and $C_1 = \{v_1\}$. The choice of elimination sequence will govern the number of fill-ins in the triangulation, and hence the size of the state space of the elimination tree. Reasonable choices can usually be made by a one-step-look-ahead search. Notice that if one knows an optimal triangulation G^c of G^m, then a perfect numbering of the nodes of G^c could be used as an elimination sequence for G^m. Step 4 may be done in time typically linear in k (but possibly as bad as $O(k^2)$), by the simple expedient of finding in each cluster set C_i the first node v_{e_i}, say, that was eliminated after v_i and then joining C_i to C_{e_i}.

3.3 Comparing Elimination and Junction Trees

In an elimination tree, the set of cluster sets contains the set of cliques of the triangulated graph together with some other sets. Hence in terms of storage requirements for potentials on the sets, elimination trees are less efficient than junction trees. Sometimes they can be very bad, as shown with the following example.

Suppose the original graph G or its moral graph G^m is a complete graph of k nodes, each of which represents a binary random variable. Then there are no fill-in edges to be added as G^m is already triangulated, and the junction tree is a single clique containing all k nodes of G and having a total state space of size 2^k. Now consider the elimination tree, made by using an elimination sequence $v_k, v_{k-1}, \ldots, v_1$. This will yield k cluster sets, with $C_j = \cup_{i=1}^{j} \{v_i\}$, having a total state space size given by $2 + 2^2 + \cdots + 2^k = 2(2^k - 1)$. That is, it requires almost double the storage requirements of the junction tree. Actually things are worse than this, because we have not taken into account the $k-1$ separators between adjacent clusters which have total state space size $2 + 2^2 + \cdots + 2^{k-1} = 2^k - 2$, thus leading to a factor of almost three in the storage requirements. However it should be emphasized that this is a worst case scenario, and in most applications the overhead is a small fraction of the total state space of the corresponding junction tree.

Now suppose that the variables of G are continuous rather than being discrete, with G now representing a multivariate Gaussian distribution. To represent this distribution in the junction tree will require k values for the means of each variable, and a further $k(k+1)/2$ values for the symmetric covariance matrix, making a total of $k(k+3)/2$ values to be stored. In the corresponding elimination tree, as we shall see, only univariate regressions are stored. Hence in C_1 we store two numbers (a mean and a variance), in C_2 we store three numbers (a mean, one coefficient and a variance), ... , and in C_k we store $k+1$ numbers, (a mean, $k-1$ coefficients, and a variance). Adding all these together we have a total of $2+3+\cdots+(k+1) = k(k+3)/2$ values to be stored, the same as for the junction tree. Thus the use of the elimination tree does not introduce duplication in the values to be stored, in contrast to the discrete case. Hence the elimination tree is *almost* as efficient as the junction tree in storage requirements (only almost, because there will be some extra bookkeeping needed to keep track of which variables are in each of the cluster sets).

3.4 Strongly Rooted Trees

For purely discrete Bayesian networks and purely continuous multivariate Gaussian Bayesian networks the junction or elimination trees described above may be used for exact propagation algorithms, with any clique or cluster set being chosen as the root to which messages are collected to and distributed from. For the conditional Gaussian networks in which both discrete and continuous variables appear, Lauritzen (1992) used the notion of a *marked graph* (Leimer, 1989) to define the structure of a *strongly rooted junction tree* in order to have a manageable propagation scheme which handles the asymmetry between the discrete and continuous variables. This structure is retained in Lauritzen and Jensen (2001). Here we use a similar structure based on elimination trees, which we shall call a *strongly rooted elimination tree*. We shall assume without loss of generality that the graph G of the Bayesian network is connected. Then a strongly rooted elimination tree may be formed in the same way as a standard elimination tree provided that, in the elimination sequence used, all of the continuous variables occur before any of the discrete variables. The cluster sets are joined up as before, and the last cluster formed is taken to be the strong root. (If G has more than one connected component, then we form a strong elimination tree for each component; it can then be useful to introduce an empty cluster set connected to each of the strong roots of the individual elimination trees, and make this the strong root of the forest of elimination trees.)

The reason for using a strong elimination tree will become apparent when we discuss the initialization of the tree and propagation on it. For a more efficient computation scheme (from a storage requirement viewpoint) is it convenient to use a tree structure that is intermediate between a junction tree and an elimination tree, a structure which we call a *strong semi-elimination tree*, introduced in the next section.

3.5 From Elimination Tree to Junction Tree

Given an elimination ordering for a graph G, one can construct a triangulated graph G^c, use maximum cardinality search to find the cliques, and then organize the cliques into a junction tree. Alternatively, one could take the elimination tree and remove the redundant cluster sets that are subsets of cliques, by repeated application of the following result due to Leimer (1989) (see also Lemma 2.13 of Lauritzen (1996) or Lemma 4.16 of Cowell et al. (1999)):

Lemma 3.1 *Let $C_1,\ldots,C_k$ be a sequence of sets having the running intersection property. Assume that $C_t \subseteq C_p$ for some $t \neq p$ and that p is minimal with this property for fixed t. Then:*

(i) *If $t > p$, then $C_1,\ldots,C_{t-1},C_{t+1},\ldots,C_k$ has the running intersection property.*

(ii) *If $t < p$, then $C_1,\ldots,C_{t-1},C_p,C_{t+1},\ldots,C_{p-1},C_{p+1},\ldots,C_k$ has the running intersection property.*

The preceding Lemma operates on sets, but in the elimination tree we also have links between sets, which will need to be rearranged if a set if deleted. The following algorithm, based on Lemma 3.1 makes a single pass through the cluster sets of an elimination tree to produce a junction tree, it assumes that the elimination tree is connected. It is convenient to make the edges of the elimination tree directed edges, with directions pointing away from the root C_1. We may then talk of parents (pa) and children (ch) of the cluster sets in the tree in the obvious manner.

Algorithm 3.2 (Elimination tree to Junction tree)

1. *Initialize:*
   - *An ordered list L of the cluster sets $C_1,C_2,\ldots,C_k$ derived from the elimination ordering $v_k,v_{k-1},\ldots v_1$ having the running intersection property, and joined to form an elimination tree.*
   - *An ordered list J, initially empty.*
2. *While L is non-empty do:*
   - *Remove the first element C_t from L;*
   - *If C_t is a clique then append it to the end of J, otherwise:*
     (a) find $C_p \in ch(C_t)$ such that p is minimized;
     (b) Remove C_p from L;
     (c) Set $pa(C_p) = pa(C_t)$;
     (d) In each cluster $c \in ch(C_t) \setminus C_p$ replace C_t in $pa(c)$ with C_p;
     (e) Add the elements of $ch(C_t) \setminus C_p$ to the set $ch(C_p)$;
     (f) Put C_p at the front of L;
     (g) Discard C_t.

It is left to the reader to verify that this repeatedly applies Step (ii) of Lemma 3.1, with Steps 2(c)-(e) updating the connections in the tree. When the algorithm terminates the list J contains the cliques in running intersection order, and the parent and child sets of these cliques contain the links required to make a strong junction tree. An example illustrating the steps in Algorithm 3.2 is shown in Figure 1.

Although the message passing algorithms presented below will work on a strong elimination tree, to optimize the storage requirements it is better to work on a *strong semi-elimination tree*. This is a strong elimination tree in which the purely discrete clusters that are subsets of other purely discrete clusters have been removed, with links among the remaining clusters suitably adjusted. Algorithm 3.3 produces a strong semi-elimination tree from a strong elimination tree.

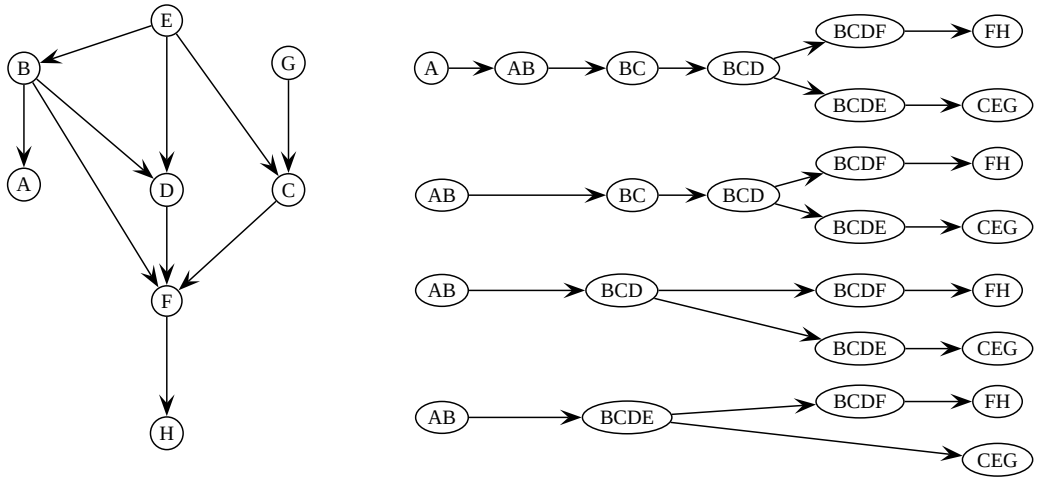

Figure 1: Illustration of Algorithm 3.2. On the left a Bayesian network, and top right the elimination tree obtained using the elimination sequence of reverse alphabetical ordering. In the top tree we first remove the redundant cluster $C_t = \{A\}$ having the unique child cluster $C_p = \{AB\}$, yielding the second tree in which $\{AB\}$ now has no parent. Then the redundant cluster $C_t = \{BC\}$ is removed: it has the unique child cluster $C_p = \{BCD\}$, and so this now inherits $\{BC\}$'s parent $\{AB\}$. Finally the redundant cluster $C_t = \{BCD\}$ is removed. It has two child clusters, of these $C_p = \{BCDE\}$ because E is eliminated after F. C_p has its parent changed to $\{AB\}$ (Step 2b) and is itself made the new parent of $\{BCDF\}$ (Step 2c). It inherits the extra child $\{BCDF\}$ from C_t (Step 2d) to yield the bottom tree, which after dropping directions on the edges gives the junction tree.

Algorithm 3.3 (Strong elimination tree to strong semi-elimination tree)

1. *Initialize:*
   - *An ordered list L of the cluster sets $C_1, C_2, \ldots, C_k$ derived from the strong elimination ordering $v_k, v_{k-1}, \ldots v_1$ having the running intersection property.*
   - *An ordered list J, initially empty.*
2. *While L is non-empty do:*
   - *Remove the first element C_t from L;*
   - *If C_t contains a continuous variable, or C_t is purely discrete and not a subset of another purely discrete cluster, append it to J, otherwise:*
     - *Find $C_p \in ch(C_t)$ such that p is minimized; (note that C_p will be purely discrete and $C_t \subset C_p$), and then:*
       (a) Remove C_p from L;
       (b) In C_p set $pa(C_p) = pa(C_t)$;
       (c) In each cluster $c \in ch(C_t) \setminus C_p$ replace C_t in $pa(c)$ with C_p;
       (d) Add the elements of $ch(C_t) \setminus C_p$ to the set $ch(C_p)$;
       (e) Put C_p at the front of L;
       (f) Discard C_t.

4. Computations on a Simple Network

The computational scheme to be presented here is more complicated than for the purely discrete case. Although both start with moralizing the Bayesian network, for CG networks a strong triangulation is required, leading to a strong elimination tree. For discrete networks, after the junction tree of cliques has been made, the initialization stage is quite straightforward, consisting of: (i) setting all entries in all of the tables (potentials) in the cliques and separators of the junction tree to unity; (ii) a multiplication of each conditional probability table of the Bayesian network into any one suitable clique table (one whose clique variables contains the variables of the conditional probability table); and (iii) propagation on the junction tree to yield clique and separator tables storing marginal distributions. For the CG networks we use the same initialization process for the discrete part of the elimination tree. However on the continuous part of the tree things are more complicated, and its initialization is perhaps the most important and complicated part of the computational scheme of this paper. In the initialization phase, CG regressions are passed between continuous clusters, and so we introduce a list structure called a *postbag* to store such messages that are to be passed. In addition we introduce for each continuous cluster another list structure called an *lp-potential* that will on completion of the initialization phase store the conditional density of the elimination variable associated with the cluster (conditional on the remaining variables in the cluster). Each regression of the Bayesian network is initially allocated to either some *lp-potential* or *postbag* by rules given below. During the initialization process the contents of the *lp-potential*'s and *postbag*'s may be modified by an operation that we call EXCHANGE (see Section 5.3), which is equivalent to an arc-reversal in the Bayesian network. These EXCHANGE operations have to done in a correct order, which requires a *postbag*'s contents to be sorted.

Evaluating node marginals is also more complicated. For discrete variables this proceeds as in the purely discrete case, by a marginalization of the tables in the discrete clusters in the elimination tree. For continuous variables we use the PUSH operation introduced by Lauritzen and Jensen (2001), which is a sequence of EXCHANGE operations carried out among regressions in continuous clusters along a path in the tree, to obtain for a continuous variable its marginal density conditional on a set of discrete variables. When combined with the marginal of those discrete variables we obtain the marginal density of the continuous variable—typically a mixture. The PUSH operation is also used as part of the process of entering evidence about continuous variables.

In this section we use a simple example to illustrate the steps of constructing a tree from a Bayesian network, initializing the tree, and finding the marginal of a continuous node, before giving the formulation of these algorithmic steps for a general network in the subsequent section. Some terminology is also introduced that will be used later.

4.1 Specification

Figure 2 shows a small mixed Bayesian network consisting of three discrete random variables, shown as square-shaped nodes, and three continuous real-valued random variables, shown as circles. It follows from the structure of the Bayesian network that the joint distribution of the discrete

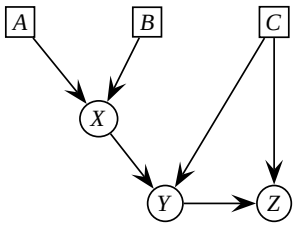

Figure 2: A mixed Bayesian network: square nodes represent discrete variables, circular nodes continuous variables.

variables is $P(A=a,B=b,C=c) = P(A=a)P(B=c)P(C=c)$, where the marginal probability distributions on each individual variable are assumed specified. To complete the probabilistic specification, we require the set of linear regressions (with the α's, β's and σ's being constants):

$$\begin{aligned} \mathcal{L}(X \mid A=a,B=b) &= \mathcal{N}(\alpha_{X:ab}, \sigma^2_{X:ab}) \\ \mathcal{L}(Y \mid X=x,C=c) &= \mathcal{N}(\alpha_{Y:c} + \beta_{Y:c}x, \sigma^2_{Y:c}) \\ \mathcal{L}(Z \mid Y=y,C=c) &= \mathcal{N}(\alpha_{Z:c} + \beta_{Z:c}y, \sigma^2_{Z:c}) \end{aligned}$$

So for example, if all discrete variables are binary, four regressions are required for X, and two each for Y and Z. This set of linear regressions defines the joint density of the continuous variables conditional on the discrete variables.

4.2 Making the Elimination Tree

The first stage is to transform the Bayesian network into the tree on which the message passing takes place. There are 36 possible elimination sequences that could be applied to the moral graph (3!

ways of eliminating the continuous variables first, followed by 3! ways of eliminating the discrete variables); here we shall use a sequence in which Y is eliminated first, then X. This is not the most computationally efficient sequence (eliminating Z first will ensure that Z does not appear in a cluster with X), but helps illustrate the operations of the propagation algorithm. In Figure 3 we show various trees that may be formed based on this elimination sequence; for our propagation algorithm we shall use the middle tree, which has the strong root $\{ABC\}$.

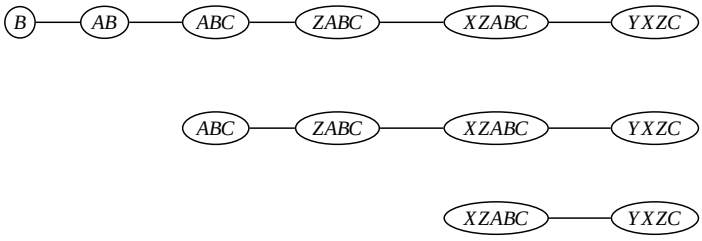

Figure 3: Strong elimination tree (top) with strong root $\{B\}$, strong semi-elimination tree with strong root $\{ABC\}$ (middle) and strong junction tree (bottom) with strong root $\{XZABC\}$, using the elimination sequence $YXZCAB$ applied to the moralized graph derived from the network in Figure 2. Separators are not shown.

4.3 Assignment of Potentials

The next stage is to assign the conditional probabilities and densities of the Bayesian network to the tree so that the tree contains a representation of the joint density of all of the variables. For purely discrete Bayesian networks, and in the formulations of mixed networks of Lauritzen (1992) and Lauritzen and Jensen (2001), each conditional distribution of a discrete node given its parents may be assigned to any clique in the junction tree that contains the family of the node, and similarly for the continuous variables the conditional density of a node given its parents is assigned to any clique containing the family of the node.

In contrast we apply a more restrictive assignment: for continuous variables the conditional density of a node given its parents is assigned to any cluster containing the family of the node, but the conditional distribution of a discrete node given its parents may be assigned to any cluster set that contains the family of the node provided that the cluster set does not contain any continuous variables. There always exists such a cluster set, because discrete variables are not eliminated until after the continuous variables are eliminated.

As a consequence of this assignment, the subtree consisting of those clusters containing only discrete variables contains all of the information required to reconstruct the joint marginal of the discrete variables; this part of the tree shall be referred to as the *discrete part*. Similarly, the clusters having the continuous variables hold representations of all of the densities with which one can construct the joint density of the continuous variables conditional on the discrete variables; this part of the tree will be called the *continuous part*. The set of discrete clusters that are neighbours to the continuous part will be called the *boundary*. Thus in Figure 3 the discrete part of the middle tree consists of the set $\{ABC\}$, the continuous part consists of the sets $\{ZABC\}, \{XZABC\}$ and $\{YXZC\}$, and the boundary consists of the single set $\{ABC\}$. We assign probability tables and regressions as

follows:

Cluster	Regressions
ABC	$P(A), P(B), P(C)$
$ZABC$	$-$
$XZABC$	$\mathcal{L}(X \mid A,B)$
$YXZC$	$\mathcal{L}(Z \mid Y,C), \mathcal{L}(Y \mid X,C)$

Our aim is that when initialization is complete, on the continuous part of the tree we have the following regressions:

Cluster	Regressions
$ZABC$	$\mathcal{L}(Z \mid A,B,C)$
$XZABC$	$\mathcal{L}(X \mid Z,A,B,C)$
$YXZC$	$\mathcal{L}(Y \mid X,Z,C)$

and in general we aim to represent in each continuous cluster set the conditional densities of the eliminated node given the remaining variables in the cluster set. This corresponds to the set-chain representation (Lauritzen and Spiegelhalter, 1988), at least on the continuous part of the tree. Note that we do not explicitly store potentials on separators between continuous cluster sets. In contrast for the discrete part of the tree we will retain separators, and perform usual sum-propagation to find the sum-marginal potential representation (see Cowell et al. (1999), Chapter 6); however in this example there is no such separator.

We move from the initial assignment of regressions to this (partial) set-chain representation by local message passing as follows. First, in the cluster set $\{YXZC\}$ we rearrange the pair of regressions as follows,

$$\mathcal{L}(Z \mid Y,C=c), \mathcal{L}(Y \mid X,C=c) \rightarrow \mathcal{L}(Z,Y \mid X,C=c) \rightarrow \mathcal{L}(Y \mid Z,X,C=c), \mathcal{L}(Z \mid X,C=c)$$

for each value that C can take. This rearrangement corresponds to the arc-reversal of the directed edge from Y to Z in Figure 2 (that is, an application of Bayes' theorem). In the expression on the right the regressions $\mathcal{L}(Y \mid ZXC)$ are retained in the cluster set. The regressions $\mathcal{L}(Z \mid X,C)$, which are independent of Y, are forwarded to the neighbouring cluster set towards the root, here $\{XZABC\}$. After this rearrangement we have the following represented on the continuous part of the tree:

Cluster	Regressions
$ZABC$	$-$
$XZABC$	$\mathcal{L}(X \mid A,B), \mathcal{L}(Z \mid X,C)$
$YXZC$	$\mathcal{L}(Y \mid X,Z,C)$

Next, the regressions in the cluster $\{XZABC\}$ are modified as follows:

$$\begin{aligned}
&\mathcal{L}(X \mid A=a,B=b), \mathcal{L}(Z \mid X,C=c) \\
&\qquad \rightarrow \mathcal{L}(X,Z \mid A=a,B=b,C=c) \\
&\qquad \rightarrow \mathcal{L}(X \mid Z,A=a,B=b,C=c), \mathcal{L}(Z \mid A=a,B=b,C=c).
\end{aligned}$$

The regressions $\mathcal{L}(X \mid Z,A,B,C)$ are retained in the cluster $\{XZABC\}$, and the regressions $\mathcal{L}(Z \mid A,B,C)$ are forwarded to the cluster $\{ZABC\}$, and we are done.

Note that it is not necessary to form the intermediate joint density implied by, for example, $\mathcal{L}(X,Z \mid A,B,C)$. Instead, the algebraic EXCHANGE formulae (see Section 5.3) may be applied to

pass directly from one pair of regressions to another, even in the case that some variances are zero (corresponding to deterministic linear relationship between continuous variables). These formulae are essentially equivalent to Theorem 1 of Shachter and Kenley (1989). By working directly and only with linear regressions, instead of multivariate conditional Gaussian densities, the need for matrix algebra and evaluation of determinants is avoided. Another advantage of the EXCHANGE formulae, as emphasized by Shachter and Kenley (1989), is that they should not lead through roundoff error to negative values of variances, something that could happen when manipulating multivariate densities.[2]

4.4 Calculating Marginals

One of the prime applications of the propagation algorithms in Bayesian networks is to evaluate marginal distributions of variables, perhaps conditional on values (evidence, findings) of some other variables in the network. We illustrate the procedure for the simple example, beginning with the case that no variable in the network has evidence.

Suppose that we wish to evaluate the marginal density of Z. Formally this may be written as

$$\begin{aligned} f_Z(z) &= \sum_{a,b,c} \int_{x=-\infty}^{\infty} \int_{y=-\infty}^{\infty} f_{XYZ|ABC}(x,y,z \,|\, a,b,c) p(a,b,c) dy dx, \\ &= \sum_{a,b,c} \int_{x=-\infty}^{\infty} f_{XZ|ABC}(x,z \,|\, a,b,c) p(a,b,c) dx \\ &= \sum_{a,b,c} f_{Z|ABC}(z \,|\, a,b,c) p(a,b,c), \end{aligned}$$

where $f_{XYZ|ABC}(x,y,z|a,b,c)$ is the multivariate normal density of the continuous variables given the values of the discrete variables, etc. Now on our elimination tree we have a representation of $f_{XYZ|ABC}(\cdot)$ in terms of the three linear regressions $\mathcal{L}(Y \,|\, X,Z,C)$, $\mathcal{L}(X \,|\, Z,A,B,C)$ and $\mathcal{L}(Z \,|\, A,B,C)$, and the last of these combined with the joint distribution $P(A,B,C)$ stored in the discrete cluster $\{ABC\}$ is sufficient to evaluate the marginal density of Z, which will thus be a mixture of normal densities.

Now suppose we wish to evaluate the marginal density of Y. Formally this is given by

$$\begin{aligned} f_Y(y) &= \sum_{a,b,c} \int_{x=-\infty}^{\infty} \int_{z=-\infty}^{\infty} f_{XYZ|ABC}(x,y,z \,|\, a,b,c) p(a,b,c) dz dx, \\ &= \sum_{a,b,c} \int_{x=-\infty}^{\infty} \int_{z=-\infty}^{\infty} f_{Y|XZC}(y \,|\, x,z,c) f_{X|ZABC}(x \,|\, z,a,b,c) \times \\ &\qquad f_{Z|ABC}(z \,|\, a,b,c) p(a,b,c) dz dx, \end{aligned} \tag{1}$$

where in (1) we have written down the factorization of the joint density as available on the tree. We wish to evaluate this by local message passing. To do this we rearrange the current factorization into a more suitable form. First we take the pair of densities $f_{Y|XZC}(y \,|\, x,z,c)$, $f_{X|ZABC}(x \,|\, z,a,b,c)$ and use the EXCHANGE operation to rewrite these as the pair $f_{X|YZABC}(x \,|\, y,z,a,b,c)$, $f_{Y|ZABC}(y \,|\, z,a,b,c)$

2. Or so in theory! In implementing the algorithms described in this paper, the author came across the problem that when subtracting one zero double precision from another, the result was zero but with the negative bit set, and so was treated by the compiled program as a negative number! Tracking down this 'bug' took the author a couple of days.

which leaves the product density unchanged. Now integrating over X (which gives unity) leaves the following expression to evaluate:

$$f_Y(y) = \sum_{a,b,c} \int_{z=-\infty}^{\infty} f_{Y|ZABC}(y|z,a,b,c)\, f_{Z|ABC}(z|a,b,c)\, p(a,b,c)dz,$$

We now apply the EXCHANGE operation again to the two densities in this expression to yield

$$f_Y(y) = \sum_{a,b,c} \int_{z=-\infty}^{\infty} f_{Z|YABC}(z|y,a,b,c)\, f_{Y|ABC}(y|a,b,c)\, p(a,b,c)dz,$$

in which the Z integral gives unity, leaving the desired marginal

$$f_Y(y) = \sum_{a,b,c} f_{Y|ABC}(y|a,b,c)\, p(a,b,c).$$

This sequence may be described in terms of a local message passing as follows: (i) take the regression stored in the cluster $\{YXZC\}$ and pass it to $\{XZABC\}$; (ii) perform the EXCHANGE operation on the pair of regressions stored in $\{XZABC\}$; (iii) take the resulting regression that has Y in the head and pass it it to $\{ZABC\}$; (iv) perform the EXCHANGE operation on the pair of regressions now stored in $\{ZABC\}$; (v) combine the resulting regression that has Y in the head with the joint distribution $P(A,B,C)$ that may be obtained from the boundary set to yield the marginal of Y.

This process of rearranging the regressions involving Y is an example of the PUSH operation introduced by Lauritzen and Jensen (2001). (Their PUSH operation can act more generally on a group of variables, but because we are working with an elimination tree we only require it to act on one variable at a time.) We say that the variable Y has been PUSHED to the boundary. The PUSH operation leaves unchanged the overall joint density of the continuous variables conditional on the discrete variables.

Lauritzen and Jensen also used the PUSH operation to incorporate evidence on continuous variables, and we shall follow them. Thus suppose that we wish to evaluate the marginal density of Z given $Y = y^*$. The first step is to PUSH the variable Y to the boundary, and when it arrives there substitute $Y = y^*$ in all cluster sets in which it appears: In the cluster neighbouring the boundary Y appears in the head and the tail is empty of continuous variables, and so this yields a likelihood term for each configuration of the discrete variables that is passed to (that is, multiplied into the discrete potential stored in) the boundary set. The tree then stores the following representations:

Cluster	Regressions and Distributions
ABC	$P(A,B,C,Y=y^*)$
$ZABC$	$\mathcal{L}(Z\|A,B,C,Y=y^*)$
$XZABC$	$\mathcal{L}(X\|A,B,C,Z,Y=y^*)$
$YXZC$	–

We may now take the CG regressions stored in $\{ZABC\}$ and combine them with the marginal $P(A,B,C,Y=y^*)$ to obtain, after normalization, the marginal density of Z given the evidence. To obtain the marginal density of X, we would PUSH the regressions in $\{XZABC\}$ to the boundary. Evidence about the discrete variables may also be entered, but only on the discrete part of the tree. For evidence on several continuous variables it is convenient to enter the evidence one variable

at a time. For example, to enter additional evidence that $X = x^*$, we first PUSH the regression $\mathcal{L}(X \mid A,B,C,Z,Y = y^*)$ to the boundary, and then substitute $X = x^*$, yielding the representation:

Cluster	Regressions and Distributions
ABC	$P(A,B,C,X = x^*,Y = y^*)$
$ZABC$	$\mathcal{L}(Z \mid A,B,C,X = x^*,Y = y^*)$
$XZABC$	–
$YXZC$	–

from which the marginal of Z given the evidence may be found. Notice that the likelihood of the evidence is found, as usual, as the normalization constant when all evidence has been collected to the strong root.

5. The General Local Propagation Scheme

The example in the previous section has illustrated the main steps in initialization and evidence propagation, and evaluation of marginal densities of variables. However the example is too simple to exhibit all of the subtleties involved in the general procedure. In this section we assume that we have a strong elimination tree, and that it is connected. If the tree is disconnected, then the scheme described below may be applied to each connected component separately. The algorithms will also work for a strong semi-elimination tree, and in implementations this might be preferred on efficiency grounds; no details of the following algorithms depend on which of the two types of tree is used.

5.1 Some Notation and Terminology

Let us suppose that the original conditional-Gaussian Bayesian network has $n > 0$ continuous variables $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ and $m > 0$ discrete variables $\Delta = \{\delta_1, \ldots, \delta_m\}$. Also suppose that the variables have been numbered so that the elimination ordering of the continuous variables to make the strong elimination tree is $(\gamma_n, \gamma_{n-1}, \ldots, \gamma_1)$. The ordering of the discrete variables is unimportant for our purposes, except that it should lead to a computationally efficient tree. We denote the set of continuous cluster sets by $\mathcal{C}_\Gamma$, with $C_\Gamma(i) \in \mathcal{C}_\Gamma$ denoting the cluster set associated with eliminating the continuous variable γ_i. Let $\mathcal{S}_\Gamma$ denote the separators adjacent to continuous cluster sets, with $S_\Gamma(i) \in \mathcal{S}_\Gamma$ denoting the separator between $C_\Gamma(i)$ and its neighbouring cluster set in the direction of the strong root. Note that if the neighbouring cluster is part of the boundary (that is, purely discrete), then $S_\Gamma(i)$ will be purely discrete. We will denote the set of purely discrete clusters by $\mathcal{C}_\Delta$, and the set of separators between purely discrete clusters by $\mathcal{S}_\Delta$.

In the present propagation scheme, the conditional distribution of the continuous variables given the discrete variables is maintained in factored form by sets of univariate regressions. The messages passed between the continuous clusters consists of such sets. To facilitate discussion of this we introduce data structures to store such sets of regressions.

In each continuous cluster we retain, for each configuration of the discrete variables in the cluster, two list structures to store zero or more linear regressions, one that we call the *postbag*, the other we call the *lp-potential* (*lp* is short for *linear predictor*). On the separators $\mathcal{S}_\Gamma$ we retain only the *postbag*. On the clusters $\mathcal{C}_\Delta$ and separators $\mathcal{S}_\Delta$ of the discrete part of the tree we store the usual tables (called potentials) of discrete junction tree propagation algorithms.

5.2 Pre-initializing the Tree

After constructing the tree from the Bayesian network, the first task is to allocate the conditional probability tables and CG regressions to the clusters of the tree. We adopt a more restrictive allocation than Lauritzen and Jensen (2001) in that we restrict where the probability tables are allocated. The allocation scheme is as follows:

Algorithm 5.1 *Allocation of potentials*

Pre-initialize tree potentials

In each continuous cluster set $C_\Gamma(i) \in \mathcal{C}_\Gamma$*:*

- *Set each* lp-potential *to empty;*
- *Set each* postbag *to an empty list.*

In each $S_\Gamma(i) \in \mathcal{S}_\Gamma$*: set each* postbag *to an empty list.*

Set all table entries to unity in all potentials of the clusters $\mathcal{C}_\Delta$ *and separators* $\mathcal{S}_\Delta$ *of the discrete part of the tree.*

Allocation of probability tables

For each $X \in \Delta$*: multiply* $P(X \mid pa(X))$ *into the potential of any one discrete cluster of* $\mathcal{C}_\Delta$ *that contains* $X \cup pa(X)$*.*

Allocation of CG regressions

For each $X \in \Gamma$*, find a cluster,* $C_\Gamma(i)$ *say, which contains* $X \cup pa(X)$*, and:*

- *IF the elimination node* $\gamma(i)$ *of* $C_\Gamma(i)$ *is* X*, then add* $\mathcal{L}(X \mid pa(X))$ *to the* lp-potential*; OTHERWISE append* $\mathcal{L}(X \mid pa(X))$ *to the* postbag *of* $C_\Gamma(i)$*.*

Under this allocation the discrete part of the tree contains all of the conditional (and unconditional) probability tables of the discrete variables, and the product of the potentials in the discrete clusters yields the joint marginal distribution of the discrete variables. In the continuous part of the tree are contained all of the CG regressions of the continuous variables in the Bayesian network, hence the lists in the clusters in the continuous part of the tree represent (in factored form) the joint multivariate density of the continuous variables given the discrete variables.

Applying this to the example of Section 4, both the *postbag* and *lp-potential* of $\{ZABC\}$ were empty, the *postbag* of $\{XZABC\}$ was empty but its *lp-potential* stored $\mathcal{L}(X \mid A,B)$, whilst for $\{YXZC\}$ the *postbag* contained $\mathcal{L}(Z \mid Y,C)$ and the *lp-potential* stored $\mathcal{L}(Y \mid X,C)$.

Note that the continuous leaves of the elimination tree will have non-empty *lp-potential*s, because for each such leaf, its associated elimination node appears nowhere else in the tree, and so there is no other cluster where the conditional density of that variable can be allocated.

5.3 The EXCHANGE Operation

The basic formula required in the message passing scheme of this paper is the EXCHANGE *operation*, which is essentially Bayes' theorem. Let $Y, Z, W_1, \ldots W_l$ be continuous variables (where Y and Z do not refer to the variables in our example), and $a_0, a_1, \ldots a_l$, b, $c_0, c_1, \ldots c_l, \sigma_Y$ and $\sigma_{Z|Y}$ be constants, with $b \neq 0$, such that we have the pair of CG regressions:

$$\begin{aligned} Z \,|\, Y, W_1, \ldots W_l &\sim N(a_0 + a_1 W_1 + \cdots + a_l W_l + bY, \sigma^2_{Z|Y}), \\ Y \,|\, W_1, \ldots W_l &\sim N(c_0 + c_1 W_1 + \cdots + c_l W_l, \sigma^2_Y). \end{aligned}$$

Then the EXCHANGE operator converts these into the pair of distributions

$$Y \,|\, Z, W_1, \ldots W_l \sim N(\cdot,\cdot) \text{ and } Z \,|\, W_1, \ldots W_l \sim N(\cdot,\cdot), \tag{2}$$

that have the same joint density as the original pair. For convenience we introduce and define $W_0 \equiv 1$. We show in Appendix A that

$$Z \,|\, W_1, \ldots W_l \sim N\left(\sum_{i=0}^{l} (a_i + bc_i) W_i, \sigma^2_{Z|Y} + b^2 \sigma^2_Y\right).$$

For the conditional distribution of Y there are three cases to consider.
Case 1: $\sigma^2_Y > 0$ and $\sigma^2_{Z|Y} > 0$:

$$Y \,|\, Z, W_1, \ldots W_l \sim N\left(\frac{\sum_{i=0}^{l} \left(c_i \sigma^2_{Z|Y} - a_i b \sigma^2_Y\right) W_i + b\sigma^2_Y Z}{\sigma^2_{Z|Y} + b^2 \sigma^2_Y}, \frac{\sigma^2_Y \sigma^2_{Z|Y}}{\sigma^2_{Z|Y} + b^2 \sigma^2_Y}\right).$$

Case 2: $\sigma^2_Y > 0$ and $\sigma^2_{Z|Y} = 0$:

$$Y \,|\, Z, W_1, \ldots W_l \sim N\left(\frac{Z - \sum_{i=0}^{l} a_i W_i}{b}, 0\right).$$

Case 3: $\sigma^2_Y = 0$ and $\sigma^2_{Z|Y} \geq 0$:

$$Y \,|\, Z, W_1, \ldots W_l \sim N\left(\sum_{i=0}^{l} c_i W_i, 0\right).$$

The derivation of these formulae is straightforward and is given in Appendix A. Case 1 is equivalent to Theorem 1 of Shachter and Kenley (1989), but differing in that they use the central-moment representation of multivariate Gaussian distributions, whereas the EXCHANGE operation (like Lauritzen and Jensen (2001)) employs the raw-moment representation. Cases 2 and 3 may be obtained as mathematical limits of Case 1, however computer implementations would require these to be treated separately.

Finally, if $b = 0$ (so that $\mathcal{L}(Z \,|\, Y, W_1, \ldots W_l)$ does not depend on Y), the EXCHANGE operation merely permutes the two regressions, with Y and Z disappearing from the conditioning sets of the two regressions.

5.4 Initial Transformation of the Tree: An Example

Having allocated potentials according to Algorithm 5.1, we proceed to complete the initialization phase via local message passing to yield, on the continuous part of the tree, univariate regressions in the *lp-potential*s and empty *postbag*s, so representing, in set-chain form on the continuous part of the tree, the joint density of the continuous variables conditional on the discrete variables. In the example of Section 4 no *postbag* contained more than one regression (for each discrete configuration); in larger and more complicated networks this might not happen, and so within each continuous cluster a sequence of EXCHANGE operations may be necessary. When this happens care must be taken in the ordering of such operations. We illustrate this in an example before giving the general procedure.

The example we shall use is given as Example 3 in Lauritzen and Jensen (2001), and is shown in Figure 4. This example is chosen because Lauritzen and Jensen show that in their initialization phase several layers of recursive combinations of potentials are required. We shall see how this arises and is avoided in our propagation scheme.

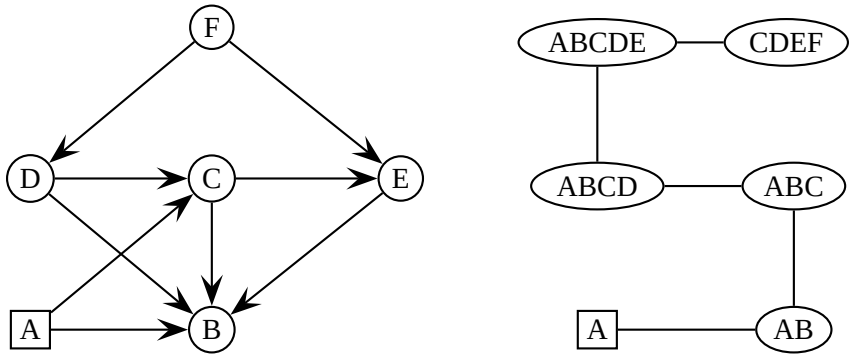

Figure 4: Mixed Bayesian network (left) with one discrete variable A, given as Example 3 in Lauritzen and Jensen (2001). In the strong elimination tree (right) the strong root is a, and the two sets $\{CDEF\}$ and $\{ABCDE\}$ would by themselves form a strong junction tree with $\{ABCDE\}$ as the strong root.

We shall use the same initial allocation of regressions as Lauritzen and Jensen, which means that $\mathcal{L}(F\,|\,-)$ is put into the *lp-potential* of cluster $\{CDEF\}$, and in its *postbag* we place $\mathcal{L}(E\,|\,CF)$ and $\mathcal{L}(D\,|\,F)$. In the *postbag* of cluster $\{ABCDE\}$ we place the regressions $\mathcal{L}(C\,|\,DA)$ and $\mathcal{L}(B\,|\,ACDE)$. The *lp-potential* of cluster $\{ABCDE\}$ is empty, as are the *lp-potential*s and *postbag*s of the three remaining continuous clusters $\{ABCD\}$, $\{ABC\}$ and $\{AB\}$. The discrete distribution $P(A)$ is allocated to the strong root $\{A\}$. This allocation is displayed in the following table.

Cluster	*lp-potential*	*postbag*
$\{CDEF\}$	$\mathcal{L}(F\,\vert -)$	$\mathcal{L}(E\,\vert CF)$, $\mathcal{L}(D\,\vert F)$
$\{ABCDE\}$	-	$\mathcal{L}(C\,\vert DA)$, $\mathcal{L}(B\,\vert ACDE)$
$\{ABCD\}$	-	-
$\{ABC\}$	-	-
$\{AB\}$	-	-
$\{A\}$	$P(A)$	

Our aim is to transform the *lp-potential* of cluster $\{CDEF\}$ into $\mathcal{L}(F\,\vert CDE)$, (because F is the elimination variable for this cluster) using the original *lp-potential*, the *postbag* contents, and EXCHANGE operations. This corresponds to arc-reversal operations on the two directed edges $F \rightarrow D$ and $F \rightarrow E$ in the original Bayesian network. Now if we first reverse the arc $F \rightarrow E$, then this will give rise to an illegal directed-cycle in the Bayesian network ($E \rightarrow F \rightarrow D \rightarrow C \rightarrow E$); this is avoided if we first reverse the arc $F \rightarrow D$. Hence our sequence of EXCHANGE operations is summarized by

$$\begin{aligned}\underbrace{\mathcal{L}(F\,\vert -),\mathcal{L}(D\,\vert F)},\mathcal{L}(E\,\vert CF) &\rightarrow \mathcal{L}(D\,\vert -),\underbrace{\mathcal{L}(F\,\vert D),\mathcal{L}(E\,\vert CF)}\\ &\rightarrow \mathcal{L}(D\,\vert -),\mathcal{L}(E\,\vert CD),\mathcal{L}(F\,\vert CDE),\end{aligned}$$

in which the pairings of the potentials in each EXCHANGE operation is indicated. We now retain $\mathcal{L}(F\,\vert CDE)$ in the *lp-potential* of cluster $\{CDEF\}$ and pass the regressions $\mathcal{L}(D\,\vert -)$ and $\mathcal{L}(E\,\vert CD)$ to the cluster $\{ABCDE\}$. Now because the variable E is the elimination variable in this cluster, this means that $\mathcal{L}(E\,\vert CD)$ is put into the *lp-potential* and $\mathcal{L}(D\,\vert -)$ is put into the *postbag*. Actually, because this cluster contains the discrete variable A, there is an *lp-potential* and a *postbag* for each configuration of A. Hence we really put a copy of $\mathcal{L}(E\,\vert CD)$ into each such *lp-potential*, and similarly a copy of $\mathcal{L}(D\,\vert -)$ into each such *postbag*. This corresponds to a trivial extension of the regressions to $\mathcal{L}(E\,\vert ACD) \equiv \mathcal{L}(E\,\vert CD)$ and $\mathcal{L}(D\,\vert A) \equiv \mathcal{L}(D\,\vert -)$. (In the following we shall assume for brevity that we are working with a particular configuration of A, to avoid repeating the phrase "for each configuration of A".)

So now in the cluster $\{ABCDE\}$ we have the following regressions stored:

lp-potential	$\mathcal{L}(E\,\vert ACD)$
postbag	$\mathcal{L}(D\,\vert A)$, $\mathcal{L}(C\,\vert DA)$, $\mathcal{L}(B\,\vert ACDE)$

In our desired set-chain representation we wish to end up with $\mathcal{L}(E\,\vert ABCD)$ in the *lp-potential*; we may use the following sequence of EXCHANGE operations,

$$\begin{aligned}\underbrace{\mathcal{L}(E\,\vert ACD),\mathcal{L}(D\,\vert A)},\mathcal{L}(C\,\vert DA),&\mathcal{L}(B\,\vert ACDE)\\ &\rightarrow \mathcal{L}(D\,\vert A),\underbrace{\mathcal{L}(E\,\vert ACD),\mathcal{L}(C\,\vert DA)},\mathcal{L}(B\,\vert ACDE)\\ &\rightarrow \mathcal{L}(D\,\vert A),\mathcal{L}(C\,\vert DA),\underbrace{\mathcal{L}(E\,\vert ACD),\mathcal{L}(B\,\vert ACDE)}\\ &\rightarrow \mathcal{L}(D\,\vert A),\mathcal{L}(C\,\vert DA),\mathcal{L}(B\,\vert ACD),\mathcal{L}(E\,\vert ABCD),\end{aligned}$$

in which the first two operations merely permute the regressions, (C and D are already conditioning variables of the regressions of E), and only the last is an application of Bayes' theorem. From this set, $\mathcal{L}(E\,\vert ABCD)$ is retained in the *lp-potential* of $\{ABCDE\}$, $\mathcal{L}(D\,\vert A)$ is put into the *lp-potential* of $\{ABCD\}$ (because D is the elimination variable of this cluster), and $\mathcal{L}(C\,\vert DA)$ and $\mathcal{L}(B\,\vert ACD)$ are put into the *postbag* of $\{ABCD\}$. Hence in the cluster $\{ABCD\}$ we have:

lp-potential	$\mathcal{L}(D\,|\,A)$
postbag	$\mathcal{L}(B\,|\,ACD)$, $\mathcal{L}(C\,|\,DA)$

Now D appears in the tails of both regressions in the *postbag*, hence care must be taken. The correct EXCHANGE sequence is:

$$\begin{aligned}\underbrace{\mathcal{L}(D\,|\,A),\mathcal{L}(C\,|\,DA)},\mathcal{L}(B\,|\,ACD) &\rightarrow \mathcal{L}(C\,|\,A),\underbrace{\mathcal{L}(D\,|\,AC),\mathcal{L}(B\,|\,ACD)} \\ &\rightarrow \mathcal{L}(C\,|\,A),\mathcal{L}(B\,|\,AC),\mathcal{L}(D\,|\,ABC)\end{aligned}$$

$\mathcal{L}(D\,|\,ABC)$ is retained in the cluster $\{ABCD\}$, $\mathcal{L}(C\,|\,A)$ is put into the *lp-potential* of the cluster $\{ABC\}$, and $\mathcal{L}(B\,|\,AC)$ is put into its *postbag*.

In the cluster $\{ABC\}$ we apply the EXCHANGE operation,

$$\mathcal{L}(C\,|\,A),\mathcal{L}(B\,|\,AC) \rightarrow \mathcal{L}(B\,|\,A),\mathcal{L}(C\,|\,AB),$$

retaining $\mathcal{L}(C\,|\,AB)$ in the *lp-potential* and putting $\mathcal{L}(B\,|\,A)$ into the *lp-potential* of cluster $\{AB\}$, and we are done.

Before proceeding in the next section to give the general initialization algorithm, we need to explain how we choose the ordering exchange operations to be performed when a *postbag* contains more than one regression. The answer is straightforward: one orders the regressions in a *postbag* by a topological ordering in the original Bayesian network of the response (head) variables in the regressions, such that in the ordering all the parents of a variable X appear to the left of X, and all child variables appear to the right of X. This ordering ensures that the sequence of EXCHANGE operations is valid. (It is essentially Proposition 2 of Shachter and Kenley (1989).)

5.5 Initial Transformation of the Tree: General Algorithm

We now present the general algorithm for initializing the CG regressions of the tree. Recall that γ_i is the elimination variable associated with the continuous cluster $C_\Gamma(i)$.

Algorithm 5.2 (Initialization of CG regressions)

1. *Given:*
   - *A mixed Bayesian network $\mathcal{B}$ with a strong elimination tree initialized according to Algorithm 5.1.*
   - *The elimination sequence $\gamma_n,\gamma_{n-1},\ldots,\gamma_1$ of the continuous nodes.*
   - *A topological ordering* TOP *of the variables in $\mathcal{B}$.*
2. *Message passing sequence:*

   For i := n step -1 until 1 do:

   For each configuration δ_i^* of the discrete variables in $C_\Gamma(i)$ do:
   - *Sort the regressions in the δ_i^** postbag *of $C_\Gamma(i)$ so that their head variables occur in the same sequence as in the topological ordering* TOP*;*
   - *While the δ_i^** postbag *of $C_\Gamma(i)$ is not empty do:*

- *Remove the first regression R in the* δ_i^* postbag *of* $C_\Gamma(i)$*;*
- *If the tail of R contains* γ_i*, then modify R and the* δ_i^* lp-potential *of* $C_\Gamma(i)$ *via the* EXCHANGE *operation, such that R does not depend on* γ_i*.*
- *For each configuration* $\delta_{j_i}^* \supseteq \delta_i^*$*, of the discrete variables in the cluster* $C_\Gamma(j_i)$ *neighbouring* $C_\Gamma(i)$ *in the direction of the strong root, put a copy of R either (1) in the* $\delta_{j_i}^*$ lp-potential *of* $C_\Gamma(j_i)$ *if the head of R is the elimination variable of* $C_\Gamma(j_i)$*, or (2) in the* $\delta_{j_i}^*$ postbag *of* $C_\Gamma(j_i)$ *otherwise.*
- *Discard R.*

Note that by the strong nature of the tree, the set of discrete variables of $C_\Gamma(i)$ is contained in the set of discrete variables of $C_\Gamma(j_i)$. Hence for each configuration $\delta_{j_i}^*$ there will be an induced sub-configuration δ_i^*.

On completion of Algorithm 5.2 all of the *postbag*s are empty, and the *lp-potential*s contain the desired set-chain CG regressions. To see this, first note that each EXCHANGE operation is a valid application of Bayes' theorem, and does not at any stage alter the joint conditional density that can be reconstructed from the regressions stored in the *lp-potential*s and *postbag*s. Secondly, when the outer *i*-th loop is performed the *lp-potential* of $C_\Gamma(i)$ is not empty, either $C_\Gamma(i)$ is a leaf cluster, in which case it started out non empty; otherwise it will have started out non empty and remained so, or it will have started out empty but then became non-empty because it received a regression from a neighbouring cluster away from the root. (The conditional density of γ_i given its parents must have been placed either in the *lp-potential* of $C_\Gamma(i)$, or in some *postbag* of a cluster on some path from $C_\Gamma(i)$ through clusters further away from the root than $C_\Gamma(i)$, and will arrive, possibly modified via EXCHANGE operations at $C_\Gamma(i)$ through application of the previous steps of the algorithm for higher values of *i*.)

5.6 Entering Evidence and the PUSH Operation

Evidence on discrete and/or continuous variables may be entered and propagated on the tree, and posterior distributions of individual nodes found. Discrete evidence is entered and propagated in the usual way, *but only on the discrete part of the tree*. To enter continuous evidence, and to find marginal densities of continuous variables, we use the PUSH operation of Lauritzen and Jensen (2001).

It is convenient to enter evidence on the continuous variables one observed variable at a time. In order to keep track of those variables that have already had their evidence entered, in each continuous cluster we retain a boolean variable—called *activeflag*—which is initially set to TRUE before any continuous evidence has been entered. A value of FALSE indicates that evidence on the elimination variable of the cluster has been entered. A TRUE value indicates that evidence concerning the elimination variable may be entered, or that the marginal density of the variable may be calculated.

Recall that the separator $S_\Gamma(i)$ between a continuous cluster set $C_\Gamma(i)$ that is a neighbour to a discrete cluster in C_Δ only contains those discrete variables in $C_\Gamma(i)$. In every such separator we store a table of real values indexed by the states of the discrete variables which we call a *weight* table.

In the algorithm below the cluster neighbouring $C_\Gamma(i)$ in the direction of the strong root will be denoted by $toroot(i)$, and is either another continuous cluster, or a purely discrete boundary cluster.

If $toroot(i)$ is continuous its index is denoted by r_i (so that $r_i \in \{1,2,\ldots,i-1\}$ and $toroot(i) \equiv C_\Gamma(r_i)$).

We now state the algorithm for entering evidence on a continuous variable γ_j (indexed according the elimination ordering) which consists of the finding that $\gamma_j = \gamma_j^*$. Basically it performs a sequence of EXCHANGE operations so that in the final regression in which γ_j is the head variable, no continuous variables appear in the tail; the substitution $\gamma_j = \gamma_j^*$ may then be performed in all regressions in which γ_j appears, and where it is the head variable this substitution forms a likelihood to be incorporated into the discrete part of the tree.

Algorithm 5.3 (The PUSH operation: Entering evidence $\gamma_j = \gamma_j^*$)

- ***Given**:*
  - *A tree with elimination sequence $\gamma_n, \gamma_{n-1}, \ldots, \gamma_1$ of the continuous variables.*
  - *The tree initialized according to Algorithm 5.2, and then possibly having had some evidence already entered and propagated—but not on the variable γ_j—according to this algorithm. (Note: all* postbags *are empty.)*
  - *Evidence $\gamma_j = \gamma_j^*$ to enter and propagate on the tree.*
- ***Step 1**: Enter $\gamma_j = \gamma_j^*$ in all regressions in which γ_j is a conditioning variable.*
  - *For $i := n$ step -1 until $j+1$ do*
    - *IF* activeflag *of $C_\Gamma(i)$ is TRUE and $\gamma_j \in C_\Gamma(i)$ substitute $\gamma_j = \gamma_j^*$ in every* lp-potential *regression of $C_\Gamma(i)$.*
- ***Step 2**: Initialize the loop for pushing γ_j towards a boundary cluster.*
  - *Set $i := j$.*
  - *For each configuration δ_i^* of the discrete variables of $C_\Gamma(i)$ move the regression in the δ_i^** lp-potential *of $C_\Gamma(i)$ into the δ_i^** postbag *of $C_\Gamma(i)$.*
  - *Set* activeflag *of $C_\Gamma(i)$ to FALSE.*
- ***Step 3**: Push γ_j towards a boundary cluster.*

  while $toroot(i)$ is not a boundary cluster do:
  - *For each configuration $\delta_{r_i}^*$ of the discrete variables of $C_\Gamma(r_i)$ (with induced configuration δ_i^* of the discrete variables of $C_\Gamma(i)$) do:*
    - *IF* activeflag *of $toroot(i)$ is FALSE, then:*
      1. *Copy the δ_i^** postbag *of $C_\Gamma(i)$ into the $\delta_{r_i}^*$* postbag *of $C_\Gamma(r_i)$;*

      OTHERWISE:
      1. *Copy the δ_i^** postbag *of $C_\Gamma(i)$ into the $\delta_{r_i}^*$* postbag *of $C_\Gamma(r_i)$*
      2. *Perform the* EXCHANGE *operation on the $\delta_{r_i}^*$* lp-potential *and* postbag *of $C_\Gamma(r_i)$ (such that the resulting* postbag *regression has γ_j as head).*
      3. *Substitute $\gamma_j = \gamma_j^*$ in the $\delta_{r_i}^*$* lp-potential *of $C_\Gamma(r_i)$;*
  - *Discard the content of every* postbag *of $C_\Gamma(i)$;*
  - *Set $i := r_i$.*

- ***Step 4****: Update the discrete part of the tree.*

    1. *For each configuration* δ_i^* *of the discrete variables of* $C_\Gamma(i)$ *substitute* $\gamma_j = \gamma_j^*$ *into the density of the regression stored in the* δ_i^* postbag *of* $C_\Gamma(i)$ *and store the result in the* δ_i^* *entry of the weight table of* $S_\Gamma(i)$.
    2. *Multiply the* weight *table of* $S_\Gamma(i)$ *into the discrete cluster potential of* $toroot(i)$*;*
    3. *Empty the* postbag *of* $C_\Gamma(i)$.

Substituting $\gamma_j = \gamma_j^*$ into a regression in which γ_j is in the tail, and γ_v is the head variable will result in another regression of γ_v on the remaining variables of the tail, affecting the expression for the mean. In the more complex substitutions of Step 4(1), there is a regression in the δ_i^* *postbag* of $C_\Gamma(i)$ of the form $\gamma_j \sim N(a_{i,j}, \sigma_{i,j}^2)$, so that the δ_i^* entry in the *weight* table will store

$$w[\delta_i^*] = \frac{\exp\left(-(\gamma_j^* - a_{i,j})^2 / 2\sigma_{i,j}^2\right)}{\sqrt{2\pi\sigma_{i,j}^2}}.$$

If $\sigma_{i,j}^2 = 0$, which could (though not necessarily must) occur if there are logical relationships on the continuous variables, this could be mathematically undefined (Lauritzen and Jensen (2001) provide an example). Should such an event occur an implementation of Algorithm 5.3 should warn the user.

To see how the algorithm works, suppose that we start with no evidence having been entered. In this case the continuous part of the tree stores a representation of the conditional density of the continuous variables given the discrete, $\mathcal{L}(\Gamma \,|\, \Delta)$. When evidence $\gamma_j = \gamma_j^*$ is entered on a continuous variable γ_j, the sequence of EXCHANGE operations and substitutions of $\gamma_j = \gamma_j^*$ leads to a factored representation $\mathcal{L}(\Gamma \setminus \{\gamma_j\} \,|\, \gamma_j^*, \Delta), \mathcal{L}(\gamma_j^* \,|\, \Delta)$ where the first term is represented in the *lp-potential*s of the continuous clusters in which the *activeflag* is TRUE, and the last term is passed as a likelihood term via a weight table to the discrete part of the tree. After standard evidence propagation on the discrete part of the tree, the latter stores a representation of $P(\Delta \,|\, \gamma_j = \gamma_j^*, \delta^*)$ or $P(\Delta, \gamma_j = \gamma_j^*, \delta^*)$ depending on whether or not the discrete potentials have been normalized to unity (δ^* represents discrete evidence). If a second piece of evidence is entered, $\gamma_k = \gamma_k^*$ say, the algorithm leads to the active continuous clusters storing a factored representation of $\mathcal{L}(\Gamma \setminus \{\gamma_j, \gamma_k\} \,|\, \gamma_j = \gamma_j^*, \gamma_k = \gamma_k^*, \Delta)$ and the further likelihood factor $\mathcal{L}(\gamma_k = \gamma_k^* \,|\, \gamma_j = \gamma_j^*, \Delta)$ being multiplied into the discrete part of the tree. At each stage the *activeflag*s which are FALSE in the continuous clusters identify, via the elimination variables of the clusters, those continuous variables about which evidence has been entered, the *lp-potential*s in the other clusters (in which the *activeflag*s are TRUE) represent the joint density of the unobserved continuous variables given the discrete variables and the observed continuous variables.

After all continuous evidence has been entered, and evidence on discrete variables has been entered and propagation performed on the discrete part of the tree, the discrete part contains a factored representation of the posterior probability of the unobserved discrete variables given the evidence on all variables.

We illustrate the algorithm by revisiting the example in Section 4. We start with no evidence on the continuous part of the tree, which is initialized thus:

Cluster	*lp-potentials*
$ZABC$	$Z \mid abc \sim \mathcal{N}(\alpha_{Z:abc}, \sigma^2_{Z:abc})$
$XZABC$	$X \mid zabc \sim \mathcal{N}(\alpha_{X:abc} + \beta_{X:Zabc} z, \sigma^2_{X:abc})$
$YXZC$	$Y \mid xzc \sim \mathcal{N}(\alpha_{Y:c} + \beta_{Y:Xc} x + \beta_{Y:Zc} z, \sigma^2_{Y:c})$

If $X = x^*$ is observed, then Algorithm 5.3 operates as follows:

1. In the cluster $\{YXZC\}$, the distribution $Y \mid xzc \sim \mathcal{N}(\alpha_{Y:c} + \beta_{Y:Xc} x + \beta_{Y:Zc} z, \sigma^2_{Y:c}) \rightarrow Y \mid x^* zc \sim \mathcal{N}(\alpha_{Y:c} + \beta_{Y:Xc} x^* + \beta_{Y:Zc} z, \sigma^2_{Y:c})$. The *activeflag* remains TRUE.

2. In $\{XZABC\}$, the *activeflag* is set to FALSE, and for each configuration *abc* the regression $\mathcal{N}(\alpha_{X:abc} + \beta_{X:Zabc} z, \sigma^2_{X:abc})$ is moved into the *abc postbag* of $\{ZABC\}$

3. In $\{ZABC\}$, for each configuration *abc* the EXCHANGE operations acts on the *abc lp-potential*, representing $\mathcal{L}(Z \mid A = a, B = b, C = c)$, and the *abc postbag* to give a new *abc lp-potential* representing $\mathcal{L}(Z \mid X = x, A = a, B = b, C = c)$ and modified *abc postbag* content representing $\mathcal{L}(X = x \mid A = a, B = b, C = c)$. Then the *lp-potential* is set to $\mathcal{L}(Z \mid X = x^*, A = a, B = b, C = c)$, and the weight table entry $w(a,b,c)$ stores the density value of $\mathcal{L}(X = x^* \mid A = a, B = b, C = c)$.

4. The *postbag*s of $\{ZABC\}$ are emptied, the *activeflag* remains TRUE.

5. In the discrete cluster $\{ABC\}$, the potential $p(a,b,c) \rightarrow p(a,b,c) w(a,b,c) \; \forall \; \{a,b,c\}$.

6. On normalizing the potential in the discrete cluster $\{ABC\}$ we obtain the probability distribution $P(A,B,C \mid X = x^*)$.

5.7 Evaluating Posterior Marginals of Individual Variables

After propagating evidence on variables as described above, finding the posterior marginal of a discrete variable, D say, proceeds in the usual way: Find a cluster set in the discrete part of the tree containing D and marginalize the joint table in that cluster set appropriately.

Finding the posterior density of an unobserved continuous variable uses the PUSH operation in a way similar to but simpler than Algorithm 5.3. Suppose the marginal density of $Y \in \Gamma$ is required. The idea is to use a sequence of EXCHANGE operations to push Y to a cluster C neighbouring the boundary, so that we have a representation of the distribution $\mathcal{L}(Y \mid \mathcal{E}_\Gamma, B)$ where $\mathcal{E}_\Gamma$ denotes the evidence on the continuous variables, and $B \subseteq \Delta$ are the discrete variables in the cluster C. From the boundary cluster the marginal $P(B \mid \mathcal{E}_{\Gamma \cup \Delta})$ may be found and then combined with $\mathcal{L}(Y \mid \mathcal{E}_\Gamma, B)$ to give the required posterior marginal of Y. The complete algorithm is given in Algorithm 5.4, which uses the same notation as Algorithm 5.3.

Algorithm 5.4 (Find the posterior density of a continuous variable)

- ***Given**:*
  - *A strong elimination tree with elimination sequence $\gamma_n, \gamma_{n-1}, \ldots, \gamma_1$ of the continuous variables.*

- *The tree initialized according to Algorithm 5.2, with evidence $\mathcal{E}_\Gamma$ entered as in Algorithm 5.3, and discrete evidence $\mathcal{E}_\Delta$ entered and propagated on the discrete part, so that the discrete clusters contain posterior distributions. (Note: all* postbag*s are empty.)*
- *Task: to find the posterior density of an unobserved continuous variable γ_j. (Note: the* activeflag *of $C_\Gamma(j)$ is TRUE.)*

- ***Step 1:*** *Initialize the loop.*
  - *For each configuration δ_j^* of the discrete variables of $C_\Gamma(j)$ copy the δ_j^** lp-potential *of $C_\Gamma(j)$ into the δ_j^** postbag *of $C_\Gamma(j)$.*
  - *Set $i := j$*

- ***Step 2:*** *Push γ_j towards a boundary cluster.*

  while $toroot(i)$ is not a boundary cluster do:
  - *For each configuration $\delta_{r_i}^*$ of the discrete variables of $C_\Gamma(r_i)$ (with induced configuration δ_i^* of the discrete variables of $C_\Gamma(i)$) do:*
    - *Copy the δ_i^** postbag *of $C_\Gamma(i)$ into the $\delta_{r_i}^*$* postbag *of $C_\Gamma(r_i)$;*
    - *IF* activeflag *of $C_\Gamma(r_i)$ is TRUE and γ_j is in the tail of the regression in the $\delta_{r_i}^*$* lp-potential *of $C_\Gamma(r_i)$, THEN use the* EXCHANGE *operation formulae to modify the $\delta_{r_i}^*$* postbag *of $C_\Gamma(r_i)$ (so that γ_j is still the head variable) but do not modify the $\delta_{r_i}^*$* lp-potential *of $C_\Gamma(r_i)$;*
  - *Empty all of the* postbag*s of $C_\Gamma(i)$;*
  - *Set $i := r_i$.*

- ***Step 3:*** *Find the marginal density.*
  1. *Marginalize the discrete potential in the boundary cluster neighbouring $C_\Gamma(i)$ to the* weight *table in the separator $S_\Gamma(i)$.*
  2. *Output the result of adding together the product of each* weight *table entry with the density of the regression stored in the corresponding* postbag *of $C_\Gamma(i)$.*
  3. *Empty all of the* postbag*s of $C_\Gamma(i)$.*

Prior to an application of this algorithm the *lp-potential*s in the active clusters represent a factorization of the joint conditional density of the unobserved continuous variables given the evidence on the continuous variables. The algorithm does not change these in any way, and so does not alter this joint conditional density, and indeed the algorithm leaves the tree ready in a state for finding the marginal density of another continuous variable. The algorithm is just using the *postbag*s as temporary storage to find, by repeated (partial) application of the EXCHANGE formulae, the marginal density of γ_j conditional on the discrete variables and the values of the observed continuous variables. Step 3 combines this with the correct posterior probability of the unobserved discrete variables of $C_\Gamma(i)$ (conditional on all evidence) to form the posterior marginal density of γ_j.

6. Comparison to Other Methods

In this section we review the propagation scheme described in this paper, and compare it to the scheme of Lauritzen and Jensen (2001) and to the work of Shachter and Kenley (1989). We then discuss some possible extensions of the scheme.

6.1 Summary of Current Scheme

In the current scheme, evidence propagation and evaluation of marginal distributions in a mixed Bayesian network $\mathcal{B}$ takes place on a strong elimination tree or strong semi-elimination tree. The tree has two distinct parts, the continuous part and the discrete part, with the strong root located in the discrete part. The continuous part is initialized using the CG regressions of $\mathcal{B}$, and represents, using a collection of univariate regressions, the density of the continuous variables conditional on the discrete variables. The discrete part represents the marginal distribution of the discrete variables, and is initialized using the discrete conditional probability tables of $\mathcal{B}$. Entering evidence on continuous variables, and evaluating marginal densities of continuous variables, uses the PUSH operation with the EXCHANGE formulae, the latter being an application Bayes' theorem. Discrete evidence is entered on the discrete part of the tree and propagated on the discrete part of the tree in the usual way. For finding marginals of continuous variables there is no DISTRIBUTE operation on the tree.

6.2 Comparison with the Lauritzen and Jensen (2001) Scheme

As discussed in Section 2 the Lauritzen and Jensen propagation scheme uses a strong junction tree architecture. Associated with each clique of the junction tree is a CG potential, which is a tuple $\phi = [p,A,B,C](H\,|\,T)$ of scalars, vectors and matrices and a partition of the variables into conditioned variables (the head) and conditioning variables (the tail). The conditional distribution or density of a variable X in $\mathcal{B}$ may be assigned to any clique or separator that contains the family of X in the Bayesian network. CG potentials may be combined but there are restrictions that have to be followed, which necessitate the introduction of the *recursive combination* of potentials to allow incoming messages to a clique to be combined correctly.

It is instructive to see how recursive combination is avoided in the current scheme, or alternatively, how to interpret recursive combination within the current scheme. For this we return to the example in Section 5.4. In Figure 4 the clusters $\{CDEF\}$ and $\{ABCDE\}$ form a strong junction tree with the latter as the strong root. In the Lauritzen and Jensen analysis, the assignment of potential to clique $\{CDEF\}$ leads to a CG potential having the head and tail structure $(DEF\,|\,C)$. This is decomposed into $(F\,|\,CED)$ and $(DE\,|\,C)$ the latter is passed to the clique $\{ABCDE\}$ to be combined with the potential $(BC\,|\,DE)$. However the heads' and tails' contents of these two potentials preclude their direct combination. Instead they must be combined recursively. The first stage is to decompose $(DE\,|\,C) \rightarrow (E\,|\,CD),(D\,|\,-)$, however this is not sufficient, as $(E\,|\,CD)$ cannot be directly combined with $(BC\,|\,DE)$. So we decompose $(BC\,|\,DE) \rightarrow (B\,|\,CDE),(C\,|\,D)$ and then we may combine the four potentials $(B\,|\,CDE)(E\,|\,CD)(C\,|\,D)(D\,|\,-)$ *in that order* to yield a potential $(BCDE\,|\,-)$.

If one compares these four potentials with the regressions stored in the cluster $\{ABCDE\}$ in Section 5.4 we see that they have the same head-and-tail structures. In the current scheme the regressions $\mathcal{L}(D\,|\,-),\mathcal{L}(E\,|\,CD)$ were passed to the cluster $\{ABCDE\}$, which stored (omitting the dependence on A) the regressions $\mathcal{L}(C\,|\,D),\mathcal{L}(B\,|\,CDE)$. These are subject to EXCHANGE operation,

dependent on the topological ordering of the head variables in the Bayesian network $\mathcal{B}$, the ordering being $D-C-E-B$.

Thus we interpret the recursive combination of potentials as a ordered factorization of potentials so their direct combinations are well defined (although it will not always decompose potentials to univariate CG potentials). The current scheme avoids the recursive combination operation because it works all the time with a factored representation, and the correct ordering of EXCHANGE operations is ensured by using the topological ordering of the variables in $\mathcal{B}$. It has echoes of *lazy-propagation* (Madsen and Jensen, 1998), in which potentials are stored in factored form when initializing a junction tree and are only combined when required; the difference is that in our scheme factored forms are always retained.

In our scheme there is no SUM-DISTRIBUTE operation on the continuous part of the tree, whereas Lauritzen and Jensen have such an operation that can be used to store weak-marginals in the separators. If weak-marginals are desired in the current scheme, they may be found using the mean and variance of the mixture marginals.

Another aspect of the Lauritzen and Jensen scheme should be mentioned, which is their operation of *minimization* of the tails of CG potentials. In their operations on CG potentials represented by tuples $\phi = [p,A,B,C](H \mid T)$, when a column of B has entries all zero for every configuration of the discrete variables in the CG potential, then that column and the associated continuous tail variable may be removed. This is a *reduction* operation, and takes place during recursive combination. In the current scheme, reduction occurs as a result of the EXCHANGE operation: the Z regression of (2) does not depend on Y, so an implementation of the EXCHANGE operation would, in taking this into account, automatically perform a reduction. The *minimization* of a potential occurs if the potential has been reduced as far as possible.

Lauritzen and Jensen also discuss the possibility of forming the marginal of a group of continuous variables. This should be possible within the present scheme, with the result being expressed as weighted sets of linear regressions. However it may be that in the message passing process more than one regression might be stored in a *postbag* (for a given configuration of discrete variables) and if so their order would be important, not however the topological ordering of the original Bayesian network variables used in Algorithm 5.2, but the (reverse of) the elimination ordering. This would be appropriate because it is a perfect numbering of the strongly triangulated graph associated with the tree. Similar considerations suggests it ought to be possible to propagate evidence on several continuous variables simultaneously. These connections are discussed in further detail in the next subsection.

6.3 Relationship to the Shachter and Kenley (1989) Scheme

In the Shachter and Kenley scheme, arc-operations are performed on a Bayesian network one pair of variables at a time, which means that it operates on pairs of linear regressions, which is like the current scheme. (Their paper is concerned with pure Gaussian networks, but this is difference is not very significant.) When several arcs need reversing in a Probabilistic Node Reduction operation (their PROPOSITION 2) the sequence of arc reversals has to follow an ordering which is equivalent to one obtained from a topological ordering of the nodes in the Bayesian network. Hence this is very similar to the sequence of EXCHANGE operations in initializing the tree described in Algorithm 5.2.

The close connections between the current scheme, and of both Shachter and Kenley (1989) and Lauritzen and Jensen (2001) are illuminated by the paper of Shachter et al. (1990). These

authors show that the inference algorithms operating on junction trees are essentially the same as node reductions algorithms operating on influence diagrams, because they are both working on the same underlying graphical structures. In the terms of the present paper, what they show is that in the elimination sequence $\gamma_n, \gamma_{n-1}, \ldots, \gamma_1$ operating on the moralized mixed Bayesian network, the cluster $C_\Gamma(i)$ is the same as that which would be obtained in the influence diagram for family of the node γ_i after all outgoing arcs have been reversed to make γ_i a barren node, if the nodes are removed from the mixed Bayesian network in the same order as the elimination sequence. To avoid directed cycles being introduced there is a restriction to the order in reversing the child arcs of a node, the restriction uses a topological ordering of the original Bayesian network. They also describe how evidence is entered—if evidence is entered on a node then arcs are reversed so that that node has no parents, and in the process child arcs from the node are removed. When this is done the likelihood associated with that node may be found. This process—which they call *evidence propagation*—is essentially the PUSH operation on a continuous variable, except that arcs are reversed only to the point that the variable has no continuous parents, and then substituting the evidence value leads to modification of those regressions in which the variable appears in the tail (corresponding to removal of those arcs from the influence diagram viewpoint) and evaluation of a likelihood to be passed on to the discrete part of the tree. The elimination tree, with its *lp-potential* and *postbag* structures, provides an organizing framework for such operations. They also describe how multiple evidence may be entered simultaneously, and their procedure should be transferable into the current scheme.

6.4 Implementation Issues

The propagation scheme presented here works by manipulating linear regressions. To aid the presentation the paper has used, like Lauritzen and Jensen (2001), a raw moment representation, for which the EXCHANGE formulae of Section 5.3 may be used. If the user wishes to implement the current scheme then one possibility would be to represent CG regressions by the tuple $\phi = [p, A, B, C](H \mid T)$ of Lauritzen and Jensen, but now A and C are scalars and B a vector (corresponding to $r = 1$) for each configuration of the discrete variables. Further restrictions are that the table p is a table of 1's if the CG regression has continuous variables, and hence are not required.

However this is not the only possibility. In the author's C++ implementation, an associative array of the form `std::map<variable,double>` is used to store coefficients of covariates in the regressions. The reduction operation is effected by a variable being removed from the associative array.

One could instead use the central-moment representation of Shachter and Kenley, all that would be required would be suitable replacements of the EXCHANGE formulae which describe Bayes' theorem. In this case the formulae in Theorem 1 of Shachter and Kenley could be used (suitably extended to take account of deterministic relationships and the configurations of the discrete variables).

A further possibility could be to use computer algebra. Each univariate regression is specified by (1) a specification of the head and tail variables and either (2a) a quadratic form in the continuous variables if the variance is strictly positive, or (2b) a linear form for deterministic relationships. These are readily represented and manipulated in computer algebra packages, and so the current scheme could be implemented in which the messages are either linear expressions or quadratic forms in the continuous variables. Reduction operations would be taken care of automatically by such computer algebra calculations.

From these comments above we see that, although the presentation in this paper has focussed on using the raw-moment representation, the propagation scheme is more general than this. The only place that the raw-moment representation has been used is in the explicit EXCHANGE formulae of Section 5.3.

7. Sampling and Mode-Finding Algorithms

Aside from finding the posterior marginals of variables, the use of the elimination tree in the current scheme facilitates other operations that may be of interest in applications. In Section 7.1 we show how to sample from the posterior distribution, and in Section 7.2 we show how to find the highest peak in the posterior mixture distribution, which could be useful as the starting point of an iterative search for the posterior mode.

7.1 Sampling from the Posterior

It may be desirable to sample from the posterior distribution of the variables in the Bayesian network given some observed evidence $\mathcal{E}_{\Gamma\cup\Delta}$. Dawid (1992) has shown how to do this for discrete networks, his method is as follows. Starting from a junction tree of cliques, and after entering and sum-propagating evidence $\mathcal{E}_\Delta$, the clique and separator tables contain posterior marginals of their variables. Suppose we label the cliques in running intersection order $C_1, C_2, \ldots, C_k$ say, with C_1 being chosen as the root clique. First one samples from the posterior marginal in C_1, to give some configuration δ_1^s. Then one samples from the posterior marginal of the variables in C_2 conditional on δ_1^s, yielding some combined configuration $\delta_1^s \cup \delta_2^s$. Then one samples the variables in C_3 conditional on $\delta_1^s \cup \delta_2^s$. Proceeding in this way will yield a sample $\delta^s = \delta_1^s \cup \delta_2^s \cup \cdots \cup \delta_k^s$ from the posterior distribution on the junction tree.

Here we present in Algorithm 7.1 a simple extension of Dawid's method to conditional-Gaussian networks. The idea is to sample the discrete variables, and then sample the remaining continuous variables one at a time in a distribute-type operation.

Algorithm 7.1 (Sampling from the posterior)

- ***Given:*** *A tree with elimination sequence $\gamma_n, \gamma_{n-1}, \ldots, \gamma_1$ of the continuous variables, which has been initialized and has had evidence propagated according to Algorithm 5.3, and any discrete evidence has also been propagated on the discrete part of the tree.*
- ***Sample:***
  - *Sample a configuration δ^s of the discrete variables Δ using the algorithm of Dawid.*
  - *For $i := 1$ step 1 until n do*
    - *IF* activeflag *of $C_\Gamma(i)$ is TRUE then find the sub-configuration $\delta_i^s \subseteq \delta^s$ of the discrete variables in $C_\Gamma(i)$, and sample γ_i from the regression stored in the δ_i^s* lp-potential *of $C_\Gamma(i)$ in which the sampled or observed values of all continuous tail variables $\in \{\gamma_1^s, \gamma_2^s, \ldots, \gamma_{i-1}^s\}$ have been substituted; denote the sampled value by γ_i^s.*
    - *OTHERWISE there is evidence $\gamma_i = \gamma_i^*$, so simply set $\gamma_i^s = \gamma_i^*$.*
- *The configuration $\{\gamma_i^s : i = 1, \ldots, n\} \cup \delta^s$ is a sample from the posterior density.*

Note that, because of the elimination tree structure employed, each simulated γ_i is sampled from a univariate normal distribution. Such simulation may be done efficiently by a variety of methods and is much simpler than sampling from a multivariate normal distribution.

7.2 Locating the Mode (Approximately)

It is sometimes of interest to locate the most probable values of the unobserved variables given evidence on observed variables. For discrete networks this may be found by local propagation as shown by Dawid (1992). For CG networks an exact solution to the problem is not known, and we do not propose a solution here. Instead in Algorithm 7.2 we propose a method that finds the component having the *highest peak* in the posterior distribution. The posterior distribution is a mixture of weighted multivariate normal densities. Each component multivariate density will attain maximum height at its mean, the height will be proportional to the weight of the component divided by the square root of the determinant of the covariance matrix of the component. These heights are compared by Algorithm 7.2. Now if the variances of the components in the mixture are small compared to the distances between their means, then the location of the component having the highest peak might be expected to be a good approximation to the posterior mode, or could be used as the starting point of an iterative search for the mode. Algorithm 7.2 is slightly more complicated than Algorithm 7.1, and cannot be used if any variances are zero. In order to keep track of the heights of each component, we need to keep a *weight* table in every continuous separator in S_Γ. Note that it is not necessary to evaluate a determinant.

Algorithm 7.2 (Highest Component Search)

- ***Given:***
  - *A tree with elimination sequence $\gamma_n, \gamma_{n-1}, \ldots, \gamma_1$ of the continuous variables, which has been initialized and had evidence propagated according to Algorithm 5.3, and any discrete evidence also been propagated on the discrete part of the tree.*
  - *All entries in all* weight *tables initialized to unity.*
- ***Highest Peak Search:***
  - *For $i := n$ step -1 until 1 do*
    1. *IF* activeflag *of $C_\Gamma(i)$ is TRUE THEN: For each configuration δ_i^* of the discrete variables in $C_\Gamma(i)$ multiply the δ_i^* entry in the* weight *table of $S_\Gamma(i)$ by $1/\sqrt{2\pi\sigma^2(\delta_i^*)}$ where $\sigma^2(\delta_i^*)$ is the variance in the regression stored in the δ_i^** lp-potential *of $C_\Gamma(i)$.*
    2. *IF $toroot(i)$ is NOT a boundary cluster, THEN: For each configuration $\delta_{r_i}^*$ of the discrete variables of $C_\Gamma(r_i)$ (with induced configuration δ_i^* of the discrete variables of $C_\Gamma(i)$), multiply the $\delta_{r_i}^*$ entry in the weight table of $S_\Gamma(r_i)$ by the δ_i^* entry in the weight table of $S_\Gamma(i)$.*
       OTHERWISE if $toroot(i)$ IS a boundary cluster, then multiply the weight table of $S_\Gamma(i)$ into the discrete potential of $toroot(i)$ in the usual way.
  - *Use the* MAX-PROPAGATE *algorithm of Dawid (1992) on the discrete part of the tree to give a* "max-configuration" δ^m *of the discrete variables.*
  - *For $i := 1$ step 1 until n do*

- *IF* activeflag *of* $C_\Gamma(i)$ *is FALSE, then there is evidence* $\gamma_i = \gamma_i^*$*, so set the peak value* $\gamma_i^m = \gamma_i^*$*;*
  OTHERWISE activeflag *of* $C_\Gamma(i)$ *is TRUE, so find the sub-configuration* $\delta_i^m \subseteq \delta^m$ *of the discrete variables in* $C_\Gamma(i)$*, and set* γ_i *to the mean of the regression stored in the* δ_i^m lp-potential *of* $C_\Gamma(i)$ *in which the peak values or observed values of all continuous tail variables* $\in \{\gamma_1^m, \gamma_2^m, \ldots, \gamma_{i-1}^m\}$ *have been substituted; denote the peak value by* γ_i^m*.*

- *The configuration* $\{\gamma_i^m : i = 1, \ldots, n\} \cup \delta^m$ *specifies the location of the highest component in the joint multivariate posterior density.*

Mathematically Algorithm 7.2 may be understood in the following manner. Let $I_\Gamma = \{I_1, I_2, \ldots, I_k\}$ denote the set of indices of the continuous nodes for which no evidence has been entered (with $I_k > I_{k-1} > \cdots > I_1$). Then the algorithm starts out with the tree storing a recursive factorization of the posterior multivariate normal mixture density in the form:

$$p(\Delta \,|\, \mathcal{E}_{\Gamma\cup\Delta}) \prod_{i \in I_\Gamma} f_i(\gamma_i \,|\, S_\gamma(i), \mathcal{E}_\Gamma)$$

where the $f_i(\cdot\,|\,\cdot)$ are appropriate CG regression densities. Given the covariates and evidence, each CG regression density is maximized at the mean, so the component having the maximum height may be obtained as a sequence of ordered maximizations:

$$\begin{aligned}
& \max_{\Gamma,\Delta} \left(p(\Delta \,|\, \mathcal{E}_{\Gamma\cup\Delta}) \prod_{j=1}^{k} f_{I_j}(\gamma_{I_j} \,|\, S_\gamma(I_j), \mathcal{E}_\Gamma) \right) \\
= & \max_{\Gamma,\Delta} \left(p(\Delta \,|\, \mathcal{E}_{\Gamma\cup\Delta}) w_{I_k}(S_\gamma(I_k), \mathcal{E}_\Gamma) \prod_{j=1}^{k-1} f_{I_j}(\gamma_{I_j} \,|\, S_\gamma(I_j), \mathcal{E}_\Gamma) \right) \\
= & \max_{\Gamma,\Delta} \left(p(\Delta \,|\, \mathcal{E}_{\Gamma\cup\Delta}) w_{I_k}(S_\gamma(I_k), \mathcal{E}_\Gamma) w_{I_{k-1}}(S_\gamma(I_{k-1}), \mathcal{E}_\Gamma) \prod_{j=1}^{k-2} f_{I_j}(\gamma_{I_j} \,|\, S_\gamma(I_j), \mathcal{E}_\Gamma) \right) \\
& \vdots \\
= & \max_{\Delta} \left(p(\Delta \,|\, \mathcal{E}_{\Gamma\cup\Delta}) \prod_{j=1}^{k} w_{I_j}(S_\gamma(I_j), \mathcal{E}_\Gamma) \right),
\end{aligned}$$

where the $w_i(\cdot)$ are the *weight* tables representing the values of the densities of the CG regressions located at the means. The algorithm accumulates the product of these values and multiplies them into the discrete part of the tree, from which standard MAX-PROPAGATION may be used to find δ^m. (Note that this accumulated product is valid because of the strong nature of the tree: a continuous cluster $C_\Gamma(r_i)$ neighbouring another continuous cluster $C_\Gamma(i)$ but closer to the strong root will contain all of the discrete variables that are in $C_\Gamma(i)$.) This information is then distributed back to locate the mean values of the continuous variables for the configuration δ^m of the discrete variables, in the final stage of Algorithm 7.2.

Finally, we mention another maximization operation called SEMIMAX-PROPAGATION that can easily be carried out in the current scheme. This consists of finding the most likely configuration

of the posterior marginal distribution of the discrete variables, that is, finding the maximum of $P(\Delta \,|\, \mathcal{E}_{\Gamma\cup\Delta})$. For this we propagate evidence as in Algorithm 5.3, and incorporate evidence on the discrete variables on the discrete part of the tree. We then perform standard MAX-PROPAGATION on the discrete part of the tree according to the algorithm of Dawid (1992). SEMIMAX-PROPAGATION was introduced and applied to forensic DNA problems involving the modelling and analysis of mixed DNA samples using CG networks by Cowell et al. (2004).

8. Summary

We have presented a local propagation scheme for conditional Gaussian Bayesian networks based on elimination trees, that combines the scheme of Lauritzen and Jensen (2001) with that of Shachter and Kenley (1989). Complex matrix algebra is avoided because operations manipulate linear regressions. The propagation scheme is not dependent on a particular implementation of the representation of linear regressions, although the paper has used one for exposition.[3] We have also introduced: an algorithm for sampling on such networks; an algorithm for finding highest peaks that could be useful either as an approximation to, or an iterative algorithm for locating, the posterior mode of the distribution; and have briefly described another operation called SEMIMAX-PROPAGATION.

Acknowledgments

The author would like to thank anonymous referees for their very helpful comments and suggestions for making improvements to this paper.

Appendix A. Derivation of EXCHANGE Formulae of Section 5.3

From the pair of normal distributions

$$\begin{aligned} Z \,|\, Y, W_1, \ldots W_l &\sim N(a_0 + a_1 W_1 + \ldots a_l W_l + bY, \sigma^2_{Z\,|\,Y}), \\ Y \,|\, W_1, \ldots W_l &\sim N(c_0 + c_1 W_1 + \ldots c_l W_l, \sigma^2_Y), \end{aligned}$$

it follows that $Y \,|\, Z, W_1, \ldots W_l$ and $Z \,|\, W_1, \ldots W_l$ are also normal distributions. The mean and variance of the latter is readily found using

$$\begin{aligned} E[Z \,|\, W_1, \ldots W_l] &= E[E[Z \,|\, Y, W_1, \ldots W_l]] = E[a_0 + a_1 W_1 + \ldots a_l W_l + bY] \\ &= \sum_{i=0}^{l} (a_i + bc_i) W_i \\ V[Z \,|\, W_1, \ldots W_l] &= E[V[Z \,|\, Y, W_1, \ldots W_l]] + V[E[Z \,|\, Y, W_1, \ldots W_l]] \\ &= E[\sigma^2_{Z\,|\,Y}] + V[a_0 + a_1 W_1 + \ldots a_l W_l + bY] \\ &= \sigma^2_{Z\,|\,Y} + b^2 \sigma^2_Y \end{aligned}$$

where we define $W_0 \equiv 1$.

There are three cases to consider in finding the conditional distribution of $Y \,|\, Z, W_1, \ldots W_l$.

3. It is also one implemented by the author.

Case 1: $\sigma_Y > 0$ **and** $\sigma^2_{Z|Y} > 0$**.**

The joint conditional density of $Y, Z \,|\, W_1, \ldots, W_l$ is

$$\begin{aligned} f_{Y,Z|W_1,\ldots,W_l}(y,z) &= f_{Z|Y,W_1,\ldots,W_l}(y,z)\, f_{Y|W_1,\ldots,W_l}(y) \\ &= \frac{1}{2\pi\sigma_{Z|Y}\sigma_Y} \exp\left(\frac{-(z-\sum_{i=0}^{l} a_i W_i - by)^2}{2\sigma^2_{Z|Y}}\right) \exp\left(\frac{-(y-\sum_{i=0}^{l} c_i W_i)^2}{2\sigma^2_Y}\right) \\ &= f_{Y|Z,W_1,\ldots,W_l}(y,z)\, f_{Z|W_1,\ldots,W_l}(z) \\ &= \frac{1}{2\pi\sigma_{Y|Z}\sigma_Z} \exp\left(\frac{-(y-\alpha-\beta z)^2}{2\sigma^2_{Y|Z}}\right) \exp\left(\frac{-(z-\sum_{i=0}^{l}(a_i+bc_i)W_i)^2}{2\sigma^2_Z}\right), \end{aligned}$$

where α, β and $\sigma^2_{Y|Z}$ are constants to be determined. The density of $Y \,|\, Z, W_1, \ldots, W_l$ may be found directly by dividing the first expression for the joint density by the density of $Z \,|\, W_1, \ldots, W_l$ (Bayes' theorem), or alternatively it may be deduced from the linear and quadratic terms in y in the exponential terms as follows. Equating the coefficients of y^2 yields

$$\frac{1}{2\sigma^2_{Y|Z}} = \frac{b^2}{2\sigma^2_{Z|Y}} + \frac{1}{2\sigma^2_Y}$$

from which it follows that the desired variance is

$$\sigma^2_{Y|Z} = \frac{\sigma^2_{Z|Y}\sigma^2_Y}{\sigma^2_{Z|Y} + b^2\sigma^2_Y}.$$

Equating the terms linear in y yields

$$\frac{\alpha+\beta z}{\sigma^2_{Y|Z}} = \frac{b\left(z-\sum_{i=0}^{l} a_i W_i\right)}{\sigma^2_{Z|Y}} + \frac{\sum_{i=0}^{l} c_i W_i}{\sigma^2_Y}$$

hence the conditional mean $\alpha + \beta Z$ is given by

$$\left(\frac{b(Z-\sum_{i=0}^{l} a_i W_i)}{\sigma^2_{Z|Y}} + \frac{\sum_{i=0}^{l} c_i W_i}{\sigma^2_Y}\right) \Bigg/ \left(\frac{\sigma^2_{Z|Y} + b^2\sigma^2_Y}{\sigma^2_{Z|Y}\sigma^2_Y}\right) = \frac{\sum_{i=0}^{l}\left(c_i\sigma^2_{Z|Y} - a_i b\sigma^2_Y\right) W_i + b\sigma^2_Y Z}{\sigma^2_{Z|Y} + b^2\sigma^2_Y}$$

Thus,

$$Y \,|\, Z, W_1, \ldots W_l \sim N\left(\frac{\sum_{i=0}^{l}\left(c_i\sigma^2_{Z|Y} - a_i b\sigma^2_Y\right) W_i + b\sigma^2_Y Z}{\sigma^2_{Z|Y} + b^2\sigma^2_Y}, \frac{\sigma^2_Y\sigma^2_{Z|Y}}{\sigma^2_{Z|Y} + b^2\sigma^2_Y}\right),$$

Case 2: $\sigma_Y > 0$ **and** $\sigma^2_{Z|Y} = 0$**.**

We may deduce that

$$Y \,|\, Z, W_1, \ldots W_l \sim N\left(\frac{Z-\sum_{i=0}^{l} a_i W_i}{b}, 0\right)$$

either by considering the limit $\sigma^2_{Z|Y} \to 0$ in Case 1, or by noting that if $\sigma^2_{Z|Y} = 0$, then

$$Z \,|\, Y, W_1, \ldots W_l \sim N(a_0 + a_1 W_1 + \ldots a_l W_l + bY, \sigma^2_{Z|Y})$$

is equivalent to

$$Z = a_0 + a_1 W_1 + \cdots + a_l W_l + bY$$

which is equivalent to the constraint

$$Y = \frac{Z - a_0 - a_1 W_1 - \cdots - a_l W_l}{b}.$$

Case 3: $\sigma_Y = 0$ and $\sigma^2_{Z|Y} \geq 0$.

We may obtain

$$Y \,|\, Z, W_1, \ldots W_l \sim N\left(\frac{Z - \sum_{i=0}^{l} a_i W_i}{b}, 0\right)$$

either as the limit $\sigma^2_Y \to 0$ of Case 1, or by noting that the deterministic constraint implied by

$$Y \,|\, W_1, \ldots W_l \sim N\left(\sum_{i=0}^{l} c_i W_i, 0\right) \equiv Y = \sum_{i=0}^{l} c_i W_i$$

will be unaffected by further conditioning on Z.

References

A. Becker and D. Geiger. A sufficiently fast algorithm for finding close to optimal clique trees. *Artificial Intelligence Journal*, 2001.

R. G. Cowell. Decision networks: a new formulation for multistage decision problems. *Research Report* **132**, Department of Statistical Science, University College London, London, United Kingdom, 1994.

R. G. Cowell, S. L. Lauritzen, and J. Mortera. Identification and separation of DNA mixtures using peak area information. Cass Statistical Science Research Report 25, City University London, 2004.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer–Verlag, 1999.

A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

U. Kjærulff. Graph triangulation — algorithms giving small total state space. *Technical Report* R 90-09, Aalborg university, Denmark, 1990.

P. Larrañaga, C. M. H. Kuijpers, M. Poza, and R. H. Murga. Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms. *Statistics and Computing*, pages 19–34, 1997.

S. L. Lauritzen. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87:1098–1108, 1992.

S. L. Lauritzen. *Graphical Models*. Oxford, United Kingdom, 1996.

S. L. Lauritzen and N. Wermuth. Mixed interaction models. Technical Report R 84-8, Institute for Electronic Systems, Aalborg University, 1984.

S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.

S. L. Lauritzen and F. Jensen. Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11:191–203, 2001.

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

H.-G. Leimer. Triangulated graphs with marked vertices. *Annals of Discrete Mathematics*, 41: 311–324, 1989.

A. L. Madsen and F. V. Jensen. Lazy propagation in junction trees. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence*, pages 362–369, San Francisco, California, 1998. Morgan Kaufmann, San Francisco, California.

K. G. Olesen and A. Madsen. Maximal prime subgraph decomposition of Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 32(21–31), 2002.

J. Pearl. *Probabilistic Inference in Intelligent Systems*. Morgan Kaufmann, San Mateo, California, San Mateo, California, 1988.

C. Raphael. Bayesian networks with degenerate Gaussian distributions. *Methodology and Computing in Applied Probability*, 5(2):235–263, 2003.

R. D. Shachter and C. Kenley. Gaussian influence diagrams. *Management Science*, 35:527–550, 1989.

R. D. Shachter, S. K. Andersen, and K. L. Poh. Directed reduction algorithms and decomposable graphs. In *In Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, July 27-29, Cambridge, MA*, pages 237–244, New York, NY, 1990. Elsevier Science Publishing Company, Inc.

R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. 13:566–579, 1984.

M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2:77–79, 1981.