Probabilistic conflicts in a search algorithm for estimating posterior probabilities in Bayesian networks

David Poole
Department of Computer Science,
University of British Columbia,
2366 Main Mall,
Vancouver, B.C., Canada V6T 1Z4
Tel: (604) 822 6254
Fax: (604) 822 5485
Email: poole@cs.ubc.ca
http://www.cs.ubc.ca/spider/poole

May 7, 1996

Abstract

This paper presents a search algorithm for estimating posterior probabilities in discrete Bayesian networks. It shows how conflicts (as used in consistency-based diagnosis) can be adapted to speed up the search. This algorithm is especially suited to the case where there are skewed distributions, although nothing about the algorithm or the definitions depends on skewness of distributions. The general idea is to forward simulate the network, based on the 'normal' values for each variable (the value with high probability given its parents). When a predicted value is at odds with the observations, we analyse which variables were responsible for the expectation failure — these form a conflict — and continue forward simulation considering different values for these variables. This results in a set of possible worlds from which posterior probabilities — together with error bounds — can be

derived. Empirical results with Bayesian networks having tens of thousands of nodes are presented.

Abbreviated title: Probabilistic conflicts for searching Bayes nets

1 Introduction

This paper is about evidential reasoning as typified by the problem of diagnosis (determining what is inside an artifact/patient based on observations) or recognition. This paper combines two approaches to model-based diagnosis, namely Bayesian networks [12, 30] and consistency-based diagnosis [13, 38, 9, 8].

Bayesian networks provide a general and natural representation for reasoning under uncertainty. They have been successfully applied to such diverse areas as medical diagnosis [15, 40, 26, 35], diagnosis of bottlenecks in computer systems [1], circuit diagnosis [12, 41], fraud detection [10] and plan recognition [36].

Implementations of Bayesian networks have been placed into three classes [30, 17]:

1. Exact methods that exploit the structure of the network to allow efficient propagation of evidence (e.g., [30, 25, 23]).

2. Stochastic simulation methods that give estimates of probabilities by generating samples of instantiations of the network (e.g., [16, 29, 21, 11, 22]).

3. Search-based approximation techniques that search through a space of possible values to estimate probabilities (e.g., [18, 4]).

The method presented in this paper falls into the last class. This paper provides a search-based technique for computing posterior probabilities in arbitrarily-structured discrete[1] Bayesian networks. The algorithm gives a way to bound the error of the probability estimates.

Developed from logical notions of diagnosis, consistency-based diagnosis is founded on the use of the *conflict* [38, 9, 8]. A conflict is a set of assumptions, the conjunction of which is inconsistent with the observations and the system description. Consistency-based diagnosis has been used in many application areas (see the

[1] All of the variables have a finite set of possible values. We do not consider variables with an infinite set of possible values.

papers in [14]). While these have been developed in the context of logical system descriptions, probabilities have been used to reduce the combinatorial explosion in the number of logical possibilities [9, 7]. This paper can be seen in two ways. One is as a way to add a notion of conflict to improve the speed and accuracy of a search algorithm for Bayesian networks. The second is as a way to extend the languages of consistency-based diagnosis to allow for probabilistic system descriptions. See Appendix A.

The problem of approximating probabilities in Bayesian networks to within any fixed error (less than 0.5) is NP-hard [3]. This means that there can be no generally efficient procedure for approximating posterior or even prior probabilities in Bayesian networks. It does not mean that there are not classes of Bayesian networks for which there are efficient algorithms. One such class is the class of singly connected Bayesian networks [30]. Another is the class of Bayesian networks with sufficiently skewed probability distributions (all probabilities in the Bayesian network are close to one or zero); the skewness of the probabilities is what is being exploited for efficiency by the algorithm in this paper (see [31]).

For practical efficiency we have to exploit some aspect of the problem. Two possibilities are to exploit structure or distributions [5]. While the efficient exact methods exploit aspects of the network structure, we instead exploit aspects of the probability distribution to gain efficiency. The exact methods work well for sparse networks (e.g., are linear for singly-connected networks [30]), but become inefficient when the networks become less sparse. They do not take the distributions into account. The method in this paper uses no information about the structure of the network, but rather has a niche for classes of problems where there are skewed distributions — conditional probabilities of variables given their parents are close to one or zero (this includes the prior probabilities of variables without parents). The algorithm is efficient for these classes of problems, but becomes very inefficient as the distributions become less extreme — see [31] for a detailed average-case complexity analysis of the simple version of the algorithm presented here (without conflicts). This algorithm should thus be seen as having an orthogonal niche to the algorithms that exploit the structure for efficiency. This paper does not consider how to exploit both structure and distributions together [5], but rather tries to see how far we can get without considering network structure.

The general idea can be stated simply. With skewed probabilities, there is a 'normal' value for each variable given its parents. By forward simulation on the network, we instantiate variables in turn to their normal value. This can be done quickly with very little bookkeeping, and when probabilities are sufficiently skewed,

the most likely world(s) contain much of the probability mass. When evidence is at odds with the predicted value, we analyse which variables are responsible for this expectation failure — these form a conflict. We then consider the alternative values for the variables in the conflict, and continue with the forward simulation. Posterior probabilities with tight error bounds can be computed from the generated assignments of values to the variables.

2 Bayesian Networks

We assume we have a set of **random variables**. Each random variable has an associated set of **values**. An **atomic proposition** is an assignment of a value to a random variable; variable X having value c is written as $X = c$. A **proposition** is made up of atomic propositions and the usual logical connectives.

A **Bayesian network** [30] is a graphical representation of (in)dependence amongst random variables. A Bayesian network is a directed acyclic graph where the nodes represent random variables[2]. 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 is that each variable is independent of its non-descendents given its parents.

Suppose we have a Bayesian network with random variables $X_1, ..., X_n$. The parents of X_i are written as $\Pi_{X_i} = \left\langle X_{i_1}, \cdots, X_{i_{k_i}} \right\rangle$. k_i is the number of parents of variable X_i.

$vals(X_i)$ is the set of possible values of random variable X_i . If $C = \left\langle X_{C_1}, \cdots, X_{C_j} \right\rangle$ is a tuple of variables, then the set of values of C is the Cartesian product:

$$vals(C) = vals(X_{C_1}) \times \cdots \times vals(X_{C_j}).$$

If $v = \left\langle v_{C_1}, \cdots, v_{C_j} \right\rangle \in vals(C)$, then $C = v$ (i.e., $\left\langle X_{C_1}, \cdots, X_{C_j} \right\rangle = \left\langle v_{C_1}, \cdots, v_{C_j} \right\rangle$) means the proposition

$$X_{C_1} = v_{C_1} \wedge \cdots \wedge X_{C_j} = v_{C_j}.$$

Associated with the Bayesian network are conditional probabilities which give the conditional probabilities of the values of X_i depending on the values of its parents Π_{X_i}. These consists of, for each $v_i \in vals(X_i)$ and $v_{i_j} \in vals(X_{i_j})$, probabilities of the form

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

[2] We will use the terms *node* (in a Bayesian network) and *random variable* interchangeably — which is meant at any time should be clear from the context.

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

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

often written as

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

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

We call an assignment of values to all the variables a **possible world**, and write '$\omega \models X_i = v_i$' if X_i is assigned value v_i in world ω. Let Ω be the set of all possible worlds. The truth value of a proposition in a possible world is determined using the standard truth tables. Possible worlds are important because the probability of any proposition can be calculated from the probabilities of possible worlds:

$$P(g) = \sum_{w \in \Omega : w \models g} P(w).$$

3 Searching possible worlds

The idea behind our search algorithm is that we estimate conditional probabilities by only enumerating a few of the possible worlds.

3.1 Ordering the variables

The first thing to do is impose a total ordering on the variables that is consistent with the ordering of the Bayesian network. We index the random variables $X_1, ..., X_n$ so 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 $\Pi_{X_i} = \left\langle X_{i_1}, \cdots, X_{i_{k_i}} \right\rangle$, 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].

[3]This search tree is the same as the probability tree of [20] and corresponds to the semantic trees used in theorem proving [2, Section 4.4], but with random variables instead of complementary literals.

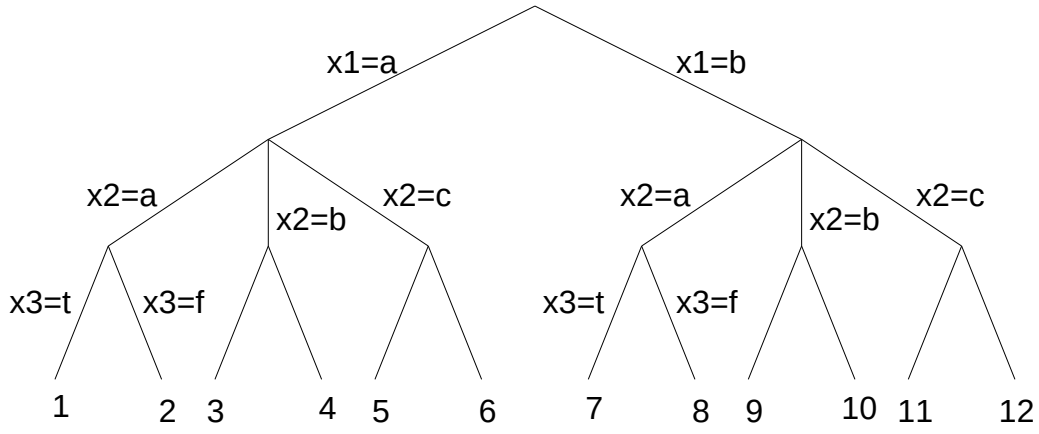

Figure 1: A search tree for three variables.

Definition 3.1 A **partial description** is a tuple of values $\langle v_1, \cdots, v_j \rangle$ where each v_i is an element of the domain of variable X_i. (i.e., $v_i \in vals(X_i)$).

Partial description $\langle v_1, \cdots, v_j \rangle$ corresponds to the variable assignment $X_1 = v_1 \wedge \cdots \wedge X_j = v_j$.

The variables of partial description $\langle v_1, \cdots, v_j \rangle$, written $vars(\langle v_1, \cdots, v_j \rangle)$ is the set $\{X_1, \ldots, X_j\}$.

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

- The root of the tree is labelled with the empty tuple $\langle\rangle$ (where $j = 0$).
- 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 labelled with tuples of the form $\langle v_1, \cdots, v_n \rangle$. These correspond to possible worlds.

We will use the terms node and partial descriptions interchangeably — which is meant at any time should be clear from the context.

For example, Figure 1 shows a search tree on three variables $x1$ (with $vals(x1) = \{a, b\}$), $x2$ (with $vals(x2) = \{a, b, c\}$) and $x3$ (with $vals(x3) = \{t, f\}$). The total

ordering is $x1 < x2 < x3$. The numbers at the bottom of the tree represent the possible worlds. For example, world 7, defined by the partial description $\langle b, a, t \rangle$, corresponds to the proposition $x1 = b \wedge x2 = a \wedge x3 = t$.

We associate a probability with each partial description and so with each node in the tree. The probability of the node labelled with $\langle v_1, \cdots, v_j \rangle$ is the probability of the corresponding proposition which is

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

This is easy to compute; given the probability of the parent of the node, it can be done in constant time.

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

Lemma 3.2 The probability of a node in the search tree 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 compute probability estimates, we expand part of the search tree, and generate some of the most likely possible worlds. Figure 2 gives a generic search algorithm that can be varied by changing which element is chosen from the queue. There are many different search methods that can be used [27].

The algorithm maintains a priority queue Q of partial descriptions. Each time through the loop an element of Q is removed; either it is a total description (i.e., where $j = n$) in which case it is added to W, the set of generated worlds, or else its children are added to the queue.

If we let the algorithm run to completion it halts, and when it halts W is the set of all partial descriptions corresponding to possible worlds. The correctness doesn't depend on the search strategy (i.e., which element is chosen from the queue at each time).

$Q := \{\langle\rangle\}$;
$W := \{\}$;
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 2: Basic search algorithm

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; we want to stop the algorithm part way through, and use the worlds generated to estimate probabilities.

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 $P(g)$ for proposition g. At any stage (at the start of an iteration of the while loop), the possible worlds can be divided into those that are in W and those that will be generated from Q.

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

We can easily compute the first of these sums, and can bound the second. The second sum is greater than or equal to zero and is less than or equal to 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

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

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

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 refinement 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 — P_Q decreases whenever an element with non-zero probability is added to W and stays the same otherwise). This thus forms the basis of an "anytime" algorithm for Bayesian networks.

4.2 Posterior Probabilities

If we want to compute the posterior probability of some g given some observations obs, we can use the definition of conditional probability,

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

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 B.

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

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

It is interesting 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.

4.3 Refinements to the Search Algorithm

There are a number of refinements that can be carried out to the algorithm of Figure 2, independently of the search strategy.

If we are trying to determine the value of $P(\alpha)$, we don't have to expand a partial description if it can be determined whether α or $\neg\alpha$ is entailed by the partial description (and so also by all of its descendents). When conditioning on our observations we can prune any partial description that is inconsistent with the observations — we know that all descendents of the partial description are inconsistent with the observations, and so the probability of the pruned node can be removed from consideration.

Figure 3 gives a refined algorithm for enumerating the possible worlds consistent with obs. Here both Q and W_{obs} are sets of pairs $\langle \pi, p\rangle$ where π is a partial description and p is the probability of π. W_{obs} is the set of the generated partial descriptions that correspond to possible worlds in which obs is true. P_Q and P_W^{obs} are the probabilities of Q and W_{obs} respectively. This algorithm shows explicitly how these can be computed.

$inconsistent(obs, \langle v_1, \cdots, v_j\rangle)$ is true if obs is inconsistent with the partial description $\langle v_1, \cdots, v_j\rangle$. If obs is a conjunction, then as this stage is not reached unless $\langle v_1, \cdots, v_{j-1}\rangle$ is consistent with obs; in this case obs is inconsistent with the partial description iff obs contains a conjunct of the form $X_j = v'_j$ where $v_j \neq v'_j$.

At any stage at the start of the while loop, this algorithm directly gives P_Q and P_W^{obs}. For any g, $P_W^{g\wedge obs}$ can be computed by testing each member of W_{obs} to see whether it is consistent with g. Alternatively, if g is known before the search is commenced, it can be incorporated into the search (each partial description in Q and W_{obs} can be marked by whether it is consistent or inconsistent with g).

$Q := \{\langle\langle\rangle, 1\rangle\}$;
$P_Q = 1$;
$W_{obs} := \{\}$;
$P_W^{obs} := 0$;
While $Q \neq \{\}$ do
choose and remove $\langle\langle v_1, \cdots, v_j\rangle, \sigma\rangle$ from Q;
if $inconsistent(obs, \langle v_1, \cdots, v_j\rangle)$
then $P_Q := P_Q - \sigma$
else if $j = n$
then $W_{obs} := W_{obs} \cup \langle\{\langle v_1, \cdots, v_j\rangle, \sigma\rangle\}$;
$P_W^{obs} := P_W^{obs} + \sigma$;
$P_Q := P_Q - \sigma$
else $Q := Q \cup \{\langle\langle v_1, \cdots, v_j, v\rangle, \sigma P(X_{j+1} = v | X_1 = v_1, \cdots, X_j = v_j)\rangle$
$: v \in vals(X_{j+1})\}$

Figure 3: Search algorithm for finding worlds in which obs is true.

For the rest of this paper we consider only the problem of generating the appropriate worlds and the error bounds, as this is the difficult computational task.

5 A Diagnosis Example

In this section we describe how the search procedure can be applied to a simple circuit diagnosis problem (as in [7]), from which we can learn what problems arise. The translation of the circuit into a Bayesian network will follow that of Geffner and Pearl [12].

The circuit is a sequence of one-bit adders, cascaded to form a multiple-bit adder. We chose this example as it is simple to extend to large systems and also because it was used in [7].

Note that there is an efficient algorithm for such an example using clique tree propagation [25, 23] that exploits the structure of the network to allow local propagation of conditioning information. A slight variant of the example would make clique tree propagation not work nearly as well. For example, if we add another circuit to the output of the adders, the algorithm in this paper would work the same,

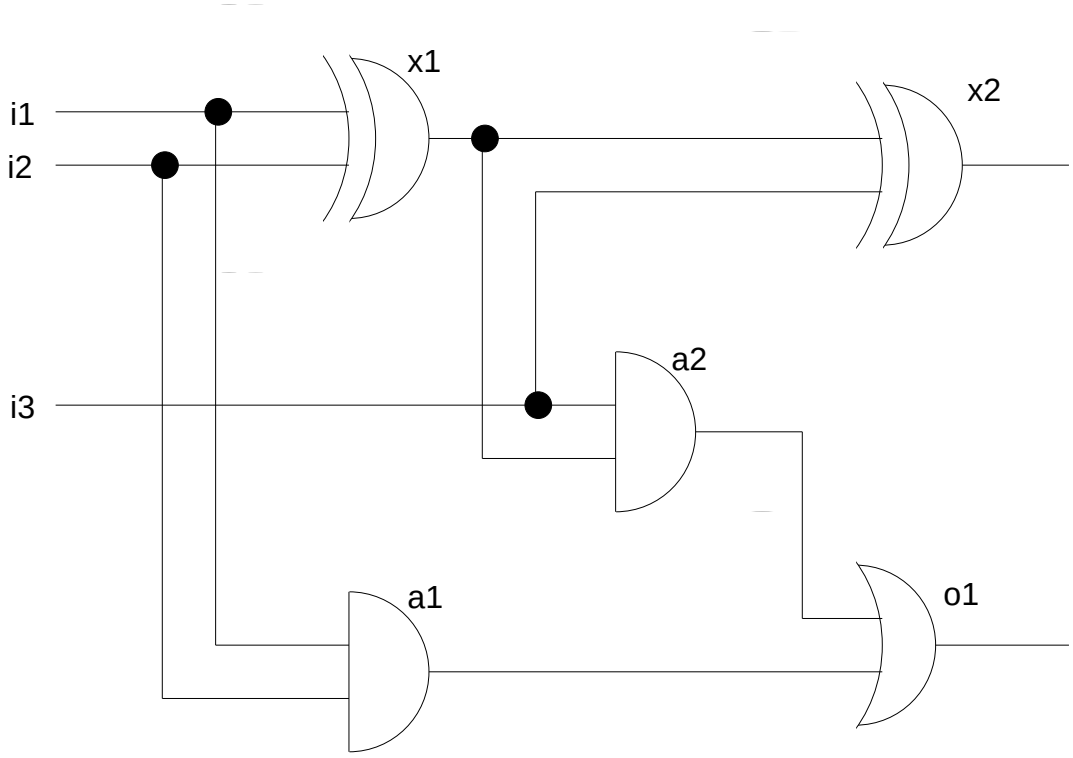

Figure 4: 1 bit adder

but the clique tree propagation would require larger cliques.

5.1 Representation

Figure 4 shows a one bit adder. Figure 5 shows a corresponding Bayesian network under the assumption that the gates fail independently.

In this Bayesian network the random variable $out\text{-}a2$ is a binary random variable with $vals(out\text{-}a2) = \{on, off\}$; $out\text{-}a2 = on$ means that the output of gate $a2$ is on, and $out\text{-}a2 = off$ means the output of gate $a2$ is off. The variables $i1$, $i2$, $i3$, $out\text{-}a1$, $out\text{-}x1$, etc., have the same values. The random variable $a2ok$ has four possible values: $vals(a2ok) = \{ok, stuck1, stuck0, ab\}$; $a2ok = ok$ means gate $a2$ is working correctly, $a2ok = stuck1$ means gate $a2$ is broken and always

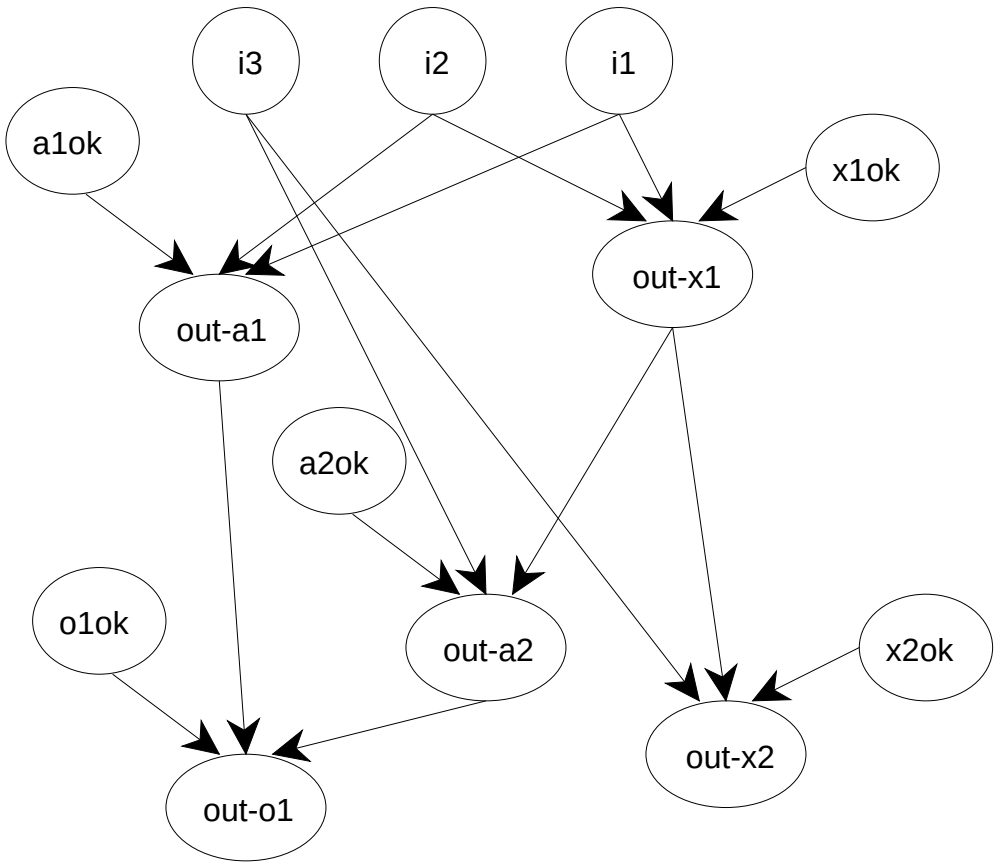

Figure 5: Bayesian network for a 1 bit adder

$a2ok$	$i3$	out-$x1$	out-$a2$	
			on	*off*
ok	on	on	1	0
ok	on	*off*	0	1
ok	*off*	on	0	1
ok	*off*	*off*	0	1
$stuck1$	–	–	1	0
$stuck0$	–	–	0	1
ab	–	–	0.5	0.5

Figure 6: Conditional probability table for variable out-$a2$.

$a2ok$			
ok	$stuck1$	$stuck0$	ab
0.99999	0.0000049	0.0000049	0.0000002

Figure 7: Conditional probability table for variable $a2ok$.

produces *on*, $a2ok = stuck0$ means gate $a2$ is broken, and always produces *off*, and $a2ok = ab$ means the gate is in an abnormal state that could produce either value. The other ok variables have the same set of values.

The value of out-$a2$ depends on the values of the variables, $i3$, out-$x1$, and $a2ok$. The conditional probabilities for the variable out-$a2$ follow the table in Figure 6. The conditional probabilities for the outputs of other gates is similar.

The value of $a2ok$ does not depend on any other variables. The values for the variable follow the table in Figure 7.[4] The probabilities for other ok variables are similar.

These one-bit adders can be cascaded to form multiple bit adders. This is done in the circuit by connecting the output of gate $o1$ in one adder to input $i3$ of the following adder. In the Bayesian network, this is done by having multiple instances

[4]The numbers are made up. It may seem as though these probabilities are very extreme, but a 1000 bit adder (with 5000 components), is only 95% reliable, if all of the gates are as reliable as that given in this table.

$out\text{-}o1_{k-1}$	$i3_k$	
	on	*off*
on	1	0
off	0	1

Figure 8: Conditional probability table for input 3 of adder k.

of the network for the one-bit adder with the value of $i3$ depending on the variable $out\text{-}o1$ for the previous instance of the adder. The table for the probabilities is given in Figure 8. The value of the output of gate $x2$ of bit k, is called the *output* of bit k; the value of the output of $o1$ is called the *carry*.

5.2 Computation

Suppose we apply the algorithm of Figure 3 to our cascaded adder example with the partial description with the highest prior probability chosen each time through the loop. First the world with all gates being ok is generated followed by the worlds with single faults, then the double stuck-at faults are generated, etc. These are pruned whenever they are found to be inconsistent with the observations. This is similar to the candidate generator phase of [7]. From this candidate generation, we can compute all of the probabilities that we need to.

To see what computational problem arises, consider a 1000-bit adder. Suppose all the inputs are zero, and all outputs, except bit k, are zero, and bit k outputs one (this example is from [7]). We first choose the most likely values of all variables (e.g., the ok state for all of the status nodes) up to the variable that represents the output of bit k. The output of bit k, which is predicted to be zero, is inconsistent with the observations. At this stage, we prune the search (Section 4.3) and consider the single-fault possible worlds. For each bit after bit k, we have already assigned a single fault (to account for the error in bit k), thus for each of these gates, we only consider the ok state. For the gates before bit k, we consider each of the single-fault states. Most of these are useless since they will need to be combined with a fault to account for the error at bit k.

Learning what we can about expectation failure and using this information for pruning the search is the basis for the conflicts developed in the following sections.

6 Search Strategy and Conflicts

The above example assumed a simple search strategy, but was not as good as it could be because we did not use the information that we discovered during the search. Here we present a solution to the search inefficiency by incorporating a notion of *conflict* analogous to that used in consistency-based diagnosis [9, 38, 8].

A 'conflict set' of Reiter [38] (a 'conflict' of de Kleer and Williams [9]) is a set of components such that, given the system description and the observations, not all of the components can be normal[5]. In the probabilistic case, 'normality' corresponds to a variable being assigned a value that maximises its probability given its parents. Conflicts corresponds to sets of variables all of which cannot be normal given the observations. We exploit the fact that some (presumably large) proportion of the probability mass on the variables in a conflict is inconsistent with the observations. In this paper we will use a probabilistic bound to define a notion of conflict that can be used in our search algorithm.

In order to make this most flexible and useful, the definitions do not appeal to normality; a conflict can be based on any values of the variables being in conflict. Like de Kleer, Mackworth and Reiter [8], we generalise conflicts to not depend on normality, although our notion of a conflict is very different to their's as it does not appeal to a logical specification of a system description, but rather extracts a conflict directly from a Bayesian network and an observation.

6.1 Bounding Functions

For each partial description $\pi \in Q$, we use an estimate of $P(\pi \wedge obs)$ rather than an estimate of $P(\pi)$ in the computation. A notion of 'conflict' will be used to refine this estimate.

Lemma 6.1

$$P(obs) \;=\; P_W^{obs} + \sum_{\pi \in Q} P(\pi \wedge obs)$$

See Appendix B for a proof of this lemma.

[5] See Appendix A for a description of the relationship between the consistency-based diagnosis work and the probabilistic framework presented here.

Lemma 6.2

$$P_W^{g \wedge obs} \leq P(g \wedge obs) \leq P_W^{g \wedge obs} + \sum_{\pi \in Q} P(\pi \wedge obs)$$

Definition 6.3 A **bounding function** for observation obs is a function f^{obs} such that if π is a partial description, $f^{obs}(\pi)$ is a number satisfying

$$P(\pi \wedge obs) \leq f^{obs}(\pi) \leq P(\pi).$$

Define the **queue mass induced by** f^{obs} to be $f_Q^{obs} = \sum_{\pi \in Q} f^{obs}(\pi)$. Bounding function f is **tighter** than bounding function f' if $f(\pi) \leq f'(\pi)$ for all π.

The following theorem is analogous to Theorem 4.3, with a similar proof.

Theorem 6.4 If f^{obs} is a bounding function for obs, then

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

The error bound is

$$\frac{f_Q^{obs}}{2(P_W^{obs} + f_Q^{obs})}$$

which is smaller than the previous estimate if $f^{obs}(\pi) < P(\pi)$ for some $\pi \in Q$. Tighter bounding functions give smaller error bounds.

In the next sections we define a notion of a conflict that allows us to tighten bounding functions.

6.2 Overview of Algorithm

The general idea behind the algorithm is that we proceed much as the algorithm in Figure 3, but choosing the partial description in the queue depending on the value of its bounding function. Initially the bounding value we use is the probability of the partial description.

When we find an expectation failure (the partial world we are considering is inconsistent with the observations), we try to extract what information we can from this expectation failure. This information is in terms of what is called a conflict. Conflicts are used to make tighter bounding functions for elements of the queue.

$Q := \{\langle\}\}$;
$W := \{\}$;
$C := \{\}$;
While $Q \neq \{\}$ do
choose and remove $\langle v_1, \cdots, v_j \rangle$ from Q;
if $inconsistent(obs, X_j = v_j)$
then $C := C \cup \{extract\_conflict(obs, \langle v_1, \cdots, v_j \rangle)\}$
else 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 9: Search algorithm for finding worlds in which conjunctive obs is true.

By choosing the most likely partial descriptions at any time, we are effectively considering the "normal" values (the values whose conditional probabilities given values for the parents are high) first, and want to extract probability bounds from these.

The revised search algorithm is shown in Figure 9. Here C is the set of conflicts that are used to define the "best" element of Q. For simplicity of exposition, we have used the simpler representation of Figure 2. The actual algorithm incorporates the improvements of Figure 3 into Figure 9.

There are a number of issues to be discussed:

1. How can a conflict be defined with only probabilistic information, and without imposing an a priori constraint that all of the probabilities are extreme (as in [34])?

2. How can conflicts be used by the search algorithm?

3. How can conflicts be discovered?

4. How does the use of conflicts affect the estimation of probabilities?

5. How much does the use of conflicts save in search time?

6. In practice, how often can we detect a small set of variables that form a conflict?

In this paper we answer all but the last of these questions. The last question we cannot answer until we have built many more systems for many diverse applications.

We assume for the rest of this paper that observations are conjunctions of assignments of values to different variables. It is straightforward to extend this analysis to the case where observations are conjunctions of disjunctions of assignments of values to the same variable (i.e., of the form $\bigwedge_i (X_i = v_1 \vee \cdots \vee X_i = v_k)$). It is not clear how the results could be extended to more general forms of observations, but it is also not clear how one could actually observe more complex formulae.

6.3 Conflicts

The main idea of a conflict is that there is some set of variables such that we can say that some proportion of the probability mass on these variables will be inconsistent with the observations. Conflicts can be used to define a tighter bounding function to prune the search (Theorem 6.12).

Definition 6.5 If C is a set of variables, define the **predecessors** of C to be

$$C^- = \{X | X \text{ is a variable, and } (\exists Y \in C \text{ such that } X < Y) \text{ and } X \notin C\}$$

where $X < Y$ means that variable X is before Y in the total ordering of variables.

We need to generalize a conflict from being a set of variables such that all of the variables being 'normal' is inconsistent the observations. The generalisation is that the sum of the probabilities of the values consistent with the observations is less than some ϵ (for all values of the ancestors of these variables). This is the basis of the following definition:

Definition 6.6 Given a Bayesian network and an observation obs, a **conflict** is a pair $\langle C, \epsilon \rangle$ where C is a tuple of variables and $0 \leq \epsilon \leq 1$ such that

$$\max_{a \in vals(C^-)} \left(\sum_{\substack{v \in vals(C) \\ consis(C = v \wedge C^- = a, obs)}} P(C = v | C^- = a) \right) \leq \epsilon$$

where $consis(C = v \wedge C^- = a, obs)$ is true if $C = v \wedge C^- = a$ is consistent with the observations. ϵ is called the **bound** of the conflict.

Conflicts $\langle C_1, \epsilon_1 \rangle$ and $\langle C_2, \epsilon_2 \rangle$ are **independent** if $C_1 \cap C_2 = \{\}$. If they are independent, there is no single variable that can account for both conflicts. A set of conflicts is independent if they are pairwise independent.

The use of independent conflicts is given by the lemma:

Lemma 6.7 If $\langle C_1, \epsilon_1 \rangle$ and $\langle C_2, \epsilon_2 \rangle$ are conflicts such that $C_1 \cap C_2 = \{\}$ then $\langle C_1 \cup C_2, \epsilon_1 \times \epsilon_2 \rangle$ is a conflict.

Example 6.8 Suppose we have a Bayesian network, where amongst the variables are $i7$ and $o7$, and suppose that $i7$ has no parents and $o7$ has $i7$ as its only parent. Suppose they are both Boolean variables that can take values from {*on*,*off*}, where the probabilities for the network are $P(i7 = on) = 0.5$, $P(o7 = on|i7 = on) = 0.1$ and $P(o7 = on|i7 = off) = 0.8$. With the observation $i7 = on \wedge o7 = on$, there are two independent conflicts, namely $\langle \langle i7 \rangle, 0.5 \rangle$ and $\langle \langle o7 \rangle, 0.1 \rangle$. These are independent and so there is a conflict $\langle \langle i7, o7 \rangle, 0.05 \rangle$.

Because of these variables, the prior probability of obs must be at most 0.05. Moreover, if π is a partial description that does not include variables $i7$ or $o7$, then $P(\pi \wedge obs) \leq P(\pi) \times 0.05$. It is this last feature that we exploit for our search algorithm.

Example 6.9 Suppose we have variables $i7$ and $o7$ as in example 6.8, but with $P(i7 = on) = 0.7$, $P(o7 = on|i7 = on) = 0.6$ and $P(o7 = on|i7 = off) = 0.9$. With the observation $i7 = on \wedge o7 = on$, there are two independent conflicts, namely $\langle \langle i7 \rangle, 0.7 \rangle$ and $\langle \langle o7 \rangle, 0.6 \rangle$. These are independent and so there is a conflict $\langle \langle i7, o7 \rangle, 0.42 \rangle$. The way we have defined the notion of a conflict does not demand that the conflicts are only for the "normal" values of the variables. The conflicts here can be discovered by our algorithm below (if we are searching for more than the most likely possible world), and can be used in exactly the same way as the more extreme conflicts.

Example 6.10 In our example of Section 5, with all inputs zero, and bit 50 having output one and all other outputs being zero, there is a conflict:
$\langle \langle$ out-$x2_{50}$, $x2ok_{50}$, $i3_{50}$, out-$o1_{49}$, $o1ok_{49}$, out-$a1_{49}$, $a1ok_{49}$, $i2_{49}$, out-$a2_{49}$, $a2ok_{49}$, out-$x1_{49}$, $x1ok_{49}$, $i1_{49}$, out-$x1_{50}$, $x1ok_{50}$, $i1_{50}$, $i2_{50}$ $\rangle$, 0.00003 $\rangle$.

Example 6.11 Suppose that in our cascaded adder example, the inputs to the circuit were observations rather than defined as part of the circuit. This is useful if we do not know the inputs at the time the circuit is built, or if some inputs are unknown

during the diagnosis. We need prior probabilities such as $P(i1_k = on) = 0.5$. For every bit k for which input 1 is known, $\langle\{i1_k\}, 0.5\rangle$ is a conflict. Although the prior probabilities of diagnoses becomes very small quickly, the use of these conflicts can prune the search as though the observations of the inputs were given as part of the network.

6.4 Refining the bounding function

We use a conflict to update the bounding function. The simplest idea[6] is that $f^{obs}(\langle v_1, \cdots, v_j\rangle)$ is the product of $P(\langle v_1, \cdots, v_j\rangle)$ and the the bound of a conflict that does not involve the variables $\{X_1, \cdots, X_j\}$. This result is formalised in the following theorem:

Theorem 6.12 Given observation obs, and a set of conflicts, the function f^{obs} defined by

$$f^{obs}(\pi) = P(\pi) \times \min\{\epsilon : \langle C, \epsilon\rangle \text{ is a conflict such that } C \cap vars(\pi) = \{\}\}$$

is a bounding function of obs, where $vars(\pi)$ is the set of variables assigned values in the partial description π.

This theorem is proved in Appendix B.

We want the bounding function to be as tight as possible, and so want the conflict with the smallest bound. Typically this conflict is the product of independent conflicts that involve variables that are after X_j in the total ordering.

A discovered conflict updates the bounding function for all the variables before (in the total variable ordering) the conflict. The bounding functions of elements of the queue evolves as computation progresses and conflicts are found.

Example 6.13 Just using the conflicts of Example 6.8, the conflict $\langle\langle i7, o7\rangle, 0.05\rangle$ means that if π_1 is a partial description on the priority queue that does not contain $i7$ or $o7$, then $f^{obs}(\pi_1) = P(\pi_1) \times 0.05$. Thus π_1 contributes much less to the error estimate of the priority queue — all of the estimated probabilities have a smaller error bound. If π_2 contains $i7$ but not $o7$, then $f^{obs}(\pi_2) = P(\pi_2) \times 0.1$.

Because the error estimates are much tighter using conflicts and the bounding function, fewer iterations are needed to obtain the same error bound.

[6] A more sophisticated version is developed in Section 6.8.

6.5 Extracting conflicts

We have now seen how to use conflicts, but it is not much use without being able to find them. Rather than building an architecture (such as an ATMS [6]) to find conflicts, we would like to extract them from our normal search. When we have predicted something which turns out to be inconsistent with the observations, we would like to learn from this, and extract conflicts from such expectation failures.

We would expect to be able to extract conflicts from expectation failures as an expectation failure gives us set of variables and values (a partial description) which are inconsistent with the observations. We will try to find a subset of these variable assignments that is also a conflict.

Definition 6.14 Partial description $\pi = \langle v_1, \cdots, v_i \rangle$ is **minimally inconsistent** with observation obs if $X_1 = v_1 \wedge \cdots \wedge X_i = v_i$ is inconsistent with obs and $X_1 = v_1 \wedge \cdots \wedge X_{i-1} = v_{i-1}$ is consistent with obs.

A minimally inconsistent partial description $\pi = \langle v_1, \cdots, v_i \rangle$ partitions the conjuncts in the observation into:

$obs^{\pi-}$, the conjunction of those variable assignments of obs involving variables before i in the total ordering. This conjunction is consistent with π, as π is minimally inconsistent.

$obs^{\pi 0}$, the variable assignment in obs involving variable X_i. $obs^{\pi 0}$ is inconsistent with π.

$obs^{\pi+}$, the conjunction of those variable assignments involving variables after i in the total ordering. This conjunction is consistent with π (as $obs^{\pi+}$ and π mention a disjoint set of variables).

Definition 6.15 $\langle C, \epsilon \rangle$ is a **counter** to formula f with respect to observation obs and partial description π if $P(f|C^- = v) \leq \epsilon$ whenever $consis(C^- = v, obs^{\pi-})$.

This notion of a counter is useful, because we can compute conflicts from counters and we can extract counters from expectation failure in our search. The following theorem shows how to extract conflicts from counters:

Theorem 6.16 If $\langle C, \epsilon \rangle$ is a counter to $obs^{\pi 0}$ with respect to observation obs and minimally inconsistent partial description π then $\langle C, \epsilon \rangle$ is a conflict with respect to obs.

For a proof see Appendix B.

Note that not all conflicts are from counters. Counters are meant to find those conflicts that can be extracted from expectation failure.

6.6 Extracting Counters

If $\pi = \langle v_1, \cdots, v_i \rangle$ is minimally inconsistent with obs then, by our assumption of the form of observations (Section 6.2), $obs^{\pi 0}$ is of the form $X_i = v$ for some v.

In this section we define the procedure $extract\_counter(X_i = v, obs, \pi)$ where $X_i = v$ is an assignment which is inconsistent with π.

We will prove that $extract\_counter(X_i = v, obs, \pi)$ will return a counter to formula $X_i = v$ with respect to observation obs and partial description π whenever $X_i = v$ is inconsistent with π.

We have to find some C so that we can bound $P(X_i = v|C^- = a)$.

If X_i has parents Π_{X_i}, we use the independence assumption of Bayesian networks. By construction we will ensure that C^- does not contain X_i or any ancestor of X_i, and so

$$P(X_i = v|C^- = a) = \sum_{u \in vals(\Pi_{X_i})} P(X_i = v|\Pi_{X_i} = u) \times P(\Pi_{X_i} = u|C^- = a)$$

For this to be small, each product should be small. We would like to use our expectation failure to lead us to conflicts with a small bound. For each $u \in vals(\Pi_{X_i})$ we make ϵ_u be a bound on the value of $P(\Pi_{X_i} = u|C^- = a)$ and C_u to be the extra elements needed to be added to the counter in order to achieve the bound. We want to construct these values so that $P(X_i = v|C^- = a)$ is small.

In order to bound $P(\Pi_{X_i} = u|C^- = a)$, consider three cases for each $u \in vals(\Pi_{X_i})$:

1. If $obs \models \Pi_{X_i} \neq u$, then let $\langle C_u, \epsilon_u \rangle = \langle \{\}, 0 \rangle$.

2. $\pi \models \Pi_{X_i} = u$. As π considered the most likely assignments of values to variables, and it did not assign v to X_i, we expect $P(X_i = v|\Pi_{X_i} = u)$ to be low. In this case, we will return $P(X_i = v|\Pi_{X_i} = u)$ as this product's contribution[7]. Let $\langle C_u, \epsilon_u \rangle = \langle \{\}, 1 \rangle$.

[7] This heuristic highlights the strength and the weakness of the approach presented in this paper. It means that we only need to consider the values assigned to variables in the current partial descrip-

3. If $\pi \not\models \Pi_{X_i} = u$, then there is some $X_{i_j} \in \Pi_{X_i}$ such that $\pi \not\models X_{i_j} = u_{i_j}$. In this case, we have a choice: we can either (a) let $\langle C_u, \epsilon_u \rangle = \langle \{\}, 1 \rangle$ or (b) choose one such X_{i_j}, and let

$$\langle C_u, \epsilon_u \rangle = extract\_counter(X_{i_j} = u_{i_j}, obs, \pi).$$

(a) will typically produce a smaller counter set, and (b) will typically produce a smaller bound. If $P(X_i = v | \Pi_{X_i} = u) = 0$, then it is clear we should do (a). If the bound returned by $extract\_counter$ is greater or equal to 1, we should also choose (a). In all our experiments these was the only cases where we chose (a).

Note that the first two cases cannot co-occur as π is consistent with the observations. Also if $\Pi_{X_i} = u$ is observed, then $\pi \models \Pi_{X_i} = u$ and we do not include the observed variable in the conflict found.

The value returned is then

$$\begin{aligned} &extract\_counter(X_i = v, obs, \pi) \\ &= \left\langle \{X_i\} \cup \bigcup_{u \in vals(\Pi_{X_i})} C_u, \sum_{u \in vals(\Pi_{X_i})} P(X_i = v | \Pi_{X_i} = u) \times \epsilon_u \right\rangle. \end{aligned}$$

Note that if X_i has no parents, then this is the degenerate form of case 2, and $extract\_counter(X_i = v, obs, \pi)$ returns $\langle \{X_i\}, P(X_i = v) \rangle$.

Figure 10 gives pseudo-code for $extract\_counter$.

Theorem 6.17 If $\pi \not\models X_i = v$ then $extract\_counter(X_i = v, obs, \pi)$ returns a counter to $X_i = v$ with respect to observation obs and partial description π.

For a proof see Appendix B.

This theorem shows that the algorithm will give a counter (and so give us a conflict), no matter which choice is made in the third case. Note that whether the conflicts returned are minimal or not does not affect the correctness of the algorithm; it only affects the efficiency. We could potentially search for the choices that

tion, and do not need to consider other variable assignments. If we tried to find a smaller bound on the contribution of the product it would mean that we need to consider values other than those we have already explored — there are only a linear (in the number of variables) number of assignments of values to variables in the current partial description, but exponentially many alternative assignments of values to variables.

function $extract\_conflict(obs, \langle v_1, \cdots, v_j \rangle)$
  $observed(X_j = v)$;
  return $extract\_counter(X_j, v, obs, \langle v_1, \cdots, v_j \rangle)$.

function $extract\_counter(X_i, v, obs, \pi)$
  $C := \{X_i\}$;
  $p := 0$;
  if $\Pi_{X_i} = \langle\rangle$ then return $\langle \{X_i\}, P(X_i = v) \rangle$ endif;
  for each $u \in vals(\Pi_{X_i})$
    if $consis(\Pi_{X_i} = u, obs)$ and $P(X_i = v | \Pi_{X_i} = u) > 0$
      then if $consis(\Pi_{X_i} = u, \pi)$
        then $p := p + P(X_i = v | \Pi_{X_i} = u)$
        else suppose $u = \left\langle v_{i_1}, \cdots, v_{i_{k_i}} \right\rangle$
          choose i_j such that $\pi \models X_{i_j} \neq v_{i_j}$;
          let $\langle C_0, P_0 \rangle := extract\_counter(X_{i_j}, v_{i_j}, obs, \pi)$;
          if $P_0 < 1$
            then $C := C \cup C_0$;
              $P := P + P(X_i = v | \Pi_{X_i} = u) \times P_0$
            else $P := P + P(X_i = v | \Pi_{X_i} = u)$
            endif
        endif
      endif
    endfor;
  return $\langle C, p \rangle$

Figure 10: Procedures $extract\_conflict$ and $extract\_counter$.

will lead to the conflict with lowest bound. Our experiments were with a greedy algorithm that chooses the first one found. There is a tradeoff between the computational effort in finding minimal conflicts, and the extra pruning that minimal conflicts allow.

Note that sometimes $extract\_counter$ may fail to find a useful counter if, for example, π contains some small probability values. This will manifest itself in returning a bound that is larger than one. The counters may also not be very useful if they are not independent of other counters found.

6.7 Empirical Case Study

The algorithms described above are independent of the search strategies used, although the conflict algorithms only make sense if we pursue the most likely alternatives at each step. This does not, however mean that we have to choose the element of the queue with the lowest bounding function at each stage. One promising idea is to use a depth-first search, always choosing the most likely child of the current partial description (i.e., hill-climbing with backtracking), adding any partial description whose bounding function is below a threshold to a pseudo queue (where we do not store the element in the queue, but keep track of the sum of the bounding functions of the elements that we throw away). We can decrease the threshold to get more accurate results. This is reminiscent of iterative-deepening search [24], but as we are not concerned with finding the most likely possible world, but a set of most likely worlds, we do not have to worry about decreasing the threshold to the maximum value it could obtain. Any threshold will give correct results — a smaller threshold will give more accurate results (and take longer). All of the results presented here are for using one threshold.

The experiments we carried out were limited to understanding the behaviour of the algorithm on cascaded n-bit adder example, with all inputs $zero$ and all output bits being $zero$, except for the output of bit k (i.e., the value of $x2_k$) which had value one. We only used the stuck-at faults and no ab faults (see Section 5), in order to make the results more comprehensible — with the use of ab faults, the results are similar to that presented here. We ran the program using a bounded depth-first search (pruning the depth-first search when the f-value gets below a threshold), generating the 5 most likely possible worlds[8]. Note that an n-bit adder

[8]These correspond to $x2ok_k = stuck1$, $x1ok_k = stuck1$, $o1ok_{k-1} = stuck1$, $a2ok_{k-1} = stuck1$ and $a1ok_{k-1} = stuck1$.

error bit	2	25	50	75	100
run time (no conflicts)	10.8	43.2	145.2	315.3	558.8
run time (with conflicts)	15.5	12.8	9.7	6.6	3.6
error (no conflicts)	0.00249	0.00937	0.0303	0.0618	0.0997
error (with conflicts)	0.00249	0.00261	0.00273	0.00285	0.00298

Figure 11: Running time and posterior error as a function of error bit in a 100-bit adder, with threshold of 0.000001 that produces 5 most likely worlds.

has $5n$ gates and corresponds to a Bayesian network with $13n$ nodes.

All times are based on a SICStus Prolog program running on a NeXTstation (a 68040-based machine). All times are in seconds. The code is available from the author.

As discussed in Section 5.2, the main problem with the search algorithm without conflicts, for our example, was how the runtime depended on the bit k that was faulty. Figure 11 shows how run time depends on the bit chosen for the program with no conflicts and for the program with conflicts. This was for the 100-bit adder (Bayesian network with 1300 nodes). The difference in times for error bit 2 indicates the overhead in using conflicts (as conflicts for this case give us nothing). This table also gives the error in posterior probability estimation. This shows the power of the use of the bounding function to give a smaller probability bound.

Consider how the program that uses conflicts runs: we pursue one world until bit k, then pursue 5 worlds separately from bits k to n. Thus we may estimate the time as proportional to $k + 5(n - k)$. This fits the experimental results extremely well.

The second experiment was with the asymptotic behaviour as the size of the network was increased. Figure 12 shows the run-time for finding the 5 most likely possible worlds, as a function of circuit size. In each of these the error bit was the middle bit of the circuit (i.e., $k = \frac{n}{2}$). This was chosen as it is the average time over all of the error bits (see Figure 11). Note the linear time that was predicted by the $k + 5(n - k)$ formula. Also the posterior error is approximately linear in the size of the circuit.

Finally, the results from double errors, are very similar. For a 100-bit adder, with ones observed at bits 30 and 70, the program took 34 seconds to find the 25 most likely possible worlds.

# bits	100	500	1000	2000	3000
# gates	500	2500	5000	10000	15000
# nodes	1300	6500	13000	26000	39000
run time	9.7	46.2	92.1	182.8	271.3
error	0.00273	0.0135	0.0267	0.0519	0.0758

Figure 12: Running time and posterior error as a function of size of multiple-bit adder for the algorithm with conflicts with threshold of 0.000001.

6.8 Distributed Conflicts

The above analysis works well when the conflicts are clustered together. This was also a property of the example of the preceding section. We prune the bounding function for all variables that come before (in the total ordering of variables) all of the variables in the conflict.

We can also prune the bounding function for variables that come between (in the total ordering of nodes) variables in a conflict. The idea is to consider the remaining probability mass that the conflict promises, and use this as the bound.

Definition 6.18 If $\pi = \langle v_1, \cdots, v_j \rangle$ is a partial description and $\langle C, \epsilon \rangle$ is a conflict then the **contribution** of C to π is

$$\left\langle C \cap \{X_{j+1}, \cdots, X_n\}, \frac{\epsilon}{\prod_{i \leq j \wedge X_i \in C} P(X_i = v_i | X_{i_1} = v_{i_1} \wedge \cdots \wedge X_{i_{k_i}} = v_{i_{k_i}})} \right\rangle$$

Note that sometimes the contribution of a conflict to a partial description may contain a bound that is greater than one. In such cases, the contribution is of no use. The theorem below explicitly allows us to ignore these contributions that do not help.

We build a bounding function from contributions of conflicts to partial descriptions. The theorem below is analogous to Theorem 6.12.

Theorem 6.19 Any function $f^{obs}(\pi)$ constructed in the following way is a bounding function for obs. For each π, select a sequence $\langle C_1, \epsilon_1 \rangle, \cdots, \langle C_k, \epsilon_k \rangle$ of contributions of conflicts to partial description π, such that each $\epsilon_i < 1$ and $C_i \cap C_j = \{\}$ for $i \neq j$. Let $f^{obs}(\pi) = P(\pi) \times \prod_{i=1}^{k} \epsilon_i$.

For a proof see Appendix B.

For example, suppose we have exactly one member X_i of a conflict that is before node X_j in the total ordering of variables and we are considering partial description π (with j elements). If X_i was assigned a high probability in π, then the rest of the probability mass of the conflict that is inconsistent with the observations must be borne by variables after X_i, and thus after X_j. If X_i was assigned a low probability in π, then proportionately less (if any at all) of the probability of the conflict can be taken into account for determining the bounding function for π.

6.9 Using overlapping conflicts

For the example of Section 6.7 we just found one conflict for each expectation failure and used it. In this example, there are multiple conflicts due to that fact that there are two ways the or-gate $o1$ could have output a one.

Example 6.20 One other conflict for Example 6.10 is:
$\langle \langle$ $out\text{-}x2_{50}$, $out\text{-}x2ok_{50}$, $out\text{-}x1_{50}$, $out\text{-}x1ok_{50}$, $i1_{50}$, $i2_{50}$, $i3_{50}$, $o1_{49}$, $o1ok_{49}$, $out\text{-}a1_{49}$, $a1ok_{49}$, $i2_{49}$, $out\text{-}a2_{49}$, $a2ok_{49}$, $i3_{49}$, $o1_{48}$, $o1ok_{48}$, $out\text{-}a1_{48}$, $a1ok_{48}$, $i2_{48}$, $out\text{-}a2_{48}$, $a2ok_{48}$, $i3_{48}$, $o1_{47}$, $o1ok_{47}$, $out\text{-}a1_{47}$, $a1ok_{47}$, $i2_{47}$, $out\text{-}a2_{47}$, $a2ok_{47}$, $i3_{47}$, $\cdots \rangle$, 0.000745 $\rangle$.

The intersection of the set of variables of the two given conflicts is $\{out\text{-}x2_{50}$, $x2ok_{50}$, $i3_{50}$, $out\text{-}o1_{49}$, $o1ok_{49}$, $out\text{-}a1_{49}$, $a1ok_{49}$, $out\text{-}a2_{49}$, $a2ok_{49}$, $out\text{-}i2_{49}$, $out\text{-}x1_{50}$, $x1ok_{50}$, $i1_{50}$, $i2_{50}\}$.

The variables that can have other values (with non-zero probability) are the ok variables. What is interesting is that the five ok variables in the intersection correspond exactly to the variables that have different values in the five most likely possible worlds. This is not a coincidence.

The intersection does not form a conflict. However, for every variable in the intersection to have a normal value, there must be a double error; there must be an abnormal probability assignment in each of the conflicts.

It may seem as though we need a new concept to characterise such sets of variables where the only worlds in which all of the variables do not have normal values have extremely low probability (corresponding to double errors). There is however, no need to do this; the current algorithm can use both conflicts so that it only considers other values for variables outside of the intersection when it is considering extremely unlikely worlds.

When we consider the bounding function for variables not in either conflict, we use the first conflict as it has the smallest bound. When we consider variables not in the intersection of the conflicts, we can use the whichever conflict the variable is not in for the bounding function. It is only when considering variables in the intersection that the bounding function is discounted by the contribution of the conflicts. Thus the general principle of using the conflicts that provide the smallest bounding functions handles the intersection of conflicts appropriately, without needing any extra machinery.

7 Comparison with other systems

The branch and bound search is very similar to the candidate enumeration of de Kleer's focusing mechanism [7]. We have considered a purely probabilistic version of de Kleer's conflicts. We have extended the language to Bayesian networks (see Appendix A). We also can bound the errors in our probabilistic estimates, which de Kleer cannot do. One of the features of our work is that finding minimal conflicts is not essential to the correctness of the program, but only to efficiency. Thus we can explore the idea of saving time by finding useful, but non-minimal conflicts quickly.

The use of search to bound the probabilities in a Bayesian network is closely related to bounded conditioning [19], where a the values for the cutset variables in a Bayesian network are enumerated, and the polytree algorithm [28] is used for the resulting singly connected networks. Instead of enumerating the variables of the cutsets, we enumerate all of the variables. This makes the algorithm much simpler, and allows for fast processing. Bounded conditioning has no analogue to the conflicts of this paper.

Shimony and Charniak [39], Poole [32] and D'Ambrosio [4] have proposed back-chaining search algorithms for Bayesian networks. None of these are nearly as efficient as the one presented here. Even if we consider finding the single most normal world, the algorithm here corresponds to forward chaining on definite clauses (see [33]), which can be done in linear time, but backward chaining has to search and takes potentially exponential time.

This paper deliberately takes the extreme position of seeing how far we can get when we exploit the distributions and not the structure of the network. Hopefully this can shed light on the algorithms that use both structure and distribution to gain efficiency (e.g., [4]).

8 Conclusion

This paper presented a simple search strategy for estimating posterior probabilities in Bayesian networks which can give a tight bound on the error. We then showed how a notion of conflict borrowed from model-based diagnosis can be used to improve efficiency and accuracy.

For most purposes (when we do not want to have very accurate probabilities), the algorithm has the following gestalt feel. We forward simulate the Bayesian network (by choosing the most likely values for each variable) until we find a predicted value that is inconsistent with the observations. When we find such an expectation failure, we extract a conflict from this failure. We only consider non-normal values for the variables in the conflict, and keep doing a forward simulation. We repeat this for each expectation failure, until we can find a consistent Bayesian network assignment (i.e., the forward simulation has assigned a value to all variables) for each of the non-normal assignments of the conflicts (removing each world that gets too unlikely). We use the worlds produced to predict conditional probabilities with a given error. When the distributions are skewed this produces small error bounds.

The main complexity is in the forward simulation which can be done in time linear in the number of variables (assuming a bounded number of parents for each variable), and finding conflicts which can be done in time linear to the size of the conflict found (finding a minimal conflict is more expensive). This needs to be done the number of times equal to the product of the size of the conflicts found. In the worst case this reduces to the algorithm without conflicts, (but with the extra (linear) cost of finding the conflicts) which has good expected complexity when there are sufficiently skewed distributions [31].

One of the aims of this work is to unify model-based diagnosis (e.g., [7]) and probabilistic modelling (e.g., [30]). Although they may look very different, a coherent synthesis is possible, which I hope I have showed in this paper.

A A Comparison of Representations

In this appendix we describe the relationship of the above definition of Bayesian networks to the formalizations of model based diagnosis of Reiter [38] and de Kleer, Mackworth and Reiter [8], and to de Kleer's incorporation of probabilities into model-based diagnosis [7].

In the frameworks of [38] and [8], a system is described in terms of a triple $\langle SD, COMPS, OBS \rangle$, where SD is the system description, $COMPS$ is the set of components and OBS is a set of observations.

In the probabilistic framework the given probabilities serve the same purpose as SD. Those worlds which are inconsistent with SD will have probability 0. Instead of writing the formula

$$\begin{aligned} out(G) = on \quad \leftarrow \quad & type(G) = and\_gate \wedge in1(G) = on \\ & \wedge in2(G) = on \wedge ok(G) \end{aligned}$$

we write[9]

$$\begin{aligned} P(out(G) = on \quad | \quad & type(G) = and\_gate \wedge in1(G) = on \\ & \wedge in2(G) = on \wedge st(G) = ok) = 1. \end{aligned}$$

The probability also places a measure over the remaining worlds. The probabilistic framework is more general in that it allows for 'noise'; for example, it allows us to state that some output is rarely true, as well as being able to state that it is never true. This is even more important for domains where there is no certain knowledge, such as medical diagnosis.

In the probabilistic framework, there is no correspondence to $COMPS$ of [38] and [8]. The observations of both frameworks, however, are identical.

A possible world here can be compared with the formula $\mathcal{D}(Cp, Cn)$, a 'state' of the system [8], which is

$$\left[\bigwedge_{c \in Cp} AB(C) \right] \wedge \left[\bigwedge_{c \in Cn} \neg AB(C) \right]$$

The main difference is that $\mathcal{D}(Cp, Cn)$ does not specify the value of all variables. A state remains agnostic about internal values (those which do not follow from the status of components). A possible world specifies not only the status of components, but of all values. For example, the value of the output of an abnormal gate may not be specified by the state of the system. However, there will be different possible worlds for each of the values of the output. We are interested in having these different possible worlds because they have different properties and

[9] Here we use the random variable (term) $st(G)$ to be the status of G. This has values, for example, ok, $stuck1$, $stuck0$, ab. Thus $ok(G)$ will be the same as $st(G) = ok$.

predictions. Even without ruling out any of the states of the components, different possible worlds may be ruled out (by having a prior probability of zero, or being inconsistent with the observations) by considering values of other variables and observations. Because he did not treat these as different worlds, de Kleer [7] had to resort to the dynamic use of Bayes rule and maximum entropy. We do not need to do this. Having a slightly smaller grain size of possible worlds means we can treat all values symmetrically. It also means that we can get good estimates of the errors in our probability estimates. Rather than just being able to return the most likely diagnoses, we can use the most likely possible worlds to estimate arbitrary conditional probabilities within some bound.

de Kleer et. al. [8] do not specify what a component is (what is a component is input to their formalism). One way we can make their framework closer to the probabilistic one is to invent new components within their framework. These components will be oracles that determine the values that are unspecified in the state. These can be compared to the use of 'stuck at zero' ($stuck0$) and 'stuck at one' ($stuck1$) failure states. This invention of 'causal hypotheses' can be done in general to produce exactly the worlds in Bayesian networks [33].

Note that because a possible world specifies the values of all variables, there is no difference between a possible world which is consistent with a formula f and one which entails f. That is, for possible world ω, $\omega \models f$ iff $\omega \not\models \neg f$ (this can be easily proved by induction on the size of formula f). In either case we say that f is true in the possible world.

A diagnosis of [8] corresponds to a possible world, with non-zero probability, in which obs is true.

de Kleer et. al. [8] consider how to characterise the set of diagnoses of the system. In the probabilistic framework, we consider what we want to do with the diagnoses. We want the diagnoses in order to make decisions. These are, for example, decisions to replace components, to seek more information, to apply treatments, to give tests, etc. Decision theory (see, e.g., [37]) gives a normative theory of what decisions to make. There is a well developed theory about what tests provide the best information and the expected value of making a test or of carrying out a particular action. In order to make good decisions we need the probability of various formulae given the observations. Approximating these probabilities within some error is the task considered in this paper.

B 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. These are $\alpha = \beta = 0$, $\alpha = 1 \wedge \beta = 0$ and $\alpha = 0 \wedge \beta = 1$, which correspond to the three values in Lemma 4.2, and the theorem follows directly from Lemma 4.2. □

Theorem 6.1

$$P(obs) \quad = \quad P_W^{obs} + \sum_{\pi \in Q} P(\pi \wedge obs)$$

Proof:

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

$$
\begin{aligned}
&= P_W^{obs} + \sum_{\pi \in Q} \left(\sum_{w \in \Omega - W : w \models obs \wedge \pi} P(w) \right) \\
&= P_W^{obs} + \sum_{\pi \in Q} P(\pi \wedge obs)
\end{aligned}
$$

□

Theorem 6.12 Given observation obs, and a set of conflicts, the function f^{obs} defined by

$$f^{obs}(\pi) = P(\pi) \times \min\{\epsilon : \langle C, \epsilon \rangle \text{ is a conflict such that } C \cap \pi = \{\}\}$$

is a bounding function of obs.

Proof: The only thing non-trivial to prove is that $P(\pi \wedge obs) \leq P(\pi) \times \epsilon$. Let $C^+ = \{X_1, \cdots, X_n\} - C - C^-$. Let $C^{-\prime} = C^- \cap \{X_{j+1}, \cdots, X_n\}$. There is a isomorphism between the set of possible worlds consistent with π and $\{\left\langle C^+, C, C^{-\prime} \right\rangle = \langle v^+, v, v^- \rangle \wedge \pi : \langle v^+, v, v^- \rangle \in vals(\left\langle C^+, C, C^{-\prime} \right\rangle)\}$.

The idea of the proof is that the variables can be partitioned into the sets C^+, C, $C^{-\prime}$ and $\{X_1, \cdots, X_j\}$. We first sum out the variables in C^+, then sum out the variables in $C^{-\prime}$.

$$
\begin{aligned}
&P(\pi \wedge obs) \\
&= \sum_{\omega \in \Omega : \omega \models \pi \wedge obs} P(\omega) \\
&= \sum_{\substack{\langle v^+, v, v^- \rangle \in vals(\left\langle C^+, C, C^{-\prime} \right\rangle) \\ consis(\left\langle C^+, C, C^{-\prime} \right\rangle = \left\langle v^+, v, v^- \right\rangle, obs)}} P(\left\langle C^+, C, C^{-\prime} \right\rangle = \left\langle v^+, v, v^- \right\rangle \wedge \pi) \\
&= \sum_{\substack{\langle v^+, v, v^- \rangle \in vals(\left\langle C^+, C, C^{-\prime} \right\rangle) \\ consis(\left\langle C^+, C, C^{-\prime} \right\rangle = \left\langle v^+, v, v^- \right\rangle, obs)}} \left(\begin{array}{l} P(C^+ = v^+ | \left\langle C, C^{-\prime} \right\rangle = \langle v, v^- \rangle \wedge \pi) \\ \times P(C = v | C^{-\prime} = v^- \wedge \pi) \\ \times P(C^{-\prime} = v^- | \pi) \times P(\pi) \end{array} \right) \\
&\leq P(\pi) \times \sum_{\substack{\langle v^+, v, v^- \rangle \in vals(\left\langle C^+, C, C^{-\prime} \right\rangle) \\ consis(\left\langle C, C^{-\prime} \right\rangle = \left\langle v, v^- \right\rangle, obs)}} \left(\begin{array}{l} P(C^+ = v^+ | \left\langle C, C^{-\prime} \right\rangle = \langle v, v^- \rangle \wedge \pi) \\ \times P(C = v | C^{-\prime} = v^- \wedge \pi) \\ \times P(C^{-\prime} = v^- | \pi) \end{array} \right)
\end{aligned}
$$

$$
\begin{aligned}
&= P(\pi) \times \sum_{\substack{\langle v,v^- \rangle \in vals(\langle C,C^{-\prime} \rangle) \\ consis(\langle C,C^{-\prime} \rangle = \langle v,v^- \rangle, obs)}} P(C = v | C^{-\prime} = v^- \wedge \pi) \times P(C^{-\prime} = v^- | \pi) \\
&\leq P(\pi) \times \sum_{\substack{v \in vals(C) \\ consis(\langle C,C^- \rangle = \langle v,a \rangle, obs)}} P(C = v | C^- = a) \\
&\leq P(\pi) \times \epsilon
\end{aligned}
$$

□

Theorem 6.16 If $\langle C, \epsilon \rangle$ is a counter to $obs^{\pi 0}$ with respect to observation obs and minimally inconsistent partial description π then $\langle C, \epsilon \rangle$ is a conflict with respect to obs.

Proof: Suppose $\langle C, \epsilon \rangle$ is a counter to $obs^{\pi 0}$ with respect to observation obs and partial description π.

$$
\begin{aligned}
\epsilon &\geq P(obs^{\pi 0} | C^- = a) \\
&= \sum_{\substack{v \in vals(C) \\ consis(C = v, obs^{\pi 0})}} P(C = v | C^- = a) \\
&\geq \sum_{\substack{v \in vals(C) \\ consis(C = v \wedge C^- = a, obs)}} P(C = v | C^- = a)
\end{aligned}
$$

□

Theorem 6.17 If $\pi \not\models X_i = v$ then $extract\_counter(X_i = v, obs, \pi)$ returns a counter to $X_i = v$ with respect to observation obs and partial description π.

Proof: First, the algorithm stops, as there are finitely many values of the parents, and so the for-loop is evaluated finitely many times, and each recursion reduces the number of parents of the node by at least one, and so there is no infinite recursion.

We can inductively assume that the theorem holds for all subcalls to $extract\_counter$.

To show that $\langle C, \epsilon \rangle$ returned by $extract\_counter(X_i = v, obs, \pi)$ satisfies $P(X_i = v | C^- = v) \leq \epsilon$ whenever $consis(C^- = v, obs^{\pi -})$.

We use the following lemma: $consis(C^- = v, obs^{\pi -})$ implies $C^- = v \models obs^{\pi -}$. This is because C does not include any variable in $obs^{\pi -}$ and $C^- = v$ assigns a value to every variable before C in the total ordering of variables. To show this, consider that a variable X_j is only added to C when $extract\_counter(X_j \cdots)$ is called, and this is never called if X_j is assigned a value in $obs^{\pi -}$. If X_j is assigned a value in $obs^{\pi -}$, then each $X_j = v$ is either inconsistent with obs or is entailed by π, and so there is no call to $extract\_counter$ with this variable.

Suppose $consis(C^- = v, obs^{\pi -})$ and X_i has parents Π_{X_i}. For each $u \in vals(\Pi_{X_i})$, we know that $P(\Pi_{X_i} = u | C^- = a) \leq \epsilon_u$ whenever $consis(C^- = v, obs^{\pi -})$ by the inductive assumption and because all probabilities are ≤ 1. Then,

$$\begin{aligned} P(X_i = v | C^- = a) &= \sum_{u \in vals(\Pi_{X_i})} P(X_i = v | \Pi_{X_i} = u) \times P(\Pi_{X_i} = u | C^- = a) \\ &\leq \sum_{u \in vals(\Pi_{X_i})} P(X_i = v | \Pi_{X_i} = u) \times \epsilon_u \\ &= \epsilon \end{aligned}$$

□

Theorem 6.19 Any function $f^{obs}(\pi)$ constructed in the following way is a bounding function for obs. For each π, select a sequence $\langle C_1, \epsilon_1 \rangle, \cdots, \langle C_k, \epsilon_k \rangle$ of contributions of conflicts to partial description π, such that each $\epsilon_i < 1$ and $C_i \cap C_j = \{\}$ for $i \neq j$. Let $f^{obs}(\pi) = P(\pi) \times \prod_{i=1}^{k} \epsilon_i$.

Proof: Let $C = C_1 \times \cdots \times C_k$, and $\epsilon = \prod_{i=1}^{k} \epsilon_i$. Let C_i' be the part of the conflict from which C_i is a contribution, which is covered in $\pi = \langle v_1, \cdots, v_j \rangle$, and ϵ_i' be the corresponding bound (i.e., $\langle C_i \cup C_i', \epsilon_i' \rangle$ forms a conflict, and $C_i' \subseteq \{X_1, \cdots, X_j\}$).

To show $P(\pi \wedge obs) \leq P(\pi) \times \epsilon$. All of the proof of Theorem 6.12 goes through up to the last step.

$$\begin{aligned} &P(\pi \wedge obs) \\ &\leq P(\pi) \times \sum_{\substack{v \in vals(C) \\ consis(\langle C, C^- \rangle = \langle v, a \rangle, obs)}} P(C = v | C^- = a) \end{aligned}$$

$$
\begin{aligned}
&= P(\pi) \times \sum_{\substack{\langle v_1,\cdots,v_k\rangle \in vals(\langle C_1,\cdots,C_k\rangle) \\ consis(\langle C_1,\cdots,C_k,C^-\rangle = \langle v_1,\cdots,v_k,a\rangle, obs)}} \prod_{i=1}^{k} P(C_i = v_i | C^- = a) \\
&= P(\pi) \times \prod_{i=1}^{k} \sum_{\substack{v_i \in vals(C_i) \\ consis(\langle C_i,C^-\rangle = \langle v_i,a\rangle, obs)}} P(C_i = v_i | C^- = a) \\
&= P(\pi) \times \prod_{i=1}^{k} \sum_{\substack{v_i \in vals(C_i) \\ consis(\langle C_i,C^-\rangle = \langle v_i,a\rangle, obs)}} P(C_i = v_i | C^- = a) \times \frac{P(C'_i = v'_i | \Pi_{C'_i} = \Pi_{v'_i})}{P(C'_i = v'_i | \Pi_{C'_i} = \Pi_{v'_i})} \\
&= P(\pi) \times \prod_{i=1}^{k} \frac{\left(\sum_{\substack{v_i \in vals(C_i) \\ consis(\langle C_i,C^-\rangle = \langle v_i,a\rangle, obs)}} P(C_i = v_i | C^- = a) \times P(C'_i = v'_i | \Pi_{C'_i} = \Pi_{v'_i}) \right)}{P(C'_i = v'_i | \Pi_{C'_i} = \Pi_{v'_i})} \\
&\leq P(\pi) \times \prod_{i=1}^{k} \frac{\epsilon'_i}{P(C'_i = v'_i | \Pi_{C'_i} = \Pi_{v'_i})} \\
&= P(\pi) \times \epsilon
\end{aligned}
$$

□

Acknowledgements

Thanks to Andrew Csinger and Michael Horsch for valuable comments on this paper. This research was supported under NSERC grant OGPOO44121, and under Project B5 of the Institute for Robotics and Intelligent Systems. The code referred to in this paper is available by anonymous FTP from `ftp.cs.ubc.ca` in the directory `/ftp/local/poole/code/BN-search`, and from the URL `http://www.cs.ubc.ca/spider/poole`.

References

[1] J. S. Breese and R. Blake. Automating computer bottleneck detection with belief nets. In S. Besnard and S. Hanks, editor, *Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence*, pages 36–45, Montreal, August 1995.

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

[3] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

[4] 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.

[5] B. D'Ambrosio. Incremental probabilistic inference. In *Proc. Ninth Conf. on Uncertainty in Artificial Intelligence*, pages 301–308, Washington, D.C., July 1993.

[6] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, March 1986.

[7] J. de Kleer. Focusing on probable diagnoses. In *Proc. 9th National Conference on Artificial Intelligence*, pages 842–848, Anahiem, Cal., July 1991.

[8] J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56:197–222, 1992.

[9] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, April 1987.

[10] K. J. Ezawa and T. Schuermann. Fraud/uncollectible debt detection using a Bayesian network based learning system: A rare binary outcome with mixed data structures. In P. Besnard and S. Hanks, editor, *Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence*, pages 157–166, Montreal, August 1995.

[11] R. Fung and B. Del Favero. Backward simulation in Bayesian networks. In R. Lopez de Mantaras and D. Poole, editor, *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence*, pages 227–234, Seattle, July 1994.

[12] H. Geffner and J. Pearl. An improved constraint-propagation algorithm for diagnosis. In *Proc. 10th International Joint Conf. on Artificial Intelligence*, pages 1105–1111, Milan, August 1987.

[13] M. R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24(1-3):411–436, December 1984.

[14] W. Hamscher, L. Console, and J. de Kleer, editors. *Readings in model-based diagnosis*. Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[15] D. Heckerman, E. Horvitz, and B. Nathwani. Toward normative expert systems: Part I. The Pathfinder project. *Methods of Information in Medicine*, 31:90–105, 1992.

[16] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In J. F. Lemmer and L. N. Kanal, editor, *Uncertainty in Artificial Intelligence 2*, pages 149–163. Elsevier Science Publishers B.V., 1988.

[17] 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.

[18] 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.

[19] E. J. Horvitz, H. J. Suermondt, and G. F. Cooper. Bounded conditioning: Flexible inference for decisions under scarse resources. In *Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence*, pages 182–193, Windsor, Ontario, August 1989.

[20] 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.

[21] T. Hrycej. Gibbs sampling in Bayesian networks. *Artificial Intelligence*, 46:351–363, 1990.

[22] M. Hulme. Improved sampling for diagnostic reasoning in Bayesian networks. In P. Besnard and S. Hanks, editor, *Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence*, pages 315–322, Montreal, August 1995.

[23] 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.

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

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

[26] B. Middleton, M. Shwe, D. E. Heckerman, M. Henrion, E. J. Horvitz, H. Lehmann, and G. F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base II: evaluation of diagnostic performance. *Methods of Information in Medicine*, 30:256–267, 1991.

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

[28] J. Pearl. Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.

[29] J. Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32(2):245–257, May 1987.

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

[31] D. Poole. Average-case analysis of a search algorithm for estimating prior and posterior probabilities in Bayesian networks with extreme probabilities. In *Proc. 13th International Joint Conf. on Artificial Intelligence*, pages 606–612, Chambery, France, August 1993.

[32] D. Poole. Logic programming, abduction and probability: A top-down anytime algorithm for computing prior and posterior probabilities. *New Generation Computing*, 11(3–4):377–400, 1993.

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

[34] D. Poole. The use of conflicts in searching Bayesian networks. In *Proc. Ninth Conf. on Uncertainty in Artificial Intelligence*, pages 359–367, Washington, DC, July 1993.

[35] M. Pradhan, G. Provan, B. Middleton, and M. Henrion. Knowledge engineering for large belief networks. In R. Lopez de Mantaras and D. Poole, editor, *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence*, pages 484–490, Seattle, July 1994.

[36] D. V. Pynadath and M. P. Wellman. Accounting for context in plan recognition, with application to traffic monitoring. In P. Besnard and S. Hanks, editor, *Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence*, pages 472–481, Montreal, auf 1995.

[37] H. Raiffa. *Decision Analysis.* Addison-Wesley, 1968.

[38] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, April 1987.

[39] 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.

[40] M. Shwe, B. Middleton, D. E. Heckerman, M. Henrion, E. J. Horvitz, H. Lehmann, and G. F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base I: Probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–255, 1991.

[41] S. Srinivas. A probabilistic approach to hierarchical model-based diagnosis. In R. Lopez de Mantaras and D. Poole, editor, *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence*, pages 538–545, Seattle, July 1994.