An Anytime Scheme for Bounding Posterior Beliefs

Bozhena Bidyuk and Rina Dechter
Donald Bren School of Information and Computer Science
University Of California Irvine
bbidyuk@ics.uci.edu
dechter@ics.uci.edu

Abstract

This paper presents an any-time scheme for computing lower and upper bounds on posterior marginals in Bayesian networks. The scheme draws from two previously proposed methods, bounded conditioning (Horvitz, Suermondt, & Cooper 1989) and bound propagation (Leisink & Kappen 2003). Following the principles of cutset conditioning (Pearl 1988), our method enumerates a subset of cutset tuples and applies exact reasoning in the network instances conditioned on those tuples. The probability mass of the remaining tuples is bounded using a variant of bound propagation. We show that our new scheme improves on the earlier schemes.

Introduction

Computing bounds on posterior marginals is a special case of approximating posterior marginals with a desired degree of precision which is NP-hard (Dagum & Luby 1993). We address this hard problem by proposing an any-time bounding framework based on two previously proposed bounding schemes, bounded conditioning and bound propagation.

Bounded conditioning (BC) (Horvitz, Suermondt, & Cooper 1989) is founded on the principle of cutset-conditioning method (Pearl 1988). Given a Bayesian network over X, evidence $E \subset X$, E=e, and a subset of variables $C \subset X \backslash E$, we can obtain exact posterior marginals by enumerating over all cutset tuples $c^i \in D(C)$, M=$|\mathcal{D}(C)|$, using the formula:

$$P(x|e) = \frac{\sum_{i=1}^{M} P(x, c^i, e)}{\sum_{i=1}^{M} P(c^i, e)} \quad (1)$$

For any assignment c=c^i, the computation of quantities $P(x, c^i, e)$ and $P(c^i, e)$ is linear in the network size if C is a loop-cutset and exponential in w if C is a w-cutset. The limitation of the cutset-conditioning method is that the number of cutset tuples M grows exponentially with the cutset size.

Horvitz, Suermondt, and Cooper (1989) observed that often a small number of tuples $h << M$ contains most of the probability mass of $P(e) = \sum_{i=1}^{M} P(c^i, e)$. Subsequently, they proposed to compute the probabilities $P(x, c^i, e)$ and $P(c^i, e)$ exactly only for the h tuples, $1 \leq i \leq h$, with high prior probabilities $P(c^i)$, while bounding the rest by their priors. Bounded conditioning (BC) was the first method to offer any-time properties and to guarantee convergence to the exact marginals with time as $h \rightarrow M$. The scheme was validated on the example of Alarm network with 37 nodes and M=108 loop-cutset tuples. Without evidence, algorithm computed small bounds intervals, ≈0.01 or less, after generating 40 cutset instances. However, with 3 and 4 nodes assigned, the bounds interval length rose to ≈ 0.15 after processing the same 40 tuples. The latter shows how BC bounds interval increases as probability of evidence $P(e)$ decreases. In fact, we can show that BC upper bound can become >1 when $P(e)$ is small (Bidyuk & Dechter 2005).

A related bounding scheme was proposed in (Poole 1996). Instead of enumerating variables of the cutset, it enumerates all variables and uses conflict-counting to update the function bounding the remaining probability mass.

Bound propagation (BdP) scheme (Leisink & Kappen 2003) obtains bounds by iteratively solving a linear optimization problem for each variable such that the minimum and maximum of the objective function correspond to lower and upper bounds on the posterior marginals. They demonstrated the performance of the scheme on the example of Alarm network, Ising grid, and regular bi-partite graphs.

In our work here, we propose a framework, which we term Any Time Bounds (ATB), that also builds upon the principles of conditioning exploring fully h cutset tuples and bounding the rest of the probability mass, spread over the unexplored tuples. The scheme improves over bounded conditioning in several ways. First, it bounds more accurately the mass of the unexplored tuples in polynomial time. Second, it uses cutset sampling (Bidyuk & Dechter 2003a; 2003b) for finding high-probability cutset tuples. Finally, utilizing an improved variant of bound propagation as a plugin within our any-time framework yields a scheme that achieves greater accuracy than either bounded conditioning or bound propagation. In general, the framework allows to plugin any bounding scheme to bound the probability mass over unexplored tuples.

Background

Definition 1 (belief networks) *Let X=$\{X_1, ..., X_n\}$ be a set of random variables over multi-valued domains*

$\mathcal{D}(X_1), ..., \mathcal{D}(X_n)$. *A* belief network (BN) *is a pair* (G, P) *where* G *is a directed acyclic graph on* X *and* $P = \{P(X_i|pa_i)\}$ *is the set of conditional probability tables (CPT) associated with each* X_i. *An evidence* e *is an instantiated subset of variables* $E \subset X$. *The parents of a variable* X_i *together with its children and parents of its children form a* Markov blanket ma_i *of node* X_i. *A graph is* **singly connected** *(also called a* **poly-tree***), if its underlying undirected graph has no cycles. A* **loop-cutset** *of a directed graph is a set of vertices that, when removed, the graph is a poly-tree.*

Definition 2 (Relevant Subnetwork) *An* irrelevant node *of a node X is a child node Y that is not observed and does not have observed descendants. The* relevant subnetwork *of X is a subnetwork obtained by removing all irrelevant nodes in the network.*

Architecture for Any-Time Bounds

Polynomial Processing Time

To obtain the any-time bounding scheme, we start with the cutset conditioning formula. Given a Bayesian network with a cutset C, let M be the total number of cutset tuples and let h be the number of generated cutset tuples, $0 < h < M$. We can assume without loss of generality that the generated h tuples are the first h cutset tuples. Then, for a variable X with $x' \in \mathcal{D}(X)$, we can re-write Eq.(1) separating the summation over the generated tuples 1 through h and the rest as:

$$P(x'|e) = \frac{\sum_{i=1}^{h} P(x', c^i, e) + \sum_{i=h+1}^{M} P(x', c^i, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{i=h+1}^{M} P(c^i, e)} \quad (2)$$

If C is a loop-cutset, $P(x', c^i, e)$ and $P(c^i, e)$, $i \leq h$, can be computed in polynomial time. The question is how to compute or bound $\sum_{i=h+1}^{M} P(x', c^i, e)$ and $\sum_{i=h+1}^{M} P(c^i, e)$ without enumerating all tuples c^i, $i > h$.

Consider a fully-expanded search tree of depth $|C|$ over the cutset search space expanded in the order C_1,...,C_l. A path from the root to the leaf at depth $|C|$ corresponds to a cutset tuple. Hence, there is a one-to-one mapping between each leaf at depth $|C|$ and a fully-instantiated cutset tuple. If we mark all the tree edges on paths that correspond to the h generated cutset tuples, then the unexpanded tuples correspond to the unmarked leaves. We can recursively trim all unmarked leaves until only leaves branching out of marked nodes remain, thus producing a *truncated search tree*.

An example of a truncated search tree is shown in Figure 1. The leaves at depth $< |C|$ in the truncated tree correspond to the partially-instantiated cutset tuples. A path from the root C_1 to a leaf C_q at depth q is a tuple denoted $c_{1:q} = \{c_1, ..., c_q\}$ which we will refer to as *truncated* or *partially-instantiated* cutset tuples. Since every node C_j in the path from root C_1 to leaf C_l can have no more than $|\mathcal{D}(C_j)| - 1$ emanating leaves,then the number of truncated tuples, denoted M', is polynomially bounded as follows:

Proposition 1 *If C is a cutset,* d *bounds the domain size, and* h *is the number of generated cutset tuples, the number of partially-instantiated cutset tuples in the truncated search three is bounded by* $O(h \cdot (d-1) \cdot |C|)$.

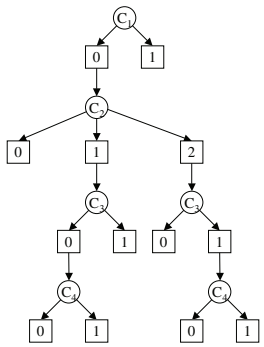

Figure 1: A search tree for cutset $C = \{C_1, ..., C_4\}$.

We enumerate all partially instantiated tuples from 1 to M' and denote the j-th tuple $c^j_{1:q_j}$, where q_j denotes the tuple's length. Clearly, the probability mass over the cutset tuples c^{h+1}, ..., c^M can be captured via the sum of the truncated tuples. Namely, $\sum_{h+1}^{M} P(c^i, e) = \sum_{j=1}^{M'} P(c^j_{1:q_j}, e)$ (similar result can be obtained for a sum of $P(x, c^i, e)$). Therefore, we can bound the unexplored tuples in Eq.(2) by bounding a polynomial number of partially-instantiated tuples.

Bounds

Replacing the summation over tuples $h+1$ through M with summation over the truncated tuples in Eq.(2), we get:

$$P(x'|e) = \frac{\sum_{i=1}^{h} P(x', c^i, e) + \sum_{j=1}^{M'} P(x', c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P(c^j_{1:q_j}, e)} \quad (3)$$

Assume we have an algorithm $\mathcal{A}$ that, for any partial assignment $c_{1:q}$, can compute lower and upper bounds, denoted $P^L_{\mathcal{A}}$ and $P^U_{\mathcal{A}}$, on $P(c_{1:q}, e)$ and $P(x, c_{1:q}, e)$.

A brute force lower bound expression using Eq.(3) can be obtained by replacing each $P(x', c^j_{1:q_j}, e)$ with its lower bound (reducing numerator) and each $P(c^j_{1:q_j}, e)$ with its upper bound (increasing denominator) yielding:

$$P(x'|e) \geq \frac{\sum_{i=1}^{h} P(x', c^i, e) + \sum_{j=1}^{M'} P^L_{\mathcal{A}}(x', c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P^U_{\mathcal{A}}(c^j_{1:q_j}, e)} \quad (4)$$

However, a tighter bound can be obtained if we apply additional transformations to Eq. (3) and prove a helpful lemma. First, we decompose $P(c^j_{1:q_j}, e)$, $0 \leq j \leq M'$, as follows:

$$P(c^j_{1:q_j}, e) = P(x', c^j_{1:q_j}, e) + \sum_{x \neq x'} P(x, c^j_{1:q_j}, e) \quad (5)$$

Replacing $P(c^j_{1:q_j}, e)$ in Eq.(3) with the right-hand size expression in Eq. (5), we obtain:

$$P(x'|e) = \frac{\sum_{i=1}^{h} P(x', c^i, e) + \sum_{j=1}^{M'} P(x', c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P(x', c^j_{1:q_j}, e) + \sum_{x \neq x'} \sum_{j=1}^{M'} P(x, c^j_{1:q_j}, e)} \quad (6)$$

We will use the following two lemmas:

Lemma 1 *Given a, b, δ>0, if* $a < b$, *then:* $\frac{a}{b} \leq \frac{a+\delta}{b+\delta}$. ■

Lemma 2 *Given positive numbers a, b, c,* c^L, c^U, *if* $a < b$ *and* $c^L \leq c \leq c^U$, *then:* $\frac{a+c^L}{b+c^L} \leq \frac{a+c}{b+c} \leq \frac{a+c^U}{b+c^U}$ ■

In Eq.(6) the sums in both numerator and denominator contain components $P(x', c^j_{1:q_j}, e)$. Hence, we can apply Lemma 2. We will obtain a lower bound by replacing each summand $P(x', c^i_{1:q_i}, e)$ in Eq.(6) with a lower bound in both numerator and denominator and replacing $P(x, c^j_{1:q_j}, e)$, $x \neq x'$, with its upper bound:

$$P(x'|e) \geq \frac{\sum_{i=1}^{h} P(x', c^i, e) + \sum_{j=1}^{M'} P^L_{\mathcal{A}}(x', c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P^L_{\mathcal{A}}(x', c^j_{1:q_j}, e) + \sum_{x \neq x'} \sum_{j=1}^{M'} P^U_{\mathcal{A}}(x, c^j_{1:q_j}, e)} \quad (7)$$

Hence, we have obtained two expressions for lower bound on $P(x'|e)$, defined in Eq. (4) and (7). In general, neither bound dominates the other although we can prove that the second bound, P^{L_2} in Eq. (7), dominates P^{L_1} if we use $\mathcal{A}$=BdP and if $|\mathcal{D}(X)|$=2 (Bidyuk & Dechter 2005).

In a similar manner, replacing $P(x', c^j_{1:q_j}, e)$ in Eq.(6) with a corresponding upper bound and $P(x, c^j_{1:q_j}, e)$, $x \neq x'$, with a lower bound, we obtain the upper bound expression:

$$P(x'|e) \geq \frac{\sum_{i=1}^{h} P(x', c^i, e) + \sum_{j=1}^{M'} P^U_{\mathcal{A}}(x', c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P^U_{\mathcal{A}}(x', c^j_{1:q_j}, e) + \sum_{x \neq x'} \sum_{j=1}^{M'} P^L_{\mathcal{A}}(x, c^j_{1:q_j}, e)} \quad (8)$$

The derivation of bounds for cutset nodes is similar (see (Bidyuk & Dechter 2005)). The outline fo the framework is given in Figure 2. Unlike the result in bounded conditioning, the resulting ATB upper bound is always ≤ 1. We can also show that lower and upper bounds derived above are always as good or better than BC bounds (Bidyuk & Dechter 2005).

Since the number of truncated cutset tuples grows polynomially with h (see Proposition 1), the ATB framework has a polynomial time complexity:

Theorem 1 (Time Complexity) *Given an algorithm deriving lower and upper bounds on probability of evidence* $P(e)$ *and* $P(x, e)$ *for any evidence* E=e *and* $\forall x \in \mathcal{D}(X)$, *in time* $O(T)$, *then* ATB *takes* $O(T \cdot h \cdot (d-1) \cdot |C|)$ *time to bound all the partially-instantiated cutset tuples.*

We refer to the proposed bounding scheme, without restricting the choice of plugin algorithm $\mathcal{A}$, as ATB framework.

Searching for High-Probability Tuples

For best results, when using ATB framework, we need to enumerate h tuples with the highest probabilities $P(c^i, e)$ in distribution $P(C, e)$. We can search for high-probability cutset tuples using, e.g., local greedy search (Kask & Dechter 1999) and branch-and-bound search. Another option is to use a stochastic simulation such as Gibbs sampling.

Any-Time Bounds Framework
Input: A belief network $\mathcal{B}$ over X, evidence $E \subset X$, cutset $C \subset X \backslash E$, truncated search tree T.
Output: lower bounds LB, upper bounds UB.
1. Generate cutset tuples:
For $i = 1$ to h do:
  $c^i \leftarrow GetNextFullyInstantiatedTuple(T)$
  compute $P(c^i, e)$
End For
2. Traverse partially-instantiated tuples:
For $j = 1$ to M' do
  $c^j_{1:q_j} \leftarrow GetNextPartiallyInstantiatedTuple(T)$
  compute bounds on $P(c^j_{1:q_j}, e)$ and $P(x_i, c^j_{1:q_j}, e)$, $\forall x_i \in X$
End For
3. Compute bounds on $P(x_i|e)$ $\forall x_i \in X$ using Eq. 4, 7, and 8.

Figure 2: Any-Time Bounds Framework Outline.

It performs a guided random walk in the multi-dimentional space of all cutset instances and can be viewed as a search algorithm looking for high-probability tuples. This is the approach we take in the current work.

We generate the cutset tuples using w-cutset sampling (Bidyuk & Dechter 2003a; 2003b) which applies Gibbs sampling over a subset of variables. We found that a brute force approach, selecting all unique cutset instances among generated samples, was not always effective. This was observed previously in (Kask & Dechter 1999). For a discrete variable C_i, the algorithm normally computes probability $P(c_i, c_{-i})$, where c_{-i}=$c \backslash c_i$, for each $c_i \in \mathcal{D}(C_i)$ yielding a sampling distribution $P(C_i|c_{-i})$. But the selected value is not always the most probable one and the tuples with higher probability may be discarded. Maintaining a list of h highest probability tuples computed during sampling (even if the Markov chain did not actually visit them), we optimize Gibbs sampling for search. Figure 3, depicting the % of explored probability mass of $P(e)$ as a function of h, shows a typical improvement in the performance on the example of Barley network with $|E|$=7 and $P(e)$=$3E$-06. Using the optimized scheme, we accumulated over 90% of the weight of $P(e)$ in a few thousand tuples in Munin3, cpcs179, and cpcs360b, and 20-30% in cpcs422b and Munin4.

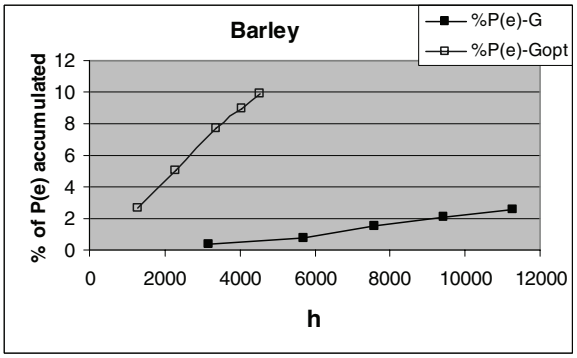

Figure 3: Performance of Gibb sampling (G) and Gibbs optimized for search (Gopt) on an instance of Barley network.

Incorporating Bound Propagation

Bound propagation (BdP) (Leisink & Kappen 2003) is an iterative algorithm that utilizes the local network structure to formulate a linear optimization (LP) problem for each variable X such that the minimum and maximum of the objective function correspond to the lower and upper bounds on posterior marginals $P(x|e)$. Initializing the bounds $P^L(x|e)$ and $P^U(x|e)$ to 0 and 1 for each variable X the algorithm solves linear minimization and maximization problems for each variable and updates the bounds until it converges.

We cannot plugin BdP directly to bound $P(c_{1:q}, e)$ because it only bounds conditional probabilities. Thus, we factorize the joint probability $P(c_{1:q}, e)$ as follows:

$$P(c_{1:q}, e) = \prod_{e_j \in E} P(e_j|e_{1:j-1}, c_{1:q})P(c_{1:q})$$

where $e_{1:j-1}$=$\{e_1, ..., e_{j-1}\}$. We know how to compute $P(c)$ if c is an assignment to all loop-cutset variables. We can also compute joint probability $P(c_{1:q})$ where $c_{1:q} = \{c_1, ..., c_q\}$, $q < |C|$, is a subset of first q cutset nodes in topological order.

Theorem 2 *If C is a topologically ordered loop-cutset of a Bayesian network and $C_{1:q} = \{C_1, ..., C_q\}$ is a subset of C, $q < |C|$, then the relevant subnetwork of $C_{1:q}$, consisting of loop-cutset nodes in subset $C_{1:q}$ and their ancestors, are singly-connected.*

Proof. We obtain a proof by showing that the relevant subnetwork of any $C_j \in C$ is singly-connected when variables $C_1, ..., C_{j-1}$ are observed. Proof is by contradiction. If the relevant subnetwork of C_j is not singly-connected, then there is a loop L with a sink S s.t. either S is observed or S has an observed descendant (otherwise S would be irrelevant). Since only variables $C_1, ..., C_j$ are observed, then observed node is C_q, $1 \leq q \leq j$. By definition of loop-cutset, $\exists C_m \in L$ s.t. $C_m \neq S$ and $C_m \in C$. Then, C_m is ancestor of C_q. Since variables are topologically ordered and all loop-cutset nodes preceding C_q are observed, then C_m must be observed, thus, breaking the loop. Contradiction.

Since the relevant subnetwork over $c_{1:q}$ is singly-connected, we can compute joint $P(c_1, ..., c_q)$ in linear time using belief propagation algorithm (Pearl 1988). We can apply BdP algorithm for each $E_j \in E$ to bound $P(e_j|e_{1:j-1}, c_{1:q})$ yielding an upper bound:

$$P^U_{BdP}(c_{1:q}, e) \triangleq \prod_{e_j \in E} P^U_{BdP}(e_j|e_{1:j-1}, c_{1:q})P(c_{1:q}) \quad (9)$$

The factorization for $P(x, c_{1:q}, e)$ is similar except it contains an additional factor $P_{BdP}(x|e, c_{1:q})$ which we also bound using BdP. The lower bound derivation is similar.

Algorithm $BdP+$

The size of the BdP linear optimization problems grows exponentially with the size of the Markov blanket. In order to limit BdP demands for memory and time, we can bound the maximum length of the Markov conditional probability table by a constant k and, thus, the maximum number of variables in LP problem. For variables, whose Markov blanket size exceeds the maximum, the lower and upper bound remain equal to their input values (usually, 0 and 1).

The performance of BdP can be improved also by restricting the Markov blanket of X_i to its relevant subnetwork. Further, if the relevant subnetwork of a node is singly-connected, then its posteriors should be computed exactly and fixed. We denote as $BdP+$ the BdP algorithm that takes advantage of the network structure as described above. (Bidyuk & Dechter 2006) show emprically that proposed improvements result in substantial gains in accuracy and speed in practice.

Algorithm $ABdP+$

When $BdP+$ is plugged into ATB, we need to bound a large number of tuples. Hence, we need to solve a large number of LP problems. Using the simplex method becomes infeasible, as our preliminary tests confirmed. Thus, we solve a relaxed LP problem, which can be described as fractional packing and covering with multiple knapsacks, using a fast greedy algorithm that finds a lower bound that is less or equal to optimal and an upper bound that is greater or equal to optimal. We denote the scheme with an approximate LP solver as $ABdP+$. Bidyuk and Dechter (2006) show emprically that $ABdP+$ achieves 10-100 times speed-up loosing only a little in the bounds accuracy.

Algorithm $BBdP+$

$ABdP+$ and $BdP+$ performance depends on the initial values of the lower and upper bounds, usually set to 0 and 1. We can boost the performance of $BdP+$ by using the bounds computed by ATB, instead of 0 and 1, to initialize its lower and upper bounds. We experiment with "boosted" $BdP+$, denoted $BBdP+$, in the empirical section.

Experiments

To evaluate the performance of our any-time bounds framework empirically, we use bound propagation with approximate LP solver, i.e., $ABdP+$, in place of algorithm $\mathcal{A}$. We refer to the resulting algorithm as ATB. We compare bounds obtained by ATB, $BdP+$, and Boosted $BdP+$ ($BBdP+$). We also computed the minimum length of BC bounds interval by plugging into the framework the brute force 0 lower bounds and prior $P(c)$ upper bounds. It remained >0.75 for all benchmarks.Where applicable, we compare our bounds with those in (Larkin 2003).

Methodology

We measure the quality of the bounds via the average length of the interval $\overline{I}$ between lower and upper bound.

We report all results for $BdP+$ "upon convergence"; its computation time is a function of k and the number of iterations needed to converge. The computation time of ATB is controlled via parameter h.

We control the time and memory of bound propagation by restricting the maximum length k of the conditional probability tables over the Markov blanket of a node. The maximum length tested was k=2^{19}. For ATB with $ABdP+$ and

$BBdP+$ k was fixed at 1025 so that ATB and $BBdP+$ average bounds length was measured as a function of h. $BBdP+$ computation time includes ATB time.

Our benchmarks are 6 networks from UAI repository listed in Table 1. Evidence is selected at random. All exact posterior marginals were obtained by bucket elimination (Dechter 1999) using min-fill heuristics. The algorithm ATB explores a subset of loop-cutset tuples in each network where the loop-cutset is obtained using mga algorithm of (Becker & Geiger 1996). The $BdP+$ algorithm uses a simplex solver from COIN-OR libraries (CLP). The experiments were conducted on 1.8Ghz CPU with 512 MB RAM.

Table 1: Benchmarks' parameters: N-number of nodes, w^*-induced width, $|LC|$-loop-cutset size, $|\mathcal{D}_{LC}|$-loop-cutset state space size, T_{BE} and T_{LC}-exact computation time via bucket elimination and loop-cutset conditioning.

network	N	w^*	$\|LC\|$	$\|\mathcal{D}_{LC}\|$	T_{BE}	T_{LC}
Alarm	37	4	5	108	0.01 sec	0.05 sec
Barley	48	7	12	>2E+6	50 sec	>22 hrs[1]
cpcs360b	360	21	26	2^{26}	20 min	> 8 hrs[1]
cpcs422b	422	22	47	2^{47}	50 min	> 2E+9 hrs[1]
Munin3	1044	7	30	$> 2^{30}$	8 sec	> 1700 hrs[1]
Munin4	1041	8	49	$> 2^{49}$	70 sec	> 1E+8 hrs[1]

Results

We summarize results for each benchmark in a chart showing the convergence of the bounds interval length with time.

Alarm network. Alarm network has 37 nodes and 108 loop-cutset tuples. Its exact posterior marginals can be obtained using either bucket elimination or cutset conditioning in less than a second. We only used this benchmark to relate to previous results of bounded conditioning and bound propagation. We experimented with 20 instances of the network with 1-4 evidence nodes. $BdP+$ obtained an average interval length of 0.44 within 3 seconds (using maximum k since all Markov blankets are small). ATB computed a more accurate bound interval of 0.31 starting with h=34, obtained within 0.039 sec, an order of magnitude faster than $BdP+$.

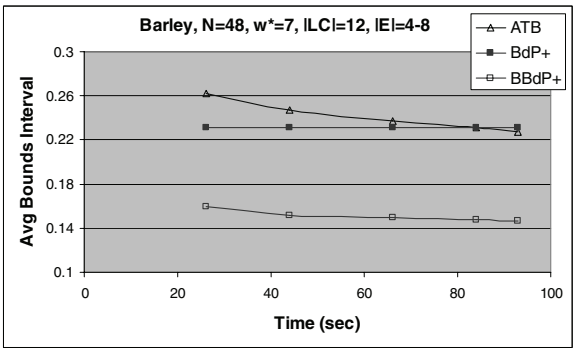

Figure 4: Barley, $\overline{I}$ as a function of time.

[1]Times are extrapolated.

Barley network. The average bounds interval length is plotted as a function of time in Figure 4. Within 100 sec, ATB processes h=4000 out of >2 million cutset tuples which account for ≈0.2% of the weight of $P(e)$. $BdP+$ yields an average bounds length of 0.23 in <2 sec for $\forall k \in [2^{10}, 2^{19}]$. ATB bounds converge with time but slowly, decreasing from 0.27 to 0.23 after 90 sec. ATB takes about 80 sec to achieve the same accuracy as $BdP+$. Using ATB results as a spring-board and having a small computation overhead, 2 sec per instance, $BBdP+$ improves considerably over both ATB and $BdP+$. It computes $\overline{I}$=0.17 in 25 sec and 0.14 in 90 sec. Although exact bucket elimination is superior (takes 50 sec), loop-cutset conditioning, using the same amount of space, takes over 22 hrs.

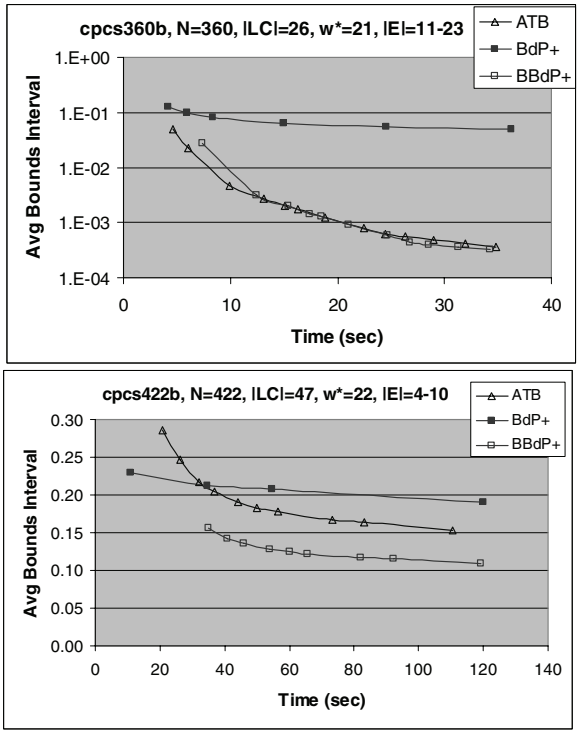

Figure 5: cpcs360b and cpcs422b, $\overline{I}$ as a function of time.

Results for CPCS networks. The results for cpcs networks are given in Figure 5, averaged over 20 instances of each network with different evidence.

cpcs360b loop-cutset has 2^{26} cutset tuples, prohibitively many for complete enumeration. Its exact computation time by bucket elimination is about 20 min. We have compared the performance of all bounding algorithms within 60 sec. ATB converged fast, covering ≈98% of probability mass at $h \approx 1000$, and outperformed $BdP+$ after 2 sec. In 60 sec, ATB computed $\overline{I}$=0.0001, two orders of magnitude smaller than $BdP+$. $BBdP+$ performance is similar to ATB. It improves on ATB results, but not enough to compensate for the time overhead. Larkin's (2003) algorithm, when applied to cpcs360b, computed $\overline{I}$=0.03 in 10 sec. Within the same time, ATB computes $\overline{I}$≈0.001. However, the comparison is not on the same instances.

cpcs422b has large induced width of w^*=22 and 2^{47} loop-cutset tuples. Computing exact marginals requires 50 min by bucket elimination and >2E+9 hrs by cutset conditioning. ATB outperforms $BdP+$ in 30 sec. Within 2 min, $BdP+$ computes $\overline{I}$=0.19 and ATB produces $\overline{I}$=0.15. $BBdP+$ computes the first data point only after 35 sec, but it is the best algorithm after that. The results of ATB and $BdP+$ are comparable to $\overline{I}$=0.15, obtained in 30 sec, in (Larkin 2003).

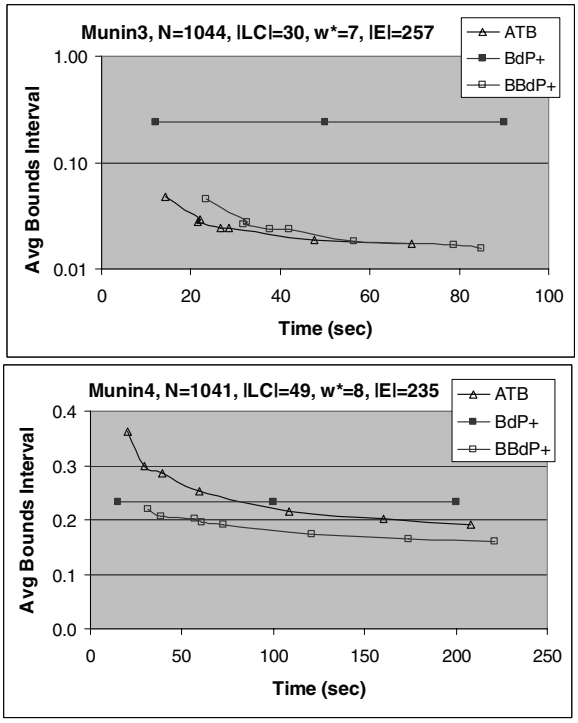

Figure 6: Munin3 and Munin4, $\overline{I}$ as a function of time.

Munin's benchmarks. Munin3 and Munin4 networks are large, with 1044 and 1041 nodes, but have small induced width of w^*=7 and w^*=8. Subsequently, exact inference by bucket elimination is easy and takes 8 sec in Munin3 and 70 sec in Munin4 (see Table 1). The empirical results are summarized in Figure 6. The behavior of the algorithms in Munin3 and Munin4 is similar. $BdP+$ computes bounds interval of 0.24 for Munin3 and 0.23 for Munin4 within 12 and 15 sec respectively. In Munin3, ATB outperforms $BdP+$ by a wide margin yielding a bounds interval of 0.05 in 12 sec. In Munin4, the loop-cutset is larger and covergence of ATB is slower. In 15 sec, ATB computes bounds interval length of 0.37. As h increases, it decreases to 0.19. ATB outperforms $BdP+$ after 90 sec. Taking ATB results as input, $BBdP+$ considerably improves ATB bounds in Munin4, compensating for the computation overhead, and outperforms both ATB and $BdP+$ timewise.

Conclusions

In this paper we present an anytime framework for bounding the posterior beliefs of every variable. The scheme is parametrized by a fixed number of cutset tuples h over which it applies exact computation using cutset conditioning. We find h high probability tuples using Gibbs sampling. We developed expressions to bound the rest of the probability mass using any off-the-shelve bounding algorithm (worst-case we can plug in 0 and priors). The resulting scheme, denoted by $ATB_h(\mathcal{A})$ (using h cutset tuples and plug-in algorithm $\mathcal{A}$), can be viewed as a boosting scheme for $\mathcal{A}$.

$ATB_h(\mathcal{A})$ can make any existing bounding scheme anytime and improve it. In this paper we focused on a specific algorithm, $\mathcal{A}$=$BdP+$, which is a variant of bound propagation. Our empirical results demonstrate the power of this anytime enhancement on several benchmarks.

Finally, we showed that for iterative bounding algorithms, such as $BdP+$, another boosting step is feasible by taking the results of $ATB_h(\mathcal{A})$ and plugging them back into $\mathcal{A}$.

Acknowledgments

This work has been partially supported by the NSF grant IIS-0412854.

References

Becker, A., and Geiger, D. 1996. A sufficiently fast algorithm for finding close to optimal junction trees. In *Uncertainty in AI*, 81–89.

Bidyuk, B., and Dechter, R. 2003a. Cycle-cutset sampling for bayesian networks. In *Canadian Conf. on AI*, 297–312.

Bidyuk, B., and Dechter, R. 2003b. Empirical study of w-cutset sampling for bayesian networks. In *UAI*, 37–46.

Bidyuk, B., and Dechter, R. 2005. An any-time scheme for bounding posterior beliefs. Technical report, UCI, http://www.ics.uci.edu/~bbidyuk/bounds.html.

Bidyuk, B., and Dechter, R. 2006. Improving bound propagation. In *European Conf. on AI (ECAI)*.

COmputational INfrastructure for Operations Research. http://www.coin-or.org.

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

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113:41–85.

Horvitz, E. J.; Suermondt, H. J.; and Cooper, G. F. 1989. Bounded conditioning: Flexible inference for decisions under scarce resources. In *Workshop on UAI*, 181–193.

Kask, K., and Dechter, R. 1999. Stochastic local search for Bayesian networks. In *Workshop on AI and Statistics*, 113–122. Morgan Kaufmann.

Larkin, D. 2003. Approximate decomposition: A method for bounding and estimating probabilistic and deterministic queries. In *Proceedings of UAI*, 346–353.

Leisink, M. A. R., and Kappen, H. J. 2003. Bound propagation. *J. of AI Research* 19:139–154.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Poole, D. 1996. Probabilistic conflicts in a search algorithm for estimating posterior probabilities in bayesian networks. *Artificial Intelligence* 88(1–2):69–100.