Exploiting Causal Independence Using Weighted Model Counting

Wei Li and **Pascal Poupart** and **Peter van Beek**
School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
{w22li, ppoupart, vanbeek}@cs.uwaterloo.ca

Abstract

Previous studies have demonstrated that encoding a Bayesian network into a SAT-CNF formula and then performing weighted model counting using a backtracking search algorithm can be an effective method for exact inference in Bayesian networks. In this paper, we present techniques for improving this approach for Bayesian networks with noisy-OR and noisy-MAX relations—two relations which are widely used in practice as they can dramatically reduce the number of probabilities one needs to specify. In particular, we present two space efficient CNF encodings for noisy-OR/MAX and explore alternative search ordering heuristics. We experimentally evaluated our techniques on large-scale real and randomly generated Bayesian networks. On these benchmarks, our techniques gave speedups of up to two orders of magnitude over the best previous approaches and scaled up to networks with larger numbers of random variables.

Introduction

Bayesian networks are a fundamental building block of many AI applications. A Bayesian network consists of a directed acyclic graph where the nodes represent the random variables and each node is labeled with a conditional probability table (CPT) which represents the strengths of the influences of the parent nodes on the child node (Pearl 1988). In general, assuming Boolean random variables, the CPT of a child node with n parents requires one to specify 2^n probabilities. This presents a practical difficulty and has led to the introduction of patterns for CPTs which require one to specify many fewer parameters (see, e.g., (Díez and Druzdzel 2006) and the references therein).

Perhaps the most widely used patterns in practice are the noisy-OR relation and its generalization, the noisy-MAX relation (Good 1961; Pearl 1988). These relations assume a form of causal independence and allow one to specify a CPT with just n parameters, where n is the number of parents of the node. The noisy-OR/MAX relations have been successfully applied in the knowledge engineering of large real-world Bayesian networks, such as the Quick Medical Reference-Decision Theoretic (QMR-DT) project (Miller, Masarie, and Myers 1986) and the Computer-based Patient Case Simulation system (Parker and Miller 1987).

We consider here the problem of exact inference in Bayesian networks which contain noisy-OR/MAX relations. One method for solving such networks is to replace each noisy-OR/MAX by its full CPT representation and then use any of the well-known algorithms for answering probabilistic queries such as variable elimination or tree clustering/jointree. However, in general—and in particular, for the networks that we use in our experimental evaluation—this method is impractical. A more fruitful approach for solving such networks is to take advantage of the semantics of the noisy-OR/MAX relations to improve both time and space efficiency (e.g., (Heckerman 1989; Olesen et al. 1989; D'Ambrosio 1994; Heckerman and Breese 1996; Zhang and Poole 1996; Takikawa and D'Ambrosio 1999; Díez and Galán 2003; Chavira, Allen, and Darwiche 2005)).

Previous studies have demonstrated that encoding a Bayesian network into a SAT-CNF formula and then performing weighted model counting using a DPLL-based algorithm can be an effective method for exact inference, where DPLL is a backtracking algorithm specialized for SAT that includes unit propagation, conflict recording and backjumping (Sang, Beame, and Kautz 2005a). In this paper, we present techniques for improving this weighted model counting approach for Bayesian networks with noisy-OR and noisy-MAX relations. In particular, we present two space efficient CNF encodings for noisy-OR/MAX which exploit its semantics. In our encodings, we pay particular attention to reducing the treewidth of the CNF formula and to directly encoding the effect of unit propagation on evidence into the CNF formula, without actually performing unit propagation. We also explore alternative search ordering heuristics for the DPLL-based backtracking algorithm.

We experimentally evaluated our techniques on large-scale real and randomly generated Bayesian networks. On these benchmarks, our techniques gave speedups of up to two orders of magnitude over the best previous approaches for Bayesian networks with noisy-OR/MAX relations and scaled up to networks with larger numbers of random variables. As well, our techniques extend the model counting approach for exact inference to networks that were previously intractable for the approach.

Background

In this section, we briefly review noisy-OR/MAX relations and the needed background on weighted model counting approaches to exact inference (for more on this latter topic see, e.g., (Chavira and Darwiche 2007)).

Let there be a noisy-OR at a node Y in a Bayesian network and let $X_1, \ldots, X_n$ be the parents of Y, where all random variables are assumed to have Boolean-valued domains. A noisy-OR relation specifies a CPT using n parameters, $q_1, \ldots, q_n$, one for each parent, where,

$$P(Y = 0 \mid X_i = 1, X_j = 0_{[\forall j, j \neq i]}) = q_i. \quad (1)$$

From these parameters, the full CPT representation of size 2^n can be generated using,

$$P(Y = 0 \mid \mathbf{x}) = \prod_{i \in T_x} q_i$$

where $T_x = \{i \mid X_i = 1\}$ and $P(Y = 0 \mid \mathbf{x}) = 0$ if T_x is empty. The noisy-MAX is a generalization of the noisy-OR to non-Boolean domains.

A Bayesian network with full CPT representations can be encoded into a SAT-CNF formula (Darwiche 2002). The encoding proceeds as follows. For each value of each node in the network, an *indicator variable* is created. Next, for each node, *indicator clauses* are generated which ensure that in each model exactly one of the corresponding indicator variables is true. Next, for each CPT and for each non-zero parameter value in the CPT, a *parameter variable* is created. Finally, for each parameter variable, a *parameter clause* is generated. A parameter clause asserts that the conjunction of the corresponding indicator variables implies the parameter variable and vice-versa. The CNF encoding of the Bayesian network can then be compiled into an arithmetic circuit to answer probabilistic queries (Darwiche 2002).

A CNF encoded Bayesian network can also be solved directly using a backtracking search algorithm by introducing weights for the propositional variables (Sang, Beame, and Kautz 2005a). The weight of a parameter variable is its corresponding probability, where $weight(v) + weight(\neg v) = 1$. The weight of an indicator variable is always 1. Let ϕ be a SAT formula and let s be an assignment of a value to every variable in the formula that satisfies the formula; i.e., s is a model of the formula. Let a *literal* be a propositional variable or the negation of a propositional variable. The weight of a formula ϕ is,

$$weight(\phi) = \sum_s \prod_{l \in s} weight(l),$$

where the sum is over all possible models and the product is over the weight of the literals in that model. Finally, let F be the CNF encoding of a Bayesian network. A general query $P(Q \mid E)$ on the network can be answered by,

$$\frac{weight(F \wedge Q \wedge E)}{weight(F \wedge E)}. \quad (2)$$

A backtracking algorithm used to enumerate the models of a CNF formula is often referred to as DPLL or DPLL-based, and usually includes such techniques as unit propagation, conflict recording and backjumping.

Related Work

In this section, we relate our work to previously proposed methods for exact inference in Bayesian networks with noisy-OR/MAX relations.

The two standard exact algorithms for Bayesian networks are variable elimination (VE) and tree clustering/jointree. Clustering algorithms are often preferred as they precompute results and so can answer queries faster. However, there are large real-world networks that clustering cannot deal with due to time and space complexities. In such networks, VE can sometimes still answer queries because it permits pruning of irrelevant variables.

Many methods have been proposed to transform a noisy-OR/MAX into a decomposable auxiliary graph by adding hidden nodes and then solving using adaptations of variable elimination or tree clustering (e.g., (Olesen et al. 1989; D'Ambrosio 1994; Heckerman and Breese 1996; Takikawa and D'Ambrosio 1999; Díez and Galán 2003)). Most recently, Díez & Galán (2003) proposed a multiplicative factorization which improves on previous work. We use their auxiliary graph as the starting point for one of our CNF encodings. In our experiments, we perform a detailed empirical comparison of their approach using variable elimination against our proposals on large Bayesian networks.

Quickscore (Heckerman 1989) was the first efficient exact inference algorithm for Boolean-valued *two-layer* noisy-OR networks. Chavira, Allen and Darwiche (2005) present a method for *multi-layer* noisy-OR networks and show that their approach is significantly faster than Quickscore on randomly generated two-layer networks. Their approach proceeds as follows: (i) transform the noisy-OR network into a Bayesian network with full CPTs using Pearl's transformation (see Figure 2), (ii) translate the network with full CPTs into CNF using a *general* encoding (see Background section), and (iii) compile the CNF into an arithmetic circuit. In our experiments, we show that our *special-purpose* encodings of noisy-OR can be more space and time efficient and scale to much harder problems.

In our work, we build upon the DPLL-based weighted model counting approach of Sang, Beame, and Kautz (2005a). Their general encoding assumes full CPTs and yields a parameter clause for each CPT parameter. However, this approach is impractical for large-scale noisy-OR networks. Our special-purpose encodings extend the weighted model counting approach for exact inference to networks that were previously intractable for the approach.

Encoding Noisy-OR/MAX into CNF

In this section, we first present our SAT-CNF encodings of the noisy-OR relation. We use as our running example the Bayesian network shown in Figure 1. We then generalize our encodings to noisy-MAX.

Weighted CNF Encoding 1

In our first weighted model encoding method (WMC1), we introduce an indicator variable I_Y for Y and an indicator variable I_{X_i} for each parent of Y. We also introduce a parameter variable P_{q_i} for each parameter q_i (see Eqn. 1) in
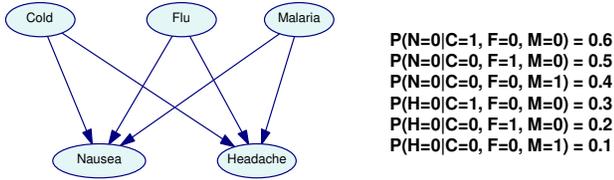
Figure 1: Example of a Bayesian network. We assume the random variables are Boolean and there is a noisy-OR at node *Nausea* and at node *Headache* with the given parameters.

the noisy-OR. The weights of these variables are as follows.

$$
\begin{aligned}
weight(I_{X_i}) = weight(I_Y) &= 1 \\
weight(P_{q_i}) &= q_i \\
weight(\neg P_{q_i}) &= 1 - q_i
\end{aligned}
$$

The noisy-OR relation can then be encoded as the formula,

$$(I_{X_1} \wedge P_{q_1}) \vee (I_{X_2} \wedge P_{q_2}) \vee \cdots \vee (I_{X_n} \wedge P_{q_n}) \Leftrightarrow I_Y. \quad (3)$$

The formula so far can be seen to be an encoding of Pearl's well-known transformation for noisy-OR (see Figure 2). Before converting the formula to CNF, we introduce an auxiliary indicator variable w_i for each conjunction such that $w_i \Leftrightarrow I_{X_i} \wedge P_{q_i}$. This dramatically reduces the number of clauses generated. The formula is then transformed into,

$$
\begin{aligned}
(\neg I_Y \quad \vee \quad & ((w_1 \vee \ldots \vee w_n) \wedge \\
& (\neg P_{q_1} \vee \neg I_{X_1} \vee w_1) \wedge \\
& (P_{q_1} \vee \neg w_1) \wedge (I_{X_1} \vee \neg w_1) \wedge \ldots \wedge \\
& (\neg P_{q_n} \vee \neg I_{X_n} \vee w_n) \wedge \\
& (P_{q_n} \vee \neg w_n) \wedge (I_{X_n} \vee \neg w_n))) \wedge \\
(I_Y \quad \vee \quad & ((\neg P_{q_1} \vee \neg I_{X_1}) \wedge \ldots \wedge \\
& (\neg P_{q_n} \vee \neg I_{X_n}))).
\end{aligned} \quad (4)
$$

The formula is not in CNF, but can be easily transformed into CNF using the distributive law. It can be seen that WMC1 can also easily encode evidence—i.e, if $I_Y = 0$ or $I_Y = 1$, the formula can be further simplified—before the final translation into CNF.

Example 1. *Consider the small network shown in Figure 1. The WMC1 encoding introduces the five Boolean indicator variables I_C, I_F, I_M, I_N, and I_H, each with weight 1; and the six parameter variables $P_{0.6}$, $P_{0.5}$, $P_{0.4}$, $P_{0.3}$, $P_{0.2}$, and $P_{0.1}$, each with weight $P_{q_i} = q_i$ and $\neg P_{q_i} = 1 - q_i$. To illustrate the encoding of evidence, suppose that nausea is present (i.e., $N = 1$) and headache is not present (i.e., $H = 0$). The corresponding constraints for the evidence are as follows.*

$$(P_{0.6} \wedge I_C) \vee (P_{0.5} \wedge I_F) \vee (P_{0.4} \wedge I_M) = 1 \quad (5)$$

$$(P_{0.3} \wedge I_C) \vee (P_{0.2} \wedge I_F) \vee (P_{0.1} \wedge I_M) = 0 \quad (6)$$

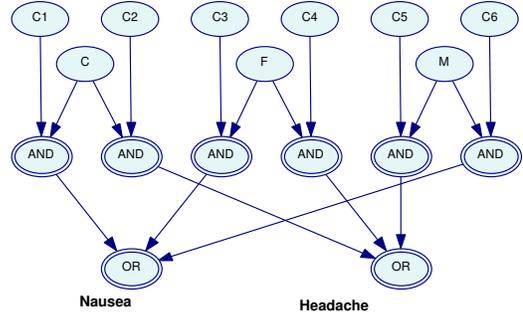*Using Equation 4, the above constraints can be converted*

Figure 2: Pearl's transformation of noisy-OR network. Nodes with double borders are deterministic nodes with the designated logical relationship.

into CNF clauses. Equation 5 gives the clauses,

$$
\begin{aligned}
& (w_1 \vee w_2 \vee w_3) \\
\wedge\ & (\neg P_{0.6} \vee \neg I_C \vee w_1) \wedge (P_{0.6} \vee \neg w_1) \wedge (I_C \vee \neg w_1) \\
\wedge\ & (\neg P_{0.5} \vee \neg I_F \vee w_2) \wedge (P_{0.5} \vee \neg w_2) \wedge (I_F \vee \neg w_2) \\
\wedge\ & (\neg P_{0.4} \vee \neg I_M \vee w_3) \wedge (P_{0.4} \vee \neg w_3) \wedge (I_M \vee \neg w_3)
\end{aligned}
$$

and constraint equation 6 gives the clauses,

$$(\neg P_{0.3} \vee \neg I_C) \wedge (\neg P_{0.2} \vee \neg I_F) \wedge (\neg P_{0.1} \vee \neg I_M).$$

Weighted CNF Encoding 2

Our second weighted model encoding method (WMC2) takes as its starting point Díez & Galán's (2003) directed auxiliary graph transformation of a Bayesian network with noisy-OR[1]. The transformation first creates a graph with the same set of nodes and arcs as the original network. Then, for each node Y with a noisy-OR relation,

- Add a hidden node Y' with the same domain as Y
- Add an arc $Y' \rightarrow Y$
- Redirect each arc $X_i \rightarrow Y$ to $X_i \rightarrow Y'$
- Associate with Y a factorization table,

	$Y' = 0$	$Y' = 1$
$Y = 0$	1	0
$Y = 1$	-1	1

This auxiliary graph is not a Bayesian network because it contains parameters which are less than 0. So the CNF encoding methods for general Bayesian networks (see Background section) cannot be applied here.

We introduce indicator variables I'_Y and I_Y for Y' and Y, and an indicator variable I_{X_i} for each parent of Y'. For each arc $X_i \rightarrow Y'$, we create two parameter variables $P^0_{X_i,Y'}$ and $P^1_{X_i,Y'}$ where the weights of these variables is given by,

$$weight(P^0_{X_i,Y'}) = 1, \quad weight(P^1_{X_i,Y'}) = q_i.$$

---

[1]The Díez & Galán (2003) transformation is a generalization to noisy-MAX of the noisy-OR transformation of Takikawa and D'Ambrosio (1999).
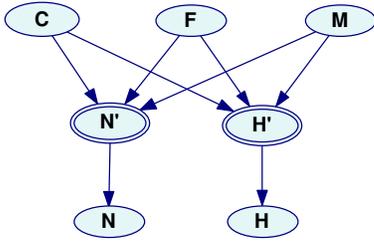
Figure 3: Díez and Galán's transformation of noisy-OR.

For each factorization table, we introduce two variables, u_Y and w_Y, where the weights of these variables are given by,

$$weight(u_Y) = 1, \quad weight(\neg u_Y) = 0,$$
$$weight(w_Y) = -1, \quad weight(\neg w_Y) = 2.$$

For the first row of a factorization table, we generate the clause,

$$(\neg I_{Y'} \vee I_Y),$$

and for the second row, we generate the clause,

$$(\neg I_{Y'} \vee \neg I_Y \vee u_Y) \wedge (I_{Y'} \vee \neg I_Y \vee w_Y).$$

Finally, for every parent X_i of Y', we generate the clauses,

$$(I_{Y'} \vee I_{X_i} \vee P^0_{X_i,Y'}) \wedge (I_{Y'} \vee \neg I_{X_i} \vee P^1_{X_i,Y'}).$$

We now have a conjunction of clauses; i.e., CNF. Once again, it can be seen that WMC2 can also easily encode evidence into the CNF formula; i.e., if $I_Y = 0$ or $I_Y = 1$, the formula can be further simplified.

Example 2. *Consider once again the network shown in Figure 1. The auxiliary graph transformation is shown in Figure 3. If we again know that $N = 0$ and $H = 1$, the WMC2 encoding results in the following formula,*

$$(I_C \vee P^0_{C,N}) \wedge (\neg I_C \vee P^1_{C,N}) \wedge$$
$$(I_F \vee P^0_{F,N}) \wedge (\neg I_F \vee P^1_{F,N}) \wedge$$
$$(I_M \vee P^0_{M,N}) \wedge (\neg I_M \vee P^1_{M,N}) \wedge$$
$$(I_{H'} \vee I_C \vee P^0_{C,H}) \wedge (I_{H'} \vee \neg I_C \vee P^1_{C,H}) \wedge$$
$$(I_{H'} \vee I_F \vee P^0_{F,H}) \wedge (I_{H'} \vee \neg I_F \vee P^1_{F,H}) \wedge$$
$$(I_{H'} \vee I_M \vee P^0_{M,H}) \wedge (I_{H'} \vee \neg I_M \vee P^1_{M,H}) \wedge$$
$$(\neg I_{H'} \vee u_H) \wedge (I_{H'} \vee w_H),$$

where the weights are given by,

$$weight(P^1_{C,N}) = 0.6 \quad weight(P^1_{C,H}) = 0.3$$
$$weight(P^1_{F,N}) = 0.5 \quad weight(P^1_{F,H}) = 0.2$$
$$weight(P^1_{M,N}) = 0.4 \quad weight(P^1_{M,H}) = 0.1$$

and all other weights are 1.

Noisy-MAX

WMC1 and WMC2 can be extended to noisy-MAX by introducing more indicator variables to represent variables with multiple values. Here, we give an overview of the extension.

The noisy-MAX model assumes that there are different causes $X_1, \ldots, X_n$ leading to a certain effect Y that might have several degrees of severity; it also assumes that the degree reached by Y is the maximum of the degrees produced by each cause if they were acting independently. The noisy-MAX model would be valid only if the effects do not accumulate with one another. Given a noisy-MAX variable Y with d_Y possible values labeled from 0 to $d_Y - 1$, and n parents $X_1, \ldots, X_n$ each with d_{X_i} possible values, the noisy-MAX can be described by two basic axioms:

1. When all the causes are absent, the effect is absent,

$$P(Y = 0 \mid X_i = 0_{[\forall i]}) = 1.$$

2. The degree (value) of Y is the maximum of the degrees produced by the X's if they were acting independently,

$$P(Y \leq y \mid \mathbf{x}) =$$
$$\prod_i P(Y \leq y \mid X_i = x_k, X_{[\forall j, j \neq i]} = 0) = q^{x_k}_{i,y}$$

For each noisy-MAX variable Y, we introduce d_Y indicator variables $I_{y0} \ldots I_{y_{d_Y-1}}$ to represent each value and $\binom{d_Y}{2} + 1$ clauses to ensure that exactly one of these variables is true. For example, if Y has three values—None, Mediocre, and Severe—we add I_{y_n}, I_{y_m} and I_{y_s} and four clauses, where

$$weight(I_{y_n}) = weight(I_{y_m}) = weight(I_{y_s}) = 1$$
$$(\neg I_{y_n} \vee \neg I_{y_m}) \wedge (\neg I_{y_n} \vee \neg I_{y_s}) \wedge (\neg I_{y_m} \vee \neg I_{y_s})$$
$$\wedge (I_{y_n} \vee I_{y_m} \vee I_{y_s})$$

For WMC1, for each parent X_i, we define parameter variables,

$$weight(P^{x_k}_{i,y_j}) = q^{x_k}_{i,y},$$

where $0 \leq j \leq d_Y - 1$, $0 \leq k \leq d_{Xi} - 1$. We also rewrite Formula 3 for each noisy-MAX variable as,

$$\bigwedge_{j=0}^{d_Y-1} \neg I_{y_j} \bigvee_{i=1}^{n} \bigvee_{k=0}^{d_{X_i}-1} (I_{i,x_k} \wedge P^{x_k}_{i,y_j})$$

For WMC2, we can introduce d_Y indicator variables for each Y and Y'. Based on the auxiliary graph we described in WMC2 above, we expand the factorization table δ_Y as a $d_Y \times d_Y$ matrix given by,

$$\delta_Y(y, y') = \begin{cases} 1, & \text{add } (\neg I_{y'} \vee \neg I_y \vee u_Y) \text{ if } y' = y \\ -1, & \text{add } (\neg I_{y'} \vee \neg I_y \vee w_Y) \text{ if } y' = y - 1 \\ 0, & \text{add } (\neg I_{y'} \vee \neg I_y \vee \neg u_Y) \text{ otherwise} \end{cases}$$

For example, the factorization table of the three value variable above is

	$Y' = y_n$	$Y' = y_m$	$Y' = y_s$
$Y = y_n$	1	0	0
$Y = y_m$	-1	1	0
$Y = y_s$	0	-1	1

The relation between X_i and Y' is represented by the clauses,

$$(\neg I_{y'_j} \vee \neg I_{x_k} \vee P^{x_k}_{i,y_j}),$$

where $0 \leq j \leq d_Y - 1$, $0 \leq k \leq d_{Xi} - 1$.

Comparison

Both WMC1 and WMC2 can answer probabilistic queries using Equation 2. Both encodings lead to quick factorization given evidence during the encoding. The clauses from negative evidence can be represented compactly in the resulting CNF, even with a large number of parents. In the WMC2 encoding, positive evidence can be represented by three Boolean variables, whereas the WMC1 encoding requires n Boolean variables, one for each parent. In WMC2, we use two parameter variables ($P^0_{X_i,Y'}$ and $P^1_{X_i,Y'}$) to represent every arc, while WMC1 only needs one.

Table 1: 500+500 Binary, two layer, noisy-OR Networks. When there is 60 positive evidence, the table lists the number of problems (/30) solved within one hour.

P+	WMC1			WMC2			ACE
	#var	w	sec	#var	w	sec	sec
30	3,686	10	0.2	6,590	11	0.1	32
35	3,716	11	0.6	6,605	11	0.2	33
40	3,746	13	21	6,620	11	0.5	33
45	3,776	14	39	6,635	13	2	36
50	3,806	19	75	6,650	13	6	41
55	3,836	22	175	6,665	16	71	166
60	3,916	24	(17)	6,680	16	(27)	(21)

We used randomly generated two-layer networks to compare the space efficiency and complexity of WMC1 and WMC2. Each random network contains 500 diseases and 500 symptoms. Each symptom has 6 possible diseases uniformly distributed in the disease set. Table 1 shows the tree width of the encoded CNF from WMC1 and WMC2. The first column shows the number of positive evidence in the symptom variables. The rest are negative symptoms. It can be seen that although WMC1 generates fewer variables than WMC2, the CNF created by WMC2 have smaller width. We compute the probability of evidence (PE) with the tree decomposition guided variable ordering (Huang and Darwiche 2003) and compare our results with ACE2[2] (a more detailed experimental analysis is given in the next section). ACE2 is the latest package which applies the general Bayesian networks encoding method to noisy-OR model and then compiles the CNF into an arithmetic circuit (Chavira, Allen, and Darwiche 2005). For ACE2, we used the best parameter settings that we could find, which were much better than the default parameters on these instances.

Experimental Evaluation

In this section, we evaluate the effectives of our encodings. We use Cachet[3] as it is currently recognized as the fastest weighted model counting solver.

We compare against ACE2 (Chavira, Allen, and Darwiche 2005). We also implemented Díez and Galán's (2003) approach, which consists of variable elimination (VE) applied to an auxiliary network that permits exploitation of causal independence. Our implementation uses Algebraic Decision

Diagrams (ADDs) (Bahar et al. 1993) as the base data structure to represent conditional probability tables. ADDs permit a compact representation by aggregating identical probability values. They also speed up computation by exploiting context-specific independence (Boutilier et al. 1996), taking advantage of determinism and caching intermediate results to avoid duplicate computation. The variable elimination heuristic that we used is a greedy one that first eliminates all variables that appear in deterministic potentials of one variable (equivalent to unit propagation) and then eliminates the variable that will create the smallest ADD with respect to the eliminated ADDs. In order to avoid creating an ADD for each variable when searching for the next variable to eliminate, the size of a new ADD is estimated by the smallest of two upper bounds: (i) the cross product of the domain size of the variables of the new ADD and (ii) the product of the sizes (e.g., number of nodes) of the eliminated ADDs.

Good variable ordering heuristics play an important role in the success of modern DPLL-based model counting solvers. Here, we evaluate two heuristics: *Variable State Aware Decaying Sum (VSADS)* and *Tree Decomposition Variable Group Ordering (DTree)*. VSADS is one of the current best performing dynamic heuristics designed for DPLL-based model counting engines (Sang, Beame, and Kautz 2005b). It can be viewed as a scoring system that attempts to satisfy the most recent conflict clauses and simultaneously makes its branching decisions based on the number of occurrences of a variable. Compared with VSADS, DTree (Huang and Darwiche 2003) can be described as a mixed variable ordering heuristic. DTree first uses a binary tree decomposition to generate ordered variable groups. Then the order of variables within a group is decided dynamically during DPLL by other dynamic heuristics.

All the experiments were performed on a Pentium workstation with a 3GHz hyper-threading CPU and 2GB RAM.

QMR-DT

Compared with randomly generated problems, QMR-DT presents a real-world inference task with various structural and sparsity properties. For example, in the empirical distribution of diseases, a small proportion of the symptoms are connected with a large number of diseases.

The network we used was aQMR-DT, an anonymized version of QMR-DT[4]. We generated symptom vectors for each experiment with k positive symptoms. For each evidence vector, we sort the symptom variables into ascending order by their parent (disease) number, chose the first k variables as positive symptoms, and then set the remaining variables to negative. The goal of the method is to generate instances of increasing difficulty.

We report the runtime to answer the probability of evidence (PE) queries. We also experimented with an implementation of Quickscore[5], but found that it could not solve any of the test cases shown in Figure 4 (the figure only shows the difficult range; the instances with fewer positive symptoms are easier for all the algorithms in the figure). The
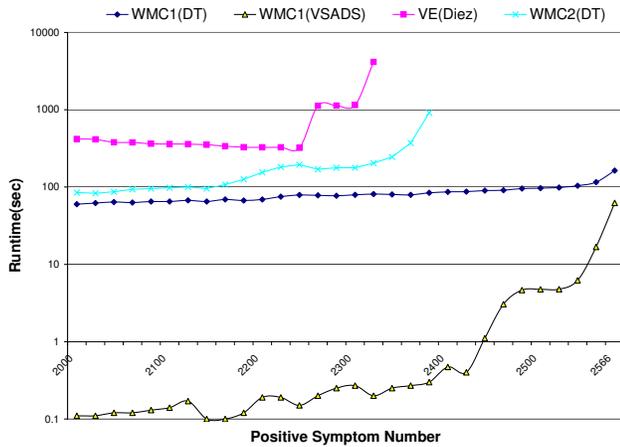
[2]http://reasoning.cs.ucla.edu/ace/

[3]http://www.cs.rochester.edu/u/kautz/Cachet/index.htm

[4]http://citeseer.ist.psu.edu/463049.html

[5]http://www.cs.ubc.ca/∼murphyk/Software/BNT/bnt.html

Figure 4: QMR-DT with 4,075 symptoms and 570 diseases.

Figure 5: Random noisy-OR network with 3,000 binary variables.

WMC-based approach also outperforms VE on QMR-DT. The model counting time of WMC1+VSADS for 2,560 positive symptoms are 25s. But this instance could not be solved within one hour by VE.

We tested different heuristics on each encoding. The runtime using WMC and DTree heuristic is the sum of two parts: the preprocessing time by DTree (Huang and Darwiche 2003) and the runtime of model counting. In this experiment, semi-static tree decomposition-based heuristic has faster run time than VSADS in the model counting process. However, the overhead of preprocessing for large size networks is too high to achieve better overall performance.

The WMC2 encoding generates twice as many variables as WMC1. Although WMC2 is more promising than WMC1 on smaller size networks (Table 1), here WMC2 is less efficient than WMC1. The overhead of the tree decomposition ordering on WMC2 encodings is also higher than on WMC1 encodings. Our results also show that dynamic variable ordering does not work well in WMC2. WMC2+VSADS cannot solve networks with more than 1,500 positive evidences.

Results also show that our approach is more efficient than ACE2. For example, using ACE2, a CNF of QMR-DT with 30 positive symptoms creates 2.8×10^5 variables, 2.8×10^5 clauses and 3.8×10^5 literals. Also, it often requires more than 1GB memory to finish the compilation process. With WMC1, the same network and the same evidence create only 4.6×10^4 variables, 4.6×10^4 clauses and 1.1×10^5 literals. Cachet only needs less than 250M memory in most cases. And in our experiments, ACE 2 cannot solve QMR-DT with more than 500 positive evidence in an hour.

Random Noisy-OR Multi-Layer

To test randomly generated multi-layer networks, we constructed a set of acyclic Bayesian networks using the same method as Díez & Galán (2003): create n binary variables; randomly select m pairs of nodes and add arcs between them, where an arc is added from X_i to X_j if $i < j$; and
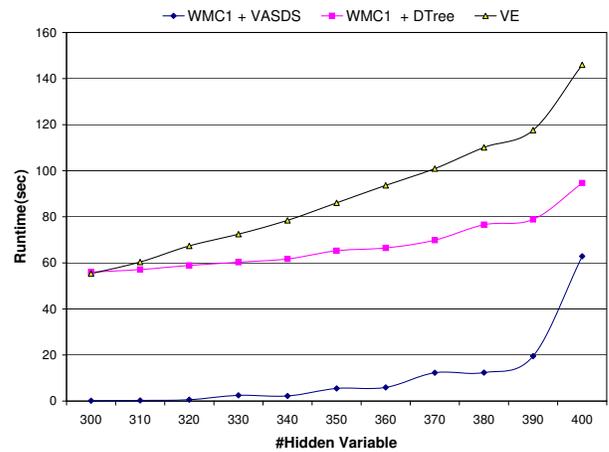
assign a noisy-OR/MAX distribution to each node with parents.

Figure 5 represents the average time to answer the probability of evidence (PE) queries, which is a function of the number of hidden variables. We repeat the experiment for a total of 50 randomly generated networks with 300 hidden variables and then increase the number of hidden variables.

The results from two layer QMR-DT and multi-layer random noisy-OR show that on average, the WMC-based approaches performed significantly better than the VE-based approach and ACE2. All the approaches benefit from the large amount of evidence, but the WMC-based approaches explore the determinism more efficiently with dynamic decomposition and unit propagation (resolution). Compared with VE, the WMC-based approaches encode the local dependencies among parameters and the evidences into clauses/constraints. The topological features of CNF, such as connectivity, can then be explored dynamically during DPLL's simplification process.

Conflict analysis based heuristics have been successfully applied in modern SAT solvers. However, conflicts rarely occur in model counting problems with large numbers of solutions as is the case in encodings of Bayesian networks. In situations where there are few conflicts, VSADS essentially makes random decisions. But here, for large Bayesian networks with large numbers of evidences, VSADS work very well because the constraints we generated from the evidence limits the number of solutions. DTree is also a good choice due to its divide-and-conquer nature. However, when we use DTree to decompose the CNF generated from QMR-DT, usually the first variable group contains more than 500 disease variables. And, the overhead of preprocessing affects the overall efficiency of this approach.

Similarly, we performed an experiment with 100 five-valued variables. We generated 50 random networks for each number of arcs. The results are displayed in Figure 6.
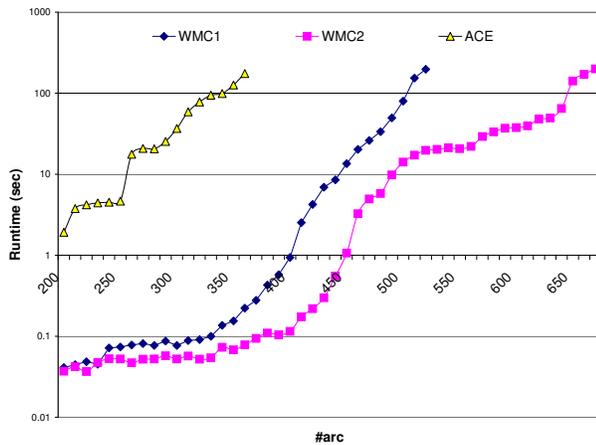
Figure 6: Random noisy-MAX network with 100 five-valued variables.

Conclusion and Future Work

Large graphical models, such as QMR-DT, are often intractable for exact inference when there is a large amount of positive evidence. We discussed multiple techniques which can exploit determinism and causal independence and make exact inference perform more efficiently on these problems. In particular, we presented two space efficient CNF encodings for noisy-OR/MAX relations. We also explored alternative search ordering heuristics for the DPLL-based backtracking algorithm on these encodings. In our experiments, we showed that together our techniques extend the model counting approach for exact inference to networks that were previously intractable for the approach. As well, our techniques gave speedups of up to two orders of magnitude over the best previous approaches for Bayesian networks with noisy-OR/MAX relations.

For future work, recent work by (Li, van Beek, and Poupart 2006) demonstrated that a DPLL procedure combined with dynamic decomposition and caching allows one to perform Bayesian inference incrementally. We intend to further apply this method to the encodings proposed in this paper. As well, future work could include developing specific SAT-encodings of other causal independence relations such as noisy-adder.

References

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *ICCAD '93*, 188–191. IEEE Computer Society Press.

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *UAI-96*, 115–123.

Chavira, M.; Allen, D.; and Darwiche, A. 2005. Exploiting evidence in probabilistic inference. In *UAI-05*, 112–127.

Chavira, M., and Darwiche, A. 2007. On probabilistic inference by weighted model counting. *Artificial Intelligence*. doi:10.1016/j.artint.2007.11.002.

D'Ambrosio, B. 1994. Symbolic probabilistic inference in large BN2O networks. In *UAI-94*, 128–135.

Darwiche, A. 2002. A logical approach to factoring belief networks. In *KR-02*, 409–420.

Díez, F. J., and Druzdzel, M. J. 2006. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01, UNED, Madrid.

Díez, F. J., and Galán, S. F. 2003. Efficient computation for the noisy MAX. *Int'l J. of Approx. Reasoning* 18:165–177.

Good, I. J. 1961. A causal calculus. *The British Journal for the Philosophy of Science* 12(45):43–51.

Heckerman, D., and Breese, J. 1996. Causal independence for probability assessment and inference using Bayesian networks. *IEEE, Systems, Man, and Cyber.* 26:826–831.

Heckerman, D. 1989. A tractable inference algorithm for diagnosing multiple diseases. In *UAI-89*, 163–172.

Huang, J., and Darwiche, A. 2003. A structure-based variable ordering heuristic for SAT. In *IJCAI-03*, 1167–1172.

Li, W.; van Beek, P.; and Poupart, P. 2006. Performing incremental Bayesian inference by dynamic model counting. In *Proceedings of the 21th National Conference on Artificial Intelligence, AAAI-06*, 1173–1179.

Miller, R. A.; Masarie, F. E.; and Myers, J. D. 1986. Quick medical reference for diagnostic assistance. *Medical Computing* 3:34–48.

Olesen, K. G.; Kjaerulff, U.; Jensen, F.; Jensen, F. V.; Falck, B.; Andreassen, S.; and Andersen, S. K. 1989. A MUNIN network for the median nerve: A case study on loops. *Appl. Artif. Intell.* 3(2-3):385–403.

Parker, R., and Miller, R. 1987. Using causal knowledge to create simulated patient cases: The CPCS project as an extension of INTERNIST-1. In *the 11th Symposium Computer Applications in Medical Care*, 473–480.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Sang, T.; Beame, P.; and Kautz, H. 2005a. Solving Bayesian networks by weighted model counting. In *AAAI-05*, 1–10.

Sang, T.; Beame, P.; and Kautz, H. A. 2005b. Heuristics for fast exact model counting. In *SAT-05*, 226–240.

Takikawa, M., and D'Ambrosio, B. 1999. Multiplicative factorization of noisy-max. In *UAI-99*, 622–630.

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