Average-case analysis of a search algorithm for estimating prior and posterior probabilities in Bayesian networks with extreme probabilities

David Poole*
Department of Computer Science,
University of British Columbia,
Vancouver, B.C., Canada V6T 1Z2
poole@cs.ubc.ca

Abstract

This paper provides a search-based algorithm for computing prior and posterior probabilities in discrete Bayesian Networks. This is an "anytime" algorithm, that at any stage can estimate the probabilities and give an error bound. Whereas the most popular Bayesian net algorithms exploit the structure of the network for efficiency, we exploit probability distributions for efficiency. The algorithm is most suited to the case where we have extreme (close to zero or one) probabilities, as is the case in many diagnostic situations where we are diagnosing systems that work most of the time, and for common-sense reasoning tasks where normality assumptions (allegedly) dominate. We give a characterisation of those cases where it works well, and discuss how well it can be expected to work on average.

1 Introduction

This paper provides a general purpose search-based technique for computing posterior probabilities in arbitrarily structured discrete[1] Bayesian networks.

Implementations of Bayesian networks have been placed into three classes [Pearl, 1988; Henrion, 1990]:

1. Exact methods that exploit the structure of the network to allow efficient propagation of evidence [Pearl, 1988; Lauritzen and Spiegelhalter, 1988; Jensen *ct* a/., 1990].
2. Stochastic simulation methods that, give estimates of probabilities by generating samples of instantiations of the network, using for example Monte Carlo techniques (see [Henrion, 1990]).
3. Search-based approximation techniques that search through a space of possible values to estimate probabilities.

At one level, the method in this paper falls into the exact class; if it is allowed to run to completion, it will have computed the exact conditional probability in a Bayesian network. It, however has the extra feature that it can be stopped before completion to give an answer, with a known error. Under certain distribution assumptions (Section C) it is shown that convergence to a small error is quick.

While the efficient exact methods exploit aspects of the network structure, we instead exploit extreme probabilities to gain efficiency. The exact methods work well for sparse networks (e.g., are linear for singly-connected networks [Pearl, 1988]), but become inefficient when the networks become less sparse. They do not take the distributions into account. The method in the paper uses no information on the structure of the network, but rather has a niche for classes of problems where there are "normality" conditions that dominate the probability tables (see Section 6). The algorithm is efficient for these classes of problems, but becomes very inefficient as the distributions become less extreme. This algorithm should thus be seen as having an orthogonal niche to the algorithms that exploit structure for efficiency.

2 Background

2.1 Probability

In this section we give a semantic view of probability theory[2] and describe the general idea behind the search method. In some sense the idea of this method has nothing to do with Bayesian networks — we just have to commit to some independence assumptions to make the algorithm more concrete.

We assume we have a set of random variables (written in upper case). Each random variable has an associated set of values; *vals(X)* is the set of possible values of variable X. Values are written in lower case. An atomic proposition is an assignment of a value to a random variable; variable *X* having value *c* is written as X = c. Each assignment of one value to each random variable is associated with a possible world. Let Ω be the set of all possible worlds. Associated with a possible world w is a measure $\mu(\omega)$, with the constraint that $\forall\omega,\ 0 \leq \mu(\omega) \leq 1$ and

$$1 = \sum_{\omega \in \Omega} \mu(\omega).$$

* Scholar, Canadian Institute for Advanced Research.

[1] All of the variables have a discrete, and here even finite, set of possible values.

[2] This could have also been presented as joint distributions, with probabilistic assignments to the possible worlds corresponding to joint distributions. If that view suits you, then please read 'possible worlds' as 'elementary events in a joint distribution' [Pearl, 1988, p. 33].

We write $\omega \models X = c$ (read "X has value c in world ω") if variable X is assigned value c in world ω. Each variable has exactly one value in a possible world; thus we can see a random variable as a set of mutually exclusive and covering propositions (i.e., the propositions of the form $X = v$ for each $v \in vals(X)$). A **proposition** is a logical formula made up of atomic propositions and the usual logical connectives. We define the truth of propositions in a world using the normal truth tables (e.g., $\omega \models \alpha \wedge \beta$ if $\omega \models \alpha$ and $\omega \models \beta$).

Probability is a function from propositions to $[0, 1]$, defined by

$$P(\alpha) = \sum_{\omega \in \Omega : \omega \models \alpha} \mu(\omega).$$

Conditional probability is defined by:

$$P(\alpha|\beta) = \frac{P(\alpha \wedge \beta)}{P(\beta)}$$

If β is a conjunction of observations, then $P(\alpha|\beta)$ is the **posterior probability** of α. $P(\alpha)$ is the **prior probability** of formula α.

2.2 Searching possible worlds

For a finite number of variables with a finite number of values, we can compute the probabilities directly, by enumerating the possible worlds. This is, however, com0-putationally expensive as there are exponentially many of these (the product of the sizes of the domains of the variables).

The idea behind the search method presented in this paper is motivated by considering the questions:

- Can we estimate the probabilities by only enumerating a few of the possible worlds?
- How can we enumerate just a few of the most probable possible worlds?
- Can we estimate the error in our probabilities?
- For what cases does the error get small quickly?
- How fast does it converge to a small error?

This paper sets out to answer these questions, for the case where the distribution is given in terms of Bayesian networks.

2.3 Bayesian Networks

A *Bayesian network* [Pearl, 1988] is a graphical representation of (in)dependence amongst random variables. A Bayesian network is a directed acyclic graph where the nodes represent random variables. If there is an arc from variable *B* to variable *A*, *B* is said to be a parent of *A*. The independence assumption of a Bayesian network says that each variable is independent of its non-descendents given its parents.

Suppose we have a Bayesian network with random variables $X_1, \ldots, X_n$. The parents of A', are written as $\Pi_{X_i} = \langle X_{i_1}, \cdots, X_{i_{k_i}} \rangle$.

Associated with the Bayesian network are conditional probability tables which gives the marginal probabilities of the values of X_i, depending on the values of its parents Π_{X_i}. This consists of, for each $v_j \in vals(X_j)$, probabilities of the form

$$P(X_i = v_i | X_{i_1} = v_{i_1} \wedge \cdots \wedge X_{i_{k_i}} = v_{i_{k_i}}).$$

For any probability distribution, we can compute a joint distribution by

$$P(X_1, \cdots, X_n) = \prod_{i=1}^{n} P(X_i | \Pi_{X_i}).$$

This is often given as the formal definition of a Bayesian network and is shorthand for the propositions that the variables have particular values. That is

$$P(X_1 = v_1 \wedge \cdots \wedge X_n = v_n) = \prod_{i=1}^{n} P(X_i = v_i | X_{i_1} = v_{i_1} \wedge \cdots \wedge X_{i_{k_i}} = v_{i_{k_i}})$$

An assignment of values to all the variables corresponds to a possible world. For the rest of this paper we will equate a possible world with an assignment of values to all of the variables, and use the two interchangeably (e.g., in using the complete description on the left hand side of the '$\models$').

3 Enumerating possible worlds in order

3.1 Ordering the variables

The first thing we do is to impose a total ordering on the variables that is consistent with the ordering of the Bayesian network. We index the random variables $X_1, \ldots, X_n$ such that the parents of a node have a lower index than the node. This can always be done as the nodes in a Bayesian network form a partial ordering. If the parents of X_i are $\Pi_{X_i} = \langle X_{i_1}, \cdots, X_{i_{k_i}} \rangle$, the total ordering preserves $i_j < i$.

3.2 Search Tree

We are now in a position to determine a search tree for Bayesian networks[3].

Definition 3.1 A **partial description** is a tuples of values $\langle v_1, \cdots, v_j \rangle$ where v_i is an element of the domain of variable X_i.

The search tree has nodes labelled with partial descriptions, and is defined as follows:

- The root is labelled with the empty tuple $\langle\rangle$.
- The children of node labelled with $\langle v_1, \cdots, v_j \rangle$ are the nodes labelled with $\langle v_1, \cdots, v_j, v \rangle$ for each $v \in vals(X_{j+1})$. In other words, the children of a node correspond to the possible values of the next variable in the total ordering.
- The leaves of the tree are tuples of the form $\langle v_1, \cdots, v_n \rangle$.

[3]This search tree is the same as the probability tree of [Howard and Matheson, 198l] and corresponds to the semantic trees used in theorem proving [Chang and Lee, 1973, Section 4.4], but with random variables instead of complementary literals.

$Q := \{\langle\rangle\}$;
$W := \{\}$;
While $Q \neq \{\}$ do
    choose and remove $\langle v_1, \cdots, v_j \rangle$ from Q;
    if $j = n$
      then $W := W \cup \{\langle v_1, \cdots, v_j \rangle\}$
      else $Q := Q \cup \{\langle v_1, \cdots, v_j, v \rangle : v \in vals(X_{j+1})\}$

Figure 1: Basic search algorithm

Partial description ($v_1, \cdots, V_j$) corresponding to the variable assignment

$$X_1 = v_1 \wedge \cdots \wedge X_j = v_j.$$

There is a one to one correspondence between leaves of the tree and possible worlds (or complete assignments to the variables).

We associate a probability with each node in the tree. The probability of the partial description (v_1, ...,v_j) is the probability of the corresponding proposition:

$$P(X_1 = v_1 \wedge \cdots \wedge X_j = v_j)$$

This is well defined as, due to our variable ordering, all of the parents of each variable X_i, $0 < i \leq j$ has a value in the partial description.

The following lemma can be trivially proved, and is the basis for the search algorithm.

Lemma 3.2 *The probability of a node is equal to the sum of the probabilities of the leaves that are descendents of the node.*

This lemma lets us bound the probabilities of possible worlds by only generating a few of the possible worlds and placing bounds on the sizes of the possible worlds we have not generated.

3.3 Searching the Search Tree

To implement the computation of probabilities, we carry out a search on the search tree, and generate some of the most likely possible worlds. There are many different search methods that can be used [Pearl. 1984].

Figure 1 gives a generic search algorithm that can be varied by changing which element is chosen from the queue. There is a priority queue Q of nodes, and a set W of generated worlds. We remove a node (e.g., the most likely); either it is a leaf (if $j = n$) in which case it is added to W or else its children are added to Q.

Note that each partial description can only be generated once. There is no need to check for multiple paths or loops in the search. This simplifies the search, in that we do not need to keep track of a CLOSED list or check whether nodes are already on the OPEN list (Q in Figure 1) [Pearl, 1984].

No matter which element is chosen from the queue at each time, this algorithm halts and when it halts *W* is the set of all tuples corresponding to possible worlds.

4 Estimating the Probabilities

If we let the above algorithm run to completion we have an exponential algorithm for enumerating the possible worlds that can be used for computing the prior probability of any proposition or conjunction of propositions. This is not, however, the point of this algorithm. The idea is that we want to stop the algorithm part way through, and determine any probability we want to compute.

We use *W,* at the start of an iteration of the while loop, as an approximation to the set of all possible worlds. This can be done irrespective of the search strategy used.

4.1 Prior Probabilities

Suppose we want to compute *P(g).* At any stage (at the start of the while loop), the possible worlds can be divided into those that are in *W* and those that will be generated from *Q*.

$$\begin{aligned} P(g) &= \sum_{w \in \Omega \wedge w \models g} P(w) \\ &= \left(\sum_{w \in W \wedge w \models g} P(w) \right) + \left(\sum_{w \in \Omega - W \wedge w \models g} P(w) \right) \end{aligned}$$

We can easily compute the first of these sums, and can bound the second. The second sum is greater than zero and is less than the sum of the probabilities of the partial descriptions on the queue (using Lemma 3.2). This means that we can bound the probabilities of a proposition based on enumerating just some of the possible worlds. Let

$$P_W^g = \sum_{w \in W \wedge w \models g} P(w)$$

$$P_Q = \sum_{t \in Q} P(t).$$

Lemma 4.1 $P_W^g \leq P(g) \leq P_W^g + P_Q$.

As the computation progresses, the probability mass in the queue P_Q approaches zero and we get a better refinements on the value of $P(g)$. Note that P_Q is monotonically non-increasing through the loop (i.e P_Q stays the same or gets smaller through the loop). This thus forms the basis of an "anytime" algorithm for Bayesian networks.

4.2 Posterior Probabilities

The above analysis was for finding the prior probability of any proposition. If we want to compute the posterior probability of some g given some observations *obs*, we can use the definition of conditional probability, and use

$$P(g|obs) = \frac{P(g \wedge obs)}{P(obs)}$$

We can estimate the conditional probability from our our estimates of $P(g \wedge obs)$ and $P(obs)$, (namely $P_W^{g \wedge obs}$ and P_W^{obs}) by noticing that each element of the queue can go towards implying $obs \wedge \neg g$, $obs \wedge g$ or $\neg obs$. We can easily prove the inequality:

Lemma 4.2

$$\frac{P_W^{g \wedge obs}}{P_W^{obs} + P_Q} \leq \frac{P_W^{g \wedge obs}}{P_W^{obs}} \leq \frac{P_W^{g \wedge obs} + P_Q}{P_W^{obs} + P_Q}$$

It can be proved that $P(g|obs)$ has the following bound:

Theorem 4.3

$$\frac{P_W^{g\wedge obs}}{P_W^{obs}+P_Q} \leq P(g|obs) \leq \frac{P_W^{g\wedge obs}+P_Q}{P_W^{obs}+P_Q}$$

For a proof see Appendix A.

If we choose the midpoint as an estimate, the maximum error is

$$\frac{1}{2}\left(\frac{P_W^{g\wedge obs}+P_Q}{P_W^{obs}+P_Q} - \frac{P_W^{g\wedge obs}}{P_W^{obs}+P_Q}\right) = \frac{P_Q}{2(P_W^{obs}+P_Q)}$$

What is interesting about this is that the error is independent of g. Thus when we are generating possible worlds for some observation, and want to have posterior estimates within some error, we can generate the required possible worlds independently of the proposition that we want to compute the probability of.

5 Search Strategies

The above analysis was independent of the search strategy (i.e., independent of which element we remove from the queue).

We can carry out various search strategies, to enumerate the most likely possible worlds. For example, [Poole, 1993] discusses a multiplicative version of A^* [Pearl, 1984], with conflicts used to refine a heuristic function.

In this paper the search strategy we use is where the element of the queue with highest prior probability is chosen at each stage. This paper does not study various search strategies, but analyses one. I make no claim that this is the best search strategy (see Section 7), but it forms a base case with which to compare other strategies.

6 Complexity

The problem of finding the posterior probability of a proposition in a Bayesian network is NP hard [Cooper, 1990]. Thus we should not expect that our algorithms will be good in the worst case. Our algorithm, when run to completion, is exponential in computing the exact prior and posterior probability of a hypothesis.

Because of the the "anytime" nature of our algorithm, which trades search time for accuracy, we should not consider run time independently of error. It is interesting to estimate how long it takes on average to get within some error, or how accurate we can expect (or guarantee as as asymptotic behaviour) to be within a certain run time.

As we have probabilities it is possible to carry out an average case complexity analysis of our algorithm.

If we make no assumptions about the probability distributions, the average case of finding the most likely explanation or prior probability within error $\epsilon < 1$ is exponential in the size n of the Bayesian network [Provan, 1988]. This can be seen by noticing that the size of complete descriptions are linear in n and so the probability of explanations is exponentially small in n. This means that we need to consider exponentially many explanations to cover any fixed proportion of the probability mass.

This is not always a reasonable distribution assumption, for example, when using this for diagnosis of a system that basically works we would like to assume that the underlying distribution is such that there is one assignment of values (the "normal values") that dominates the probability mass. This also may be appropriate for commonsense reasoning tasks where normality assumptions (allegedly) dominate (i.e., we assume that abnormality is rare [McCarthy, 1986]).

For our analysis we assume that we have extreme probabilities for each conditional probability given. For each value of the parents of variable A\, we assume that one of the values for X_i is close to one, and the other values are thus close to zero. Those that are close to one we call *normality* values; those that are close to zero we call *faults*:

Definition 6.1 If we have extreme probabilities (i.e., all of the conditional probabilities are close to one or zero), then the **faults** of partial description $\langle v_1, \cdots, v_k \rangle$ are those v_i such that $P(X_i = v_i | X_{i_1} = v_{i_1} \wedge \cdots \wedge X_{i_{k_1}} = v_{i_{k_1}})$ is close to zero.

Which values are faults is thus context dependent and depends on the values of the predecessor variables. The term "fault" is derived from its use in diagnosis, where deviation from normality is a fault. The non-monotonic reasoning community has called such things *abnormalities* [McCarthy, 1986].

Each element of the queue consists of a set of normality conditions and a set of faults. For each set of faults there can be at most one element of Q or W whose faults are exactly that set. A k-fault partial description is one with exactly k faults.

For our complexity analysis, let p ($p \approx 1$) be the probability of the least likely normality condition. We can interpret our analysis as being for the case where all of the normality conditions have value p or as a bound for the cases where all normality conditions are greater than p. Let $f = 1 - p$. f is a bound on the probability of the faults.

Let $b + 1$ be a bound on the number of values of a variable. Thus b is the maximum number of fault values of any variable given its predecessors. We assume that b is fixed (it is not a function of the number of variables).

6.1 Error convergence

Suppose we have n variables in our Bayesian network. Consider the case where we are about to choose the first k-fault partial description from the queue. We assume for this analysis that $k \ll n$.

At this time there can be no $(k+1)$-fault node on the queue (as we can only generate $(k+1)$-fault nodes from k-fault nodes and we have not yet expanded out first k-fault node), so there are at most $\binom{bn}{k}$ possible combinations of faults on the priority queue (choosing k faults from the set of bn fault assumptions). Each partial description on the queue has a probability less than f^k

(not because they all have k faults, but because something with probability less than f^k has been chosen from the priority queue and it was the element with highest probability). Thus the probability mass in the queue can be bounded by $\binom{bn}{k} f^k$. So

$$\begin{aligned} P_Q &\leq \binom{bn}{k} f^k \\ &\leq \frac{(bn)^k}{k!} f^k \\ &= \frac{b^k (nf)^k}{k!}. \end{aligned}$$

Thus, we have convergence (as the computation proceeds and k increases) when $bnf < 1$.

In order to interpret the value of nf, we use the following lemma:

Lemma 6.2 If nf is small then $nf \approx 1 - p^n$.

p^n is the probability that there are no faults. Thus $1-p^n$ is the prior probability of some fault in the system. This is low when we are diagnosing a system which works most of the time, and has rare errors. We will carry out our analysis for the class of systems where the error rate of the total system is low. That is where $\delta = nf$ is low (at least less than $\frac{1}{b}$). The analysis is parametrized by δ. This becomes the class of Bayesian networks that we analyse. N.B., with δ fixed, $f \to 0$ as $n \to \infty$.

6.2 How fast is convergence?

The time to compute the posterior probability of any formula (of bounded size) given some observations using this search algorithm is dominated by the search. This is because the search already includes checking every generated possible world. Thus we just consider generating the possible worlds.

To see how fast we reach convergence, consider how long it takes to reach the stage where we are about to choose the first k-fault hypothesis from the queue. We assume for this analysis that $k \ll n$.

There are at most $(bn+1)^k$ ways to generate k or fewer faults. For each of these combinations of faults, we go through the while-loop of Figure 1, at most n times (for the other normality assumptions). The only thing in the while loop that depends on the size of n, is adding and removing elements from the priority queue, which can be done in $\log |Q|$ time.

$$\log |Q| \leq \log \left(\frac{(bn)^k}{k!} \right) = k(\log n + \log b) - \log k! = O(\log n)$$

(for fixed b and k). Thus we can reach the stage where we are taking the first k-fault hypothesis off the queue in $O(n^{k+1} \log n)$ time.

We can combine the above results on the queue size and time to reach the stage where where we are taking the first k-fault hypothesis off the queue to give us:

Lemma 6.3 For fixed b and k, if $\delta < \frac{1}{b}$ we can attain an accuracy of $\frac{(b\delta)^k}{k!}$ in the estimate of prior probabilities in $O(n^{k+1} \log n)$ time.

For example, if $b = 1$ (there is only one fault mode) we can attain an accuracy of δ in $O(n^2 \log n)$ time. We can attain an accuracy of $\frac{\delta^2}{2}$ in $O(n^3 \log n)$ time. We can attain an accuracy of $\frac{\delta^3}{6}$ in $O(n^4 \log n)$ time.

Theorem 6.4 We can obtain an accuracy of ϵ in time

$$O\left(n^{1+\frac{\log \epsilon}{\log \delta} - k} \log n\right).$$

See Appendix A for a proof of this theorem.

6.3 Posterior Probabilities

As discussed in Section 4.2, at any stage through the loop in the algorithm of Figure 1, we can estimate the posterior probability of g given *obs* with an error that is independent of g.

To make the posterior error less than E, we require

$$\frac{P_Q}{2(P_W^{obs} + P_Q)} < \epsilon$$

this occurs when

$$P_Q < \frac{2\epsilon P_W^{obs}}{1 - 2\epsilon}$$

which can be ensured if we make sure that

$$P_Q < 2\epsilon P_W^{obs}$$

We can use the analysis for the prior probability, but multiplying the error bound by a factor that is an estimate of *P(obs)*. As it is unlikely that the observations have a low probability, it is unlikely to have a situation where the error term required is dominated by the probability of the observation. This observation is reflected in Theorem 6.5 below.

The following theorem gives a PAC (probably approximately correct) characterization of the complexity[4].

Theorem 6.5 In the space of all systems, to compute the posterior probability of any proposition (of bounded size) given observation *obs*, we can guarantee an error of less than ϵ ($\epsilon < \frac{1}{2}$), for at l e $1 - \psi$) f the cases in time

$$O\left(n^{1+\frac{\log \epsilon \psi}{\log \delta \delta} - k} \log n\right)$$

See Appendix A for a proof of this theorem.

Note that in this theorem we are considering "the space of all systems". For diagnosis, this means that we consider a random artifact. Most of these have no faults; and presumably would not be the subject of diagnosis. Thus the space of all systems is probably not the space of all systems that we are likely to encounter is a diagnostic situation. A more realistic space of systems by which to judge our average-time behaviour is the space of all broken systems, that is those that have at least one fault[5]. We are thus excluding all but b of

[4] This has the extra property that we know when we are in a case for which we cannot guarantee the error; when we have run our algorithm we know our error.

[5] It could also be argued that this is also inappropriate; we would rather consider the space of systems that exhibit faulty behaviour. This would be much harder to analyse here, as we have no notion of the observable variables developed in this paper. The space of broken devices seems like a reasonable approximation.

the systems. To exclude all but ψ of the broken systems we have to exclude all but $b\psi$ of the total number of systems. We thus have the following corollary.

Corollary 6.6 In the space of all broken systems (with at least one fault), to compute the posterior probability of any proposition (of bounded size) given observation *obs*, we can guarantee an error of less than ϵ ($\epsilon < \frac{1}{2}$), for at least $1 - \psi$ of the cases in time

$$O\left(n^{1+\frac{\log \epsilon + \log \psi}{\log bf}} \log n\right)$$

7 Refinements

There are a number of refinements that can he carried out to the algorithm of Figure 1. Some of these are straightforward, and work well. The most straightforward refinements are:

If we are trying to determine the value of $P(a)$, we can stop enumerating the partial descriptions once it can be determined whether a is true in that partial description. When conditioning on an observation we can prune any partial description that is inconsistent with the observation.

We do not really require that we find the most likely possible worlds in order, as we are just summing over them anyway. One way to improve the algorithm is to carry out a depth-first depth-bounded search. We can guess the probability of the least likely possible world we will need to generate, use this as a threshold and carry out a depth-first search pruning any partial description with probability less that this threshold. If the answer is not accurate enough, we decrease the threshold and try again. This is reminiscent of iterative deepening A* [Korf, 1985], but we can decrease the bound in larger ratios as we do not have to find the most likely possible world.

We can use conflicts [de Kleer, 1991] to form a heuristic function for a multiplicative version of A* [Poole, 1993].

See [Poole, 1993] for a Prolog implementation that incorporates these refinements.

8 Comparison with other systems

The branch and hound search is very similar to the candidate enumeration of de Kleer's focusing mechanism [de Kleer, 199I]. This similarity to a single step in de Kleer's efficient method indicates the potential of the search method. He has also been considering circuits with thousands of components, which correspond to Bayesian networks with thousands of nodes. It seems to be very promising to combine the pragmatic efficiency issues confronted by de Kleer, with the Bayesian network representation, and the error bounds obtained used in this paper.

Poole [1992a] has proposed a Prolog-like search approach that can be seen as a top-down variant of the bottom-up algorithm presented here. It is not as efficient as the one here. Even if we consider finding the single most normal world, the algorithm here corresponds to forward chaining on definite clauses (see [Poole, 1992b]), which can be done in linear time, but backward chaining has to search and takes potentially exponential time. The backward chaining approach seems more suitable however when we have a richer language [Poole, 1992b].

D'Ambrosio [1992] has a backward chaining search algorithm for "incremental term computation", where he has concentrated on saving and not recomputing shared structure in the search. This seems to be a very promising approach for when we do not have as extreme probabilities as we have assumed in this paper.

Shimony and Chamiak [1990] have an algorithm that is a backward chaining approach to finding the most likely possible world. The algorithm is not as simple as the one presented here, and has worse asymptotic behaviour (as it is a top-down approach — see above). It has not been used to find prior or posterior probabilities, nor has the average-case complexity been investigated.

This paper should be seen as a dual to the TOP-N algorithm of Henrion [1991]. We have a different niche. We take no account of the noisy-OR distribution that Henrion concentrates on.

9 Conclusion

This paper has considered a simple search strategy for computing prior and posterior probabilities in Bayesian networks. It is a general purpose algorithm, that is always correct, and has a niche where it works very well. We have characterised this niche, and have given bounds on how badly it can be expected to perform on average. How common this niche is, is, of course, an open question, but the work in diagnosis and nonmonotonic reasoning would suggest that reasoning about normality is a common task.

A Proofs

Theorem 4.3

$$\frac{P_W^{g \wedge obs}}{P_W^{obs} + P_Q} \leq P(g|obs) \leq \frac{P_W^{g \wedge obs} + P_Q}{P_W^{obs} + P_Q}$$

Proof: Consider what happens to the elements of the queue. Let α be the proportion of the possible worlds that are descendents of elements of the priority queue in which $obs \wedge \neg g$ is true. Let β be the proportion in which $obs \wedge g$ is true. Then $\alpha + \beta$ is the proportion in which *obs* is true.
As all of the possible worlds are either in W or are descendents of elements of the priority queue, we have

$$P(g|obs) = \frac{P_W^{g \wedge obs} + \beta P_Q}{P_W^{obs} + (\alpha + \beta) P_Q}$$

We want to maximise this formula under the constraints that $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$ and $0 \leq \alpha + \beta \leq 1$. There are no internal extrema in this formula, and so the maxima occur at the extremes. There are $\alpha = \beta = 0$, $\alpha = 1 \wedge \beta = 0$ and $\alpha = 1 \wedge \beta = 0$. These correspond to the three values in Lemma 4.2, and the theorem follows directly from Lemma 4.2. □

Theorem 6.4 We can obtain an accuracy of ϵ in time

$$O\left(n^{1+\frac{\log \epsilon}{\log bf}} \log n\right)$$

Proof: If we require an accuracy of ϵ, we ensure $\frac{(b\delta)^k}{k!} < \epsilon$. This can be obtained if we ensure $(b\delta)^k = \epsilon$. Solving for k, we get

$$\begin{aligned} k \log b\delta &= \log \epsilon \\ k &= \frac{\log \epsilon}{\log b\delta} \end{aligned}$$

This requires $O(n^{k+1} \log n)$ time. Substituting the value for k, the theorem follows. □

Theorem 6.5 In the space of all systems, to compute the posterior probability of any proposition (of bounded size) given observation *obs*, we can guarantee an error of less than ϵ ($\epsilon < \frac{1}{2}$), at least $1 - \psi$ of the cases in time

$$O\left(n^{1+\frac{\log \frac{1}{\psi\epsilon}}{\log b\delta}} \log n\right)$$

Proof: We can assume that the $P(obs) \geq \psi$. This will be wrong less than ψ of the cases (by definition). If we make sure that $P_Q < \frac{\psi}{2}$, then

$$\frac{\psi}{2} < P(obs) - P_Q \leq P_W^{obs}$$

To achieve error ϵ, we make sure $P_Q < \epsilon\psi$. Then, as $\epsilon\psi < 2\epsilon P_W^{obs}$, we will have $P_Q < 2\epsilon P_W^{obs}$, which, as described above implies that the error will be less than ϵ. Thus we have to ensure that $P_Q < min(\frac{\psi}{2}, \epsilon\psi) = \epsilon\psi$. By Theorem 6.4, this can be done in time

$$O\left(n^{1+\frac{\log \frac{1}{\psi\epsilon}}{\log b\delta}} \log n\right)$$

□

Acknowledgements

Thanks to Andrew Csinger, Michael Horsch, Runping Qi and Nevin Zhang for valuable comments on this paper. This research was supported under NSERC grant OGP0044121, and under Project B5 of the Institute for Robotics and Intelligent Systems.

References

[Chang and Lee, 1973] C-L Chang and R. C-T Lee. Symbolic Logical and Mechanical Theorem Proving. Academic Press, New York, 1973.

[Cooper, 1990] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. Artificial Intelligence, 42(2-3):393 405, March 1990.

[D'Ambrosio, 1992] B. D'Ambrosio. Real-time value-driven diagnosis. In Proc. Third International Workshop on the Principles of Diagnosis, pages 86-95, Rosario, Washington, October 1992.

[de Kleer, 199l] J. de Kleer. Focusing on probable diagnoses. In Proc. 9th National Conference on Artificial Intelligence, pages 842-848, Anahiem, Cai., July 1991.

[Henrion, 1990] M. Henrion. An introduction to algorithms for inference in belief nets. In M. Henrion, et al., editor, Uncertainty in Artificial Intelligence 5, pages 129-138. North Holland, 1990.

[Henrion, 1991] M. Henrion. Search-based methods to bound diagnostic probabilities in very large belief networks. In Proc. Seventh Conf. on Uncertainty in Artificial Intelligence, pages 142-150, Los Angeles, Cal., July 1991.

[Howard and Matheson, 198l] R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. Matheson, editors, The Principles and Applications of Decision Analysis, pages 720-762. Strategic Decisions Group, CA, 1981.

[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.

[Korf, 1985] K. E. Korf. Depth-first iterative deepening: an optimal admissable tree search. Artificial Intelligence, 27(1):97-109, September 1985.

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and J). J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, Series 5, 50(2):157-224, 1988.

[McCarthy, 1986] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. Artificial Intelligence, 28(1):89 116, February 1986.

[Pearl, 1984] J. Pearl. Heuristics. Addison-Wesley, Reading, MA, 1984.

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

[Poole, 1992a] D. Poole. Logic programming, abduction and probability. In International Conference on Fifth Generation Computer Systems (FGCS-92), pages 530 538, Tokyo, June 1992.'

[Poole, 1992b] D. Poole. Probabilistic Horn abduction and Bayesian networks. Technical Report 92-20, Department of Computer Science, University of British Columbia, August 1992. To appear, Artificial Intelligence 1993.

[Poole, 1993] D. Poole. The use of conflicts in searching Bayesian networks. Technical Report 93-??, Department of Computer Science, University of British Columbia, March 1993.

[Provan, 1988] G. Provan. A complexity analysis for assumption-based truth maintenance systems. In B. M. Smith and R. Kelleher, editors, Reason Maintenance Systems and Their Applications, pages 98-113. Ellis Howard, 1988.

[Shimony and Charniak, 1990] S. E. Shimony and E. Charniak. A new algorithm for finding MAP assignments to belief networks. In Proc. Sixth Conf. on Uncertainty in Artificial Intelligence, pages 98-103, Cambridge, Mass., July 1990.