Journal of Artificial Intelligence Research 1 (2009) 1-38 Submitted 10/12; published

An Anytime Boosting Scheme for Bounding Posterior Beliefs[1]

Bozhena Bidyuk BBIDYUK@GOOGLE.COM
19540 Jamboree Rd
Irvine, CA 92612

Rina Dechter DECHTER@ICS.UCI.EDU
Donald Bren School of Information and Computer Science
University Of California Irvine
Irvine, CA 92697-3425

Emma Rollon EROLLON@LSI.UPC.EDU
Donald Bren School of Information and Computer Science
University Of California Irvine
Irvine, CA 92697-3425

Abstract

This paper presents an any-time scheme for computing lower and upper bounds on the posterior marginals in Bayesian networks with discrete variables. Its power is in that it can use any available scheme that bounds the probability of evidence, enhance its performance in an anytime manner, and transform it effectively into bounds for posterior marginals. The scheme is novel in that using the cutset condition principle (Pearl, 1988), it converts a bound on joint probabilities into a bound on the posterior marginals that is tighter than earlier schemes, while at the same time facilitates anytime improved performance. At the heart of the scheme is a new data structure which facilitate the efficient computation of such a bound without enumerating all the cutset tuples. Using a variant of bound propagation algorithm (Leisink & Kappen, 2003) as the plugged-in scheme, we demonstrate empirically the value of our scheme, for bounding posterior marginals and probability of evidence.

1. Introduction

This paper addresses the problem of bounding the probability of evidence and posterior marginals in Bayesian networks with discrete variables. Deriving bounds on posteriors with a given accuracy is clearly an NP-hard problem (Abdelbar & Hedetniemi, 1998; Dagum & Luby, 1993) and indeed, most available approximation algorithms provide little or no guarantee on the quality of the approximation. Still, few approaches were presented in the past few years for bounding posterior marginals (Horvitz, Suermondt, & Cooper, 1989; Poole, 1996, 1998; Mannino & Mookerjee, 2002; Mooij & Kappen, 2008) and for bounding the probability of evidence (Dechter & Rish, 2003; Larkin, 2003; Leisink & Kappen, 2003).

In this paper[2] we develop a framework that can accept any earlier developed bounding scheme and boost its performance in an anytime manner using the cutset-conditioning principle (Pearl, 1988). To facilitate our scheme we develop an expression that effectively

1. This work was supported in part by the NSF under award numbers IIS-0331707, IIS-0412854 and IIS-0713118 and by the NIH grant R01-HG004175-02.
2. The work here was presented in part in (Bidyuk & Dechter, 2006a, 2006b)

converts bounds on the probability on evidence into bounds on posterior marginals using cutset conditioning. The expression yields an algorithm that is anytime, that can use any off the shelve bounding algorithm, and can output improved bounds on the posterior marginals and on the probability of evidence.

Given a Bayesian network defined over a set of variables $\mathcal{X}$, a variable $X \in \mathcal{X}$, and a domain value $x \in \mathcal{D}(X)$, a posterior marginal $P(x|e)$ (where e is a subset of assignments to the variables, called evidence) can be computed directly from two joint probabilities, $P(x,e)$ and $P(e)$:

$$P(x|e) = \frac{P(x,e)}{P(e)} \tag{1}$$

Given a set C=$\{C_1, ..., C_p\} \subset \mathcal{X}$ of cutset variables (e.g., a loop-cutset), we can compute the probability of evidence by enumerating over all the cutset tuples $c^i \in \mathcal{D}(C)$ over cutset variables using the formula:

$$P(e) = \sum_{i=1}^{M} P(c^i, e) \tag{2}$$

where $M = \prod_{i=1}^{p} |\mathcal{D}(C_i)|$ is the number of cutset tuples. We can also compute the posterior marginals by:

$$P(x|e) = \sum_{i=1}^{M} P(x|c^i, e) P(c^i|e) \tag{3}$$

The computation of $P(c^i, e)$ for any assignment $c = c^i$ is linear in the network size if C is a loop-cutset and exponential in w if C is a w-cutset (see definition in Section 2). The limitation of the cutset-conditioning method, as defined in Eq. (2) and (3), is that the number of cutset tuples M grows exponentially with the cutset size. Furthermore, while computing the joint probability of a single tuple $P(c,e)$ is easy due to loop-cutset, computing a single conditional $P(c|e)$ in Eq. (3) already requires enumeration of all the cutset tuples because of the need to compute the normalization constant in formula (3).

There are two basic approaches to handling the combinatorial explosion in the cutset-conditioning scheme. One is to sample over the cutset space and subsequently approximate the distribution $P(C|e)$ from the samples, as we have shown in (Bidyuk & Dechter, 2007). The second approach, which we investigate here, and which was first presented in (Horvitz et al., 1989), is to enumerate a specified constant number of cutset tuples h, over which we compute exactly the quantities $P(c^i, e)$ for $1 \leq i \leq h$, and bound the tails of the distribution (over the remaining tuples). This approach is likely to perform well if the selected h tuples contain most of the probability mass of $P(e)$.

There are several alternative formulations for bounding the posterior marginals via bounds on joint probabilities. Using P^L and P^U to denote available lower and upper bounds over joint probabilities, we can obtain naive bounds on posterior marginals from Eq. (1):

$$\frac{P^L(x,e)}{P^U(e)} \leq P(x|e) \leq \frac{P^U(x,e)}{P^L(e)}$$

which usually perform very poorly and often yield an upper bound > 1.

In their derivation Horvitz et. al (Horvitz et al., 1989) started with the standard formula shown in Eq. (3) and used the prior probabilities $P(c_i)$ to select the h tuples, hoping that this will yield high posterior probability (on the assumption that priors for all tuples can be computed and stored and this will be amortized over computation of many beliefs for many different sets of evidence). Their resulting *bounded conditioning* algorithm was shown to compute good bounds on some variables in an Alarm network (with $M = 108$). However, the algorithm does not scale well with the network size since computing all the priors over all tuples becomes impractical. Also when the probability of the evidence is small, the priors become bad predictors of the high probability tuples in $P(C|e)$ and the intervals between lower and upper bound values increase.

The expression we derive in this paper gives rise to a far improved formulation for cutset-based bounds of posterior marginals yielding our *Any Time Bounds (ATB) framework*. The generated bounds are provably tighter compared with bounded conditioning. In addition, our expression accommodates the use of any off-the shelve scheme which bounds the probability of evidence (or joint probabilities of partial tuples). Namely, it accepts any algorithm for bounding $P(e)$ and generates an algorithm that bounds the posterior marginals.

The time complexity of ATB is linear in the number of explored cutset tuples h. Thus, if the complexity of bounding $P(e)$ is $O(T)$, bounding the probability mass of the unexplored tuples is $O(T \cdot h \cdot (d-1) \cdot |C|)$ where $|C|$ is the number of variables in the cutset and d is the maximum domain size.

We evaluate our framework experimentally, using a variant of *bound propagation* algorithm (Leisink & Kappen, 2003) as the plug-in bounding scheme. *Bound propagation* computes bounds by iteratively solving a linear optimization problem for each variable where the minimum and maximum of the objective function correspond to lower and upper bounds on the posterior marginals. The performance of the scheme was demonstrated on the Alarm network, the Ising grid network, and on regular bi-partite graphs. Since bound propagation is exponential in the Markov boundary size, and since it requires solution of linear programming problems many times, its overhead as a plug-in scheme was too high and not cost-effective. We therefore developed several variants which we describe and evaluate in (Bidyuk & Dechter, 2006b).

We use Gibbs cutset sampling (Bidyuk & Dechter, 2003a, 2003b) for finding high-probability cutset tuples. Other schemes, such as stochastic local search (Kask & Dechter, 1999) can also be used. The investigation into generating high-probabiliy cutset tuples is outside the primary scope of the paper.

We prove that our bounds are superior to those obtained by *bounded conditioning* (Horvitz et al., 1989) and show empirically that ATB using a variant bound propagation as a plug-in scheme is sometimes superior to bound propagation alone, even when both are given comparable time resources. More importantly, when more time is available ATB achieves greater accuracy validating our general claim. We also demonstrate the power of ATB for boosting the probability of evidence.

The paper is organized as follows. Section 2 provides background on the previously proposed method of bounded conditioning. Section 3 defines our ATB framework and analyzes its properties. Section 4 describes the implementation details of using bound propagation as an ATB plug-in and present our empirical evaluation. Section 5 discusses related work. inally, Section 6 concludes and points out directions of future work.

2. Background

2.1 Preliminaries

In this section, we define essential terminology and provide background information on Bayesian networks.

Definition **2.1 (graph concepts)** *A* **directed graph** *is a pair $\mathcal{G}$=< $\mathcal{V}, \mathcal{E}$ >, where $\mathcal{V} = \{X_1, ..., X_n\}$ is a set of nodes and $\mathcal{E} = \{(X_i, X_j)|X_i, X_j \in \mathcal{V}\}$ is the set of edges. Given $(X_i, X_j) \in \mathcal{E}$, X_i is called a* **parent** *of X_j, and X_j is called a* **child** *of X_i. The set of X_i's parents is denoted $pa(X_i)$, or pa_i, while the set of X_i's children is denoted $ch(X_i)$, or ch_i. The family of X_i includes X_i and its parents. The* **moral graph** *of a directed graph $\mathcal{G}$ is the undirected graph obtained by connecting the parents of all the nodes in $\mathcal{G}$ and removing the arrows. A* **cycle-cutset** *of an undirected graph is a subset of nodes that, when removed, yields a graph without cycles. A* **loop** *in a directed graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ whose underlying graph is a cycle. A directed graph is acyclic if it has no directed loops. A directed graph is* **singly-connected** *(also called a* **poly-tree***), if its underlying undirected graph has no cycles. Otherwise, it is called* **multiply-connected**.

Definition **2.2 (loop-cutset)** *A vertex v is a* **sink** *with respect to a loop $\mathcal{L}$ if the two edges adjacent to v in $\mathcal{L}$ are directed into v. A vertex that is not a sink with respect to a loop $\mathcal{L}$ is called an* **allowed** *vertex with respect to $\mathcal{L}$. A* **loop-cutset** *of a directed graph $\mathcal{G}$ is a set of vertices that contains at least one allowed vertex with respect to each loop in $\mathcal{G}$.*

Definition **2.3 (Bayesian network)** *Let $\mathcal{X} = \{X_1, ..., X_n\}$ be a set of random variables over multi-valued domains $\mathcal{D}(X_1), ..., \mathcal{D}(X_n)$. A* **belief network** *$\mathcal{B}$ (Pearl, 1988) is a pair $<\mathcal{G}, \mathcal{P}>$ where $\mathcal{G}$ is a directed acyclic graph whose nodes are the variables $\mathcal{X}$ and $\mathcal{P} = \{P(X_i|pa_i) \mid i = 1, ..., n\}$ is the set of conditional probability tables (CPTs) associated with each X_i. $\mathcal{B}$ represents a joint probability distribution having the product form:*

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|pa(X_i))$$

An evidence e is an instantiated subset of variables $E \subset \mathcal{X}$.

The structure of the directed acyclic graph $\mathcal{G}$ reflects the dependencies between the variables using d-separation criterion. The parents of a variable X_i together with its children and parents of its children form a **Markov boundary** of node X_i, i.e., its minimal **Markov blanket**, as defined in (Pearl, 1988).

The most common query over belief networks is *belief updating* which is the task of computing the posterior distribution $P(X_i|e)$ given evidence e and a query variable $X_i \in \mathcal{X}$. Reasoning in Bayesian networks is NP-hard (Cooper, 1990). Finding approximate posterior marginals with a fixed accuracy is also NP-hard (Dagum & Luby, 1993; Abdelbar & Hedetniemi, 1998). When the network is a poly-tree, belief updating and other inference tasks can be accomplished in time linear in size of the input. In general, exact inference is exponential in the induced width of the network's moral graph.

Definition **2.4 (w-cutset)** *A w-cutset of a Bayesian network $\mathcal{B}$ is a subset of variables C such that, when removed from the moral graph of the network, its induced width is $\leq w$.*

Throughout the paper, we will consider a Bayesian network over a set of variables $\mathcal{X}$, evidence variables $E \subset \mathcal{X}$ and evidence $E = e$, and a cutset $C = \{C_1, ..., C_p\} \subset \mathcal{X} \backslash E$. Lower-case $c = \{c_1, ..., c_p\}$ will denote an arbitrary instantiation of cutset C, and $M = |\mathcal{D}(C)| = \prod_{C_i \in C} |\mathcal{D}(C_i)|$ will denote the number of different cutset tuples.

2.2 Bounded Conditioning

Bounded conditioning (BC) is an any-time scheme for computing posterior bounds in Bayesian networks proposed by Horvitz et al. (1989). It is derived from the loop-cutset conditioning method (see Eq. (3)). Given a node $X \in \mathcal{X}$ and a domain value $x \in \mathcal{D}(X)$, they derive bounds from the following formula:

$$P(x|e) = \sum_{i=1}^{M} P(x|c^i, e)P(c^i|e) = \sum_{i=1}^{h} P(x|c^i, e)P(c^i|e) + \sum_{i=h+1}^{M} P(x|c^i, e)P(c^i|e) \tag{4}$$

The hard-to-compute $P(c^i|e)$ is replaced for $i \leq h$ with a normalization formula:

$$P(x|e) = \frac{\sum_{i=1}^{h} P(x|c^i, e)P(c^i, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{i=h+1}^{M} P(c^i, e)} + \sum_{i=h+1}^{M} P(x|c^i, e)P(c^i|e) \tag{5}$$

BC computes exactly $P(c^i, e)$ and $P(x|c^i, e)$ for the h cutset tuples with the highest prior weight $P(c^i)$ and bounds the rest.

The lower bound is obtained from Eq. (5) by replacing $\sum_{i=h+1}^{M} P(c^i, e)$ in the denominator on the first summand with the sum of priors $\sum_{i=h+1}^{M} P(c^i)$ and simply dropping the sum on the right:

$$P_{BC}^{L}(x|e) \triangleq \frac{\sum_{i=1}^{h} P(x, c^i, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{i=h+1}^{M} P(c^i)} \tag{6}$$

The upper bound is obtained from Eq. (5) by replacing $\sum_{i=h+1}^{M} P(c^i, e)$ in the denominator with a zero, replacing $P(x|c^i, e)$ for $i > h$ with its upper bound of 1, and replacing $P(c^i|e)$ for $i > h$ with a derived upper bound (not provided here):

$$P_{BC}^{U}(x|e) \triangleq \frac{\sum_{i=1}^{h} P(x, c^i, e)}{\sum_{i=1}^{h} P(c^i, e)} + \frac{\sum_{i=h+1}^{M} P(c^i)}{\sum_{i=1}^{h} P^L(c^i|e) + 1 - \sum_{i=1}^{h} P^U(c^i|e)}$$

Applying definitions for $P^L(c^i|e) = \frac{P(c^i,e)}{\sum_{i=1}^{h} P(c^i,e) + \sum_{i=h+1}^{M} P(c^i)}$ and $P^U(c^i|e) = \frac{P(c^i,e)}{\sum_{i=1}^{h} P(c^i,e)}$ from Horvitz et al. (1989), we get:

$$P_{BC}^{U}(x|e) \triangleq \frac{\sum_{i=1}^{h} P(x, c^i, e)}{\sum_{i=1}^{h} P(c^i, e)} + \frac{(\sum_{i=h+1}^{M} P(c^i))(\sum_{i=1}^{h} P(c^i, e) + \sum_{i=h+1}^{M} P(c^i))}{\sum_{i=1}^{h} P(c^i, e)} \tag{7}$$

The bounds expressed in Eq. (6) and (7) converge to the exact posterior marginals as $h \to M$. However:

Theorem **2.1 (bounded conditioning bounds interval)** *The interval between lower and upper bounds computed by bounded conditioning is lower bounded by the probability mass of prior distribution $P(C)$ of the unexplored cutset tuples:*

$$\forall h, P^U_{BC}(x|e) - P^L_{BC}(x|e) \geq \sum_{i=h+1}^{M} P(c^i)$$

Proof. See Appendix A. $\square$

3. Architecture for Any-Time Bounds

In this section, we describe our Any-Time Bounds (ATB) framework. It builds on the same principles as bounded conditioning. Namely, given a cutset C and some method for generating h cutset tuples, the probabilities of the h tuples are evaluated exactly and the rest are upper and lower bounded. However, ATB bounds can be improved by using a different plug-in for bounding participating joint probabilities and computes tighter bounds than BC even when using priors to bound $P(c, e)$.

For the rest of the section, $c_{1:q} = \{c_1, ..., c_q\}$ with $q < |C|$ denotes a generic partial instantiation of the first q variables in C, while $c^i_{1:q}$ indicates a particular partial assignment.

Given h cutset tuples, $0 \leq h \leq M$, that we assume without loss of generality to be the first h tuples according to some enumeration order, a variable $X \in \mathcal{X} \backslash E$ and $x \in \mathcal{D}(X)$, we can rewrite Eq. (3) as:

$$P(x|e) = \frac{\sum_{i=1}^{M} P(x, c^i, e)}{\sum_{i=1}^{M} P(c^i, e)} = \frac{\sum_{i=1}^{h} P(x, c^i, e) + \sum_{i=h+1}^{M} P(x, c^i, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{i=h+1}^{M} P(c^i, e)} \tag{8}$$

The probabilities $P(x, c^i, e)$ and $P(c^i, e)$, $1 \leq i \leq h$, can be computed in polynomial time if C is a loop-cutset and in time and space exponential in w if C is a w-cutset. The question is how to compute or bound $\sum_{i=h+1}^{M} P(x, c^i, e)$ and $\sum_{i=h+1}^{M} P(c^i, e)$ in an efficient manner.

Our approach first replaces the sums over the tuples c^{h+1},...,c^M with a sum over a polynomial number (in h) of partially-instantiated tuples. From that, we develop new expressions for lower and upper bounds on posterior marginals as a function of the lower and upper bounds on the joint probabilities $P(x, c_{1:q}, e)$ and $P(c_{1:q}, e)$. We assume in our derivation that there is an algorithm $\mathcal{A}$ that can compute those bounds, and refer to them as $P^L_{\mathcal{A}}(x, c_{1:q}, e)$ (resp. $P^L_{\mathcal{A}}(c_{1:q}, e)$) and $P^U_{\mathcal{A}}(x, c_{1:q}, e)$ (resp. $P^U_{\mathcal{A}}(c_{1:q}, e)$), respectively. In our experiments, algorithm A will be the bounded propagation algorithm (Leisink & Kappen, 2003).

3.1 Bounding the Number of Processed Tuples

Consider a fully-expanded search tree of depth $|C|$ over the cutset search space expanded in the order C_1,...,C_p. A path from the root to the leaf at depth $|C|$ corresponds to a full cutset tuple. If we mark all the tree edges on paths that correspond to the first h generated cutset tuples, then the unexpanded tuples c^i, $i > h$, correspond to unmarked leaves. We can obtain the *truncated search tree* by trimming all the unmarked branches.

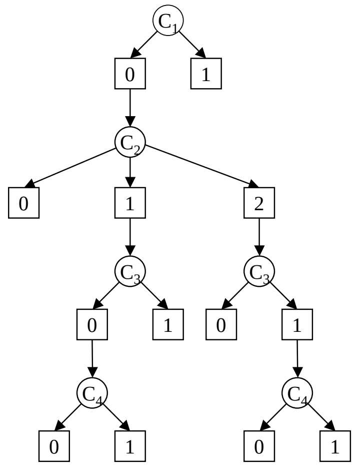


Figure 1: A search tree for cutset $C = \{C_1, ..., C_4\}$.

Definition **3.1 (truncated search tree)** *Given a search tree T covering the search space $\mathcal{H}$ over variables $\mathcal{Y} = \{Y_1, \ldots, Y_m\} \subseteq \mathcal{X}$, a* **truncated search tree** *relative to a subset $S = \{y^1, ..., y^t\} \subset \mathcal{D}(Y_1) \times ... \times \mathcal{D}(Y_m)$ of full assignments, is obtained by marking the edges on all the paths appearing in S and removing all unmarked edges and nodes except those emanating from marked nodes.*

Let $S = \{c^1, \ldots, c^h\}$. Clearly, the leaves at depth $q < |C|$ in the truncated tree relative to S correspond to the partially instantiated cutset tuples $c_{1:q}$ which are not extended to full cutset assignments.

Example **3.1** *Consider a Bayesian network $\mathcal{B}$ with cutset variables C=$\{C_1, ..., C_4\}$, domain values $\mathcal{D}(C_1)$=$\mathcal{D}(C_3)$=$\mathcal{D}(C_4)$=$\{0,1\}$, $\mathcal{D}(C_2)$=$\{0,1,2\}$, and four fully-instantiated tuples $\{0,1,0,0\}$, $\{0,1,0,1\}$, $\{0,2,1,0\}$, $\{0,2,1,1\}$. Figure 1 shows its truncated search tree, where the remaining partially instantiated tuples are $\{0,0\}$, $\{0,1,1\}$, $\{0,2,0\}$, and $\{1\}$.*

Proposition **3.1** *Let C be a cutset, d be the maximum domain size, and h be the number of generated cutset tuples, then the number of partially-instantiated cutset tuples in the truncated search tree is bounded by $O(h \cdot (d-1) \cdot |C|)$.*

Proof. Since every node in the path from the root C_1 to a leaf C_p can have no more than $(d-1)$ emanating leaves, the theorem clearly holds. $\square$

Let M' be the number of truncated tuples. We can enumerate the partially instantiated tuples, denoting the j-th tuple as $c^j_{1:q_j}$, $1 \leq j \leq M'$, where q_j is the tuple's length. Clearly, the probability mass over the cutset tuples c^{h+1}, ..., c^M can be captured by a sum over the truncated tuples. Namely:

Proposition **3.2**

$$\sum_{i=h+1}^{M} P(c^i, e) = \sum_{j=1}^{M'} P(c^j_{1:q_j}, e) \tag{9}$$

$$\sum_{i=h+1}^{M} P(x, c^i, e) = \sum_{j=1}^{M'} P(x, c^j_{1:q_j}, e) \tag{10}$$

□

Therefore, we can bound the sums over the tuples $h+1$ through M in Eq. (8) by bounding a polynomial number of partially-instantiated tuples as follows,

$$P(x|e) = \frac{\sum_{i=1}^{h} P(x, c^i, e) + \sum_{j=1}^{M'} P(x, c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P(c^j_{1:q_j}, e)} \tag{11}$$

3.2 Bounding the Probability over the Truncated Tuples

In the following, we develop lower and upper bound expressions used by ATB.

3.2.1 Lower Bounds

First, we decompose $P(c^j_{1:q_j}, e)$, $0 \leq j \leq M'$, as follows. Given a variable $X \in \mathcal{X}$ and a distinguished value $x \in \mathcal{D}(X)$:

$$P(c^j_{1:q_j}, e) = \sum_{x' \in D(X)} P(x', c^j_{1:q_j}, e) = P(x, c^j_{1:q_j}, e) + \sum_{x' \neq x} P(x', c^j_{1:q_j}, e) \tag{12}$$

Replacing $P(c^j_{1:q_j}, e)$ in Eq. (11) with the right-hand side of Eq. (12), we get:

$$P(x|e) = \frac{\sum_{i=1}^{h} P(x, c^i, e) + \sum_{j=1}^{M'} P(x, c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P(x, c^j_{1:q_j}, e) + \sum_{j=1}^{M'} \sum_{x' \neq x} P(x', c^j_{1:q_j}, e)} \tag{13}$$

We will use the following two lemmas:

Lemma **3.1** *Given positive numbers $a > 0$, $b > 0$, $\delta \geq 0$, if $a < b$, then:* $\frac{a}{b} \leq \frac{a+\delta}{b+\delta}$. □

Lemma **3.2** *Given positive numbers a, b, δ, δ^L, δ^U, if $a < b$ and $\delta^L \leq \delta \leq \delta^U$, then:*

$$\frac{a+\delta^L}{b+\delta^L} \leq \frac{a+\delta}{b+\delta} \leq \frac{a+\delta^U}{b+\delta^U}$$

□

The proof of both lemmas is straight forward.

Lemma 3.2 says that if the sums in numerator and denominator have some component δ in common, then replacing δ with a larger value δ^U in both the numerator and the denominator yields a larger fraction. Replacing δ with a smaller value δ^L in both places yields a smaller fraction.

Observe now that in Eq. (13) the sums in both the numerator and the denominator contain $P(x, c^j_{1:q_j}, e)$. Hence, we can apply Lemma 3.2. We will obtain a lower bound by replacing $P(x, c^j_{1:q_j}, e)$, $1 \leq j \leq M'$, in Eq. (13) with corresponding lower bounds in both numerator and denominator, yielding:

$$P(x|e) \geq \frac{\sum_{i=1}^{h} P(x, c^i, e) + \sum_{j=1}^{M'} P^L_{\mathcal{A}}(x, c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P^L_{\mathcal{A}}(x, c^j_{1:q_j}, e) + \sum_{j=1}^{M'} \sum_{x' \neq x} P(x', c^j_{1:q_j}, e)} \tag{14}$$

Subsequently, we replace each $\sum_{x' \neq x} P(x', c_{1:q}, e)$ with its upper bound in Eq. 14 (increasing denominator), yielding:

$$P(x|e) \geq \frac{\sum_{i=1}^{h} P(x, c^i, e) + \sum_{j=1}^{M'} P^L_{\mathcal{A}}(x, c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P^L_{\mathcal{A}}(x, c^j_{1:q_j}, e) + \sum_{j=1}^{M'} UB[\sum_{x' \neq x} P(x', c^j_{1:q_j}, e)]} \triangleq P^L_{\mathcal{A}}(x|e) \tag{15}$$

where upper bound UB can be obtained as follows:

$$UB[\sum_{x' \neq x} P(x', c^j_{1:q_j}, e)] \triangleq \min \left\{ \begin{array}{l} \sum_{x' \neq x} P^U_{\mathcal{A}}(x', c^j_{1:q_j}, e) \\ P^U_{\mathcal{A}}(c^j_{1:q_j}, e) \end{array} \right. \tag{16}$$

The value $\sum_{x' \neq x} P^U_{\mathcal{A}}(x', c^j_{1:q_j}, e)$ is, obviously, an upper bound of $\sum_{x' \neq x} P(x', c^j_{1:q_j}, e)$, and it provides a good upper bound for variables with domain size two since $\sum_{x' \neq x} P(x', c^j_{1:q_j}, e) = P(x', c^j_{1:q_j}, e) \leq P(c^j_{1:q_j}, e)$. The value $P^U_{\mathcal{A}}(c^j_{1:q_j}, e)$ is also an upper bound since $\sum_{x' \neq x} P(x', c^j_{1:q_j}, e) \leq P(c^j_{1:q_j}, e) \leq P^U_{\mathcal{A}}(c^j_{1:q_j}, e)$, and it may provide a better bound for variables with larger domain sizes.

Please note that the expression in the Eq. (15) above also provides an any-time lower bound on the joint probability $P(x, e)$ and can be used to compute a lower bound on the probability of evidence. In general, a lower bound denoted $P^L_{\mathcal{A}}(e)$ is obtained by:

$$P(e) \geq \sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P^L_{\mathcal{A}}(c^j_{1:q_j}, e) \triangleq P^L_{\mathcal{A}}(e) \tag{17}$$

3.2.2 Upper Bound

The upper bound expression can be obtained in a similar manner. Since both numerator and denominator in Eq. (13) contain addends $P(x, c^j_{1:q_j}, e)$, using Lemma 3.2 we replace each $P(x, c^j_{1:q_j}, e)$ with an upper bound $P^U_{\mathcal{A}}(x, c^j_{1:q_j}, e)$ yielding:

$$P(x|e) \leq \frac{\sum_{i=1}^{h} P(x, c^i, e) + \sum_{j=1}^{M'} P^U_{\mathcal{A}}(x, c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P^U_{\mathcal{A}}(x, c^j_{1:q_j}, e) + \sum_{x' \neq x} \sum_{j=1}^{M'} P(x', c^j_{1:q_j}, e)} \tag{18}$$

Subsequently, replacing each $P(x, c^j_{1:q_j}, e)$, $x \neq x'$, with a lower bound (reducing denominator), we obtain a new upper bound expression on $P(x|e)$:

$$P(x|e) \leq \frac{\sum_{i=1}^{h} P(x, c^i, e) + \sum_{j=1}^{M'} P^U_{\mathcal{A}}(x, c^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P^U_{\mathcal{A}}(x, c^j_{1:q_j}, e) + \sum_{j=1}^{M'} \sum_{x' \neq x} P^L_{\mathcal{A}}(x', c^j_{1:q_j}, e)} \triangleq P^U_{\mathcal{A}}(x|e) \tag{19}$$

Similar to the lower bound, the numerator in the upper bound expression $P^U_{\mathcal{A}}(x|e)$ provides an any-time upper bound on the joint probability $P(x, c^i, e)$ which can be generalized to upper bound the probability of evidence:

$$P(e) \leq \sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'} P^U_{\mathcal{A}}(c^j_{1:q_j}, e) \triangleq P^U_{\mathcal{A}}(e) \tag{20}$$

The derived bounds $P^L_{\mathcal{A}}(x|e)$ and $P^U_{\mathcal{A}}(x|e)$ are never worse than those obtained by bounded conditioning, as we will show in Section 3.4.

3.3 Algorithmic Description

Figure 2 summarizes the any-time bounding scheme ATB. In steps 1 and 2, we generate h fully-instantiated cutset tuples and compute exactly the probabilities $P(c^i, e)$ and $P(X, c^i, e)$ for $i \leq h$, $\forall X \in \mathcal{X} \backslash (C \cup E)$, using the bucket-elimination algorithm. In step 3, we compute bounds on the partially instantiated tuples using algorithm $\mathcal{A}$. In step 4, we compute the lower and upper bounds on the posterior marginals using expressions (15) and (19), respectively.

Example **3.2** *Consider again the Bayesian network $\mathcal{B}$ described in Example 3.1. Recall that $\mathcal{B}$ has a cutset $C = \{C_1, ..., C_4\}$ with domains $\mathcal{D}(C_1) = \mathcal{D}(C_3) = \mathcal{D}(C_4) = \{0, 1\}$ and $\mathcal{D}(C_2) = \{0, 1, 2\}$. The total number of cutset tuples is $M = 24$. Let $X \notin C$ be a variable in $\mathcal{B}$ with domain $\mathcal{D}(X) = \{x, x'\}$. We will compute bounds on $P(x|e)$. Assume we generated*

```
Any-Time Bounds Architecture
Input: A belief network ($\mathcal{B}$), variables $\mathcal{X}$, evidence $E \subset \mathcal{X}$, cutset $C \subset \mathcal{X} \backslash E$, constant $h$,
algorithm $\mathcal{A}$ for computing lower and upper bounds on joint probabilities.
Output: lower bounds $P^L$, upper bounds $P^U$.
1. Generate $h$ cutset tuples.
2. Compute:
        $S \leftarrow \sum_{i=1}^{h} P(c^i, e)$
        $S_x \leftarrow \sum_{i=1}^{h} P(x, c^i, e), \quad \forall x \in \mathcal{D}(X), \quad \forall X \in \mathcal{X} \backslash (C \cup E)$
3. Traverse partially-instantiated tuples:
    3.1 Generate the truncated tree associated with the $h$ tuples and let $c^1_{1:q_1}, ..., c^{M'}_{1:q_{M'}}$ be
        the $M'$ partial assignments.
    3.2 $\forall x \in \mathcal{D}(X), \quad \forall X \in \mathcal{X} \backslash (C \cup E)$, compute:
        $LB_{\mathcal{A}}(x) \leftarrow \sum_{j=1}^{M'} P^L_{\mathcal{A}}(x, c^j_{1:q_j}, e)$
        $UB'_{\mathcal{A}}(x) \leftarrow \sum_{j=1}^{M'} UB[\sum_{x' \neq x} P_{\mathcal{A}}(x', c^j_{1:q_j}, e)]$
        $UB_{\mathcal{A}}(x) \leftarrow \sum_{j=1}^{M'} P^U_{\mathcal{A}}(x, c^j_{1:q_j}, e)$
4. Compute bounds:
        $P^L_{\mathcal{A}}(x|e) = \frac{S_x + LB_{\mathcal{A}}(x)}{S + LB_{\mathcal{A}}(x) + UB'_{\mathcal{A}}(x)}$
        $P^U_{\mathcal{A}}(x|e) = \frac{S_x + UB_{\mathcal{A}}(x)}{S + UB_{\mathcal{A}}(x) + LB_{\mathcal{A}}(x)}$
```

Figure 2: Any-Time Bounds Architecture

the same four cutset tuples ($h = 4$) as before:

$$c^1 = \{C_1 = 0, C_2 = 1, C_3 = 0, C_4 = 0\} = \{0, 1, 0, 0\}$$
$$c^2 = \{C_1 = 0, C_2 = 1, C_3 = 0, C_4 = 1\} = \{0, 1, 0, 1\}$$
$$c^3 = \{C_1 = 0, C_2 = 2, C_3 = 1, C_4 = 0\} = \{0, 2, 1, 0\}$$
$$c^4 = \{C_1 = 0, C_2 = 2, C_3 = 1, C_4 = 1\} = \{0, 2, 1, 1\}$$

The corresponding truncated search tree is shown in Figure 1. For the tuple $\{0, 1, 0, 0\}$, we compute exactly the probabilities $P(x, C_1 = 0, C_2 = 1, C_3 = 0, C_4 = 0, e)$ and $P(C_1 = 0, C_2 = 1, C_3 = 0, C_4 = 0)$. Similarly, we obtain exact probabilities $P(x, C_1 = 0, C_2 = 1, C_3 = 0, C_4 = 1)$ and $P(C_1 = 0, C_2 = 1, C_3 = 0, C_4 = 1)$ for the second cutset instance $\{0, 1, 0, 1\}$. Since $h = 4$, $\sum_{i=1}^{h} P(x', c^i, e)$ and $\sum_{i=1}^{h} P(c^i, e)$ are:

$$\sum_{i=1}^{4} P(x, c^i, e) = P(x, c^1, e) + P(x, c^2, e) + P(x, c^3, e) + P(x, c^4, e)$$
$$\sum_{i=1}^{4} P(c^i, e) = P(c^1, e) + P(c^2, e) + P(c^3, e) + P(c^4, e)$$

The remaining partial tuples are: $c^1_{1:2} = \{0, 0\}$, $c^2_{1:3} = \{0, 1, 1\}$, $c^3_{1:3} = \{0, 2, 0\}$, and $c^4_{1:1} = \{1\}$. Since these 4 tuples are not full cutsets, we compute bounds on their joint probabilities.

Using the same notation as in Figure 2, the sums over the partially instantiated tuples will have the form:

$$UB_{\mathcal{A}}(x) \triangleq P^U_{\mathcal{A}}(x, c^1_{1:2}, e) + P^U_{\mathcal{A}}(x, c^2_{1:3}, e) + P^U_{\mathcal{A}}(x, c^3_{1:3}, e) + P^U_{\mathcal{A}}(x, c^4_{1:1}, e)$$

$$LB_{\mathcal{A}}(x) \triangleq P^L_{\mathcal{A}}(x, c^1_{1:2}, e) + P^L_{\mathcal{A}}(x, c^2_{1:3}, e) + P^L_{\mathcal{A}}(x, c^3_{1:3}, e) + P^L_{\mathcal{A}}(x, c^4_{1:1}, e)$$

From Eq. (19) we get:

$$P^U_{\mathcal{A}}(x|e) = \frac{\sum_{i=1}^4 P(x, c^i, e) + UB_{\mathcal{A}}(x)}{\sum_{i=1}^4 P(c^i, e) + UB_{\mathcal{A}}(x) + LB_{\mathcal{A}}(x')}$$

From Eq. (15) we get:

$$P^L_{\mathcal{A}}(x|e) = \frac{\sum_{i=1}^4 P(x, c^i, e) + LB_{\mathcal{A}}(x)}{\sum_{i=1}^4 P(c^i, e) + LB_{\mathcal{A}}(x) + UB_{\mathcal{A}}(x')}$$

The total number of tuples processed is $M' = 4 + 4 = 8 < 24$.

3.4 ATB Framework's Properties

In this section we analyze the time complexity of the ATB framework, evaluate its worst-case lower and upper bounds, and the monotonicity properties of its bounds interval (as a function of h).

Theorem **3.1 (complexity)** *Given an algorithm $\mathcal{A}$ that computes lower and upper bounds on joint probabilities $P(c_{1:q_i}, e)$ and $P(x, c_{1:q_i}, e)$ in time $O(T)$, and a loop-cutset C, $P^L_{\mathcal{A}}(x|e)$ and $P^U_{\mathcal{A}}(x|e)$ are computed in time $O(h \cdot N + T \cdot h \cdot (d-1) \cdot |C|)$ where d is the maximum domain size and N is the problem input size.*

Proof. Since C is a loop-cutset, the exact probabilities $P(c^i, e)$ and $P(x, c^i, e)$ can be computed in time $O(N)$. From Proposition 3.1, there are $O(h \cdot (d-1) \cdot |C|)$ partially-instantiated tuples. Since algorithm $\mathcal{A}$ computes upper and lower bounds on $P(c^j_{1:q_j}, e)$ and $P(x, c^j_{1:q_j}, e)$ in time $O(T)$, the bounds on partially-instantiated tuples can be computed in time $O(T \cdot h \cdot (d-1) \cdot |C|))$. □

Let the plug-in algorithm $\mathcal{A}$ be a brute-force algorithm, denoted BF, that trivially instantiates $P^L_{BF}(x, c^j_{1:q_j}, e) = 0$, $P^U_{BF}(x, c^j_{1:q_j}, e) = P(c^j_{1:q_j})$, and $UB[\sum_{x' \neq x} P(x', c^j_{1:q_j}, e)] = P(c^j_{1:q_j})$. Then, from Eq. (15):

$$P^L_{BF}(x|e) \triangleq \frac{\sum_{i=1}^h P(x, c^i, e)}{\sum_{i=1}^h P(c^i, e) + \sum_{j=1}^{M'} P(c^j_{1:q_j})} = \frac{\sum_{i=1}^h P(x, c^i, e)}{\sum_{i=1}^h P(c^i, e) + \sum_{j=h+1}^{M} P(c^j)} \tag{21}$$

while from Eq. (19):

$$P^U_{BF}(x|e) \triangleq \frac{\sum_{i=1}^h P(x, c^i, e) + \sum_{j=1}^{M'} P(c^j_{1:q_j})}{\sum_{i=1}^h P(c^i, e) + \sum_{j=1}^{M'} P(c^j_{1:q_j})} = \frac{\sum_{i=1}^h P(x, c^i, e) + \sum_{j=h+1}^{M} P(c^j)}{\sum_{i=1}^h P(c^i, e) + \sum_{j=h+1}^{M} P(c^j)} \tag{22}$$

Assuming that algorithm $\mathcal{A}$ computes bounds at least as good as those computed by BF, $P^L_{BF}(x|e)$ and $P^U_{BF}(x|e)$ are the worst-case bounds computed by ATB.

Now, we are ready to compute an upper bound on the ATB bounds interval:

Theorem **3.2 (ATB bounds interval upper bound)** *ATB bounds interval length is upper bounded by a monotonic non-increasing function of h:*

$$P^U_{\mathcal{A}}(x|e) - P^L_{\mathcal{A}}(x|e) \leq \frac{\sum_{j=h+1}^{M} P(c^j)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=h+1}^{M} P(c^j)} \triangleq I_h$$

Proof. See Appendix C. $\square$

Next we show that ATB lower and upper bounds are as good or better as the bounds computed by BC, as stated in the following two theorems.

Theorem **3.3 (tighter lower bound)** $P^L_{\mathcal{A}}(x|e) \geq P^L_{BC}(x|e)$.

Proof. $P^L_{BF}(x|e)$ is the worst-case lower bound computed by ATB. Since $P^L_{BF}(x|e) = P^L_{BC}(x|e)$, and $P^L_{\mathcal{A}}(x|e) \geq P^L_{BF}(x|e)$, then $P^L_{\mathcal{A}}(x|e) \geq P^L_{BC}(x|e)$. $\square$

Theorem **3.4 (tighter upper bound)** $P^U_{\mathcal{A}}(x|e) \leq P^U_{BC}(x|e)$.

Proof. $P^U_{BF}(x|e)$ is the worst-case upper bound computed by ATB. Since $P^U_{BF}(x|e) \leq P^U_{BC}(x|e)$ due to lemma 3.1, it follows that $P^U_{\mathcal{A}}(x|e) \leq P^U_{BC}(x|e)$. $\square$

4. Experimental Evaluation

The purpose of the experiments is to evaluate the performance of our ATB framework on the probabilitic tasks of single-variable posterior marginals and probability of evidence. The experiments on the first task were conducted on 1.8Ghz CPU with 512 MB RAM, while the experiments on the second task were conducted on 2.66GHz CPU with 2Gb RAM.

Recall that ATB has a control parameter h that fixes the number of cutset tuples for which the algorithm computes its exact joint probability. Given a fixed h, the quality of the bounds will presumably depend on the ability to select h high probability cutset tuples. In our implementation, we use an optimized version of Gibbs sampling, that during the sampling process maintains a list of the h tuples having the highest joint probability. As noted, other schemes should be considered for this subtask as part of the future work. We obtain the loop-cutset using *mga* algorithm (Becker & Geiger, 1996).

In the following, we describe Bound Propagation and its variants, which we use as a plug-in algorithm $\mathcal{A}$ and also as a stand-alone bounding scheme, and report the experimental results for each task.

4.1 Bound Propagation

Bound Propagation (BdP) (Leisink & Kappen, 2003) is an iterative algorithm that bounds the posterior marginals of a variable. BdP formulates a linear optimization problem for each variable $X \in X$. The minimum and maximum of the objective function correspond to the lower and upper bound on the posterior marginal $P(x|e)$, $x \in \mathcal{D}(X)$. The size of the linear optimization problems grows exponentially with the size of the Markov boundary of each variable.

We cannot plug-in directly BdP into ATB to bound $P(c_{1:q}, e)$ because it only bounds conditional probabilities. Thus, we factorize $P(c_{1:q}, e)$ as follows:

$$P(c_{1:q}, e) = \prod_{e_j \in E} P(e_j|e_1, \ldots, e_{j-1}, c_{1:q})P(c_{1:q})$$

where $c_{1:q}$ is a subset of the first q cutset nodes in topological order. Each factor $P(e_j|e_1, \ldots, e_{j-1}, c_{1:q})$ can be bounded by BdP, while $P(c_{1:q})$ can be computed exactly since the relevant subnetwork over $c_{1:q}$ is singly connected. Let P^L_{BdP} and P^U_{BdP} denote the lower and upper bounds computed by BdP on some marginal. The bounds BdP computes on the joint probability are,

$$\prod_{e_j \in E} P^L_{BdP}(e_j|e_1, \ldots, e_{j-1}, c_{1:q})P(c_{1:q}) \leq P(c_{1:q}, e) \leq \prod_{e_j \in E} P^U_{BdP}(e_j|e_1, \ldots, e_{j-1}, c_{1:q})P(c_{1:q})$$

Note that BdP has to bound a large number of tuples when plugged-in into ATB, and therefore, solve a large number of linear optimization problems. Our preliminary tests showed that plugging BdP into ATB became timewise infeasible. Instead, we developed and used an improved version of BdP called *ABdP+* (Bidyuk & Dechter, 2006b) as a plug-in algorithm $\mathcal{A}$, which was more cost-effective in terms of accuracy and time overhead.

ABdP+ is build upon another version of BdP, called *BdP+* proposed in (Bidyuk & Dechter, 2006b). Since in our experiments we consider *BdP+* as a stand-alone algorithm to bound marginals, we first describe *BdP+* and then *ABdP+*.

BdP+ exploits the structure of the network to restrict the computation of $P(x|e)$ to the relevant subnetwork of variable X instead of using the whole Markov boundary. Moreover, *BdP+* controls the overhead using a parameter k which specifies the maximum subnetwork size used in the linear optimization problems. For variables whose subsetwork size exceeds k, their lower and upper bound values remain 0 and 1, respectively. Our initial evaluation showed that the overhead of ATB using *BdP+* as plug-in algorithm $\mathcal{A}$ was still high.

ABdP+ includes the same enhancements as *BdP+*, but solves the linear optimization problem for each variable using an approximation algorithm. This implies that we obtain bounds faster, but they are not as accurate. Roughly, the relaxed linear optimization problem can be described as a fractional packing and covering with multiple knapsacks, and solved by a fast greedy algorithm (see (Bidyuk & Dechter, 2006b) for more details). Note that since *ABdP+* is parameterized by k, ATB using *ABdP+* as plug-in algorithm $\mathcal{A}$ has two control parameters: h and k.

network	N	w*	|LC|	$\|\mathcal{D}(\mathbf{LC})\|$	$\|E\|$	Time(BE)	Time(LC)
Alarm	37	4	5	108	1-4	0.01 sec	0.05 sec
Barley	48	7	12	>2E+6	4-8	50 sec	>22 hrs[1]
cpcs54	54	15	15	32768	2-8	1 sec	22 sec
cpcs179	179	8	8	49152	12-24	2 sec	37 sec
cpcs360b	360	21	26	2^{26}	11-23	20 min	> 8 hrs[1]
cpcs422b	422	22	47	2^{47}	4-10	50 min	$> 2 \times 10^9$ hrs[1]
Munin3	1044	7	30	$> 2^{30}$	257	8 sec	> 1700 hrs[1]
Munin4	1041	8	49	$> 2^{49}$	235	70 sec	$> 1 \times 10^8$ hrs[1]

Table 1: Complexity characteristics of the benchmarks from the UAI repository: N-number of nodes, w^*-induced width, $|LC|$-number of nodes in a loop-cutset, $|\mathcal{D}(LC)|$-loop-cutset state space size, Time(BE) is the exact computation time via bucket elimination, Time(LC) is the exact computation time via loop-cutset conditioning. The results are averaged over a set of network instances with different evidence. Evidence nodes and their values are selected at random.

4.2 Bounding Single-Variable Marginals

4.2.1 Algorithms and Benchmarks

We compare the performance of the following four algorithms: ATB and $BdP+$, as described in the previous section; BC using the brute force 0 lower bounds and prior $P(c)$ upper bounds; and, a combination of ATB and $BdP+$, denoted as $BBdP+$, where we use the bounds computed by ATB as initial bounds for $BdP+$. Note that, given fixed values of h and k, $BBdP+$ will always compute tighter bounds than either ATB and $BdP+$. Our goal is to analyze its trade-off between the increase of the bounds' accuracy and the computation time overhead. We also compare with *Approximate Decomposition* (AD) (Larkin, 2003) whenever it is feasible and relevant. In (Bidyuk, 2006), we provide additional comparison with various refinements described earlier in (Bidyuk & Dechter, 2006b).

We tested our framework on four different benchmarks: *Alarm*, *Barley*, *CPCS*, and *Munin*. *Alarm network* is a model for monitoring patients undergoing surgery in an operating room (OR) for emergencies that would be handled by the OR anesthesiologist (Beinlich, Suermondt, Chavez, & Cooper, 1989). *Barley network* is a part of the decision-support system for growing malting barley (Kristensen & Rasmussen, 2002). *CPCS networks* are derived from the Computer-Based Patient Care Simulation system and based on INTERNIST-1 and Quick Medical Reference Expert systems (Pradhan, Provan, Middleton, & Henrion, 1994). We experiment with cpcs54, cpcs179, cpcs360b, and cpcs422b networks. *Munin networks* is a part of the expert system for computer-aided electromyography (Andreassen, Jensen, Andersen, Falck, Kjaerulff, Woldbye, Srensen, Rosenfalck, & Jensen, 1990). We experiment with Munin3 and Munin4 networks. For each network, we generated 20 different sets of evidence variables picked at random. For Barley network, we select evidence variables as defined by (Kristensen & Rasmussen, 2002).

Table 1 summarizes the characteristic of each network. For each one, the table specifies the number of variables N, the induced width w^*, the size of loop cutset $|LC|$, the number of loop-cutset tuples $|\mathcal{D}(LC)|$, and the time needed to compute the exact posterior marginals

1. Times are extrapolated.

by bucket-tree elimination (exponential in the induced width w^*), and by cutset conditioning (exponential in the size of loop-cutset).

Computing the posterior marginals exactly is easy in Alarm network, cpcs54, and cpcs179 using either bucket elimination or cutset conditioning since they have small induced width and a small loop-cutset. We include those benchmarks as a proof of concept only. Several other networks, Barley, Munin3, and Munin4, also have small induced width and, hence, their exact posterior marginals can be obtained by bucket elimination. However, since ATB is linear in space, it should be compared against linear-space schemes such as cutset-conditioning. From this perspective, Barley, Munin3, and Munin4 are hard. For example, Barley network has only 48 variables, its induced width is $w^* = 7$, and exact inference by bucket elimination takes only 30 seconds. Its loop-cutset contains only 12 variables, but the number of loop-cutset tuples exceeds 2 million because some variables have large domain sizes (up to 67 values). Enumerating and computing all cutset tuples, at a rate of about 1000 tuples per second, would take over 22 hours. Similar considerations apply in case of Munin3 and Munin4 networks.

4.2.2 Measures of Performance

We measure the quality of the bounds via the average length of the interval between lower and upper bound:

$$\overline{I} = \frac{\sum_{X \in \mathcal{X}} \sum_{x \in \mathcal{D}(X)} (P^U(x|e) - P^L(x|e))}{\sum_{X \in \mathcal{X}} |\mathcal{D}(X)|}$$

We approximate posterior marginal as the midpoint between lower and upper bound in order to show whether the bounds are well-centered around the posterior marginal $P(x|e)$. Namely:

$$\hat{P}(x|e) = \frac{P^U_{\mathcal{A}}(x|e) + P^L_{\mathcal{A}}(x|e)}{2}$$

and then measure the average absolute error Δ with respect to that approximation:

$$\Delta = \frac{\sum_{X \in \mathcal{X}} \sum_{x \in \mathcal{D}(X)} |P(x|e) - \hat{P}(x|e)|}{\sum_{X \in \mathcal{X}} |\mathcal{D}(X)|}$$

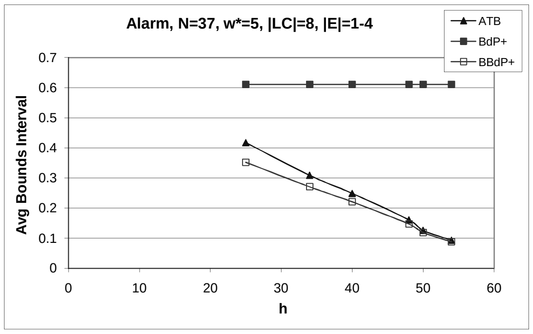
Finally, we report $\%P(e) = \frac{\sum_{i=1}^{h} P(x,c^i,e)}{P(e)} \times 100\%$ that was covered by the explored cutset tuples. Notably, in some benchmarks, a few thousand cutset tuples is enough to cover $> 90\%$ of $P(e)$.

4.2.3 Results

We summarize the results for each benchmark in a tabular format and charts. We highlight in bold face the first ATB data point where the average bounds interval is as good or better than $BdP+$. The charts show the convergence of the bounds interval length as a function of h and time.

For ATB and $BBdP+$ the maximum Markov CPT size was fixed at $k = 2^{10}$. For $BdP+$, we vary the maximum length k from $k = 2^{14}$ to $k = 2^{19}$. In the tables, we report the best result obtained $BdP+$ and its computation time so that it appears as constant with respect to h. However, when we plot accuracy against time, we include $BdP+$ bounds

Alarm, N=37, w^*=5, $\|LC\|$=8, $\|D_{LC}\|$=108, $\|E\|$=1-4										
		BdP+			*ATB*			*BBdP+*		
h	%P(e)	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*
25	86	0.61	0.21	*4.3*	**0.41**	0.12	**0.038**	0.35	0.10	*3.4*
34	93	0.61	0.21	*4.3*	0.31	0.09	*0.039*	0.27	0.08	*2.3*
40	96	0.61	0.21	*4.3*	0.25	0.07	*0.044*	0.22	0.06	*2.1*
48	97	0.61	0.21	*4.3*	0.24	0.05	*0.051*	0.15	0.04	*1.5*
50	98	0.61	0.21	*4.3*	0.16	0.04	*0.052*	0.12	0.03	*1.2*
54	99	0.61	0.21	*4.3*	0.13	0.03	*0.059*	0.09	0.02	*0.86*



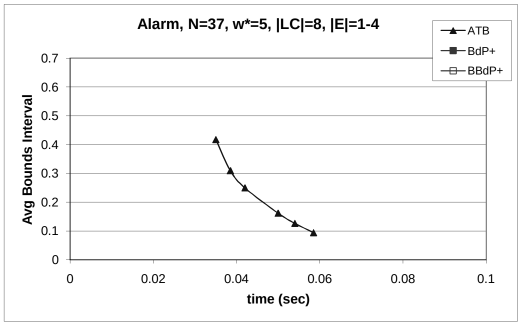


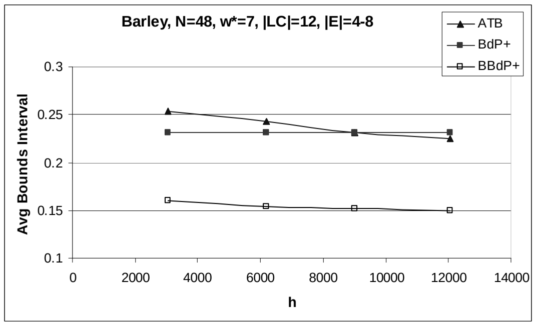
Figure 3: Results for Alarm network. The table reports the average bounds interval $\overline{I}$, average error Δ, computation time (in seconds), and percent of probability of evidence $P(e)$ covered by the fully-instantiated cutset tuples as a function of h. We highlight in bold face the first ATB data point where the average bounds interval is as good or better than $BdP+$. The charts show the convergence of the bounds interval length as a function of h and time.

obtained using smaller parameer k. Note that $BdP+$ only depends on k, not on h. The computation time of $BBdP+$ includes the ATB plus the $BdP+$ time.

For all benchmarks, the computed bounds interval length using BC remained close to 0.75 for Munin benchmarks and 0.95 for the others; hence, those results are omitted in the remainder of the section.

Alarm network. Figure 3 reports the results. As expected, the average bounds interval generated by ATB and $BBdP+$ decreases as the number of cutset tuples h increases, demonstrating the any-time property of ATB with respect to h. Given a fixed h, $BBdP+$ has a very significant overhead in time with respect to ATB (two orders of magnitude for values of h smaller than 54). However, the improvement in its accuracy is very moderate increasing with h. ATB outperforms $BdP+$, computing more accurate bounds starting with the first data point of $h = 25$. Observe that at that point, the mean interval $\overline{I}_{ATB}$ is 0.41, while the computation time is 0.038 seconds, an order of magnitude less than $BdP+$. The extreme efficiency of ATB in terms of time is clearly shown in the right chart, where only the line corresponding to ATB appears. Finally, note that the enumeration of less than the 25% of the total number of cutset tuples covers a percentage of 99% of the $P(e)$. This fact suggests that schemes based on cutset conditioning should be very suitable for this benchmark. However, recall that BC was only able to obtain a bound interval of 0.95,

Barley, N=48, w^*=7, $\lvert LC\rvert$=12, $\lvert D_{LC}\rvert > 2 \times 10^6$, $\lvert E\rvert$=4-8										
		BdP+			*ATB*			BBdP+		
h	%P(e)	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*
562	1	0.23	0.07	*1.7*	0.279	0.097	*9*	0.167	0.047	*10*
1394	3	0.23	0.07	*1.7*	0.263	0.090	*23*	0.162	0.045	*25*
2722	6	0.23	0.07	*1.7*	0.247	0.084	*43*	0.154	0.042	*45*
4429	14	0.23	0.07	*1.7*	0.235	0.079	*65*	0.147	0.040	*67*
6016	22	0.23	0.07	*1.7*	**0.230**	0.078	**86**	0.145	0.040	*88*
7950	33	0.23	0.07	*1.7*	0.228	0.077	*99*	0.145	0.040	*101*
9297	40	0.23	0.07	*1.7*	0.224	0.075	*111*	0.143	0.039	*113*
12478	52	0.23	0.07	*1.7*	0.219	0.073	*139*	0.142	0.038	*141*



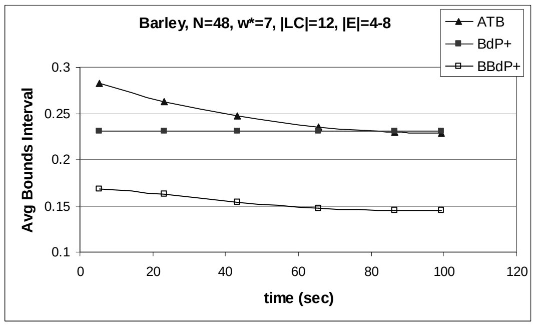


Figure 4: Results for Barley network. The table reports the average bounds interval $\overline{I}$, average error Δ, computation time (in seconds), and percent of probability of evidence $P(e)$ covered by the fully-instantiated cutset tuples as a function of h. We highlight in bold face the first ATB data point where the average bounds interval is as good or better than $BdP+$. The charts show the convergence of the bounds interval length as a function of h and time.

which demonstrates the value of correctly bounding the partially-instantiated cutset tuples.

Barley network. Figure 4 reports the results. ATB and $BBdP+$ improve as h increases. However, the improvement is quite moderate while very time consuming due to more uniform shape of the distribution $P(C|e)$ as reflected by the very small % of $P(e)$ covered by explored tuples (only 1% for 562 tuples and only 52% for 12478 tuples). For example, the average ATB (resp. $BBdP+$) bounds interval decreases from 0.279 (resp. 0.167), obtained in 9 (resp. 10) seconds, to 0.219 (resp. 0.142) obtained in 139 (resp. 141) seconds. Given a fixed h, $BBdP+$ substantially improves ATB bounds with little time overhead (2 seconds in general). Namely, in this benchmark, $BBdP+$ computation time is dominated by ATB computation time. Note that the computation time of the stand-alone $BdP+$ algorithm is less than 2 seconds. Within that time, $BdP+$ yields an average interval length of 0.23, while ATB and $BBdP+$ spend 86 and 10 seconds, respectively, to obtain the same quality bounds. However, the any-time behaviour of the latter algorithms allows them to improve with time, a very desirable characteristic when computing bounds.

Moreover, note that its overhead in time with respect to ATB is completely negligible.

cpcs54, N=54, $\lvert LC\rvert$=15, w^*=15, $\lvert D_{LC}\rvert$=32678, $\lvert E\rvert$=2-8										
		BdP+			*ATB*			BBdP+		
h	%P(e)	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*
513	10	0.35	0.02	*24*	0.51	0.027	*0.9*	0.34	0.011	*3.1*
1114	19	0.35	0.02	*24*	0.45	0.023	*1.5*	0.32	0.010	*3.1*
1581	29	0.35	0.02	*24*	0.42	0.021	*1.9*	0.31	0.009	*3.4*
1933	34	0.35	0.02	*24*	0.40	0.020	*2.2*	0.30	0.009	*3.6*
2290	40	0.35	0.02	*24*	0.38	0.019	*2.4*	0.30	0.008	*3.9*
2609	46	0.35	0.02	*24*	0.37	0.018	*2.7*	0.29	0.007	*4.0*
3219	53	0.35	0.02	*24*	**0.34**	0.016	**3.2**	0.27	0.007	*4.5*
3926	59	0.35	0.02	*24*	0.31	0.014	*3.8*	0.25	0.006	*5.2*
6199	63	0.35	0.02	*24*	0.23	0.010	*5.9*	0.20	0.006	*6.6*
7274	68	0.35	0.02	*24*	0.20	0.008	*6.9*	0.17	0.006	*7.3*

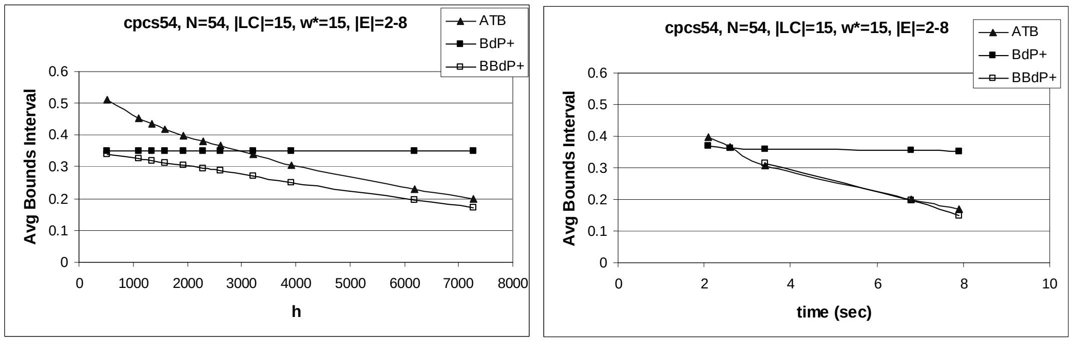


Figure 5: Results for cpcs54 network. The table reports the average bounds interval $\overline{I}$, average error Δ, computation time (in seconds), and percent of probability of evidence $P(e)$ covered by the fully-instantiated cutset tuples as a function of h. We highlight in bold face the first ATB data point where the average bounds interval is as good or better than *BdP+*. The charts show the convergence of the bounds interval length as a function of h and time.

CPCS networks. Figure 5 to Figure 7 show the results for cpcs54, cpcs179, cpcs360b and cpcs422b, respectively. The behaviour of the algorithms in all networks is very similar. As in the previous benchmarks, ATB and $BBdP+$ bounds interval decreases as h increases. Given a fixed h, $BBdP+$ computes slightly better bounds intervals than ATB in all networks but cpcs179, where the improvement is very notorious. For all networks, $BBdP+$ has overhead in time with respect to ATB. This overhead is constant for all values of h and for all networks except for cpcs54, for which the overhead decreases as h increases. ATB and $BBdP+$ outperform $BdP+$. Both algorithms compute the same bound interval length as $BdP+$, improving the computation time in one order of magnitude. Consider for example cpcs422b, a challenging instance for any inference scheme as it has relatively large indcued width and loop-cutset size. ATB outperforms $BdP+$ after 50 seconds starting with h = 1181, and $BBdP+$ outperforms $BdP+$ in 37 seconds starting with h = 253. ($BdP+$ convergence is shown in the plot, but only the best result is reported in the table).

(Larkin, 2003) reported bounds on cpcs360b and cpcs422b using AD algorithm. For the first network, AD achieved bounds interval length of 0.03 in 10 seconds. Within the same time, ATB computes an average bounds interval of $\approx$ 0.005. For cpcs422b, AD achieved

cpcs179, N=179, w^*=8, $\|LC\|$=8, $\|D_{LC}\|$=49152, $\|E\|$=12-24										
		BdP+			*ATB*			BBdP+		
h	%P(e)	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*
242	70	0.15	0.05	*20*	0.22	0.067	*4*	0.092	0.029	*11*
334	75	0.15	0.05	*20*	**0.12**	**0.033**	**6**	0.054	0.016	*13*
406	78	0.15	0.05	*20*	0.09	0.024	*7*	0.037	0.010	*13*
574	82	0.15	0.05	*20*	0.07	0.018	*9*	0.029	0.008	*15*
801	85	0.15	0.05	*20*	0.05	0.014	*10*	0.022	0.006	*17*
996	87	0.15	0.05	*20*	0.04	0.010	*12*	0.017	0.005	*18*
1285	88	0.15	0.05	*20*	0.03	0.006	*13*	0.012	0.003	*20*
1669	90	0.15	0.05	*20*	0.02	0.003	*16*	0.007	0.002	*22*

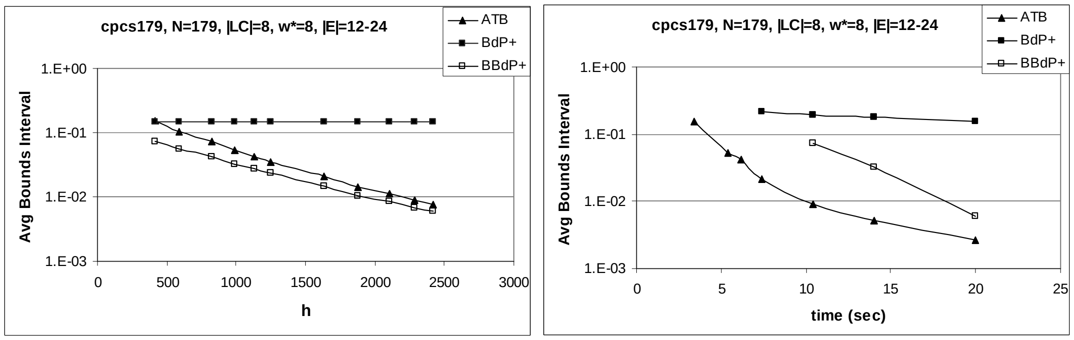
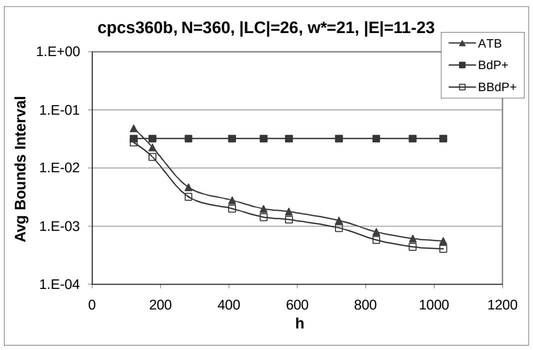


Figure 6: Results for cpcs179 network. The table reports the average bounds interval $\overline{I}$, average error Δ, computation time (in seconds), and percent of probability of evidence $P(e)$ covered by the fully-instantiated cutset tuples as a function of h. We highlight in bold face the first ATB data point where the average bounds interval is as good or better than $BdP+$. The charts show the convergence of the bounds interval length as a function of h and time.

bounds interval of 0.15, obtained in 30 seconds. Within the same time, ATB and $BBdP+$ obtain comparable results computing average bounds interval of 0.24 and 0.15, respectively. It is important to note that the comparison is not on the same instances since the evidence nodes are not the same. Larkin's code was not available for further experiments.

Munin networks. Figure 9 reports the results for both Munin networks. Let us first consider Munin3 network. Given a fixed h, ATB and $BBdP+$ compute almost identical bound intervals with $BBdP+$ having a noticiable time overhead. Note that the two curves in the chart showing convergence as a function of h are very close and hard to distinguish, while the points of $BBdP+$ in the chart showing convergence as a function of time are shifted to the right with respect to the ones of ATB. ATB is clearly superior to $BdP+$ both in accuracy and time. $BdP+$ computes bounds interval of 0.24 within 12 seconds, while ATB computes bounds interval of 0.050 in 8 seconds. In Munin4, given a fixed h, $BBdP+$ computes tighter bounds than ATB with some time overhead. However, the improvement decreases as h increases as shown by the convergence of both curves either as a function of h and time. Since the loop-cutset size is large, the convergence of ATB is relatively slow. $BdP+$ computes bounds interval of 0.23 within 15 seconds, while ATB and $BBdP+$ computes bounds of the same quality within 54 and 21 seconds, respectively.

cpcs360b, N=360, $w^* = 21$, $\|LC\| = 26$, $\|D_{LC}\|$=2^{26}, $\|E\|$=11-23										
		BdP+			*ATB*			BBdP+		
h	%P(e)	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*
121	83	0.027	0.009	*55*	0.0486	1.6E-2	5	0.0274	1.0E-2	*7*
282	92	0.027	0.009	*55*	**0.0046**	9.0E-4	**10**	0.0032	8.5E-4	*12*
501	96	0.027	0.009	*55*	0.0020	3.6E-4	*15*	0.0014	3.5E-4	*17*
722	97	0.027	0.009	*55*	0.0012	2.4E-4	*19*	0.0009	2.3E-4	*21*
938	98	0.027	0.009	*55*	0.0006	8.4E-5	*25*	0.0004	7.8E-5	*27*
1168	98	0.027	0.009	*55*	0.0005	7.5E-5	*29*	0.0004	6.9E-5	*31*
1388	99	0.027	0.009	*55*	0.0004	5.9E-5	*35*	0.0003	5.4E-5	*37*
1582	99	0.027	0.009	*55*	0.0003	5.3E-5	*39*	0.0002	4.8E-5	*41*
1757	99	0.027	0.009	*55*	0.0003	4.7E-5	*43*	0.0002	4.4E-5	*46*



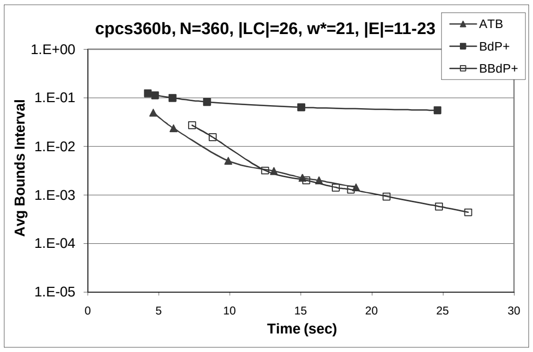


Figure 7: Results for cpcs360b. The table reports the average bounds interval $\overline{I}$, average error Δ, computation time (in seconds), and percent of probability of evidence $P(e)$ covered by the fully-instantiated cutset tuples as a function of h. We highlight in bold face the first ATB data point where the average bounds interval is as good or better than $BdP+$. The charts show the convergence of the bounds interval length as a function of h and time.

4.3 Bounding the Probability of Evidence

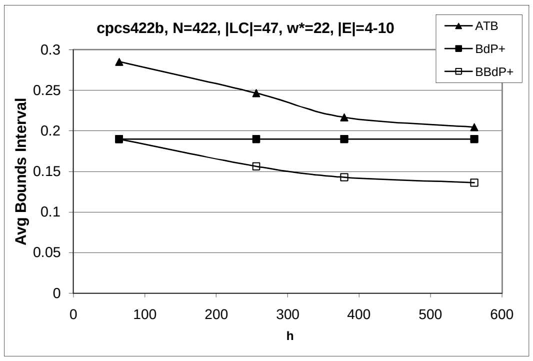
4.3.1 Algorithms and Benchmarks

We compare the performance of the following three algorithms that can bound $P(e)$: ATB, Mini-bucket Elimination (MBE) (Dechter & Rish, 2003), and Variable Elimination and Conditioning (VEC). For ATB, we test different configurations of the control parameters (h, k). Note that when $h = 0$, ATB is equivalent to its plug-in algorithm $\mathcal{A}$, which in our case is $ABdP+$.

MBE is a general bounding algorithm for graphical model problems. In particular, given a Bayesian network, MBE computes lower and upper bound on the probability of evidence. MBE has a control parameter z, that allows trading time and space for accuracy. As the value of the control parameter z increases, the algorithm computes tighter bounds using more time and space, which is exponential in z.

VEC is an algorithm that combines conditioning and variable elimination. It is based on the w-cutset conditioning scheme. Namely, the algorithm conditions or instantiates enough variables so that the remaining problem conditioned on the instantiated variables can be solved exactly using Bucket Elimination (Dechter, 1999). The exact probability of evi-

cpcs422b, N=422, $w^* = 22$, $\|LC\| = 47$, $\|D_{LC}\|=2^{47}$, $\|E\|$=4-10										
		BdP+			ATB			BBdP+		
h	%P(e)	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*
64	1.7	0.19	0.06	*120*	0.28	0.100	*21*	0.19	0.056	*23*
256	2.0	0.19	0.06	*120*	0.24	0.090	*26*	0.15	0.050	*35*
379	2.6	0.19	0.06	*120*	0.22	0.078	*32*	0.14	0.049	*41*
561	2.9	0.19	0.06	*120*	0.20	0.073	*36*	0.13	0.046	*46*
861	3.4	0.19	0.06	*120*	0.19	0.068	*43*	0.12	0.044	*54*
1181	4.5	0.19	0.06	*120*	**0.18**	0.064	**50**	0.12	0.041	*60*
1501	5.4	0.19	0.06	*120*	0.17	0.062	*56*	0.12	0.041	*65*
2427	8.0	0.19	0.06	*120*	0.16	0.058	*73*	0.12	0.039	*82*
3062	9.5	0.19	0.06	*120*	0.16	0.057	*83*	0.12	0.038	*92*
4598	12.2	0.19	0.06	*120*	0.16	0.053	*110*	0.11	0.036	*120*



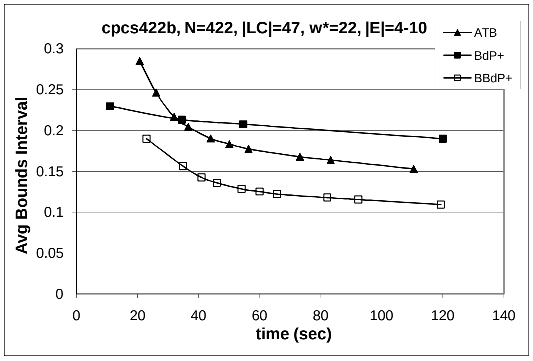


Figure 8: Results for cpcs422b. The table reports the average bounds interval $\overline{I}$, average error Δ, computation time (in seconds), and percent of probability of evidence $P(e)$ covered by the fully-instantiated cutset tuples as a function of h. We highlight in bold face the first ATB data point where the average bounds interval is as good or better than $BdP+$. The charts show the convergence of the bounds interval length as a function of h and time.

dence can be computed by summing over the exact solution output by bucket elimination for all possible instantiations of the w-cutset. When VEC is terminated before completion, it outputs a partial sum yielding a lower bound on the probability of evidence. The implementation of VEC is publicly available[1].

We tested ATB for bounding $P(e)$ on three different benchmarks: *Two-layer Noisy-Or*, *grids* and *coding* networks. All instances are included in the UAI08 evaluation[2].

In *two-layer noisy-or networks*, variables are organized in two layers where the ones in the second layer have 10 parents. Each probability table represents a noisy OR-function. Each parent variable y_j has a value $P_j \in [0..P_{noise}]$. The CPT for each variable in the second layer is then defined as, $P(x = 0|y_1, \ldots, y_P) = \prod_{y_j=1} P_j$ and $P(x = 1|y_1, \ldots, y_P) = 1 - P(x = 0|y_1, \ldots, y_P)$. We experiment with a class of problems called *bn2o* instances in the UAI08.

1. http://graphmod.ics.uci.edu/group/Software
2. http://graphmod.ics.uci.edu/uai08/Evaluation/Report

network	**N**	**w***	**|LC|**	**|$\mathcal{D}$(LC)|**	**|E|**	**Time(BE)**	**Time(LC)**	
bn2o-15-30-15	45	22	24	2^{24}	15	14.51	17.4 hrs	
bn2o-15-30-20	50	25	26	2^{26}	20	174.28	93.2 hrs	
bn2o-15-30-25	55	24	25	2^{25}	25	66.23	75.76 hrs	
Grids	10	16×16	22	116	2^{116}	1	27.59	$> 2 \times 10^{93}$ hrs[1]
Grids	10	20×20	29	185	2^{185}	1	out	$> 2 \times 10^{66}$ hrs[1]
Grids	26×26	40	325	2^{325}	1	out	$> 2 \times 10^{306}$ hrs[1]	
Grids	42×42	70	863	2^{863}	1	out	$> 2 \times 10^{844}$ hrs[1]	
coding	512	54-61	59-64	2^{59}-2^{64}	256	out		

Table 2: Complexity characteristics of the benchmarks from the UAI repository: N-number of nodes, w^*-induced width, $|LC|$-number of nodes in a loop-cutset, $|\mathcal{D}(LC)|$-loop-cutset state space size, Time(BE) is the exact computation time via bucket elimination, Time(LC) is the exact computation time via loop-cutset conditioning. The results are averaged over the set of network instances of each benchmark.

In *grid networks*, variables are organized as an $M \times M$ grid. We experiment with *grids2* instances, as they were called in UAI08, which are characterized by two parameters (M, D), where D is the percentage of determinism (i.e., the percentage of values in all CPTs assigned to either 0 or 1). For each parameter configuration, there are samples of size 10 generated by randomly assigning value 1 to one leaf node. In UAI08 competition, these instances were named D-M-I, where I is the instance number.

Coding networks can be represented as a four layer Bayesian network having M nodes in each layer. The second and third layer correspond to input information bits and parity check bits respectively. Each parity check bit represents an XOR function of input bits. Input and parity check nodes are binary while the output nodes are real-valued. We consider the BN_126 to BN_134 instances in the UAI08 evaluation. Each one has $M = 128$, 4 parents for each node and channel noise variance ($\sigma = 0.40$). These networks are very hard. Actually, exact results are not available.

Table 2 summarizes the characteristics of each network. For each one, the table specifies the number of variables N, the induced width w^*, the size of loop cutset $|LC|$, the number of loop-cutset tuples $|\mathcal{D}(LC)|$, and the time needed to compute the exact posterior marginals by bucket-tree elimination (exponential in the induced width w^*), and by cutset conditioning (exponential in the size of loop-cutset). An 'out' indicates that bucket-tree elimination is unfeasible in terms of memory demands. Note that the characteristics of grid networks only depend on its size, but not on the percentage of determinism; and the characteristics of all coding networks are the same.

For our purposes, we consider VEC as another exact algorithm to compute the exact $P(e)$ in the first and second benchmarks, and as a lower bounding technique for the third benchmark. We fix the control parameter z of MBE and the w-cutset of VEC so that the algorithms require less than 1.5GB of space.

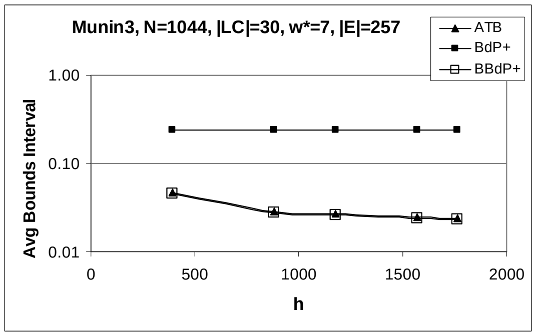
1. Times are extrapolated.

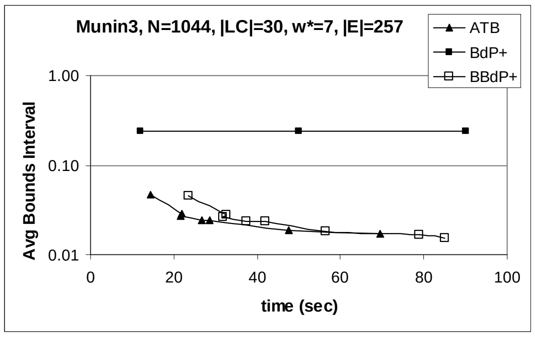
4.3.2 Results

We summarize the results for each benchmark in a tabular format. The tables report the bounds and computation time (in seconds) for each compared algorithm. For ATB, we report results by varying the values of the control parameters (h, k). In particular, we consider values of h in the range 4 to 200, and values of k in the set $\{2^{10}, 2^{12}, 2^{14}\}$. By doing so, we analyze the impact of each control parameter in the performance of the algorithm. Grey areas in the tables correspond to (h, k) configurations that cannot be compared due to computation time.

MUNIN3

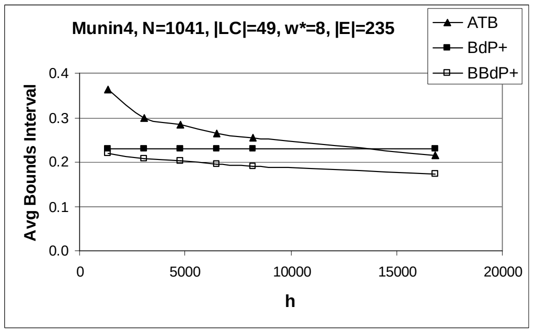
Munin3, N=1044, w^*=7, $\lvert LC\rvert$=30, $\lvert E\rvert$=257										
		BdP+			ATB			BBdP+		
h	%P(e)	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time(sec)*
196	64	0.24	0.1	*12*	**0.050**	0.020	**8**	0.048	0.020	*16*
441	72	0.24	0.1	*12*	0.030	0.011	*12*	0.029	0.012	*20*
882	78	0.24	0.1	*12*	0.025	0.009	*18*	0.025	0.009	*26*
1813	79	0.24	0.1	*12*	0.020	0.007	*32*	0.019	0.007	*40*
2695	80	0.24	0.1	*12*	0.018	0.006	*46*	0.017	0.007	*54*
2891	81	0.24	0.1	*12*	0.017	0.006	*49*	0.016	0.006	*57*
3185	82	0.24	0.1	*12*	0.014	0.005	*54*	0.014	0.005	*62*
3577	82	0.24	0.1	*12*	0.013	0.004	*68*	0.012	0.004	*76*
4312	83	0.24	0.1	*12*	0.011	0.004	*80*	0.010	0.004	*88*





MUNIN4

Munin4, N=1041, w^*=8, $\lvert LC\rvert$=49, $\lvert E\rvert$=235										
		BdP+			ATB			BBdP+		
h	%P(e)	$\overline{I}$	Δ	*time(sec)*	$\overline{I}$	Δ	*time*	$\overline{I}$	Δ	*time(sec)*
245	1	0.23	0.1	*15*	0.39	0.16	*14*	0.24	0.102	*21*
441	7	0.23	0.1	*15*	0.32	0.13	*17*	0.22	0.095	*24*
1029	11	0.23	0.1	*15*	0.28	0.12	*34*	0.21	0.089	*44*
2058	17	0.23	0.1	*15*	**0.25**	0.11	**54**	0.19	0.082	*65*
3087	20	0.23	0.1	*15*	0.22	0.11	*83*	0.18	0.077	*91*
5194	24	0.23	0.1	*15*	0.21	0.09	*134*	0.17	0.072	*145*



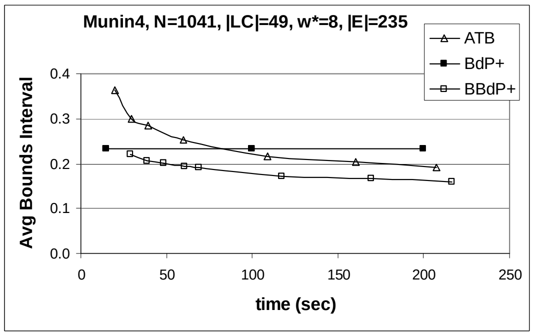


Figure 9: Results for munin3 and munin4. The tables report the average bounds interval $\overline{I}$, average error Δ, computation time (in seconds), and percent of probability of evidence $P(e)$ covered by the fully-instantiated cutset tuples as a function of h. We highlight in bold face the first ATB data point where the average bounds interval is as good or better than $BdP+$. The charts show the convergence of the bounds interval length as a function of h and time.

bn2o, $\lvert E \rvert = 15, 20$															
Inst.	P(e)	h	%P(e)	ATB($h, k = 2^{10}$)			ATB($h, k = 2^{12}$)			ATB($h, k = 2^{14}$)			MBE(z=18)		
				LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time
bn2o-30-15-150															
		4	3.93E-04	5.73E-10	5.32E-01	1.981	5.86E-10	4.81E-01	8.206	6.03E-10	4.36E-01	38.157			
1a	5.85E-05	50	0.1	1.23E-07	1.10E-01	31.948	1.24E-07	9.42E-02	128.74				9.14E-06	4.80E-04	2.5
		200	0.25	3.48E-07	5.70E-02	103.397									
		4	6.55E-03	3.06E-04	9.26E-01	2.013	4.28E-04	9.26E-01	8.314	4.95E-04	9.26E-01	38.313			
1b	0.565652	50	0.12	4.14E-03	8.64E-01	30.857	4.87E-03	8.63E-01	124.05				0.172774	1.42	2.46
		200	0.46	1.54E-02	8.26E-01	101.759									
		4	2.62E-03	2.00E-11	1.25E-01	2.012	2.01E-11	1.07E-01	8.3	2.05E-11	8.51E-02	38.205			
2a	4.02E-07	50	0.02	1.50E-10	1.00E-02	28.766	1.52E-10	8.85E-03	115.14				8.42E-10	2.11E-05	2.44
		200	0.32	1.73E-09	4.04E-03	88.81									
		4	8.06E-03	6.96E-03	7.99E-01	2.012	8.56E-03	7.98E-01	8.346	9.17E-03	7.98E-01	38.251			
2b	0.541111	50	0.21	5.46E-02	7.75E-01	29.296	6.08E-02	7.74E-01	114.73				0.0264676	1.83916	2.46
		200	1.11	1.12E-01	7.51E-01	89.935									
		4	2.16E-01	2.90E-07	1.68E-01	1.981	2.90E-07	1.46E-01	8.268	2.90E-07	1.38E-01	38.235			
3a	1.18E-04	50	1.04	1.66E-06	4.57E-02	26.457	1.65E-06	4.19E-02	103.35				4.35E-07	1.46E-03	2.60
		200	3.58	5.29E-06	2.67E-02	73.772									
		4	7.56E-02	1.10E-03	7.70E-01	1.997	1.14E-03	7.69E-01	8.299	1.20E-03	7.69E-01	38.204			
3b	0.188686	50	0.47	6.78E-03	6.27E-01	24.71	7.10E-03	6.26E-01	94.98				0.0308901	0.813878	2.45
		200	1.44	2.13E-02	5.36E-01	69.358									
bn2o-30-20-200															
		4	3.53E-03	5.40E-12	1.59E-02	3.416	5.40E-12	1.53E-02	15.756	5.40E-12	1.42E-02	67.158			
1a	1.36E-07	50	0.05	9.12E-11	1.82E-03	61.916	9.12E-11	1.58E-03	263.58				2.36E-15	3.32E-04	3.46
		200	1.88	2.80E-09	5.70E-04	195.454									
		4	1.20E-02	1.07E-04	7.32E-01	3.354	1.09E-04	7.32E-01	15.662	1.09E-04	7.32E-01	67.659			
1b	0.156537	50	0.14	3.33E-03	6.72E-01	63.617	3.55E-03	6.68E-01	279.05				9.84E-04	1.99457	3.48
		200	0.43	1.14E-02	5.99E-01	218.183									
		4	1.26E-02	3.78E-11	1.59E-02	3.447	3.79E-11	1.56E-02	15.709	3.80E-11	1.51E-02	70.497			
2a	2.24E-07	50	0.41	1.35E-09	3.27E-03	51.605	1.34E-09	3.11E-03	210.78				4.42E-15	8.03E-05	3.46
		200	1.41	4.53E-09	2.42E-03	169.027									
		4	2.00E-02	6.39E-03	8.33E-01	3.417	7.32E-03	8.32E-01	15.709	7.98E-03	8.32E-01	68.484			
2b	0.276951	50	0.43	3.03E-02	7.69E-01	50.669	3.26E-02	7.66E-01	197.48				2.31E-05	2.99178	3.48
		200	1.62	5.85E-02	6.91E-01	145.346									
		4	1.79E-03	8.29E-14	1.79E-03	3.416	8.30E-14	1.79E-03	15.756	8.30E-14	1.79E-03	68.235			
3a	2.37E-09	50	0.06	2.22E-12	1.07E-04	57.876	2.21E-12	1.06E-04	236.48				5.18E-13	1.74E-06	3.48
		200	0.09	3.56E-12	3.29E-05	198.183									
		4	2.17E-04	4.47E-05	9.72E-01	3.417	5.08E-05	9.72E-01	15.693	6.16E-05	9.72E-01	67.985			
3b	0.480395	50	0.11	5.41E-02	9.29E-01	64.476	5.86E-02	9.28E-01	276.59				5.26E-03	1.89902	3.48
		200	0.66	1.13E-01	8.81E-01	194.252									

Table 3: Results on bn2o networks. The table shows the LB and UB computed by ATB varying the number of cutset tuples h and the maximum length k of the conditional probability tables over the Markov boundary.

bn2o, $\lvert E\rvert = 25$															
Inst.	P(e)	h	%P(e)	ATB($h, k = 2^{10}$)			ATB($h, k = 2^{12}$)			ATB($h, k = 2^{14}$)			MBE(z=18)		
				LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time
bn2o-30-25-250															
		4	3.69E-04	1.30E-14	6.60E-02	5.788	1.30E-14	6.46E-02	22.153	1.30E-14	4.84E-02	98.828			
1a	2.96E-09	50	0.01	3.74E-13	3.28E-03	118.717	3.73E-13	2.79E-03	428.61				1.73E-16	3.11E-06	4.15
		200	0.06	2.04E-12	1.10E-03	395.51									
		4	1.61E-02	4.26E-04	8.11E-01	5.834	5.69E-04	8.10E-01	22.245	6.16E-04	8.10E-01	99.575			
1b	0.151829	50	0.22	4.65E-03	7.18E-01	120.494	6.66E-03	7.15E-01	436.99				1.42E-03	1.39152	4.52
		200	1.07	1.31E-02	6.48E-01	380.799									
		4	3.58E-04	1.77E-12	1.98E-01	5.804	1.77E-12	1.92E-01	22.23	1.77E-12	1.69E-01	99.138			
2a	2.44E-07	50	1.24E-03	5.70E-12	4.54E-02	111.712	5.70E-12	3.96E-02	398.23				1.77E-12	1.22E-05	4.20
		200	0.07	1.84E-10	2.17E-02	401.594									
		4	1.78E-02	5.28E-04	7.63E-01	5.787	5.94E-04	7.63E-01	22.199	6.34E-04	7.63E-01	99.201			
2b	0.308949	50	0.19	5.41E-03	7.36E-01	106.782	6.13E-03	7.36E-01	373.59				7.16E-03	1.75661	4.16
		200	0.65	1.43E-02	7.07E-01	367.179									
		4	6.02E-05	1.66E-16	1.14E-01	5.819	1.66E-16	1.13E-01	22.215	1.66E-16	8.09E-02	98.717			
3a	2.76E-10	50	0.01	4.27E-14	1.94E-02	119.434	4.26E-14	1.62E-02	427.33				1.27E-15	4.99E-07	4.06
		200	0.20	5.52E-13	7.10E-03	409.05									
		4	6.53E-03	1.00E-03	7.96E-01	5.85	1.15E-03	7.95E-01	22.23	1.25E-03	7.95E-01	97.983			
3b	0.468007	50	0.45	4.18E-02	7.73E-01	106.033	4.78E-02	7.71E-01	351.56				3.49E-03	2.75533	4.07
		200	1.52	8.47E-02	7.47E-01	336.556									

Table 4: Results on bn2o networks. The table shows the LB and UB computed by ATB varying the number of cutset tuples h and the maximum length k of the conditional probability tables over the Markov boundary.

Two-layer noisy-or networks. Tables 3 and 4 show the results. As expected, the quality of the bounds produced by ATB increase when the value of the control parameters (h, k) increases. We observe that the best bounds are obtained when fixing h to the highest value (i.e., 200) and k to the smallest value (i.e., 2^{10}). However, the increase in the value of h leads to higher computation times than when increasing the value of k. When taking time into account, comparing configurations with similar time (see $(h = 50, k = 2^{10})$ and $(h = 4, k = 2^{14})$, and $(h = 200, k = 2^{10})$ and $(h = 50, k = 2^{12})$, respectively), we observe that the configuration with the highest value of h and the smallest value of k outperforms the other ones.

When compared with MBE, there is no clear superior approach. The accuracy of the algorithms depends on whether we look at upper or lower bounds. When considering upper bounds, ATB outperforms MBE for all instances $1b$, $2b$ and $3b$. Note that for those instances, MBE computes worse upper bounds than the trivial one (i.e., greater than 1). However, for instances $1a$, $2a$ and $3a$, MBE computes tighter upper bounds than ATB. For lower bounds, in general ATB outperforms MBE for instances with 20 and 25 evidence variables, while MBE is more accurate for instances having 15 evidence variables. Regarding computation time, ATB is definitely slower than MBE.

Grid networks. Table 5 reports the results. The first thing to observe is that MBE computes completely uninformative bounds. In this case, the any-time behaviour of ATB is not effective either. The increase of the value of its control parameters (h, k) does not affect its accuracy. Since the Markov boundary in grid networks is relatively small, the smallest tested value of k is higher than its Markov boundary size which explains the independence on k. Another reason for its ineffectiveness may be the high percentage of determinism in these networks. It is known that sampling methods are inefficient in the presence of determinism. As a consequence, the percentage of probability mass accumulated in the h sampled tuples is not significant, which cancels the benefits of computing exact probability of evidence for that subset of tuples. Therefore, in such cases a more sophisticated sampling scheme should be used, as for example (Gogate & Dechter, 2007). Consequently, for these deterministic grids, ATB's performance is controlled totally by its bound propagation plugged-in algorithm.

Grids2, $\|E\| = 1$															
(M, D)	P(e)	h	%P(e)	ATB($k = 2^{10}$,h)			ATB($k = 2^{12}$,h)			ATB($k = 2^{14}$,h)			MBE		
				LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	UB	Time
$(16, 50)$	0.61724	4	1.57E-14	0.312738	0.828553	1.36	0.312738	0.828553	1.36	0.312738	0.828553	1.36			
		100	3.50E-11	0.312738	0.828553	56.77	0.312738	0.828553	56.80				0	5.13	16.24
		200	4.22E-11	0.312738	0.828552	110.75									
$(20, 50)$	0.4441	4	1.07E-24	0.176514	0.493885	4.84	0.176514	0.493885	4.82	0.176514	0.493885	4.82			
		100	1.57E-21	0.176514	0.493885	207.86	0.176514	0.493885	203.27				0	12411.60	38.86
		200	1.13E-20	0.176514	0.493885	411.82									
$(20, 75)$	0.4843	4	1.25E-09	0.210608	0.745416	2.60	0.210608	0.745416	2.59	0.210608	0.745416	2.61			
		100	2.40E-09	0.210608	0.745416	80.66	0.210608	0.745416	80.12				0	114654.95	38.80
		200	2.89E-09	0.210608	0.745416	156.13									
$(26, 75)$	0.6579	4	3.88E-19	0.0505681	0.933801	6.27	0.0505681	0.933801	6.27	0.0505681	0.933801	6.27			
		100	7.32E-19	0.0505681	0.933801	268.494	0.0505681	0.933801	270.15				0	> 1E+10	84.00
		200	1.55E-18	0.0505681	0.933801	534.46									
$(26, 90)$	0.8206	4	3.47E-08	0.185838	0.894316	2.29	0.185838	0.894316	2.29	0.185838	0.894316	2.29			
		100	3.41E-06	0.185839	0.894316	84.90	0.185839	0.894316	84.00				0	> 1E+10	87.01
		200	8.38E-06	0.185839	0.894316	163.53									
$(42, 90)$	0.4933	4	8.65E-29	0.0048445	0.917501	10.13	0.0048445	0.917501	10.13	0.0048445	0.917501	10.13			
		100	2.32E-25	0.0048445	0.917501	436.10	0.0048445	0.917501	438.50				0	> 1E+10	110.18
		200	3.48E-25	0.0048445	0.917501	865.64									

Table 5: Results on grid networks. The table shows the LB and UB computed by ATB varying the number of cutset tuples h and the maximum length k of the conditional probability tables over the Markov boundary.

coding, $\|E\| = 256$												
Inst.	h	ATB($k = 2^{10}$,h)			ATB($k = 2^{14}$,h)			MBE(z=22)			VEC	
		LB	UB	Time	LB	UB	Time	LB	UB	Time	LB	Time
	4	1.931E-76	1.520E-41	50.14	1.931E-76	1.520E-41	348.74					
BN_126	50	1.872E-69	2.543E-42	631.52				1.44E-139	1.45E-044	142.80	9.18 E-102	1900
	150	1.892E-58	1.256E-42	1441.75								
	4	5.304E-60	2.277E-43	54.85	5.304E-60	2.277E-43	398.66					
BN_127	50	1.217E-58	2.257E-44	426.26				1.62E-134	1.04E-045	164.44	5.30E-115	1900
	150	1.597E-58	1.901E-44	946.33								
	4	7.231E-54	1.635E-42	85.71	7.231E-54	1.635E-42	582.84					
BN_128	50	4.859E-48	7.190E-43	637.23				1.19E-144	5.10E-043	124.64	1.98E-112	1900
	150	4.859E-48	1.446E-43	1225.00								
	4	1.464E-72	8.155E-45	50.48	1.464E-72	8.155E-45	362.34					
BN_129	50	1.504E-64	2.114E-45	585.19				2.83E-139	4.76E-043	144.44	1.47E-115	1900
	150	8.490E-64	5.432E-46	1400.44								
	4	4.738E-65	2.870E-44	47.22	4.738E-65	2.870E-44	324.95					
BN_130	50	6.312E-63	2.966E-45	619.17				1.12E-132	1.99E-045	112.78	1.33E-96	1900
	150	3.665E-58	2.280E-45	1299.18								
	4	1.949E-60	1.250E-44	52.81	1.949E-60	1.250E-44	366.88					
BN_131	50	2.269E-54	3.684E-45	484.84				2.25E-141	3.21E-045	119.46	3.16E-102	1900
	150	2.269E-54	1.009E-45	1276.21								
	4	2.283E-79	6.326E-44	50.84	2.283E-79	6.326E-44	362.80					
BN_132	50	3.562E-67	1.032E-44	689.28				2.82E-134	2.28E-048	108.72	8.85E-111	1900
	150	1.462E-66	8.094E-45	1627.15								
	4	1.625E-56	2.748E-42	53.20	1.625E-56	2.748E-42	398.27					
BN_133	50	1.115E-54	2.372E-43	671.80				1.83E-136	4.08E-045	147.34	1.89E-109	1900
	150	2.319E-54	9.505E-44	1846.83								
	4	8.956E-63	1.803E-43	47.69	8.956E-63	1.803E-43	355.68					
BN_134	50	1.181E-62	8.625E-45	606.47				1.93E-148	3.99E-045	162.98	4.19E-111	1900
	150	6.060E-57	4.828E-45	1412.53								

Table 6: Results on coding networks. The table shows the LB and UB computed by ATB varying the number of cutset tuples h and the maximum length k of the conditional probability tables over the Markov boundary.

Coding networks. Table 6 shows the results. We do not report the percentage of $P(e)$ covered by the fully-instantiated cutset tuples because the exact $P(e)$ is not available. We set the time limit of VEC to 1900 seconds (i.e., the maximum computation time required by running ATB in these instances). We only report the results for $k = 2^{10}$ and $k = 2^{14}$ because the increase in the value of k was not effective and did not result in increased accuracy. In this case, the accuracy of ATB increases as the value of h increases. Comparing ATB with the other algorithms we have to distinguish between lower and upper bounds. Regarding lower bounds, ATB clearly outperforms MBE and VEC in all instances. Indeed, the lower bound computed by MBE and VEC is very loose. Regarding upper bounds, $ATB(h = 150, k = 2^{10})$ outperforms MBE in three instances (i.e., BN_128, BN_129 and BN_131). When taking time into account ATB only outperforms MBE in instance BN_129.

5. Related Work

There are three earlier approaches based on the same principle as ATB: Poole's algorithm (Poole, 1996), Bounded Conditioning (BC) (Horvitz et al., 1989) which we described before, and bounded recursive decomposition (Monti & Cooper, 1996). In all cases the computation of the bounds is divided into an exact inference over a subset of tuples and a bounding scheme over the sum of probabilities over the rest of tuples. Similar to ATB, Poole's scheme is based on a tree structure which is partially explored. However, his search tree corresponds to the state space of the whole network and hence, it is exponential in the network size. The tree structure used by our approach corresponds to the state space of the loop-cutset variables and hence, it is exponential only in the loop-cutset size. Also, Poole *updates* the bounding function when a tuple with probability 0 (i.e., a conflict) is discovered. As discussed in Section 2.2, BC is also based in the loop-cutset condition principle, but there are two main differences with respect to ATB: (i) the probability mass of the missing tuples is bounded via prior probabilities; and (ii) the upper bound expression is looser, which we proved. Bounded recursive decomposition (Monti & Cooper, 1996) uses Markov simulation (Pearl, 1988) to generate highly probable instantiations of the network nodes, similar to ATB, and bounds the missing elements with 0 and prior values in which it resembles Poole's algoarithm and bounded conditiong. Unlike ATB, bounded recursive decomposition requires instantiation of all variables in the network and relies on priors to guide the simulation. Our algorithm uses Gibbs sampling on a cutset which is likely to be more accurate at selecting high probability tuples in presence of evidence. ATB subsumes all three algorithms offering a unifying approach to bounding posteriors with any-time properties, able to improve its bounds by investing more time and exploring more cutset tuples.

There are a number of alternative approaches for computing bounds on marginals. (Poole, 1998) proposed *context-specific* bounds obtained from simplifying the conditional probability tables. The method performs a variant of bucket elimination where intermediate tables are collapsed by grouping some probability values together. However, since the method was validated only on a small car diagnosis network with 10 variables, it is hard to draw any conclusions. (Larkin, 2003) also obtains bounds by simplifying intermediate probability tables in the variable elimination order. He solves an optimization problem to find a table decomposition that minimizes the error. (Kearns & Saul, 1999, 1998) proposed a specialized *large deviation bounds* approach for layered networks, while (Mannino & Mookerjee, 2002) suggested an elaborate bounding scheme with non-linear objective functions. (Jaakkola & Jordan, 1999) proposed a variational method for computing lower and upper bounds on posterior marginals in Noisy-Or networks and evaluated its performance in the case of diagnostic QMR-DT network. More recent approaches include (Tatikonda, 2003; Taga & Mase, 2006; Ihler, 2007; Mooij & Kappen, 2008), aiming to bound the error of Belief Propagation marginals. The first two approaches are exponential in the size of the Markov boundary. The third approach is linear in the size of the network, but it is only formulated for pairwise interactions. Finally, the fourth algorithm is exponential in the number of domain values. Comparing and relating to these approaches is a subject for future work.

It is important to note that our approach offers an any-time framework for computing bounds where any of the above bounding algorithms can be used to bound joint probabilities for partially-instantiated tuples within ATB and will therefore improve the performance of any bounding scheme.

6. Summary and Conclusions

The task of bounding the likelihood of conditional or joint probability distribution is known to be very hard. While individual, single-principle methods start to emerge, it is clear that methods that pool together a variety of principles so that they all cooperate in this hard task, are needed. The current paper provides a framework that facilitates the collaboration principle.

It defines an any-time bounding scheme (ATB) that exploits the cutset-conditioning scheme to control the trade-off between time and accuracy. Given a cutset C (a subset of network variables $\mathcal{X}\backslash E$), it generates a subset of cutset tuples, computing exactly their probabilities, and then uses an off the shelve bounding scheme to bound the tails of the distribution $P(c, e)$ over the unexplored cutset tuples. We proved that our scheme is superior to the earlier approach of bounded conditioning (Horvitz et al., 1989).

We evaluated ATB empirically using a variant of bound propagation (Leisink & Kappen, 2003) called $ABdP+$, evaluated in (Bidyuk & Dechter, 2006b), as a plug-in algorithm. We demonstrated on a set of benchmarks that ATB's bounds converge as the number of computed cutset tuples increases, and that ATB always outperforms the $BdP+$ variant of BdP (Bidyuk & Dechter, 2006b) given enough explored cutset tuples. Moreover, in many cases, ATB computed more accurate bounds faster than $BdP+$. We demonstrated the power of our scheme for both computing posterior marginals as well as for the probability of evidence.

We also showed that the bounds computed by ATB can further boost the performance of bound propagation algorithm by using them as initial bounds yielding algorithm $BBdP+$. The experiments demonstrated that $BBdP+$ sometimes explore the time-accuracy trade-off more effectively than ATB alone. However, the outcome depends on both the network properties and the shape of the distribution $P(C|e)$. In particular, when the distribution $P(C|e)$ contains a small number of high probability tuples, as in the case of cpcs54, cpcs179, and cpcs360 networks, ATB is a clear winner as it converges very fast.

There are many options for improving the performance of ATB. We have looked at one possible instantiation of the plug-in algorithm $\mathcal{A}$. Other approximation algorithms can be tried offering different time/accuracy trade-offs. In particular, we plan to investigate the effectiveness of ATB using MBE as plug-in algorithm.

Appendix A. Analysis of Bounded Conditioning

Theorem **2.1** The interval between lower and upper bounds computed by bounded conditioning is lower bounded by the probability mass of prior distribution $P(C)$ of the unexplored cutset tuples: $\forall h, P^U_{BC}(x|e) - P^L_{BC}(x|e) \geq \sum_{i=h+1}^{M} P(c^i)$.

Proof.

$$\begin{aligned}
P^U_{BC}(x|e) - P^L_{BC}(x|e) &= \frac{\sum_{i=h+1}^{M} P(c^i)(\sum_{i=1}^{h} P(c^i,e) + \sum_{i=h+1}^{M} P(c^i))}{\sum_{i=1}^{h} P(c^i,e)} \\
&+ \frac{\sum_{i=1}^{h} P(x,c^i,e)}{\sum_{i=1}^{h} P(c^i,e)} - \frac{\sum_{i=1}^{h} P(x,c^i,e)}{\sum_{i=1}^{h} P(c^i,e) + \sum_{i=h+1}^{M} P(c^i)} \\
&\geq \frac{\sum_{i=h+1}^{M} P(c^i)(\sum_{i=1}^{h} P(c^i,e) + \sum_{i=h+1}^{M} P(c^i))}{\sum_{i=1}^{h} P(c^i,e)} \\
&= \sum_{i=h+1}^{M} P(c^i) + \frac{(\sum_{i=h+1}^{M} P(c^i))^2}{\sum_{i=1}^{h} P(c^i,e)} \geq \sum_{i=h+1}^{M} P(c^i)
\end{aligned}$$

□

Appendix B. Bounding posteriors of cutset nodes

So far, we only considered computation of posterior marginals for variable $X \in \mathcal{X} \backslash (C \cup E)$. Now we focus on computing bounds for a cutset node $C_k \in C$. Let $c'_k \in \mathcal{D}(C)$ be some value in domain of C_k. Then, we can compute exact posterior marginal $P(c_k|e)$ using Bayes formula:

$$P(c'_k|e) = \frac{P(c'_k,e)}{P(e)} = \frac{\sum_{i=1}^{M} \delta(c'_k,c^i)P(c^i,e)}{\sum_{i=1}^{M} P(c^i,e)} \tag{23}$$

where $\delta(c'_k,c^i)$ is a Dirac delta-function so that $\delta(c'_k,c^i) = 1$ iff $c^i_k = c'_k$ and $\delta(c'_k,c^i) = 0$ otherwise. To simplify notation, let $Z = C \backslash Z$. Let M_k denote the number of tuples in state-space of Z. Then we can re-write the numerator as:

$$\sum_{i=1}^{M} \delta(c'_k,c^i)P(c^i,e) = \sum_{i=1}^{M_k} P(c'_k,z^i,e)$$

and the denominator can be decomposed as:

$$\sum_{i=1}^{M} P(c^i,e) = \sum_{c_k \in \mathcal{D}(C_k)} \sum_{i=1}^{M_k} P(c'_k,z^i,e)$$

Then, we can re-write the expression for $P(c'_k|e)$ as follows:

$$P(c'_k|e) = \frac{\sum_{i=1}^{M_k} P(c'_k, z^i, e)}{\sum_{c_k \in \mathcal{D}(C_k)} \sum_{i=1}^{M_k} P(c_k, z^i, e)} \tag{24}$$

Let h_{c_k} be the number of full cutset tuples where $c^i_k = c_k$. Then, we can decompose the numerator in Eq. (24) as follows:

$$\sum_{i=1}^{M_k} P(c'_k, z^i, e) = \sum_{i=1}^{h_{c'_k}} P(c'_k, z^i, e) + \sum_{i=h_{c'_k}+1}^{M_k} P(c'_k, z^i, e)$$

Similarly, we can decompose the sums in the denominator:

$$\sum_{c_k \in \mathcal{D}(C_k)} \sum_{i=1}^{M_k} P(c_k, z^i, e) = \sum_{c_k \in \mathcal{D}(C_k)} \sum_{i=1}^{h_{c_k}} P(c_k, z^i, e) + \sum_{c_k \in \mathcal{D}(C_k)} \sum_{i=h_{c_k}+1}^{M_k} P(c_k, z^i, e)$$

After decomposition, the Eq. (24) takes on the form:

$$P(c'_k|e) = \frac{\sum_{i=1}^{h_{c'_k}} P(c'_k, z^i, e) + \sum_{i=h_{c'_k}+1}^{M_k} P(c'_k, z^i, e)}{\sum_{c_k \in \mathcal{D}(C_k)} \sum_{i=1}^{h_{c_k}} P(c_k, z^i, e) + \sum_{c_k \in \mathcal{D}(C_k)} \sum_{i=h_{c_k}+1}^{M_k} P(c_k, z^i, e)} \tag{25}$$

Now, for conciseness, we can group together all fully instantiated tuples in the denominator:

$$\sum_{c_k \in \mathcal{D}(C_k)} \sum_{i=1}^{h_{c_k}} P(c_k, z^i, e) = \sum_{i=1}^{h} P(c^i, e)$$

Then, Eq. (25) transforms into:

$$P(c'_k|e) = \frac{\sum_{i=1}^{h_{c'_k}} P(c'_k, z^i, e) + \sum_{i=h_{c'_k}+1}^{M_k} P(c'_k, z^i, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{i=h_{c_k}+1}^{M_k} \sum_{c_k \in \mathcal{D}(C_k)} P(c_k, z^i, e)} \tag{26}$$

Now, we can replace each sum $\sum_{i=h_{c'_k}+1}^{M_k}$ over unexplored cutset tuples with a sum over the partially-instantiated cutset tuples. Denoting as $M'_{c_k} = M_k - h_{c_k} + 1$ the number of partially instantiated cutset tuples for $C_k = c_k$, we obtain:

$$P(c'_k|e) = \frac{\sum_{i=1}^{h_{c'_k}} P(c'_k, z^i, e) + \sum_{j=1}^{M'_{c'_k}} P(c'_k, z^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'_{c_k}} \sum_{c_k \in \mathcal{D}(C_k)} P(c_k, z^j_{1:q_j}, e)} \tag{27}$$

In order to obtain lower and upper bounds formulation, we will separate the sum of joint probabilities $P(c'_k, z^j_{1:q_j}, e)$ where $C_k = c'_k$ from the rest:

$$P(c'_k|e) = \frac{\sum_{i=1}^{h_{c'_k}} P(c'_k, z^i, e) + \sum_{j=1}^{M'_{c'_k}} P(c'_k, z^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'_{c'_k}} P(c'_k, z^j_{1:q_j}, e) + \sum_{j=1}^{M'_{c_k}} \sum_{c_k \neq c'_k} P(c_k, z^j_{1:q_j}, e)} \tag{28}$$

In the expression above, probabilities $P(c_k, z^i, e)$ and $P(c^i, e)$ are computed exactly since they correspond to full cutset instantiations. Probabilities $P(c_k, z^i_{1:q_i}, e)$, however, will be bounded since only partial cutset is observed. Observing that both numerator and denominator have component $P(c'_k, z^i_{1:q_i}, e)$ and replacing it with an upper bound $P^U(c'_k, z^i_{1:q_i}, e)$ in both numerator and denominator, we will obtain an upper bound on $P(c'_k|e)$ due to Lemma 3.2:

$$P(c'_k|e) \leq \frac{\sum_{i=1}^{h_{c'_k}} P(c'_k, z^i, e) + \sum_{j=1}^{M'_{c'_k}} P^U_{\mathcal{A}}(c'_k, z^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'_{c'_k}} P^U_{\mathcal{A}}(c'_k, z^j_{1:q_j}, e) + \sum_{j=1}^{M'_{c_k}} \sum_{c_k \neq c'_k} P(c_k, z^j_{1:q_j}, e)} \tag{29}$$

Finally, replacing $P(c_k, z^j_{1:q_j}, e)$, $c_k \neq c'_k$, with a lower bound (also increasing fraction value), we obtain:

$$P(c'_k|e) \leq \frac{\sum_{i=1}^{h_{c'_k}} P(c'_k, z^i, e) + \sum_{j=1}^{M'_{c_k}} P^U_{\mathcal{A}}(c'_k, z^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'_{c_k}} P^U_{\mathcal{A}}(c'_k, z^j_{1:q_j}, e) + \sum_{j=1}^{M'_{c_k}} \sum_{c_k \neq c'_k} P^L_{\mathcal{A}}(c_k, z^j_{1:q_j}, e)} = P^U_c \tag{30}$$

The lower bound derivation is similar. We start with Eq. (28) and replace $P(c'_k, z^i_{1:q_i}, e)$ in numerator and denominator with a lower bound. Lemma 3.2 guarantees that the resulting fraction will be a lower bound on $P(c'_k|e)$:

$$P(c'_k|e) \geq \frac{\sum_{i=1}^{h_{c'_k}} P(c'_k, z^i, e) + \sum_{j=1}^{M'_{c'_k}} P^L_{\mathcal{A}}(c'_k, z^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'_{c'_k}} P^L_{\mathcal{A}}(c'_k, z^j_{1:q_j}, e) + \sum_{j=1}^{M'_{c_k}} \sum_{c_k \neq c'_k} P(c_k, z^j_{1:q_j}, e)} \tag{31}$$

Finally, replacing $\sum_{c_k \neq c'_k} P(c_k, z^j_{1:q_j}, e)$ in Eq. (31) with an upper bound, we obtain the lower bound P^L_c:

$$P(c'_k|e) \geq \frac{\sum_{i=1}^{h_{c'_k}} P(c'_k, z^i, e) + \sum_{j=1}^{M'_{c'_k}} P^L_{\mathcal{A}}(c'_k, z^j_{1:q_j}, e)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=1}^{M'_{c'_k}} P^L_{\mathcal{A}}(c'_k, z^j_{1:q_j}, e) + \sum_{j=1}^{M'_{c_k}} UB[\sum_{c_k \neq c'_k} P(c_k, z^j_{1:q_j}, e)]} = P^L_c \tag{32}$$

where

$$UB[\sum_{c_k \neq c'_k} P(c_k, z^j_{1:q_j}, e)] = \min \left\{ \begin{array}{l} \sum_{c_k \neq c'_k} P^U_{\mathcal{A}}(c_k, z^j_{1:q_j}, e) \\ P^U_{\mathcal{A}}(z^j_{1:q_j}, e) \end{array} \right.$$

The lower bounds P^L_c are respective cutset equivalents of the lower bound P^L obtained in Eq. (15).

With respect to computing bounds on $P(c'_k, z_{1:q}, e)$ in Eq. (30) and (32) in practice, we distinguish two cases. We demonstrate them on the example of upper bound.

In the first case, each partially instantiated tuple $c_{1:q}$ that includes node C_k, namely $k \leq q$, can be decomposed as $c_{1:q} = z_{1:q} \bigcup c'_k$ so that:

$$P^U(c'_k, z_{1:q}, e) = P^U(c_{1:q}, e)$$

The second case concerns the partially instantiated tuples $c_{1:q}$ that do not include node C_k, namely $k > q$. In that case, we compute upper bound by decomposing:

$$P^U(c_k', z_{1:q}, e) = P^U(c_k|c_{1:q})P^U(c_{1:q}, e)$$

Appendix C. ATB Properties

Theorem **3.2** *ATB* bounds interval length is upper bounded by a monotonic non-increasing function of h:

$$P^U_{\mathcal{A}}(x|e) - P^L_{\mathcal{A}}(x|e) \leq \frac{\sum_{j=h+1}^{M} P(c^j)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=h+1}^{M} P(c^j)} \triangleq I_h$$

Proof. The upper bound on the bounds interval follows from the fact that, $P^U_{\mathcal{A}}(x|e) - P^L_{\mathcal{A}}(x|e) \leq P^U_{BF}(x|e) - P^L_{BF}(x|e)$ and from the definitions of brute force lower and upper bounds given by Eq. (21) and (22). We only need to prove that the upper bound is monotonously non-increasing as a function of h.

$$I_{h-1} = \frac{\sum_{j=h}^{M} P(c^j)}{\sum_{i=1}^{h-1} P(c^i, e) + \sum_{j=h}^{M} P(c^j)} = \frac{P(c^h) + \sum_{j=h+1}^{M} P(c^j)}{\sum_{i=1}^{h-1} P(c^i, e) + P(c^h) + \sum_{j=h+1}^{M} P(c^j)}$$

Since $P(c^h) \geq P(c^h, e)$, then replacing $P(c^h)$ with $P(c^h, e)$ and applying Lemma 3.1, yields:

$$\begin{aligned} I_{h-1} &\geq \frac{P(c^h, e) + \sum_{j=h+1}^{M} P(c^j)}{\sum_{i=1}^{h-1} P(c^i, e) + P(c^h, e) + \sum_{j=h+1}^{M} P(c^j)} = \frac{P(c^h, e) + \sum_{j=h+1}^{M} P(c^j)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=h+1}^{M} P(c^j)} \\ &\geq \frac{\sum_{j=h+1}^{M} P(c^j)}{\sum_{i=1}^{h} P(c^i, e) + \sum_{j=h+1}^{M} P(c^j)} = I_h \end{aligned}$$

Thus, $I_{h-1} \geq I_h$. $\Box$

References

Abdelbar, A. M., & Hedetniemi, S. M. (1998). Approximating MAPs for belief networks is NP-hard and other theorems. *Artificial Intelligence*, *102*, 21–38.

Andreassen, S., Jensen, F., Andersen, S., Falck, B., Kjaerulff, U., Woldbye, M., Srensen, A., Rosenfalck, A., & Jensen, F. (1990). Munin - an expert EMG assistant. In Desmedt, J. E. (Ed.), *Computer-Aided Electromyography and Expert Systems, ch. 21*, chap. 21. Elsevier Science Publishers, Amsterdam.

Becker, A., & Geiger, D. (1996). A sufficiently fast algorithm for finding close to optimal junction trees. In *Proceedings of Uncertainty in AI*, pp. 81–89.

Beinlich, I., Suermondt, G., Chavez, R., & Cooper, G. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Second European Conference on AI and Medicine, Berlin, 1989.* Springer–Verlag.

Bidyuk, B. (2006). *Exploiting Graph Cutsets for Sampling-Based Approximations in Bayesian Networks. Ph.D. Thesis.* Ph.D. thesis, University of California, Irvine.

Bidyuk, B., & Dechter, R. (2003a). Cycle-cutset sampling for Bayesian networks. In *Proceedings of the 16th Canadian Conference on Artificial Intelligence*, pp. 297–312.

Bidyuk, B., & Dechter, R. (2003b). Empirical study of w-cutset sampling for Bayesian networks. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 37–46. Morgan Kaufmann.

Bidyuk, B., & Dechter, R. (2006a). An anytime scheme for bounding posterior beliefs. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pp. 1095–1100.

Bidyuk, B., & Dechter, R. (2006b). Improving bound propagation. In *European Conf. on AI (ECAI)*, pp. 342–346.

Bidyuk, B., & Dechter, R. (2007). Cutset sampling for bayesian networks. *JAIR*, *28*, 1–48.

Cooper, G. (1990). The computational complexity of probabilistic inferences. *Artificial Intelligence*, *42*, 393–405.

Dagum, P., & 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.

Dechter, R., & Rish, I. (2003). Mini-buckets: A general scheme for bounded inference. *Journal of the ACM*, *50*, 107–153.

Gogate, V., & Dechter, R. (2007). Samplesearch: A scheme that searches for consistent samples. In *In 11th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Horvitz, E., Suermondt, H., & Cooper, G. (1989). Bounded conditioning: Flexible inference for decisions under scarce resources. In *Workshop on Uncertainty in Artificial Intelligence*, pp. 181–193.

Ihler, A. (2007). Accuracy bounds for belief propagation. In *Proceedings of UAI 2007.*

Jaakkola, T. S., & Jordan, M. I. (1999). Variational probabilistic inference and the qmr-dt network. *Journal of Artificial Intelligence Research*, *10*, 291–322.

Kask, K., & Dechter, R. (1999). Stochastic local search for Bayesian networks. In Heckerman, D., & Whittaker, J. (Eds.), *Workshop on AI and Statistics*, pp. 113–122. Morgan Kaufmann.

Kearns, M., & Saul, L. (1998). Large deviation methods for approximate probabilistic inference, with rates of convergence. In *Proc. of Uncertainty in AI*, pp. 311–319. Morgan Kaufmann.

Kearns, M., & Saul, L. (1999). Inference in multilayer networks via large deviation bounds. *Advances in Neural Information Processing Systems*, *11*, 260–266.

Kristensen, K., & Rasmussen, I. (2002). The use of a Bayesian network in the design of a decision support system for growing malting Barley without use of pesticides. *Computers and Electronics in Agriculture*, *33*, 197–217.

Larkin, D. (2003). Approximate decomposition: A method for bounding and estimating probabilistic and deterministic queries. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 346–353.

Leisink, M. A. R., & Kappen, H. J. (2003). Bound propagation. *Journal of Artificial Intelligence Research*, *19*, 139–154.

Mannino, M. V., & Mookerjee, V. S. (2002). Probability bounds for goal directed queries in Bayesian networks. *IEEE Transactions on Knowledge and Data Engineering*, *14*(5), 1196–1200.

Monti, S., & Cooper, G. (1996). Bounded recursive decomposition: a search-based method for belief network inference under limited resources. *International Journal of Approximate Reasoning*, *15*, 49–75.

Mooij, J. M., & Kappen, H. J. (2008). Bounds on marginal probability distributions. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 1105–1112.

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.

Poole, D. (1998). Context-specific approximation in probabilistic inference. In *Proc. of Uncertainty in Artificial Intelligence (UAI)*, pp. 447–454.

Pradhan, M., Provan, G., Middleton, B., & Henrion, M. (1994). Knowledge engineering for large belief networks. In *Proceedings of 10th Conference on Uncertainty in Artificial Intelligence, Seattle, WA*, pp. 484–490.

Taga, N., & Mase, S. (2006). Error bounds between marginal probabilities and beliefs of loopy belief propagation algorithm. In *MICAI 2006: Advances in Artificial Intelligence, 5th Mexican International Conference on Artificial Intelligence, Apizaco, Mexico, November 13-17, 2006, Proceedings*, pp. 186–196.

Tatikonda, S. C. (2003). Convergence of the sum-product algorithm. In *Proceedings 2003 IEEE Information Theory Workshop*, pp. 222–225.