AI Magazine Volume 20 Number 2 (1999) (© AAAI)

Inference in Bayesian Networks

Bruce D'Ambrosio

■ A Bayesian network is a compact, expressive representation of uncertain relationships among parameters in a domain. In this article, I introduce basic methods for computing with Bayesian networks, starting with the simple idea of summing the probabilities of events of interest. The article introduces major current methods for exact computation, briefly surveys approximation methods, and closes with a brief discussion of open issues.

In rare cases, primarily involving terminological information or other artificially constructed domains, one has the opportunity to determine categorically the truth of a proposition based on prior knowledge and current observation. Often, truth is more elusive, and categorical statements can only be made by judgment of the likelihood or other ordinal attribute of competing propositions. Probability theory is the oldest and best-understood theory for representing and reasoning about such situations, but early AI experimental efforts at applying probability theory were disappointing and only confirmed a belief among AI researchers that those who worried about numbers were "missing the point."[1] The *point*, so succinctly stated in Newell and Simon's physical symbol system hypothesis,[2] was that structure was the key, not the numeric details. The problem: The core around which a probabilistic approach revolves is the joint-probability distribution (JPD). Unfortunately, for domains described by a set of discrete parameters, the size of this object and the complexity of reasoning with it directly can both be exponential in the number of parameters.

A popular simplification was the *naïve Bayes's model*. This model assumes that the probability distribution for each observable parameter (that is, the probability of each value in the domain of the parameter) depends only on the root cause and not on the other parameters. Simplifying assumptions such as naïve Bayes's permitted tractable reasoning but were too extreme: They again provided no mechanism for representing the qualitative structure of a domain.

About 10 years ago, probability, and especially decision theory, began to attract renewed interest within the AI community, which was the result of a felicitous combination of obstacle and opportunity: The issue of ordering possible beliefs, both for belief revision and for action selection, was seen as increasingly important and problematic, and at the same time, dramatic new developments in computational probability and decision theory directly addressed perceived shortcomings. The key development was the discovery that a relationship could be established between a well-defined notion of conditional independence in probability theory and the absence of arcs in a directed acyclic graph (DAG). This relationship made it possible to express much of the structural information in a domain independently of the detailed numeric information, in a way that both simplifies knowledge acquisition and reduces the computational complexity of reasoning. The resulting graphic models have come to be known as Bayesian networks.

An example Bayesian network is shown in figure 1. I shortly examine the formal semantics of the graph, but intuitively, it is a representation of the following model: Sneezing can be "caused by" cold or an allergic reaction,[3] allergic reaction can be "caused by" the presence of a cat, and furniture scratches can be "caused by" a cat. For reasons we see later, a Bayes's net must be a DAG. Still, we could add several more arcs without violating the acyclicity requirement. The missing arcs encode information: The chance I have a cold is unrelated to the presence of a cat, colds don't cause furniture scratches, cat's don't cause sneezing except through initiation of an allergic reaction, and sneezing and scratches are unrelated except through their shared possible causes. We have described the structure of the model

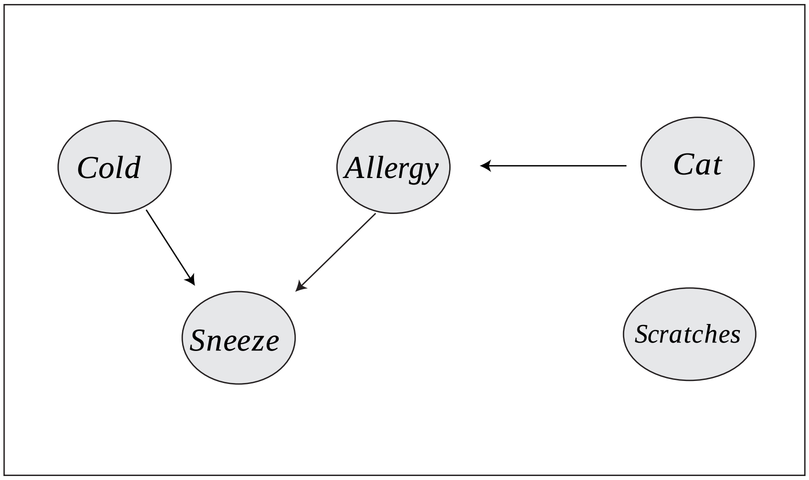

Figure 1. Simple Bayes's Net.

without specifying a single number and have done so in an intuitive and easily understandable form.

The probability model (the JPD across the parameters) is completed by specifying a set of local probability distributions, one associated with each node in the graph. That is, the breakthrough was the discovery that the graph and the set of local distributions associated with it uniquely determine the joint distribution. The number of probability values needed is linear in the number of parameters and exponential in the number of immediate parents of a parameter.[4] Thus, if the graph is sparse, few numbers will be needed to complete the model. It is believed that for domains in which humans can reason effectively, the graph is often sparse.

This article is concerned with *inference*, that is, computational methods for deriving answers to queries given a probability model expressed as a Bayesian network. We see that this sparseness of the graph can be exploited to make reasoning tractable, even when the number of parameters grows quite large. I begin with a review of the kinds of question that can be asked of such a model (that is, the *tasks*) and of the basic computational approaches to inference, survey the applicability of approaches to the tasks, and conclude with a discussion of open problems in inference.

Bayesian Networks— The Representation

I previously described the Bayes's net in figure 1 using the term *cause* to describe the meaning of arcs. Graphs are often constructed causally, that is, starting from an expert's understanding of the causal relationships among parameters. Formally, however, no notion of causality is needed or used. Begin with the chain rule or probability: For any (all) ordering of the parameters of a probability model, it is always true that the JPD can be written as the product over all parameters of the conditional distribution of each parameter conditioned on all parameters that precede it in the ordering. A Bayesian network simply encodes the fact that most of these distributions are independent of most of the predecessors. For each parameter in turn, we select a subset of the parameters that precede it in the ordering and that, if known, render its probability distribution independent of the remaining parameters that precede it. This subset defines the incoming arcs into the corresponding node in the graph.

For example, consider a health problem in which we are trying to build a model relating (1) cold (the illness), (2) the presence of a cat, (3) allergic reaction, (4) sneezing, and (5) furniture scratches. Assume the ordering we establish is as given. Cold has no parameters preceding it, so it must be a root in the graph. Let's say we believe the presence of a cat is independent of cold—in this case, cat presence is also a root in the graph.[5] That is, the lack of an arc between cold and cat is an assertion that my belief in whether or not there is a cat present is unchanged if I notice that I have a cold (in the presence of no other information). Allergy has two parameters that precede it: (1) cold and (2) cat. I am allergic to cats (and have four, sigh),[6] so allergic reaction definitely is not independent of cat presence but is unaffected by whether or not I have a cold, so I draw an arc from cat to allergy. Sneeze is an interesting parameter. It seems to depend on all three of its predecessors in the ordering. That is, I am certainly more likely to be sneezing if I have a cold, also if I am currently suffering an allergic reaction. I am also more likely to sneeze if there is a cat around.

However, let's assume that if I already know that I am currently having an allergic reaction, then knowing that there is a cat doesn't increase my belief that I will sneeze. This might arise, for example, if I believe that the *causal mechanism* by which cats induce sneezing in me is through initiation of an allergic reaction. In this case, I need only draw arcs from cold and allergic reaction; I can omit the arc from cat to sneeze. Notice that I say can, not must. It is never wrong to add an arc, but the goal is to construct as sparse a graph as possible, consistent with the realities of the domain. Omission of the arc from cat to sneeze is a statement that sneezing is conditionally independent of cat presence given knowledge of whether or not I have a cold and an allergic reaction. Finally, let's say that I believe the probability of furniture scratches is independent of cold, sneeze, and allergic reaction, given cat (again, this statement is that once I know whether or not a cat is present, knowing whether or not I have a cold, am sneezing, or have active allergies will not further change my belief that I am likely to find scratches on the furniture). Our final network, then, is as shown in figure 1.

The graph is a visual statement of a set of independence relations. Critical, as we saw in the construction phase, are the arcs that are missing. Many find it convenient to interpret the arcs as representing causality, as I suggested earlier. However, formally, they are simply statements about the structure of the JPD. We could have chosen a different ordering of the variables and gotten a different graph. There is, however, at least in "natural" domains, an apparent relation between Bayes's nets and causality: Graphs drawn in the natural, or causal, direction tend to be sparse. Consider, for example, the simple three-node network relating cold, allergy, and sneeze. If we had chosen to order sneeze first, the graph would have been fully connected, as shown in figure 2. The apparent loss of independence is easy to see: If I am sneezing, then the chance that I am having an allergic reaction is certainly reduced if I know I have a cold.

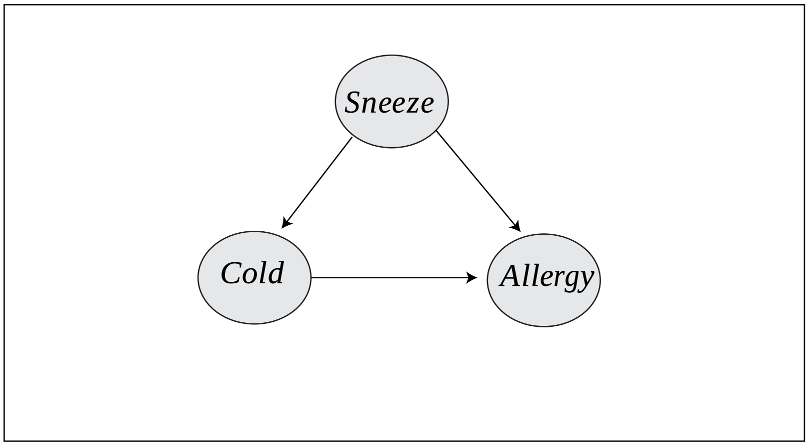

Figure 2. Noncausal Ordering.

Why do we care if the graph is sparse? Sparseness can be exploited in both knowledge acquisition and inference. The graph specifies a factorization of the joint, in particular, given the graph in figure 1.

$$\begin{aligned} &P(Cold, Allergy, Sneeze, Cat, Scratches) \\ &\quad = P(Scratches \mid Cat) \\ &\quad * P(Sneeze \mid Cold, Allergy) \\ &\quad * P(Allergy \mid Cat) * P(Cold) * P(Cat) \end{aligned}$$

That is, we need only specify the conditional probability of each parameter given its immediate parents in the graph (or the marginal for parameters at roots of the graph). Details of why this is so are beyond the scope of this article, but the ability to recover the JPD from the local distributions, plus the graph topology, is the fundamental theorem of Bayes's nets—Given the conditional independence implied by the missing arcs in the graph, the JPD (or, simply, joint) can be recovered from the previous factorization. Thus, as long as the necessary conditionals and marginals are locally coherent, we are guaranteed to have defined a unique, globally coherent joint. Bayes's nets separate structure from numerics in a simple, intuitive graphic form. Further, they dramatically reduce the magnitude of the knowledge-acquisition problem ("Where do you get all those numbers?"): Instead of needing an exponential number of numbers, we now only need to acquire a number of numeric parameters linear in the number of variables (although still exponential in the number of immediate parents; I return to this point later). Thus, Bayes's nets directly respond to the criticisms of probability theory leveled a generation ago.

Inference Tasks

To this point, we have an expressive, concise representation that is purportedly easy to acquire because of a good cognitive match, but how can we use it? We could use the chain rule, recover the full joint, and answer questions from the full joint, as I now demonstrate. For example, consider table 1 for our example problem.

This table, if all the parts were combined using the chain rule described earlier, yields the joint distribution for the domain. The joint enumerates all possible states of the system. The joint will have 32 entries, 1 for each element in the cross-product of the variable domains. I list it in table 2.

Answering a query is, from the perspective of the joint, simply a matter of summing the values as appropriate subset cells of this table. Some important queries are described in the following paragraphs:

Single marginal: A *marginal probability* is the probability of some subset of the parameters in a model. The *prior* (before any evidence) *marginal probability* of, for example, cold = true, is simply the sum of the values in all the cells labeled with cold = true. In this case, the value is

$$\begin{aligned} &P(cold = true) \\ &\quad = \quad .00001 + .00001 + .00112 + \ldots + .00010 \\ &\quad = \quad .05 \end{aligned}$$

Cold = False	Cold = True
.95	.05

Cat Presence = False	Cat Presence = True
.98	.02

Cat Presence	Allergic Reaction = False	Allergic Reaction = True
False	.95	.05
True	.25	.75

Cold	Allergic Reaction	Sneeze = False	Sneeze = True
False	False	.99	.01
False	True	.3	.7
True	False	.2	.8
True	True	.1	.9

Cat Presence	Scratches = False	Scratches = True
False	.95	.05
True	.5	.5

Table 1. Numeric Details of Simple Bayes's Net.[1]

Cold	Cat	Allergy	Sneeze = False & Scratches = False	Sneeze = False & Scratches = True	Sneeze = True & Scratches = False	Sneeze = True & Scratches = True
True	True	False	.00001	.00001	.00012	.00012
True	True	True	.00001	.00004	.00035	.00035
True	False	False	.00884	.00047	.03538	.00186
True	False	True	.00047	.00002	.00186	.00010
False	True	False	.00071	.00071	.00166	.00166
False	True	True	.00214	.00214	.00499	.00499
False	False	False	.83183	.04378	.00840	.00044
False	False	True	.04378	.00230	.00044	.00002

Table 2. Joint Probability Distribution Function Induced by Simple Bayes's Net.

Formally, we could write such a computation as

$P(cold = \text{true})$
$\quad = \Sigma_{Scratches,Cat,Sneeze,Allergy}$
$\quad P(Scratches \mid Cat)$
$\quad * P(Sneeze \mid Cold = True, Allergy)$
$\quad * P(Allergy \mid Cat) * P(Cold = True)$
$\quad * P(Cat)$

where the multiplications reconstruct the joint and the summation adds elements in the appropriate cells, as before. A *prior probability*, then, is simply the probability over the entire set of possible situations. If we want the probability distribution across *cold*, we simply scan the entire joint, adding entries into the answer for *P*(*Cold* = *True*) or *P*(*Cold* = *False*) according to their labels. In this case, we simply recover the prior probability for cold, as expected.

An important situation arises when, after constructing such a table, we learn that certain subsets of cells are impossible. For example, suppose we start sneezing. Then it is certainly the case that none of the cells (situations) labeled with sneeze = false apply at present; so, in determining the posterior probability that cold = true given sneeze = true, we only sum cells labeled with cold = true and sneeze = true and renormalize (divide by the sum of all cells labeled with sneeze = true regardless of their cold label). Renormalizing enforces the constraint that the domain is a partition of the space of possibilities, and therefore, the probability distribution over the domain must sum to one. In this case, we have

.00012 + .00012 +. 00035 +. 00035 +. 03538
+ .00186 + .00186 + .00010

.00012 + … + .00010 + .00166 + .00166
+ .00499 + .00499 + .00840 + .00044
+ .00044 + .00002
= .04014 / .06274 = .63

Again, we can write this computation as

$P(Cold = True \mid Sneeze = True)$
$\quad = (\Sigma_{Scratches,Cat,Allergy}\, P(Scratches \mid Cat)$
$\quad * P(Sneeze = True \mid Cold = True, Allergy)$
$\quad * P(Allergy \mid Cat) * P(Cold = True) * P(Cat))$

$/ \; (\Sigma_{Scratches,Cat,Allergy,Cold}\, P(Scratches \mid Cat)$
$\quad * P(Sneeze = True \mid Cold, Allergy)$
$\quad * P(Allergy \mid Cat) * P(Cold) * P(Cat))$

Subjoint: Similarly, we might want to know the JPD function (JPDF) across a subset of the parameters. The procedure is the same—we simply sum all the cells of the JPDF that are labeled with the same value for the parameters we care about. For example, we might want to know the prior probability that we have a cold and that we are having an allergy attack:

$P(Cold = True \; \& \; Allergy = True)$
= .00001 + .00004 + .00035 + .00035
+ .00047 + .00002 + .00186 + .00010
= .0032

and can be written as

$P(Cold = True \; \& \; Allergy = True)$
$\quad = \Sigma_{Scratches,Cat,Allergy}\, P(Scratches \mid Cat)$
$\quad * P(Sneeze \mid \text{Cold} = True, Allergy = True)$
$\quad * P(Allergy = True \mid Cat) * P(Cold = True)$
$\quad * P(Cat)$

Evidence is handled as before.

All marginals: An interesting query is the computation of the marginal probability of all parameters rather than merely a single one. Although we could repeatedly apply the simple technique described earlier for each parameter, quite a bit of redundant computation would be involved. It would be more efficient to first compute the joint, then establish separate answer cells for each parameter and pass over the joint only once, adding each cell of the joint into the appropriate answer cell for each parameter as we go. Notice that this single-pass optimization is a generalization of the computation of the distribution across a single parameter, where we suggested the same procedure.

Arbitrary subset of queries: A natural generalization of the all-marginals query is to query for an arbitrary subset of subjoints (for example, {*P*(*Cold* & *Allergy*), *P*(*Cat*)}).

Conditional: Another useful query is the conditional. We have already seen an instance of a conditional query when we wanted to compute a marginal probability given evidence. More generally, we can ask for a conditional distribution, such as *P*(*Cold* | *Sneeze*). This query is just four primitive queries, *P*(*Cold* = *True* | *Sneeze* = *True*), *P*(*Cold* = *False* | *Sneeze* = *True*), and so on, but again one might hope for a more efficient method than performing the four separate computations.

Boolean: We just discussed subjoint queries and saw that they are essentially conjunctive queries. More generally, we can ask any Boolean query, such as $P((Cold = True \wedge Allergy = False) \vee (Cold = True \wedge Cat = False))$?[7] As before, the answer is simply the sum of the values in every cell of the joint that satisfies the query condition. In this case, we get

$P((Cold = True \wedge Allergy = False)$
$\quad \vee (Cold = True \wedge Cat = False)$
= row 2 + row 3 + row 4
= .00001 + .00001 + .00012 + .00012
+ .00012 + .00002 + .00186 + .00010
+ .00884 + .00047 + .03538 + .00186
= .04926

The expression corresponding to this query is straightforward but beyond the scope of this survey.

Cat Presence	Scratches = False	Scratches = True
False	.931	.049
True	.01	.01

Table 3. Probability of Scratches on Furniture Given Cat Presence.

Cat Presence = False	Cat Presence = True
.98	.02

Table 4. Probability of Cat Presence.

MPE: The *most probable explanation* (MPE) is the label on the cell with the highest probability. In this case, it is

Cold = False, Cat = False, Scratches = False,
Allergy = False, Sneeze = False

$P = .83183$

If we observe that we are sneezing, then we eliminate all cells with sneeze = false, and the remaining cell with the largest value is

Cold = True, , Cat = False, Scratches = False,
Allergy = False, Sneeze = True
$P = .03538$ (prior), $.03538 / .06274 = .56$ (posterior)

A variant of this request is to ask for the *m* most likely rather than the single most likely.

Maximum a posteriori probability (MAP): Often, we are interested in the MPE, not of the full joint but of a subjoint. For example, we might want to determine the most likely joint instantiation of cold and allergy. This is simply the label of the cell in the subjoint ($P(Cold, Allergy)$) with the largest value. Again, the most likely instantiation can be in the prior or with respect to evidence.

The previous queries are the basic queries that one can ask of a JPD. A variety of application-specific queries can be composed of these, including various forms of sensitivity analysis such as expected entropy and others as well as decision analyses including expected utility and value of information. *Sensitivity analysis* answers the question of how dependent the marginal distribution is over parameter *X* dependent on the value taken by parameter *Y*. *Value of information* is a decision-theoretic analysis of the question in a decision context of how much we should be willing to pay to obtain information about parameter *Y*. Sensitivity analysis and value of information are beyond the scope of this overview.

Inference Methods

To this point, I have introduced a factored representation of a JPD and argued that it made specification of such a distribution a tractable task. I then reviewed the basic information one could derive from such an object. However, the exposition depended on first reconstructing the full joint from the factored representation. Such explicit reconstruction is feasible only for toy problems. We now turn to the core topic of this article, an introduction to practical methods for inference in Bayesian networks. I begin our review with, and spend most of our time on, exact methods. I then survey approximate methods and conclude with a brief discussion of open research topics in inference.

Exact

Let's begin our examination of tractable inference methods by reexamining our first example query. We said it could be formulated as

$$\begin{aligned}&P(Cold)\\ &\quad = \Sigma_{Scratches,Cat,Sneeze,Allergy}\, P(Scratches \mid Cat)\\ &\quad * P(Sneeze \mid Cold, Allergy) * P(Allergy \mid Cat)\\ &\quad * P(Cold) * P(Cat)\end{aligned}$$

The sum and product operations being performed here are real-number operations, so the normal associativity, commutivity, and distributivity properties apply. Thus, we can rewrite the previous expression as

$$\begin{aligned}&P(Cold = true)\\ &\quad = \Sigma_{Sneeze,Allergy}\, P(Sneeze \mid Cold = True, Allergy)\\ &\quad * P(Cold = True) * (\Sigma_{Cat}\, P(Allergy \mid \mathrm{Cat})\\ &\quad * (\Sigma_{Scratches}\, P(Scratches \mid \mathrm{Cat}) * P(Cat)))\end{aligned}$$

The meaning of a summation within an expression might seem unclear, but it is quite straightforward. Consider $\Sigma_{Scratches}\, P(Scratches \mid Cat) * P(Cat)$. We can understand it constructively. First, $P(Scratches \mid Cat) * P(Cat)$ is simply a two-dimensional table, where each entry is the product of an entry from $P(Scratches \mid Cat)$ with the entry from $P(Cat)$ (table 3).

Next, the summation operator reduces this to a one-dimensional table by summing over values of scratches (table 4).

In this case (no evidence on either scratches or cat presence), the resulting table is simply the prior on cat, which is not always the case. Renormalization is not performed until all combination is complete, so these intermediate tables are not true probability distributions. They are sometimes referred to as generalized distributions or potentials.

Although the original form of the expression requires an explicit reconstruction of the full joint table, in the rewritten form no intermediate result table is more than three variables. The problem of determining the optimal evaluation form for a query was formalized as the *optimal factoring problem* by Zhaoyu Li in 1991 (Li 1994) and shown to be NP-hard by Mark Bloemke in 1998 (Bloemke and Valtorta 1998).

Further consideration of the previous expression reveals a few interesting observations: First, legal rearrangements correspond to those in which we marginalize out, or sum over, a parameter only after combining all distributions naming the parameter into a single intermediate result. This is a simple consequence of the laws of associativity and commutivity. Second, the expression $P(\mathit{Scratches} \mid \mathit{Cat})$ contributes nothing to the query, which is obvious if we remember that probabilistic coherence requires that each "row" of a conditional distribution sum to one. Stating this independence of the query from $P(\mathit{Scratches} \mid \mathit{Cat})$ somewhat differently,

$$P(\mathit{Cat}) = \Sigma_{\mathit{Scratches}}\, P(\mathit{Scratches} \mid \mathit{Cat}) * P(\mathit{Cat})$$

There are linear-time methods for determining the set of parameters relevant to a query, given a set of evidence (see Geiger [1989]). These methods can be thought of as methods for identifying a subgraph of the original graph relevant to the query. For the remainder of this article, I ignore this issue and concentrate on the more difficult task of performing efficient inference given a (sub)network.

Factoring

One class of methods for answering individual queries directly uses the method sketched earlier. Because the problem of finding an optimal evaluation form is intractable, simple greedy heuristics are typically applied. A very simple one, the *set-factoring* heuristic, developed by Li (1994), operates by selecting pairs of distributions to combine. It begins with the initial set of distributions to be combined. It selects a pair to combine, removes from the initial set of distributions, and places the result of the combination back into the set. The process iterates until there is only one distribution left, which is the answer to the query. The set-factoring heuristic is simply to pick the pair to combine next that has the smallest result table. Ties are broken by selecting the pair combination that marginalizes over the most parameters. As stated previously, a parameter can be marginalized, if it does not appear in the query, when it does not appear in any distributions except the two being combined. As an example, consider our initial query for $P(\mathit{Cold})$. Five distributions are in the initial set, so there are 10 possible pairs (order doesn't matter). However, for simplicity, I consider only those pairs that share at least one parameter:

1. For our first step, the candidate pairs are

1. $P(\mathit{Cat}) * P(\mathit{Scratch} \mid \mathit{Cat}) \Rightarrow P'(\mathit{Cat})$

2. $P(\mathit{Cat}) * P(\mathit{Allergy} \mid \mathit{Cat}) \Rightarrow P'(\mathit{Allergy}, \mathit{Cat})$

3. $P(\mathit{Scratch} \mid \mathit{Cat}) * P(\mathit{Allergy} \mid \mathit{Cat}) \Rightarrow P'(\mathit{Allergy} \mid \mathit{Cat})$

4. $P(\mathit{Sneeze} \mid \mathit{Cold}, \mathit{Allergy}) * P(\mathit{Allergy} \mid \mathit{Cat}) \Rightarrow P'(\mathit{Allergy} \mid \mathit{Cat}, \mathit{Cold})$

5. $P(\mathit{Sneeze} \mid \mathit{Cold}, \mathit{Allergy}) * P(\mathit{Cold}) \Rightarrow P'(\mathit{Cold} \mid \mathit{Allergy})$

The first choice has the smallest result, so it is selected. The two input distributions are removed from the set, and the result is added.

2. Regenerating the set of candidate pairs, we now obtain

1. $P'(\mathit{Cat}) * P(\mathit{Allergy} \mid \mathit{Cat}) \Rightarrow P'(\mathit{Allergy})$

2. $P(\mathit{Sneeze} \mid \mathit{Cold}, \mathit{Allergy}) * P(\mathit{Allergy} \mid \mathit{Cat}) \Rightarrow P'(\mathit{Allergy} \mid \mathit{Cat}, \mathit{Cold})$

3. $P(\mathit{Sneeze} \mid \mathit{Cold}, \mathit{Allergy}) * P(\mathit{Cold}) \Rightarrow P'(\mathit{Cold} \mid \mathit{Allergy})$

The first choice again has the smallest result, so it is selected.

3. Again regenerating the set of candidate pairs, we obtain

4. $P(\mathit{Sneeze} \mid \mathit{Cold}, \mathit{Allergy}) * P'(\mathit{Allergy}) \Rightarrow P'(\mid \mathit{Cold})$

5. $P(\mathit{Sneeze} \mid \mathit{Cold}, \mathit{Allergy}) * P(\mathit{Cold}) \Rightarrow P'(\mathit{Cold} \mid \mathit{Allergy})$

The first choice has the smaller result, so it is selected.

4. There are now only two distributions in the set, $P(\mathit{Cold})$ and $P'(\mid \mathit{Cold})$.[8] These two are combined.

The overall computation was

$$\begin{aligned} &P(\mathit{Cold}) \\ &\quad = P(\mathit{Cold}) \\ &\quad * \Sigma_{\mathit{Allergy},\mathit{Sneeze}}\, (P(\mathit{Sneeze} \mid \mathit{Cold}, \mathit{Allergy}) \\ &\quad * \Sigma_{\mathit{Cat}}\, (P(\mathit{Allergy} \mid \mathit{Cat}) \\ &\quad * \Sigma_{\mathit{Scratches}}\, (P(\mathit{Scratches} \mid \mathit{Cat}) * P(\mathit{Cat})))) \end{aligned}$$

Direct factoring methods have been developed for all the single-query tasks, including MPE and MAP (Li 1993). However, until recently, it has been unclear how to apply direct factoring methods to multiple query tasks (Bloemke 1998).

Variable Elimination

The previous method is very simple to understand, is fairly simple to implement, and offers excellent performance. Further, its performance is limited only by the quality of the heuristic used. That is, the method is inherent-

Because the problem of finding an optimal evaluation form is intractable, simple greedy heuristics are typically applied. A very simple one, the set-factoring heuristic, developed by Li (1994), operates by selecting pairs of distributions to combine

ly capable of expressing the optimal computation for any single query, whether for a single parameter or an arbitrary subjoint. However, although it seems natural, it somewhat obscures the real problem. It turns out that the hard problem is identifying an order in which to marginalize the parameters not in the query. Once that has been determined, it is simple (polynomial time) to determine an optimal combination sequence. Bucket elimination (Dechter 1996) is a family of algorithms using this approach. The idea, like that of direct application of factoring, is simple. First, establish an elimination ordering for the parameters not in the result. Then, for each parameter in the elimination ordering in turn, (1) remove and gather all distributions indexed by the parameter; (2) combine them, marginalizing over the parameter; and (3) place the result in the distribution set.

Let's review our original example from the variable elimination perspective. Suppose for the moment that we establish the following elimination order for the *P(Cold)* query: scratch, cat, allergy, sneeze. Then the computation would be as follows:

1. Gather all distributions indexed over *Scratch*, combine, and sum out *Scratch*: $\Sigma_{Scratches}$ $(P(Scratches \mid Cat))$. Call the result $P'(\mid Cat)$.

2. Gather all distributions indexed over *Cat*, combine, and sum out *Cat:* Σ_{Cat} $(P(Allergy \mid Cat) * P'(\mid Cat) * P(Cat))$. Call the result $P'(Allergy)$.

3. Gather all distributions indexed over *Allergy*, combine, and sum out *Allergy:* $\Sigma_{Allergy}$ $(P(Sneeze \mid Cold, Allergy) * P'(Allergy))$. Call the result $P'(Sneeze \mid Cold)$.

4. Gather all distributions indexed over *Sneeze*, combine, and sum out *Sneeze:* Σ_{Sneeze} $(P'(Sneeze \mid Cold))$.

One advantage of focusing on the problem of parameter-elimination ordering is the availability of results from multiple disciplines. The problem of determining an elimination ordering for nodes in a graph has relevance to a number of problems, including constraint satisfaction and the solving of sparse systems of equations. Although the general problem is NP-hard, a variety of approximate methods have been developed. *Minimum deficiency ordering* (MDO) is typical of the widely used class of simple greedy heuristics and performs well for Bayes's net variable elimination.[9] An ordering is established by choosing to eliminate last the parameter with the fewest connections to other nodes. Once the node is chosen, its neighbors are pairwise connected, and then it is removed from the graph. The process is then repeated until all nodes are eliminated. Note that the final parameter-elimination ordering is the reverse of the order in which nodes are removed from the graph. An excellent review of heuristics for generating elimination orderings can be found in Kjaerulff (1992). See also Becker (1996) for nearly optimal methods and Jensen (1994).

Parameter elimination fills roughly the same niche as direct factoring: It is intended for single queries (single parameter or subjoint) and is available in modified form for the MAP or MPE query.

Finally, we can ask whether or not the optimal computation is expressible in bucket-elimination form. Surprisingly, the answer is no but only by a constant factor. To see why not, consider figure 3.

Let us examine a query for $P(G)$. Assuming all parameters are binary valued, there are a number of equivalent optimal computations, one of which is

$$P(G) = \Sigma_{A,B,C,D,E,F} P(G \mid A, B, C, D, E, F) * ((P(A) * P(B)) * P(C)) * (P(D) * (P(E) * P(F))$$

However, it is not clear how we can obtain this computation using parameter elimination. We can't eliminate G; it is the query parameter. Any other parameter we choose to eliminate first requires that we combine the marginal for that parameter with the conditional for G, a clearly suboptimal choice. The computational significance of this point is minor: In this example, the optimal computation is only a constant factor better than one obtained by eliminating A or B first. Further, it seems easily fixed. The primary reason for pointing this suboptimality out is to show that optimality of inference in Bayes's nets is not well understood, and open questions remain.

Junction Trees

To this point, I have presented exact methods for answering any single marginal or subjoint query on a Bayes's net and stated (without demonstration) that these methods easily extend to Boolean, MPE, and MAP queries. What remains is to discuss processing for multiple queries. Consider our original network and a query for $P(Cold)$. We might use the following rearrangement of the marginalization of the chain-rule reconstruction of the joint:

$$\begin{aligned} &P(Cold) \\ &\quad = \Sigma_{Allergy} ((\Sigma_{Sneeze} P(Sneeze \mid Cold, Allergy) \\ &\quad * P(Cold)) * (\Sigma_{Cat} (P(Allergy \mid Cat) \\ &\quad * (\Sigma_{Scratches} P(Scratches \mid Cat) * P(Cat))))) \end{aligned}$$

It is a bit difficult to see the structure of the computation in linear form. We can display this computation as a tree (figure 4).

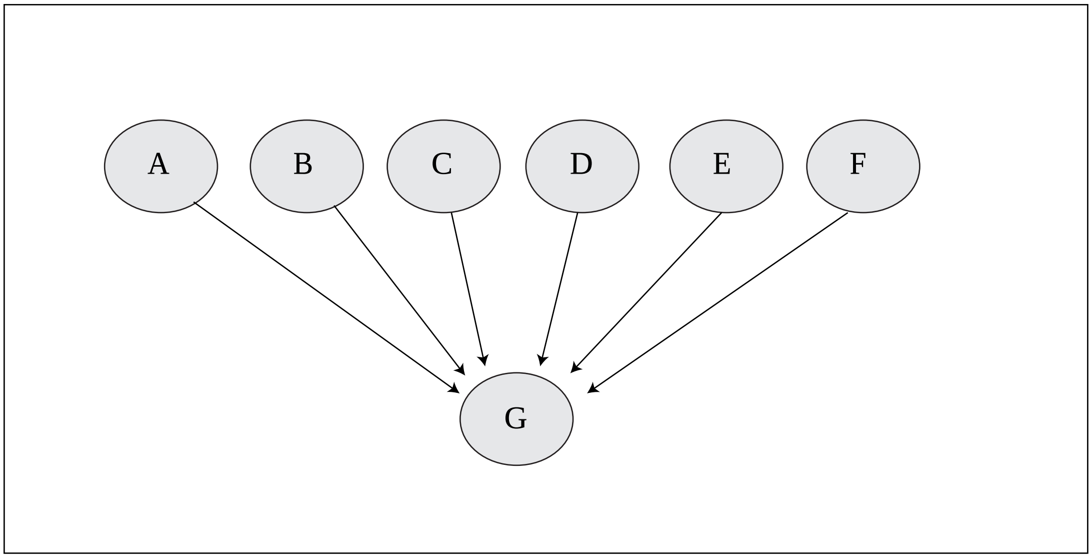

Figure 3. Problematic Bayes's Net.

Now consider a query for $P(Cat)$. Suppose we build the following computation:

$$\begin{aligned} &P(Cat) \\ &\quad = ((\Sigma_{Scratches} P(Scratches \mid Cat) * P(Cat)) \\ &\quad * (\Sigma_{Allergy}\ P(Allergy \mid Cat) \\ &\quad * (\Sigma_{Sn,Co}\ (P(Sneeze \mid Cold, Allergy) \\ &\quad * P(Cold))))) \end{aligned}$$

Again, we can display this computation as a tree (figure 5).

There is quite a bit of overlap in the two computations. In fact, if we align them together, we obtain the following information flows (figure 6).

Although figure 6 informally indicates some commonality in the two computations, it is a bit vague about how such commonality might actually be exploited. Let's examine how we might exploit this commonality more generally. Further, let's assume our goal is to share a single copy of each of the three core "tables":

$$\{(P(Sn \mid Co, Al) * P(Co)), P(Al \mid \mathrm{Ca}), (P(Sc \mid \mathrm{Ca}) * P(Ca))\}$$

Then we might use the following computational sequence:

1. Compute the three tables, $P(Sn \mid Co, Al)$ * $P(Co)$, $P(Al \mid Ca)$, and $P(Sc \mid Ca)$ * $P(Ca)$ (note the second one involves only a single distribution and, thus, no actual computation).

2. Marginalize the bottom table as indicated by the upward arrow (*Sn, Co*), and multiply the result by the middle table. Replace the middle table with the result.

3. Marginalize the middle table as shown in the next arrow (*Al*), and multiply the top table by the result. Replace the top table with the result.

4. Now, to get *P(Cat)*, simply marginalize the top table over *Sc*. Note that we have exactly reconstructed the original computation sequence for $P(Cat)$.

5. Next, we will compute $P(Cold)$. However, notice that the table at the top now has the wrong information in it. To recover the original table needed for the $P(Cold)$ computation, we need to divide out the message we just multiplied. We can then marginalize over *Sc*, as specified on the upper downward-pointing arrow and multiply the result by the middle table. Again, we replace the middle table with the result of this multiplication.

6. Once again, if we examine the computational history of the middle table, it includes too much information. We again can recover the needed table by dividing out the "upward" message we previously multiplied it by and marginalize out *Ca* as specified in the middle downward-pointing arrow.

7. We then multiply the bottom table by the earlier message. Finally, we can obtain $P(Cold)$ by marginalizing out *Al, Sn*, as shown in the bottom downward-pointing arrow. Again, we have largely reproduced the original computation of $P(Cold)$.

We could have avoided the need for division by keeping the original tables, at the cost of a higher storage-space requirement.

Junction Trees: In the previous computational sequence, we generated a unified computational structure for multiple queries by first determining structures for each of the individual queries, then merging them. I close our discussion of exact inference with the most well-known method of inference in Bayesian networks, a general procedure for constructing a single computational structure

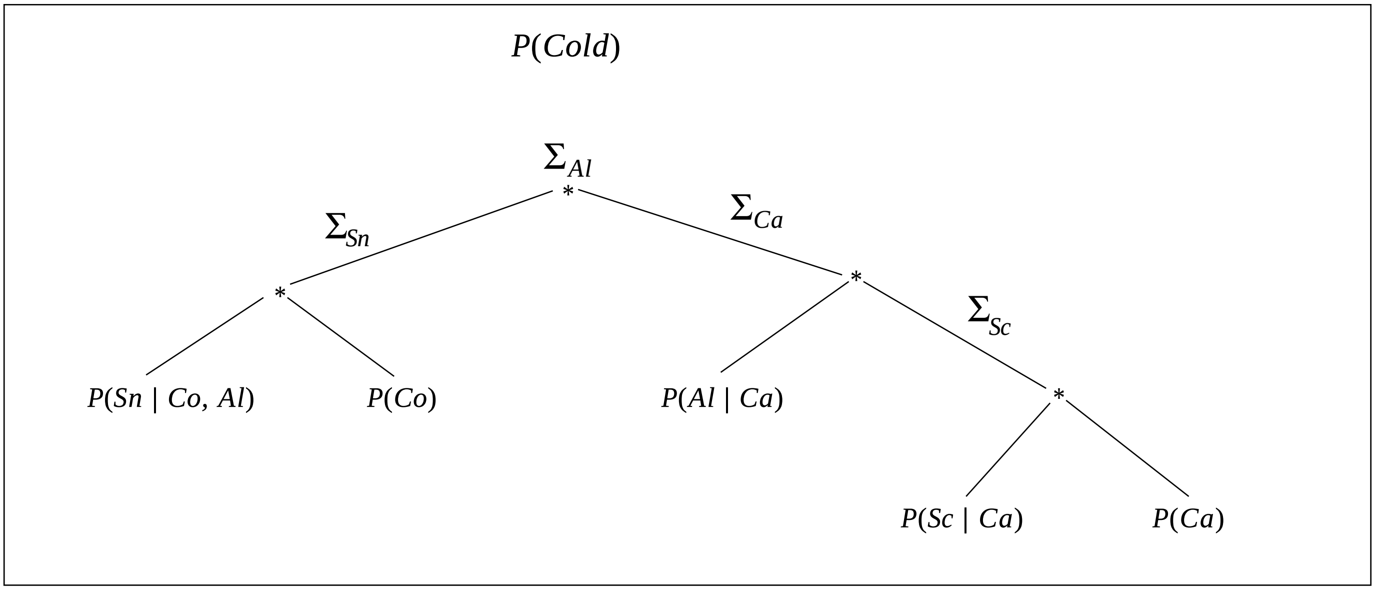

Figure 4. Computation Trees for P(Cold).

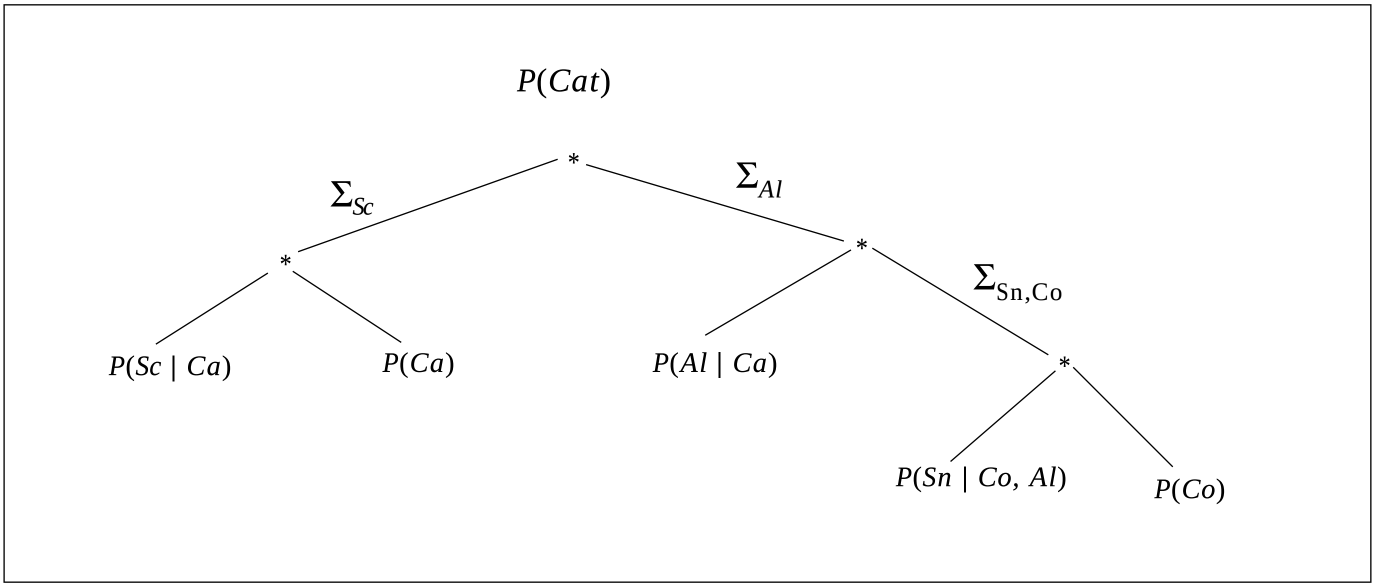

Figure 5. Computation Tree for P(Cat).

capable of computing the answer to multiple queries, without the need for first computing separate structures for each. A *junction tree* is a computational structure for the *all-marginals* task, where the query set is the set of all single-parameter marginals (Lepar 1998; Jensen 1990a, 1990b; Lauritzen 1988). I present junction trees as directly derived from an elimination ordering.[10] To demonstrate the concepts involved, I present figure 7.

Using the minimum deficiency ordering (MDO) heuristic sketched earlier, one-elimination ordering, we can obtain *A, B, C, D, E, F*. Now, let's actually eliminate each variable in turn, noticing the other parameters involved at each step. Remember that to eliminate a parameter, we must first combine all remaining distributions indexed on the parameter. Eliminating all parameters involves the following steps:

1. To eliminate *A*, we need to combine *P*(*A*), *P*(*B* | *A*), and *P*(*C* | *A*). Thus, the set of parameters mentioned is {*A*, *B*, *C*}. After this combination and elimination (marginalization) of *A*, the result is *P*(*B*, *C*), and the full set of remaining distributions is {*P*(*B*, *C*), *P*(*D* | *B*), *P*(*E* | *C*), *P*(*F* | *D*, *E*)}.

2. Next, to eliminate *B*, we combine *P*(*B*, *C*) with *P*(*D* | *B*), so the set of parameters mentioned is {*B*, *C*, *D*}, and the set of remaining

distributions is $\{P(D , C), P(E \mid C), \mathrm{P}(F \mid D, E)\}$.

3. Eliminating C requires that we combine $P(D, C)$ with $P(E \mid C)$ to yield $P(D, E)$ and involves $\{C, D, E\}$. The set of remaining distributions is $\{P(D, E), P(F \mid D, E)\}$.

4. Finally, eliminating D requires combining $P(D, E)$ with $P(F \mid D, E)$ and involves $\{D, E, F\}$.

Because we now have only a single distribution, we can stop—the remaining eliminations are simply marginalizations of an already-computed subjoint and, thus, are trivial. Now, let's arrange the sets of variables involved in the computations in a graph, where the nodes are the sets of parameters, and the arcs show data flow in the elimination computation. We obtain the junction tree in figure 8.

We see that the essential property of the nodes (from our perspective focusing on the joint as the central object) is that they are the sets of parameters that are involved in eliminating a parameter from the joint.[11] An interesting property of a junction tree is that if a parameter is present in two nodes, it will be present in all nodes on the path between the two.

How can we use such a structure for the computation of all marginals? Remember our earlier example of the parallels between the two queries $P(Cold)$ and $P(Cat)$ in our original net. First, we populate the nodes of the previous graph with products of subsets of the distributions in the original Bayes's net. Second, we use a two-phase computation—(1) inward and then (2) outward—in correspondence with the two directions of flow of computation in our earlier example. Specifically, we do the following:

1. Place each distribution in the original Bayes's net into any node that includes all the parameters indexing the distribution (or, more simply, into the node corresponding to the elimination step in which it was consumed).

2. For each node, form the product of all distributions it contains. If it contains none, initialize it to 1.

3. Pick any node as the root. From each leaf, send a message to its immediate parent. The message to be sent from a leaf is the marginal of its table, marginalizing out any parameter not in the *separator* (label on the arc, intersection of the sets of parameters on either side of the arc). When a parent receives a message from a child, it multiplies its table by the message. When a parent receives messages from all children, it performs a marginalization according to the separator on the arc to its parent and sends the corresponding message up.

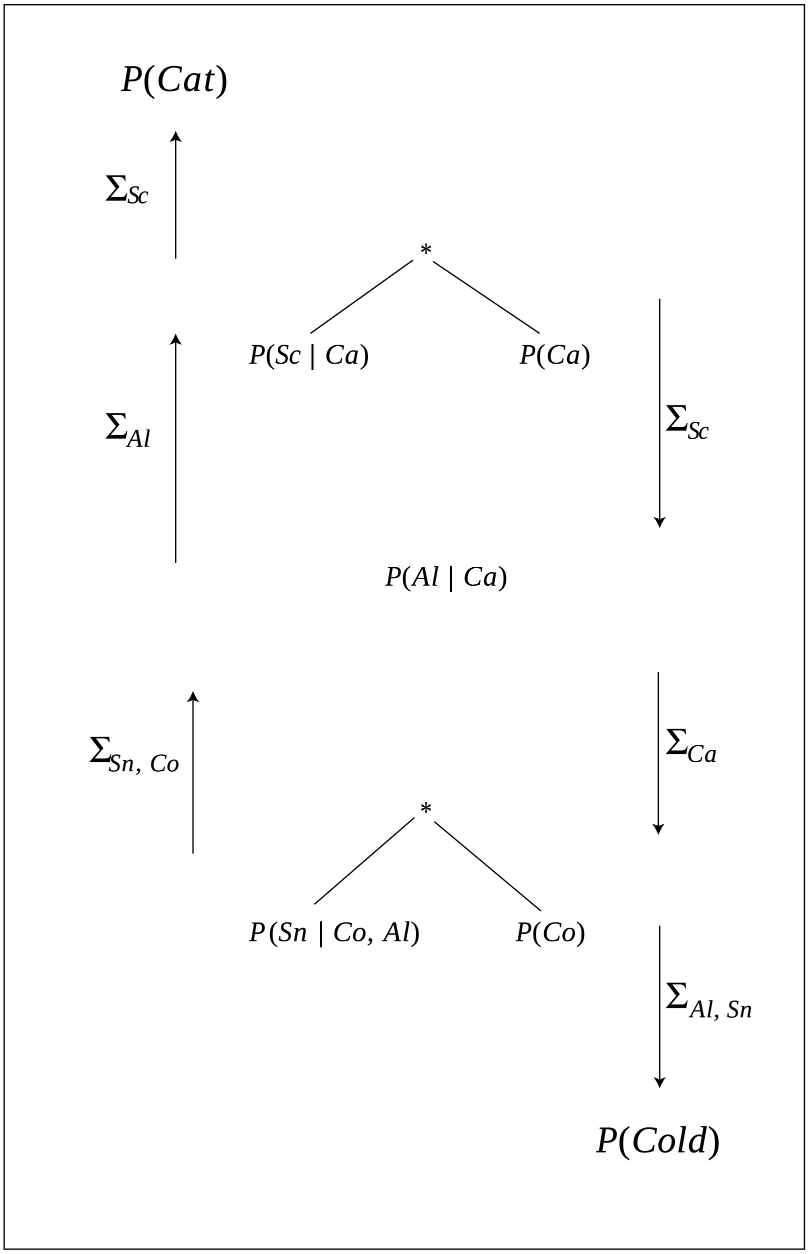

Figure 6. Information Flows in Merged Computation.

4. Once the root receives messages from all its children, the process is reversed. That is, the root sends a message down each arc to the corresponding child. However, the root must not include information already sent up by that child; so, it must first divide by the message previously sent from that child.[12] As before, each child, when it receives a message from its parent, multiplies its table by the message and sends outgoing messages to its children, first, for each child, dividing by the message it received from that child.

5. Once the message passing is complete, marginals for parameters are available by simply

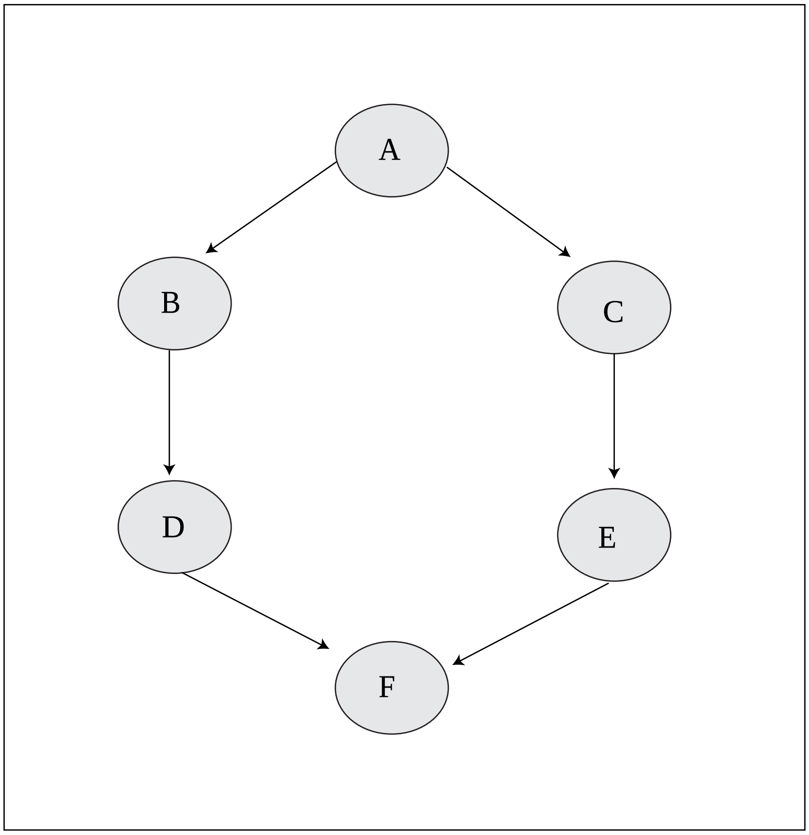

Figure 7. Sample Bayes's Net for Junction Tree.

marginalizing out the other parameters in the table for any node containing the parameter of interest.

With a bit of thought, it should become apparent that one could reconstruct each marginal computation from the previous computation. A junction tree is built from a specific elimination ordering, but it gives a graphic representation of a partial order on a possible elimination ordering, including at least one in which each parameter occurs last (and, thus, can be the subject of a single-marginal query). The partial ordering has the property that rearrangement preserves the groupings of parameters in intermediate results and, thus, maximizes structure sharing for the all-marginals task. For example, let's look at the second node (*BCD*). We follow the computation, assuming *CDE* is chosen as root.

1. *BCD* is initialized to $P(D \mid B)$.
2. *ABC* will send its contents, marginalized over *A*; so, *BCD* now contains $P(D \mid B) * \Sigma_A P(S) * P(B \mid A) * P(C \mid A)$.
3. *BCD* sends to *CDE* the message $\Sigma_B P(D \mid B) * \Sigma_A P(S) * P(B \mid A) * \mathrm{P}(C \mid \mathrm{A})$. *CDE* multiplies its contents by this message, but because it will divide it back out before sending a message back to *BCD*, we can ignore this multiplication for our current purposes.
4. *CDE* initially contains $P(E \mid C)$ and gets the message $\Sigma_F P(F \mid D, E)$ from *DEF* and multiplies it by current contents, which yields $P(E \mid C) * \Sigma_F P(F \mid D, E)$.
5. *CDE* now sends to *BCD* the message $\Sigma_E P(E \mid C) * \Sigma_F P(F \mid D, E)$. *BCD* multiplies by its current contents, yielding $(P(D \mid B) * \Sigma_A P(S) * P(B \mid A) * P(\mathrm{C} \mid A)) * (\Sigma_E P(E \mid C) * \Sigma_F P(F \mid D, E))$. However, this is exactly $P(BCD)$!

The computation is similar for other nodes in the junction tree. Regardless of the node chosen as root, at the end of the computation, each node holds the joint across the parameters associated with it. Individual parameter marginals can be obtained by marginalizing the table at any node containing the parameter of interest.

The method is elegant, and simple, and was the first and still the only well-understood, efficient, provably correct method for concurrently computing multiple queries. Furthermore, its graph-theoretic origins provide a number of useful heuristics for finding efficient elimination orderings. However, as with parameter elimination, we can ask about optimality. If we construct the effective computations for each node in the junction tree, we see that they are very closely related and are effectively close variants of the elimination ordering we used to generate the tree. Is there any reason to believe that a single elimination ordering is optimal across all single-marginal queries? To say this another way, although we understand that the method is correct, is it capable of expressing the optimal computation for the query (the set of all posterior single-parameter marginals)? This turns out to be a rather difficult question to answer and seems to depend in part on aspects of the query I have not yet discussed. Recent results indicate there might be important cases where the use of a single-elimination ordering is suboptimal.[13] Further, the method does not generalize naturally to answering queries about subjoints. Nonetheless, junction-tree propagation in its variant forms (Lepar 1998; Madsen 1998; Kjaerulff 1997) is currently the dominant method for exact inference in Bayes's nets. The algorithms are reasonably well understood, good descriptions and efficient implementations are widely available, and the complexity of inference does not depend on the particular query or pattern of evidence.

Approximate

It would be wonderful if exact algorithms were adequate for all inference tasks. However, as network size scales, exact inference times grow unpredictably. Relatively small changes in network topology can transform a relatively simple problem into an intractable one. As a result, research into approximation methods has been quite active. The methods fall into two broad categories: (1) approximate inference methods and (2) model reduction methods. The latter methods apply exact methods to a reduced (and hopefully simpler) form of the network. Because of the wide variety of methods in use, I do not attempt to describe them all in detail but survey a few of the most well-known families of methods.

Approximate Inference

Simulation: The most well-known approximate methods are based on *simulation*, that is, using the Bayes's net to generate randomly selected instantiations of the set of parameters and then counting the number of instantiations of interest. For example, imagine we want to determine the probability of finding scratches on the furniture, and for this example, let's imagine we have a simple two-parameter model (a subset of our first example containing just cat and scratch).

1. Randomly sample from *Cat* according to its prior probability (say, we choose *False*).
2. Next, randomly sample from *Scratch*, using *P(Scratch | Cat = False)*. If the result is *Scratch = True*, tally one for *Scratch = True*; else tally one for *Scratch = False*.
3. Repeat the process until bored.
4. Assume the final tallies are {104, 96} for *Scratch = True* and *False*, respectively. The estimate for the distribution over *P(Scratch)* is, therefore, {104/200, 96/200}.

There are a bewilderingly large variety of variants of this scheme, both in the uncertainty in AI literature (Fung 1994; Shachter 1989; Henrion 1988; Pearl 1987) and in the traditional statistical literature. All suffer from three problems: (1) the basic approximation problem (for example, the problem of determining if a probability is less than a specified value) is NP-hard (Dagum 1993), (2) error decreases as the square of the number of samples, and (3) unexpected evidence on nonroot nodes can reduce the number of useful samples collected. The last is particularly problematic and has largely restricted the use of such methods to the computation of priors; see Fung (1994).

Search

An alternate approach to approximation is to rely on the chain rule as in exact inference but to sum only a subset of the relevant cells of the JPDF. This can give good estimates in short time when (1) a small subset of the cells contain most of the probability mass and (2) can easily be found (Huang 1996; D'Ambrosio 1993; Poole 1993, 1992; Henrion 1991). As an example, consider our initial query, *P(Cold = True)*. Examining the JPDF table presented earlier, we see that the two largest cells have values .83 (*Cold = False, …*) and .035 (*Cold = True, …*). Using just these two, we can estimate the *P(Cold = True)* as .035/(.83 + .035) = .04, quite close to the true value of .05.

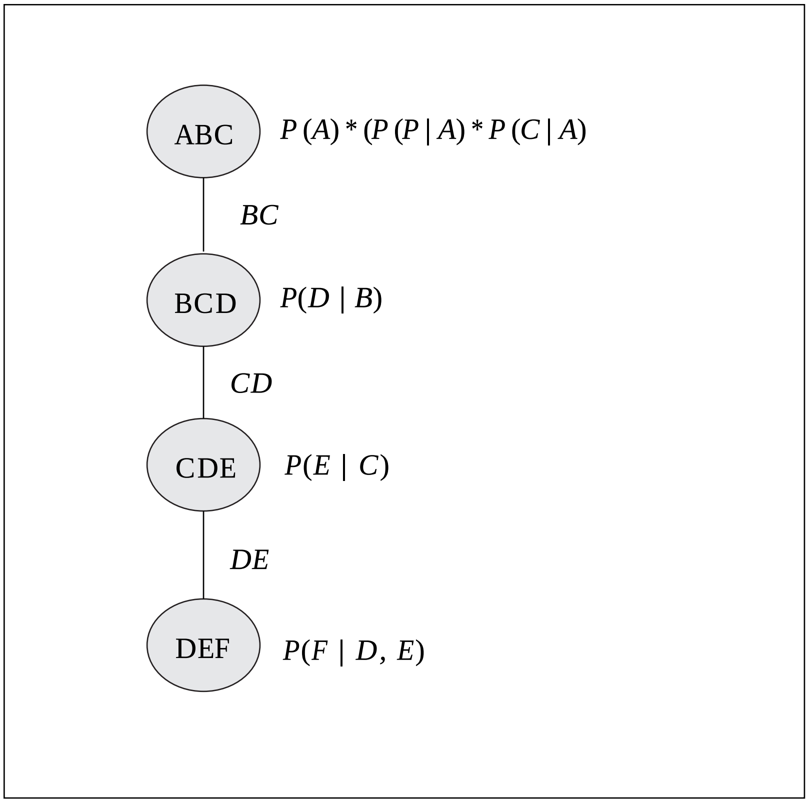

Figure 8. Junction Tree.

One application that often meets the requirements of this method is diagnosis (D'Ambrosio 1996). However, Druzdel (1994) has shown that the first requirement, at least, might be far more commonly met than might be expected. Search can be used for all queries, including MPE and MAP. One disadvantage of search is that like exact inference, the boundaries between reasonable and unreasonable computation time tend to be sharp and unpredictable. Deterministic search for large cells has bad worst-case performance. Further, it is rather complex to implement efficiently; so, it is rarely used in practice.

Model Reduction

A variety of methods have been used to find or construct a simplified network as a method of attacking difficult inference problems. These methods include reduction of the domains of parameters

(D'Ambrosio 1996; Wellman 1994; Chang 1989), elimination of irrelevant arcs (Dechter 1997; Kjaerulff 1994), local or partial evaluation (Draper 1994; Horvitz 1989), variational methods for fitting simpler parameterized models (Jaakola and Jordan 1996), and qualitative methods (Liu 1998; Goldzmidt 1995; Wellman 1990). None of these methods is in widespread use.

Challenges, Other Topics, Current Research

There are other aspects of task definition besides the question being asked. In particular, I argued (D'Ambrosio 1993) that in general, inference should be incremental with respect to query, evidence, precision, and even restricted forms of model reformulation. However, there is little known about how such incrementality would affect the tractability or suitability of Bayes's net inference for various tasks.

Local Structure

Bayesian networks exploit *value-independent conditional independence,* that is, conditional independence properties that hold at the parameter level. However, often there are conditional independence relations that hold only for specific values of a parent or child (D'Ambrosio 1991). Poole (1997), Koller (1996), and Geiger (1991) have addressed ways of representing and exploiting these relations within the Bayesian network framework.

A second kind of independence below the topological level often occurs among the parents of a parameter. Often, the effect of each parent can be modeled independently of the state of the other parents. One very important such example is the *noisy-or*, in which we can quantify independently for each parent the causal power of the parent on the child. Such models can be exploited in both network construction and inference (Zhang 1998; D'Ambrosio 1994; Srinivas 1993; Agosta 1991; D'Ambrosio 1991; Henrion 1991; Heckerman 1989), although optimal exploitation is still an open issue (Takikawa 1998).

Continuous Variables

The focus of this article is simple Bayes's nets over discrete parameters. There are several methods for incorporating continuous variables within Bayesian networks, including both exact (Chang 1991; Olesen 1991; Lauritzen 1990) and approximate (mostly simulation-based) methods.

Dynamic Networks

Dynamic probabilistic networks are compact, factored representations of Markov processes. There has been some study of inference methods, both exact (Xiang 1995) and approximate (Boyen 1998). The most common applications include projection of future values and the integration of observations over time. Koller argues that stochastic approximation methods are necessary because independence relationships among variables in a static, factored state model tend to disappear when reasoning over time. As a result, exact methods are exponential in the number of variables in the state model despite the factored representation of the prior density.

Summary

Early attempts to use probability theory in AI led to frustration, largely a result of the inability to represent and exploit the structure of large probabilistic models. The development of the Bayesian network, a factorized representation of a probability model that explicitly captures much of the structure typical in human-engineered models, has enabled direct application to problems previously beyond the reach of rigorous probabilistic methods. In this article, I introduced the basic problems and methods of reasoning with a Bayesian network from the perspective of implicitly recovering the joint distribution defined by such a network. I attempted to illustrate that differing methods make differing assumptions about the nature of the actual computational task and the structure of the model. Finally, I concluded with a brief survey of some still-open topics in inference in Bayes's nets.

Acknowledgments

This article benefited greatly from discussions with Anders Madsen and review comments from Peter Haddawy.

Notes

[1] An example is PROSPECTOR, not that PROSPECTOR was disappointing in its performance but that the use of probability theory seemed one of the obstacles, rather than contributors, to its success.

[2] Roughly, the ability to reason by construction and manipulation of structures of symbols is both necessary and sufficient for intelligent behavior.

[3] I do not attempt to define causality, nor do I mention it again once we move to a more formal level.

[4] For discrete variables, the size of a distribution is exponential in the number of dimensions or indexes.

[5] Mathematically, a Bayesian network simply presents a probability distribution in factorized form

and so is blissfully ignorant of the philosophical wars between “Bayesians” and “frequentists.” However, as a knowledge engineering device, it is often used to express prior, judgmental, or subjective belief.

6. In this case, I am building this net to help me understand circumstances I find myself in, not as an arbitrary member of the general population; so my personal sensitivity to cats is the appropriate knowledge to encode.

7. Note that the two sides of the Or are not mutually exclusive because there is a nonzero probability of *Allergy = False* when *Cat = True*. Note also that each cell is counted only once, regardless of how many disjuncts it satisfies.

8. *P'*(| *Cold*) is simply a table with one column (that is, no conditioned variables) and several rows, one for each domain value of *Cold*. This subject would be a boring object if it were a probability distribution because each row would have to sum to one. However, remember that it is not a probability distribution.

9. This information cam from Rina Dechter through personal communication.

10. What follows is not the typical way of presenting junction trees. However, I believe it provides a better intuition of the relationship of the structure to the essential problems of inference in Bayes's nets than the standard graph-theoretic presentation.

11. I ignore the case where multiple parameters are eliminated simultaneously. It introduces only minor additional complexity but would obscure the discussion.

12. Details of this division vary with the specific algorithm; see, for example, Lauritzen (1988) or Jensen (1990b).

13. Here, recent in-preparation work is represented with Anders Madsen and with Masami Takikawa.

References

Agosta, M. 1991. Conditional Inter-Causally Interdependent Node Distributions. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence,* 9–16. San Francisco, Calif.: Morgan Kaufmann.

Becker, A., and Geiger, D. 1996. A Sufficiently Fast Algorithm for Finding Close to Optimal Junction Trees. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence,* 81–89. San Francisco, Calif.: Morgan Kaufmann.

Bloemeke, M., and Valtorta, M. 1998. A Hybrid Algorithm to Compute Marginal and Joint Beliefs in Bayesian Networks and Its Complexity. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence,* 16–23. San Francisco, Calif.: Morgan Kaufmann.

Boutillier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-Specific Independence in Bayesian Networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence,* 115–123. San Francisco, Calif.: Morgan Kaufmann.

Boyen, X., and Koller, D. 1998. Tractable Inference for Complex Stochastic Processes. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence,* 33–42. San Francisco, Calif.: Morgan Kaufmann.

Chang, K. C., and Fung, R. M. 1991. Symbolic Probabilistic Inference with Continuous Variables. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*. San Francisco, Calif.: Morgan Kaufmann.

Chang, K. C., and Fung, R. M. 1989. Node Aggregation for Distributed Inference in Bayesian Networks. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 265–270. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Cooper, G. 1990. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artificial Intelligence* 42(2–3): 393–406.

Cooper, G. 1988. A Method for Using Belief Networks as Influence Diagrams. Paper presented at the 1988 Workshop on Uncertainty in AI, 19–21 August, Minneapolis, Minnesota.

Dagum, P., and Luby, M. 1993. Approximating Probabilistic Inference in Bayesian Belief Networks Is NP-Hard. *Artificial Intelligence* 60(1): 141–153.

D'Ambrosio, B. 1994. SPI in Large BN2O Networks. In *Tenth Annual Conference on Uncertainty on AI,* eds. D. Poole and R. Lopez de Mantaras, 128–135. San Francisco, Calif.: Morgan Kaufmann.

D'Ambrosio, B. 1993. Incremental Probabilistic Inference. In *Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence*, 301–308. San Francisco, Calif.: Morgan Kaufmann.

D'Ambrosio, B. 1992. Value-Driven Real-Time Diagnosis. Paper presented at the Third International Workshop on the Principles of Diagnosis, 12–14 October, Rosario, Washington.

D'Ambrosio, B. 1991. Local Expression Languages for Probabilistic Dependence. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence,* 95–102. San Francisco, Calif.: Morgan Kaufmann.

Darwiche, A., and Provan, G. 1997. A Standard Approach for Optimizing Belief Network Inference Using Query DAGs. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence,* 116–123. San Francisco, Calif.: Morgan Kaufmann.

Dechter, R. 1997. Mini-Buckets: A General Scheme for Generating Approximations in Automated Reasoning. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1297–1302. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Dechter, R. 1996. Bucket Elimination: A Unifying Framework for Probabilistic Inference. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence,* eds. E. Horvitz and F. Jensen, 211–219. San Francisco, Calif.: Morgan Kaufmann.

Draper, D. 1994. Localized Partial Evaluation of Belief Networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence,* 170–177. San Francisco, Calif.: Morgan Kaufmann.

Druzdel, M. 1994. Some Properties of Joint Probability Distributions. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence,* 187–194. San Francisco, Calif.: Morgan Kaufmann.

Fung, R., and Del Favero, B. 1994. Backward Simulation in Bayesian Networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence,* 227–234. San Francisco, Calif.: Morgan Kaufmann.

Geiger, D., and Heckerman, D. 1991. Advances in Probabilistic Reasoning. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, 118-126. San Francisco, Calif.: Morgan Kaufmann.

Geiger, D.; T. Verma, T.; and Pearl, J. 1989. D-Separation: From Theorems to Algorithms. Paper presented at the Fifth Workshop on Uncertainty in AI, 18–20 August, Windsor, Ontario, Canada.

Goldszmidt, M. 1995. Fast Belief Update Using Order of Magnitude Probabilities. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 208–216. San Francisco, Calif.: Morgan Kaufmann.

Heckerman, D. 1989. A Tractable Inference Algorithm for Diagnosing Multiple Diseases. Paper presented at the Fifth Conference on Uncertainty in AI, 18–20 August, Windsor, Ontario, Canada.

Henrion, M. 1991. Search-Based Methods to Bound Diagnostic Probabilities in Very Large Belief Nets. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, 142–150. San Francisco, Calif.: Morgan Kaufmann.

Henrion, M. 1988. Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling. In *Uncertainty in Artificial Intelligence 2*, eds. J. Lemmer and L. Kanal, 149–163. New York: Elsevier Science.

Horvitz, E.; Suermondt, H. J.; and Cooper, G. 1989. Bounded Conditioning: Flexible Inference for Decisions under Scarce Resources. Paper presented at the Fifth Conference on Uncertainty in AI, 18–20 August, Windsor, Ontario, Canada.

Huang, K., and Henrion, M. 1996. Efficient Search-Based Inference for Noisy-Or Belief Networks: Top Epsilon. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, 325–331. San Francisco, Calif.: Morgan Kaufmann.

Jaakola, T., and Jordan, M. 1996. Computing Upper and Lower Bounds on Likelihoods in Intractable Networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, 340–348. San Francisco, Calif.: Morgan Kaufmann.

Jensen, F.; Lauritzen, S.; and Olesen, K. 1990. Bayesian Updating in Causal Probabilistic Networks by Local Computation. *Computational Statistics Quarterly* 4:269–282.

Jensen, F.; Olesen, K.; and Andersen, S. 1990. An Algebra of Bayesian Belief Universes for Knowledge-Based Systems. *Networks* 20(5): 637–659.

Jensen, F., and Jensen, F. 1994. Optimal Junction Trees. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 360–366. San Francisco, Calif.: Morgan Kaufmann.

Kim, J. H., and Pearl, J. 1983. A Computational Model for Causal and Diagnostic Reasoning in Inference Engines. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 190–193. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Kjaerulff, U. 1997. Nested Junction Trees. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, 294–301. San Francisco, Calif.: Morgan Kaufmann.

Kjaerulf, U. 1994. Reduction of Computational Complexity in Bayesian Networks through Removal of Weak Dependencies. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 374–382. San Francisco, Calif.: Morgan Kaufmann.

Lauritzen, S. L. 1990. Propagation of Probabilities, Means, and Variances in Mixed Graphical Association Models, Research Report, R 90-18, Institute for Electronic Systems, Aalborg University.

Lauritzen, S., and Spiegelhalter, D. 1988. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society* B50(2): 157–224.

Lepar, V., and Shenoy, P. 1998. A Comparison of Lauritzen-Spiegelhalter, Hugin, and Shenoy-Shafer Architectures for Computing Marginals of Probability Distributions. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, 328–337. San Francisco, Calif.: Morgan Kaufmann.

Li, Z., and D'Ambrosio, B. 1994. Efficient Inference in Bayes's Nets as a Combinatorial Optimization Problem. *International Journal of Approximate Reasoning* 10(5).

Li, Z., and D'Ambrosio, B. 1993. An Efficient Approach for Finding the MPE in Belief Networks. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, 342–349. San Francisco, Calif.: Morgan Kaufmann.

Liu, C., and Wellman, M. 1998. Using Qualitative Relationships for Bounding Probability Distributions. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, 346–353. San Francisco, Calif.: Morgan Kaufmann.

Madsen, A., and Jensen, F. 1998. Lazy Propagation in Junction Trees. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, 362–369. San Francisco, Calif.: Morgan Kaufmann.

Olesen, K. G. 1991. Causal Probabilistic Networks with both Discrete and Continuous Variables, Research Report R 91-29, Institute for Electronic Systems, Aalborg University.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. San Francisco, Calif.: Morgan Kaufmann.

Pearl, J. 1987a. Distributed Revision of Composite Beliefs. *Artificial Intelligence* 33(2): 173–216.

Pearl, J. 1987b. Evidential Reasoning Using Stochastic Simulation of Causal Models. *Artificial Intelligence* 32(2): 245–258.

Peot, M. A. 1991. Fusion and Propagation with Multiple Observations in Belief Networks. *Artificial Intelligence* 48(3): 299–318.

Pewnnrock, D., and Wellman, M. 1996. Towards a Market Model of Bayesian Inference. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, 405–413. San Francisco, Calif.: Morgan Kaufmann.

Poole, D. 1997. Probabilistic Partial Evaluation: Exploiting Rule Structure in Probabilistic Inference. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1284–1291. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Poole, D. 1993. The Use of Conflicts in Searching Bayesian Networks. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, 359–367. San Francisco, Calif.: Morgan Kaufmann.

Poole, D. 1992. Search in Bayesian Horn Clause Networks. Paper presented at the Third International Workshop on Principles of Diagnosis, 12–14 October, Rosario, Washington.

Shachter, R. 1988. Probabilistic Inference and Inference Diagrams. *Operations Research* 36(6): 589-604.

Shachter, R., and Peot, M. 1989. Evidential Reasoning Using Likelihood Weighting. Paper presented at the Fifth Workshop on Uncertainty in Artificial Intelligence, 18–20 August, Windsor, Ontario, Canada.

Srinivas, S. 1993. A Generalization of the Noisy-Or Model. In *Proceedings of the Ninth Annual Conference on Uncertainty on AI*, 208–218. San Francisco, Calif.: Morgan Kaufmann.

Takikawa, M. 1998. Representation and Inference in Bayesian Networks. Ph.D. thesis, Department of Computer Science, Oregon State University.

Wellman, M. 1990. Fundamental Concepts of Qualitative Probabilistic Networks. *Artificial Intelligence* 44(3): 257–302.

Wellman, M., and Liu, C. 1994. State-Space Abstraction for Anytime Evaluation of Probabilistic Networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 567–573. San Francisco, Calif.: Morgan Kaufmann.

Xiang, Y. 1995. Optimization of Inter-Subnet Belief Updating in Multiply Sections Bayesian Networks. In *Proceedings of the Eleventh Conference on Uncertainty in AI*, 565–574. San Francisco, Calif.: Morgan Kaufmann.

Zhang, N. 1994. Intercausal Independence and Heterogeneous Factorization. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 606–614. San Francisco, Calif.: Morgan Kaufmann.

Bruce D'Ambrosio received his Ph.D. from the University of California at Berkeley in 1986. He currently splits his time between Oregon State University, where he is an associate professor in the Computer Science Department, and Information Extraction and Transport, Inc., where he is a principal scientist. D'Ambrosio's current research areas include representation and inference in Bayesian networks and advanced applications in diagnosis and dynamic situation modeling. His e-mail address is dambrosi@cs.orst.edu.