To appear, *Proc. 15th International Joint Conference on AI (IJCAI-97)*, Nagoya, Japan, August, 1997

Probabilistic Partial Evaluation: Exploiting rule structure in probabilistic inference*

David Poole
Department of Computer Science
University of British Columbia
2366 Main Mall, Vancouver, B.C., Canada V6T 1Z4
poole@cs.ubc.ca
http://www.cs.ubc.ca/spider/poole

Abstract

Bayesian belief networks have grown to prominence because they provide compact representations of many domains, and there are algorithms to exploit this compactness. The next step is to allow compact representations of the conditional probability tables of a variable given its parents. In this paper we present such a representation in terms of parent contexts and provide an algorithm that exploits this compactness. The representation is in terms of rules that provide conditional probabilities in different contexts. The algorithm is based on eliminating the variables not needed in an answer in turn. The operations for eliminating a variable correspond to a form of partial evaluation, where we are careful to maintain the probabilistic dependencies necessary for correct probabilistic inference. We show how this new method can exploit more structure than previous methods for structured belief network inference.

1 Introduction

Probabilistic inference is important for many applications in diagnosis, perception, and anywhere there is uncertainty about the state of the world from observations. Belief (Bayesian) networks [Pearl, 1988] are a representation of independence amongst random variables. They are of interest because the independence is useful in many domains, they allow for compact representations of problems of probabilistic inference, and there are algorithms to exploit the compact representations.

Recently there has been work to extend belief networks by allowing more structured representations of the conditional probability of a variable given its parents. This has been in terms of either causal independencies [Heckerman and Breese, 1994; Zhang and Poole, 1996] or by exploiting finer grained contextual independencies inherent in stating the conditional probabilities in terms of rules [Poole, 1993] or trees [Boutilier *et al.*, 1996]. In this paper we show how algorithms for efficient inference in belief networks can be extended to also exploit the structure of the rule-based representations.

In the next section we introduce belief networks, a rule-based representation for conditional probabilities, and an algorithm for belief networks that exploits the network structure. We then show how the algorithm can be extended to exploit the rule-based representation. We present an example in detail and show how it is more efficient than previous proposals for exploiting structure.

2 Background

2.1 Belief Networks

A belief network [Pearl, 1988] is a DAG, with nodes labelled by random variables. We use the terms node and random variable interchangeably. Associated with a random variable x is its frame, $val(x)$, which is the set of values the variable can take on. For a variable x, let π_x be the parents of x in the belief network. Associated with the belief network is a set of probabilities of the form $P(x|\pi_x)$, the conditional probability of each variable given its parents (this includes the prior probabilities of those variables with no parents).

A belief network represents a particular independence assumption: each node is independent of its non-descendents given its parents. Suppose the variables in a belief network are $x_1, \ldots, x_n$ where the variables are ordered so that the parents of a node come before the node in the ordering. Then the independence of a belief network means that:

$$P(x_i|x_{i-1} \ldots x_1) = P(x_i|\pi_{x_i})$$

By the chain rule for conjunctions we have

$$\begin{aligned} P(x_1, \ldots, x_n) &= \prod_{i=1}^{n} P(x_i|x_{i-1} \ldots x_1) \\ &= \prod_{i=1}^{n} P(x_i|\pi_{x_i}) \end{aligned}$$

This is often given as the formal definition of a belief network.

Example 2.1 Consider the belief network of Figure 1. This represents a factorization of the joint probability distribution:

$$P(a, b, c, d, e, y, z)$$

*This work was supported by Institute for Robotics and Intelligent Systems, Project IC-7 and Natural Sciences and Engineering Research Council of Canada Research Grant OGPOO44121. Thanks to Holger Hoos and Mike Horsch for comments.

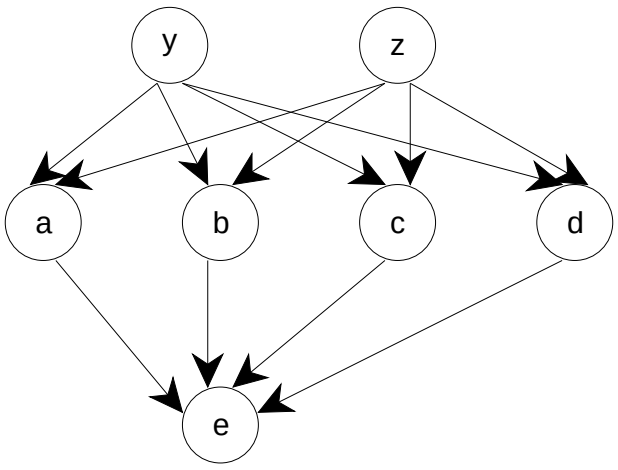

Figure 1: Simple Belief Network

$$= P(e|abcd)P(a|yz)P(b|yz) \\ P(c|yz)P(d|yz)P(y)P(z)$$

If the variables are binary, the first term, $P(e|abcd)$, requires the probability of e for all 16 cases of assignments of values to a, b, c, d.

2.2 Contextual Independence

Definition 2.2 Given a set of variables C, a **context** on C is an assignment of one value to each variable in C. Usually C is left implicit, and we simply talk about a context. Two contexts are **incompatible** if there exists a variable that is assigned different values in the contexts; otherwise they are **compatible**.

Boutilier *et al.* [1996] present a notion of contextually independent that we simplify. We use this definition for a representation that looks like a belief network, but with finer-grain independence that can be exploited.

Definition 2.3 [Boutilier *et al.*, 1996] Suppose X, Y and C are disjoint sets of variables. X and Y are **contextually independent** given context $c \in val(C)$ if $P(X|Y{=}y_1 \wedge C{=}c) = P(X|Y{=}y_2 \wedge C{=}c)$ for all $y_1, y_2 \in val(Y)$ such that $P(Y{=}y_1 \wedge C{=}c) > 0$ and $P(Y{=}y_2 \wedge C{=}c) > 0$.

Definition 2.4 Suppose we have a total ordering of variables. Given variable x_i, we say that $c \in val(C)$ where $C \subseteq \{x_{i-1} \ldots x_1\}$ is a **parent context** for x_i if x_i is contextually independent of $\{x_{i-1} \ldots x_1\} - C$ given c.

What is the relationship to a belief network? In a belief network, the rows of a conditional probability table for a variables form a set of parent contexts for the variable. However, often there is a much smaller set of smaller parent contexts that covers all of the cases.

A **minimal parent context** for variable x_i is a parent context such that no subset is also a parent context.

Example 2.5 [1]variable e are a, b, c, d, y, z. It could be the case that the set of minimal parent contexts for e are $\{\{a, b\}, \{a, \overline{b}\}, \{\overline{a}, c\}, \{\overline{a}, \overline{c}, d, b\}, \{\overline{a}, \overline{c}, d, \overline{b}\}, \{\overline{a}, \overline{c}, \overline{d}\}\}$. The probability of a given values for its predecessors can be reduced to the probability of a given a parent context. For example:

$$P(e|\overline{a}, b, c, \overline{d}, y, \overline{z}) = P(e|\overline{a}, c)$$

In the belief network, the parents of e are a, b, c, d, and would, in the traditional representation, require $2^4 = 16$ numbers instead of the 6 needed above. Adding an extra variable as a parent to e doubles the size of the table representation, but if it is only relevant in a very restricted context it may only increase the size of the rule based representation by one.

For each variable x_i and for each assignment $x_{i-1}{=}v_{i-1}, \ldots, x_1{=}v_1$ of values to its preceding variables, there is a parent context $\pi_{x_i}^{v_{i-1} \ldots v_1}$. Given this, the probability of an assignment of a value to each variable is given by:

$$\begin{aligned} & P(x_1{=}v_1, \ldots, x_n{=}v_n) \\ &= \prod_{i=1}^{n} P(x_i{=}v_n | x_{i-1}{=}v_{i-1}, \ldots, x_1{=}v_1) \\ &= \prod_{i=1}^{n} P(x_i{=}v_i | \pi_{x_i}^{v_{i-1} \ldots v_1}) \end{aligned} \quad (1)$$

This looks like the definition of belief network, but which variables act as the parents depends on the values. The numbers required are the probability of each variable for each of its minimal parent contexts. There can be many fewer minimal parent contexts that the number of assignments to parents in a belief network.

Before showing how the structure of parent contexts can be exploited in inference, there are a few properties to note:

The set of minimal parent contexts is covering, in the sense that for each assignment of values to the variables before x_i in the ordering with non-zero probability, there is a minimal context that is a subset.

The minimal parent contexts are not necessarily pairwise incompatible: it is possible to have two minimal parent contexts whose conjunction is consistent. This can only occur when the probability of the variable given the compatible contexts is the same, in which case it doesn't matter which parent context is chosen in the above formula.

The minimal parent contexts can often, but not always, be represented as a decision tree [Boutilier *et al.*, 1996] where the contexts correspond to the paths to the roots in the tree. The operations we perform don't necessarily preserve the tree structure. Section 4.1 shows how we can do much better than the analogous tree-based formulation of the algorithm.

2.3 Rule-based representations

We write the probabilities in contexts as rules,

$$x_i{=}v \leftarrow x_{i_1}{=}v_{i_1} \wedge \ldots \wedge x_{i_{k_i}}{=}v_{i_{k_i}} : p$$

where $x_{i_1}{=}v_{i_1} \wedge \ldots \wedge x_{i_{k_i}}{=}v_{i_{k_i}}$ forms a parent context of x_i and $0 \leq p \leq 1$ is a probability.

This rule can be interpreted in at least two ways:

[1]In this and subsequent examples, we assume that variables are Boolean. If x is a variable, $x = true$ is written as x and $x = false$ is written as $\overline{x}$.

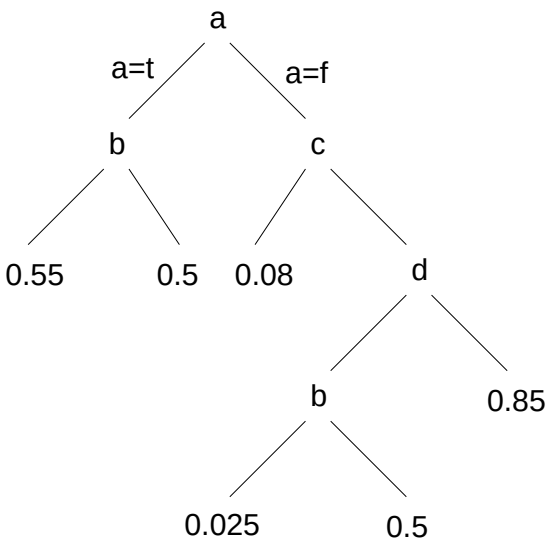

Figure 2: A tree-structured representation of the conditional probability function for e given its parents.

In the first, this rule simply means the conditional probability assertion:

$$\forall V_i \; P(x_i{=}v | x_{i_1}{=}v_{i_1} \wedge \ \ldots \wedge x_{i_{k_i}}{=}v_{i_{k_i}} \wedge Y_i{=}V_i) = p$$

where $Y_i = \{x_{i-1}, \ldots, x_1\} - \{x_{i_1}, \ldots, x_{i_{k_i}}\}$.

The second interpretation [Poole, 1993] is as a set of definite clauses, with "noise" terms in the body. The noise terms are atoms that are grouped into independent **alternatives** (disjoint sets) that correspond to random variables. In this interpretation the above rule is interpreted as the clause:

$$x_i{=}v \leftarrow x_{i_1}{=}v_{i_1} \wedge \ \ldots \wedge x_{i_{k_i}}{=}v_{i_{k_i}} \wedge n\_i^v_{v_{i_1},\ldots,v_{i_{k_i}}}$$

where $n\_i^v_{v_{i_1},\ldots,v_{i_{k_i}}}$ is a noise term, such that, for each tuple of values $\langle v_{i_1}, \ldots, v_{i_{k_i}}\rangle$, the noise terms for different values for v are grouped into an alternative, and the different alternatives are independent. $P(n\_i^v_{v_{i_1},\ldots,v_{i_{k_i}}}) = p$. This interpretation may be helpful as the operations we consider can be seen as instances of resolution on the logical formula. One of the main advantages of rules is that there is a natural first-order version, that allows for the use of logical variables.

Example 2.6 Consider the belief network of Figure 1. Figure 2 gives a tree-based representations for the conditional probability of e given its parents. In this tree, nodes are labelled with parents of e in the belief network. The left hand child corresponds to the variable being true, and the right hand node to the variable being false. The leaves are labelled with the probability that e is true. For example $P(e{=}t | a{=}t \wedge b{=}f) = 0.5$, irrespectively of the value for c or d.

These trees can be translated into rules:[2]

$$e \leftarrow a \wedge b : 0.55 \quad (2)$$
$$e \leftarrow a \wedge \overline{b} : 0.5 \quad (3)$$
$$e \leftarrow \overline{a} \wedge c : 0.08 \quad (4)$$
$$e \leftarrow \overline{a} \wedge \overline{c} \wedge d \wedge b : 0.025 \quad (5)$$
$$e \leftarrow \overline{a} \wedge \overline{c} \wedge d \wedge \overline{b} : 0.5 \quad (6)$$
$$e \leftarrow \overline{a} \wedge \overline{c} \wedge \overline{d} : 0.85 \quad (7)$$

Note that the parent contexts are exclusive and covering.

Assume the corresponding rules for b are:

$$b \leftarrow \overline{y} : 0.27 \quad (8)$$
$$b \leftarrow y \wedge z : 0.77 \quad (9)$$
$$b \leftarrow y \wedge \overline{z} : 0.17 \quad (10)$$

Definition 2.7 Suppose R is a rule

$$x_i{=}v \leftarrow x_{i_1}{=}v_{i_1} \wedge \ \ldots \wedge x_{i_{k_i}}{=}v_{i_{k_i}} : p$$

and y is a context on Y such that $\{x_i, x_{i_1}, \ldots, x_{i_{k_i}}\} \subseteq Y \subseteq \{x_1, \ldots, x_n\}$. We say that R is **applicable** in context y if y assigns v to x_i and for each i_j assigns v_{i_j} to x_{i_j}.

Lemma 2.8 If the bodies for the rules are exclusive the probability of any context on $\{x_1, \ldots, x_n\}$ is the product of the probabilities of the rules that are applicable in that context.

For each x_i, there is exactly one rule with x_i in the head that is applicable in the context. The lemma now follows from equation (1).

In general we allow conjunctions on the left of the arrow. These rules have the obvious interpretation. Section 3.2 explains where these rules arise.

2.4 Belief network inference

The aim of probabilistic inference is to determine the posterior probability of a variable or variables given some observations. In this section we outline a simple algorithm for belief net inference called VE [Zhang and Poole, 1996] or bucket elimination for belief assessment, BEBA [Dechter, 1996], that is based on the ideas of SPI [Shachter *et al.*, 1990]. This is a query oriented algorithm that exploits network structure for efficient inference, similarly to clique tree propagation [Lauritzen and Spiegelhalter, 1988; Jensen *et al.*, 1990]. One difference is the factors represent conditional probabilities rather than the marginal probabilities the cliques represent.

Suppose we want to determine the probability of variable x given evidence e which is the conjunction of assignments to some variables $e_1, \ldots, e_s$, namely $e_1{=}o_1 \wedge \ \ldots \wedge e_s{=}o_s$.

[2] We only specify the positive rules on our examples. For each rule of the form:

$$a \leftarrow b : p$$

we assume there is also a rule of the form

$$\overline{a} \leftarrow b : 1 - p$$

We maintain both as, when we have evidence (Section 3.3), they may no longer sum to one.

Then:

$$P(x|e_1{=}o_1 \wedge \ldots \wedge e_s{=}o_s) = \frac{P(x \wedge e_1{=}o_1 \wedge \ldots \wedge e_s{=}o_s)}{P(e_1{=}o_1 \wedge \ldots \wedge e_s{=}o_s)}$$

Here $P(e_1{=}o_1 \wedge \ldots \wedge e_s{=}o_s)$ is a normalizing factor. The problem of probabilistic inference can thus be reduced to the problem of computing the probability of conjunctions. Let $\{y_1, \ldots, y_k\} = \{x_1, \ldots, x_n\} - \{x\} - \{e_1, \ldots, e_s\}$, and suppose that the y_i's are ordered according to some elimination ordering. To compute the marginal distribution, we sum out the y_i's in order. Thus:

$$P(x \wedge e_1{=}o_1 \wedge \ldots \wedge e_s{=}o_s) = \sum_{y_k} \cdots \sum_{y_1} P(x_1, \ldots, x_n)_{\{e_1=o_1 \wedge \ldots \wedge e_s=o_s\}}$$

$$= \sum_{y_k} \cdots \sum_{y_1} \prod_{i=1}^{n} P(x_i|\pi_{x_i})_{\{e_1=o_1 \wedge \ldots \wedge e_s=o_s\}}$$

where the subscripted probabilities mean that the associated variables are assigned the corresponding values in the function.

Thus probabilistic inference reduces to the problem of summing out variables from a product of functions. To sum out a variable y_i from a product, we distribute all of the factors that don't involve the variable out of the sum. Suppose $f_1, \ldots, f_k$ are some functions of the variables that are multiplied together (initially these are the conditional probabilities), then

$$\sum_{y_i} f_1 \ldots f_k = f_1 \ldots f_m \sum_{y_i} f_{m+1} \ldots f_k$$

where $f_1 \ldots f_m$ are those functions that don't involve y_i, and $f_{m+1} \ldots f_k$ are those that do involve y_i. We explicitly construct a representation for the new function $\sum_{y_i} f_{m+1} \ldots f_k$, and continue summing out the remaining variables. After all the y_i's have been summed out, the result is a function on x that is proportional to x's posterior distribution.

Unfortunately space precludes a more detailed description; see [Zhang and Poole, 1996; Dechter, 1996] for more details.

3 Probabilistic Partial Evaluation

Partial evaluation [Lloyd and Shepherdson, 1991] is a technique for removing atoms from a theory. In the simple case for non-recursive theories, we can, for example partially evaluate b, in the clauses:

$$e \leftarrow b \wedge a$$
$$b \leftarrow y \wedge z$$

by resolving on b resulting in the clause:

$$e \leftarrow y \wedge z \wedge a$$

The general idea of the structured probabilistic inference algorithm is to represent conditional probabilities in terms of rules, and use the VE algorithm with a form of partial evaluation to sum out a variable. This returns a new set of clauses. We have to ensure that the posterior probabilities can be extracted from the reduced rule set.

The units of manipulation are finer grained than the factors in VE or the buckets of BEBA; what is analogous to a factor or a bucket consists of sets of rules. Given a variable to eliminate, we can ignore (distribute out) all of the *rules* that don't involve this variable.

The input to the algorithm is: a set of rules representing a probability distribution, a query variable, a set of observations, and an elimination ordering on the remaining variables.

At each stage we maintain a set of rules with the following **program invariant**:

> The probability of a context on the non-eliminated variables can be obtained by multiplying the probabilities associated with rules that are applicable in that context. Moreover for each assignment, and for each non-eliminated variable there is only one applicable rule with that variable in the head.

The algorithm is made up of the following primitive operations that locally preserve this program invariant:[3]

Variable partial evaluation (VPE). Suppose we are eliminating e, and have rules:

$$a \leftarrow b \wedge e : p_1 \quad (11)$$
$$a \leftarrow b \wedge \overline{e} : p_2 \quad (12)$$

such that there are no other rules that contain e in the body whose context is compatible with b. For each rule for e:

$$e \leftarrow h : p_3 \quad (13)$$
$$\overline{e} \leftarrow h : p_4 \quad (14)$$

where b and h are compatible contexts, we create the rule:

$$a \leftarrow b \wedge h : p_1 p_3 + p_2 p_4 \quad (15)$$

This is done for all pairs of rules with e in the head and body. The original rules that contain e are removed.

To see why this is correct, consider a context c that includes a, b, and h, but doesn't give a value for e. Then $P(c) = P(c \wedge e) + P(c \wedge \overline{e})$. $P(c \wedge e) = pP(a|b \wedge e)P(e|h)$, for some product p of terms that don't include e. Similarly $P(c \wedge \overline{e}) = pP(a|b \wedge \overline{e})P(\overline{e}|h)$, for the same value p. Thus we have $P(c) = p(p_1 p_3 + p_2 p_4)$. Because of the structure of Rule (15), it is only chosen for contexts with a, b, and h true, and it is the only rule with head a in such contexts.

Rule Splitting. If we have a rule

$$a \leftarrow b : p_1 \quad (16)$$

We can replace it by its split on variable d, forming rules:

$$a \leftarrow b \wedge d : p_1 \quad (17)$$
$$a \leftarrow b \wedge \overline{d} : p_1 \quad (18)$$

[3] To make this presentation more readable we assume that each variable is Boolean. The extension to the multi-valued case is straightforward. Our implementation uses multi-valued variables.

Combining Heads. If we have two rules:

$$a \leftarrow c : p_1 \quad (19)$$

$$b \leftarrow c : p_2 \quad (20)$$

such that a and b refer to different variables, we can combine them producing:

$$a \wedge b \leftarrow c : p_1 p_2 \quad (21)$$

Thus in the context with a, b, and c all true, the latter rule can be used instead of the first two. We show why we may need to do this in Section 3.2.

In order to see the algorithm, let's step through some examples to show what's needed and why.

Example 3.1 Suppose we want to sum out b given the rules in Example 2.6. b has one child e in the belief network, and so b only appears in the body of rules for e. Of the six rules for e, two don't contain b (rules (4) and (7)), and so remain. The first two rules that contain b can be treated separately from the other two as they are true in different contexts. VPE of rules (2) and (3) with rule (8), results in:

$$e \leftarrow a \wedge \overline{y} : 0.27 \times 0.55 + (1 - 0.27) \times 0.5$$

Summing out b results in the following representation for the probability of e. (You can ignore these numbers, it is the structure of the probability tables that is important.)

$$e \leftarrow a \wedge \overline{y} : 0.5135$$
$$e \leftarrow a \wedge y \wedge z : 0.5385$$
$$e \leftarrow a \wedge y \wedge \overline{z} : 0.5085$$
$$e \leftarrow \overline{a} \wedge c : 0.08$$
$$e \leftarrow \overline{a} \wedge \overline{c} \wedge d \wedge \overline{y} : 0.37175$$
$$e \leftarrow \overline{a} \wedge \overline{c} \wedge d \wedge y \wedge z : 0.13425$$
$$e \leftarrow \overline{a} \wedge \overline{c} \wedge d \wedge y \wedge \overline{z} : 0.41925$$
$$e \leftarrow \overline{a} \wedge \overline{c} \wedge \overline{d} : 0.85$$

Thus we need 16 rules (including rules for the negations) to represent how e depends on its parents once b is summed out. This should be contrasted with the table of size 64 that is created for VE or in clique tree propagation.

3.1 Compatible Contexts

The partial evaluation needs to be more sophisticated to handle more complicated cases than summing out b, which only appears at the root of the decision tree and has only one child in the belief network.

Example 3.2 Suppose, instead of summing out b, we were to sum out d where the rules for d were of the form:

$$d \leftarrow z : 0.29 \quad (22)$$

$$d \leftarrow \overline{z} \wedge y : 0.79 \quad (23)$$

$$d \leftarrow \overline{z} \wedge \overline{y} : 0.59 \quad (24)$$

The first three rules for e (rules (2)-(4)) don't involve d, and remain as they were. Variable partial elimination is not directly applicable to the last three rules for e (rules (5)-(7)) as they don't contain identical contexts apart from the variable being eliminated. It is simple to make the variable partial elimination applicable by splitting rule (7) on b resulting in the two rules:

$$e \leftarrow \overline{a} \wedge \overline{c} \wedge \overline{d} \wedge b : 0.85 \quad (25)$$

$$e \leftarrow \overline{a} \wedge \overline{c} \wedge \overline{d} \wedge \overline{b} : 0.85 \quad (26)$$

Rules (25) can be used with rule (5) in a variable partial evaluation, and (26) can be used with rule (6). The two rules corresponding variable partial evaluation with rule (22) are:

$$e \leftarrow \overline{a} \wedge \overline{c} \wedge z \wedge b : 0.025 \times 0.29 + 0.85 \times (1 - 0.29)$$
$$e \leftarrow \overline{a} \wedge \overline{c} \wedge z \wedge \overline{b} : 0.5 \times 0.29 + 0.85 \times (1 - 0.29)$$

Four other rules are created by combining with the other rules for d.

In general, you have to split rules with complementary literals and otherwise compatible, but not identical, contexts. You may need to split the rules multiple times on different atoms. For every pair of such rules, you create the number of rules equal to the size of the union of the literals in the two rules minus the number of literals in the intersection.

3.2 Multiple Children

One problem remains: when summing out a variable with multiple children in the belief network, using the technique above, we can't guarantee to maintain the loop invariant. Consider the belief network of Figure 1. If you were to sum out y, the variables a, b, c, and d become mutually dependent. Using the partial evaluation presented so far, the dependence is lost, but it is crucial for correctness.

To overcome this, we allow multiple variables in the head of clauses. The rules imply different combinations of the truth of the variables in the heads of clauses.

Example 3.3 Consider a belief network with a and b are the only children of y, and y is their only parent, and y has a single parent z. Suppose we have the following rules involving a, b, and y:

$$a \leftarrow y : 0.02 \quad (27)$$

$$a \leftarrow \overline{y} : 0.22 \quad (28)$$

$$b \leftarrow y : 0.77 \quad (29)$$

$$b \leftarrow \overline{y} : 0.27 \quad (30)$$

$$y \leftarrow z : 0.3 \quad (31)$$

We could imagine variable partial elimination on the rules for a with rule (31), and the rules for b with rule (31), resulting in:

$$b \leftarrow z : 0.27 \times 0.3 + 0.77 \times (1 - 0.3)$$
$$a \leftarrow z : 0.02 \times 0.3 + 0.22 \times (1 - 0.3)$$

However, this fails to represent the dependency between a and b that is induced by eliminating y.

We can, however, combine rules (27) and (29) resulting in the four rules:

$$a \wedge b \leftarrow y : 0.02 \times 0.77 \quad (32)$$

$$a \wedge \overline{b} \leftarrow y : 0.02 \times (1 - 0.77) \quad (33)$$

$$\overline{a} \wedge b \leftarrow y : (1 - 0.02) \times 0.77 \quad (34)$$

$$\overline{a} \wedge \overline{b} \leftarrow y : (1 - 0.02) \times (1 - 0.77) \quad (35)$$

Similarly, we can combine rules (28) and (30), resulting in four rules including:

$$a \wedge b \leftarrow \overline{y} : 0.22 \times 0.27 \quad (36)$$

which can be combined with rule (32) giving

$$a \wedge b \leftarrow z : 0.0462 \quad (37)$$

Note that the rules with multiple elements in the head follow the same definition as other rules.

3.3 Evidence

We can set the values of all evidence variables before summing out the remaining non-query variables (as in VE). Suppose $e_1{=}o_1 \wedge \ldots \wedge e_s{=}o_s$ is observed. There are three cases:

- Remove any rule that contains $e_i{=}o_i'$, where $o_i \neq o_i'$ in the head or the body.
- Remove any term $e_i{=}o_i$ in the body of a rule.
- Replace any $e_i{=}o_i$ in the head of a rule by $true$.

Rules with $true$ in the head are treated as any other rules, but we never resolve on $true$. When combining heads containing $true$, we can use the equivalence: $true \wedge a \equiv a$.

Example 3.4 Suppose $\overline{d}$ is observed. The rules for e become:

$$e \leftarrow a \wedge b : 0.55 \quad (38)$$
$$e \leftarrow a \wedge \overline{b} : 0.5 \quad (39)$$
$$e \leftarrow \overline{a} \wedge c : 0.08 \quad (40)$$
$$e \leftarrow \overline{a} \wedge \overline{c} : 0.85 \quad (41)$$

The rules (22)–(24) for d become:

$$true \leftarrow z : 0.29 \quad (42)$$
$$true \leftarrow \overline{z} \wedge y : 0.79 \quad (43)$$
$$true \leftarrow \overline{z} \wedge \overline{y} : 0.59 \quad (44)$$

d doesn't appear in the resulting theory.

3.4 Extracting the answer

Once evidence has been incorporated into the rule-base, the program invariant becomes:

> The probability of the evidence conjoined with a context c on the non-eliminated variables can be obtained by multiplying the probabilities associated with rules that are applicable in context c.

Suppose x is the query variable. After setting the evidence variables, and summing out the remaining variables, we end up with rules of the form:

$$x \leftarrow true : p_1$$
$$true \leftarrow x : p_3$$
$$\overline{x} \leftarrow true : p_2$$
$$true \leftarrow \overline{x} : p_4$$

The probability of $x \wedge e$ is obtained by multiplying the rules of the first two forms. The probability of $\overline{x} \wedge e$ is obtained by multiplying the rules of the last two forms. Then

$$P(x|e) = \frac{P(x \wedge e)}{P(x \wedge e) + P(\overline{x} \wedge e)}.$$

3.5 The Algorithm

We have now seen all of the components of the algorithm. It remains to put them together. We maintain the loop invariant of Section 3.4.

The top-level algorithm is the same as VE:

To compute $P(x|e_1{=}o_1 \wedge \ldots \wedge e_s{=}o_s)$
given elimination ordering $y_1, \ldots, y_k$:
1. Set the evidence variables as in Section 3.3.
2. Sum out $y_1, \ldots, y_k$ in turn.
3. Compute posterior probability as in Section 3.4

The only tricky part is in summing out variables.

To sum out variable y_i:
1. {*Rule splitting for combining heads*}
   for each pair of rules $h_1 \leftarrow b_1 : p_1$ and $h_2 \leftarrow b_2 : p_2$
   such that b_1 and b_2 both contain y_i
   and $h_1 \wedge b_1$ and $h_2 \wedge b_2$ are compatible,
   but not identical
   split each rule on variables in body of the other rule.
   {*Following 1, all rules with y_i in the body that are applicable in the same context have identical bodies.*}
2. {*Combining heads*}
   for each pair of rules $h_1 \leftarrow b : p_1$ and $h_2 \leftarrow b : p_2$
   such that b contains y_i
   and h_1 and h_2 are compatible
   replace them by the rule $h_1 \wedge h_2 \leftarrow b : p_1 p_2$
   {*Following 2, for every context, there is a single rule with y_i in the body that is applicable in that context.*}
3. {*Rule splitting for variable partial evaluation*}
   for every pair of rule of the form $h \leftarrow y_i{=}v_i \wedge b_1 : p_1$
   and $h \leftarrow y_i{=}v_i' \wedge b_2 : p_2$, where $v_i' \neq v_i$
   and b_1 and b_2 are comparable and not identical
   split each rule on atoms in body of the other rule.
   {*Following 3, all rules with complementary values for the y_i, but otherwise compatible bodies have otherwise identical. bodies and identical heads*}
4. {*Variable partial evaluation*}
   for each set of rules: $h \leftarrow y_i{=}v_k \wedge b : p_k$
   where the v_k are all of the values for y_i
   for each set of rules $h_2 \wedge y_i{=}v_k \leftarrow b_2 : q_k$
   such that $h \wedge b$ and $h_2 \wedge b_2$ are compatible
   create the rule $h \wedge h_2 \leftarrow b \wedge b_2 : \sum_k p_k q_k$.
5. {*Clean up*}
   Remove all rules containing y_i.

4 Comparison with other proposals

In this section we compare standard belief network algorithms, other structured algorithms and the new probabilistic partial evaluation algorithm. Example 2.6 is particularly illuminating because other algorithms do very badly on it.

Under the elimination ordering b, d, c, a, y, z, to find the prior on e, the most complicated rule set created is the rule set for e given in Example 3.1 with 16 rules (including the rules for the negations). After summing out d there are also 16 rules for e. After summing out c there are 14 rules for e, and after summing out a there are 8 rules for e. Observations simplify the algorithm as they mean fewer partial evaluations.

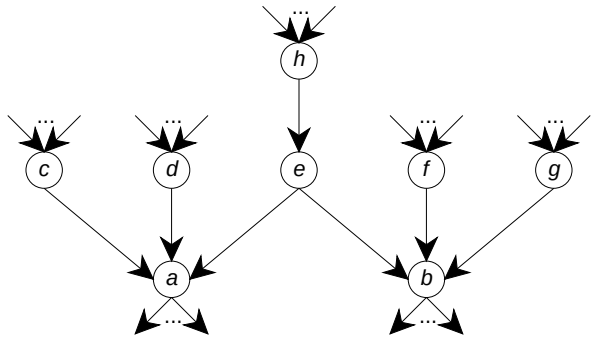

Figure 3: Exemplar for a node with multiple children: e to eliminate.

In contrast, VE requires a functor with table size 64 after b is summed out. Clique tree propagation constructs two cliques, one containing y, z, a, b, c, d of size $2^6 = 64$, and the other containing a, b, c, d, e of size 32. Neither takes the structure of the conditional probabilities into account.

Note however, that VE and clique tree propagation manipulate tables which can be indexed much faster than we can manipulate rules. There are cases where the rule-base exponentially is smaller than the tables (where added variables are only relevant in narrow contexts). There are other cases where we require as many rules as there are entries in the table (we never require more), in which case the overhead for manipulating rules will not make us competitive with the table-based methods. Where real problems lie in this spectrum is still an open question.

Boutilier *et al.* [1996] present two algorithms to exploit structure. For the network transformation and clustering method, Example 2.6 is the worst case; no structure can be exploited after triangulation of the resulting graph. (The tree for e in Example 2.6 is structurally identical to the tree for $X(1)$ in Figure 2 of [Boutilier *et al.*, 1996]). The structured cutset conditioning algorithm does well on this example. However, if the example is changed so that there are multiple (disconnected) copies of the same graph, the cutset conditioning algorithm is exponential in the number of copies, whereas the probabilistic partial evaluation algorithm is linear.

This algorithm is most closely related to the tree-based algorithms for solving MDPs [Boutilier *et al.*, 1995], but these work with much more restricted networks and with stringent assumptions on what is observable.

4.1 Why not trees?

It may be thought that the use of rules is a peculiarity of the author and that one may as well just use a tree-based representation. In this section I explain why the rule-based version presented here can be much more efficient than a tree-based representation.

Figure 3 shows an exemplar for summing out a variable with multiple children. The ancestors of c, d, f, g, and h are not shown. They can be multiply connected. Similarly the descendents of a and b are not shown.

Suppose we were to sum out e. Once e is eliminated, a and b become dependent. In VE and bucket elimination we form a factor containing all the remaining variables. This factor represents $P(a, b|c, d, f, g, h)$. One could imagine a version of VE that builds a tree-based representation for this factor. We show here how the rule-based version is exploiting more structure than this.

Suppose e is only relevant to a when d is true, and e is only relevant to b when f is true. In this case, the only time we need to consider the dependence between a and b is when both d and f are true. For all of the other contexts, we can treat a and b as independent. The algorithm does this automatically. Consider the following rules for a:

$$a \leftarrow d \wedge e : p_1 \quad (45)$$
$$a \leftarrow d \wedge \overline{e} : p_2 \quad (46)$$
$$a \leftarrow \overline{d} \wedge c : p_3 \quad (47)$$
$$a \leftarrow \overline{d} \wedge \overline{c} : p_4 \quad (48)$$

Consider the rules for b:

$$b \leftarrow f \wedge e : p_5 \quad (49)$$
$$b \leftarrow f \wedge \overline{e} : p_6 \quad (50)$$
$$b \leftarrow \overline{f} \wedge g : p_7 \quad (51)$$
$$b \leftarrow \overline{f} \wedge \overline{g} : p_8 \quad (52)$$

Consider the rules for e:

$$e \leftarrow h : p_9 \quad (53)$$
$$e \leftarrow \overline{h} : p_{10} \quad (54)$$

The first thing to note is that the rules that don't mention e are not affected by eliminating e. Thus rules (47), (48), (51), and (52) remain intact after eliminating e.

Rules (45) and (49) are both applicable in a context with a, d, e, b and f true. So we need to split them, according to the first step of the algorithm, creating:

$$a \leftarrow d \wedge e \wedge f : p_1 \quad (55)$$
$$a \leftarrow d \wedge e \wedge \overline{f} : p_1 \quad (56)$$
$$b \leftarrow d \wedge f \wedge e : p_5 \quad (57)$$
$$b \leftarrow \overline{d} \wedge f \wedge e : p_5 \quad (58)$$

We can combine rules (55) and (57) forming:

$$a \wedge b \leftarrow d \wedge e \wedge f : p_1 p_5 \quad (59)$$
$$a \wedge \overline{b} \leftarrow d \wedge e \wedge f : p_1(1 - p_5) \quad (60)$$
$$\overline{a} \wedge b \leftarrow d \wedge e \wedge f : (1 - p_1)p_5 \quad (61)$$
$$\overline{a} \wedge \overline{b} \leftarrow d \wedge e \wedge f : (1 - p_1)(1 - p_5) \quad (62)$$

Note also that rules (56) and (58), don't need to be combined with other rules. This reflects the fact that we only need to consider the combination of a and b for the case where both f and d are true.

Similarly we can split rules (46) and (50), and combine the compatible rules, giving rules for the combination of a and b in the context $d \wedge \overline{e} \wedge f$, rules for a in the context $d \wedge \overline{e} \wedge \overline{f}$ and rules for b in the context $\overline{d} \wedge f \wedge \overline{e}$.

Finally we can now safely replace e by its rules; all of the dependencies have been eliminated. The resultant rules encode the probabilities of $\{a, b\}$ in the contexts $d \wedge f \wedge h$ and $d \wedge f \wedge \overline{h}$, (8 rules). For all other contexts we can consider a and b separately. There are rules for a in the contexts $\overline{d} \wedge c$ (rule (47)), $\overline{d} \wedge \overline{c}$ (rule (48)), $d \wedge \overline{f} \wedge h$, and $d \wedge \overline{f} \wedge \overline{h}$, with the last two resulting from combining rule (56), and an analogous rule created by splitting rule (46), with rules (53) and (54) for e). Similarly there are rules for b in the contexts $\overline{f} \wedge g$, $\overline{f} \wedge \overline{g}$, $\overline{d} \wedge f \wedge h$, and $\overline{d} \wedge f \wedge \overline{h}$. The total number of rules (including rules for the negations) is 24.

One could imagine using VE or BEBA with tree-structures probability tables. This would mean that, once e is eliminated, we need a tree representing the probability on both a and b. This would entail multiplying out the rules that were not combined in the rule representation, for example the distribution on a and b the contexts $\overline{d} \wedge c \wedge \overline{f} \wedge g$. This results in a tree with 72 probabilities at leaves. Without any structure, VE or BEBA needs a table with $2^7 = 128$ values.

Unlike VE or BEBA, we need the combined effect on a and b only for the contexts where e is relevant to both a and b. For all other contexts, we don't need to combine the rules for a and b. This is important as combining the rules is the primary source of combinatorial explosion. By avoiding combining rules, we can have a huge saving when the variable to be summed out appears in few contexts.

5 Conclusion

This paper has presented a method for computing the posterior probability in belief networks with structured probability tables given as rules. This algorithm lets us maintain the rule structure structure, only combining contexts when necessary.

The main open problem is in finding good heuristics for elimination orderings. Finding a good elimination ordering is related to finding good triangulations in building compact junction trees, for which there are good heuristics [Kjærulff, 1990; Becker and Geiger, 1996]. These are not directly applicable to probabilistic partial evaluation, as an important criteria in this case is the exact form of the rules, and not just the graphical structure of the belief network.

The two main extensions to this algorithm are to multi-valued random variables and to allow logical variables in the rules. Both extensions are straightforward.

One of the main potential benefits of this algorithm is in approximation algorithms, where the rule bases allows fine-grained control over distinctions. Complementary rules with similar probabilities can be collapsed into a simpler rule. This can lead to more compact rule bases, and reasonable posterior ranges [Poole, 1997].

References

[Becker and Geiger, 1996] A. Becker and D. Geiger. A sufficiently fast algorithm for finding close to optimal junction trees. In E. Horvitz and F. Jensen, editor, *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence (UAI-96)*, pages 81–89, Portland, Oregon, 1996.

[Boutilier *et al.*, 1995] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proc. 14th International Joint Conf. on Artificial Intelligence (IJCAI-95)*, pages 1104–1111, Montreal, Quebec, 1995.

[Boutilier *et al.*, 1996] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In E. Horvitz and F. Jensen, editor, *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence (UAI-96)*, pages 115–123, Portland, Oregon, 1996.

[Dechter, 1996] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In E. Horvits and F. Jensen, editor, *Proc. Twelfth Conf. on Uncertainty in Artificial Intelligence (UAI-96)*, pages 211–219, Portland, Oregon, 1996.

[Heckerman and Breese, 1994] D. Heckerman and J. Breese. A new look at causal independence. In *Proc. of the Tenth Conference on Uncertainty in Artificial Ingelligence*, pages 286–292, 1994.

[Jensen *et al.*, 1990] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quaterly*, 4:269–282, 1990.

[Kjærulff, 1990] U. Kjærulff. Triangulation of graphs - algorithms giving small total state space. Technical Report R 90-09, Department of Mathematics and Computer Science, Strandvejen, DK 9000 Aalborg, Denmark, 1990.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50(2):157–224, 1988.

[Lloyd and Shepherdson, 1991] J.W. Lloyd and J.C. Shepherdson. Partial evaluation in logic programming. *Journal of Logic Programming*, 11:217–242, 1991.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[Poole, 1993] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[Poole, 1997] D. Poole. Exploiting contextual independence and approximation in belief network inference. *Technical Report*, 1997. `http://www.cs.ubc.ca/spider/poole/abstracts/approx-pa.html`.

[Shachter *et al.*, 1990] R. D. Shachter, B. D. D'Ambrosio, and B. D. Del Favero. Symbolic probabilistic inference in belief networks. In *Proc. 8th National Conference on Artificial Intelligence*, pages 126–131, Boston, 1990. MIT Press.

[Zhang and Poole, 1996] N.L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, December 1996.