Improving Bound Propagation

Bozhena Bidyuk[1] and **Rina Dechter**[2]

Abstract. This paper extends previously proposed bound propagation algorithm [11] for computing lower and upper bounds on posterior marginals in Bayesian networks. We improve the bound propagation scheme by taking advantage of the directionality in Bayesian networks and applying the notion of relevant subnetwork. We also propose an approximation scheme for the linear optimization subproblems. We demonstrate empirically that while the resulting bounds loose some precision, we achieve 10-100 times speedup compared to original bound propagation using a simplex solver.

1 Introduction

Using Bayesian networks to model the problems arising in practical applications requires answering queries regarding the probability of an outcome given observations, namely, computing posterior marginals. Computing exact posteriors is NP-hard. Computing bounds on posterior marginals is a special case of approximating posterior marginals with the desired degree of precision which is also NP-hard [6, 2]. Previously proposed methods include bounded conditioning [7], search with conflict-counting [14], "context-specific" bounds [15], *large deviation bounds* for layered networks [8, 9], bounds for goal directed queries [12], and a scheme bounding exact computation in bucket elimination [10]. None of the methods dominates the rest as they offer different accuracy and speed trade-offs.

We focus on a recently proposed **bound propagation** (BdP) algorithm [11], applicable to both Bayesian networks and Markov random fields. The algorithm iteratively solves a linear optimization problem for each variable such that the minimum and maximum of the objective function correspond to lower and upper bounds on the variable's posterior marginals. The lower and upper bounds are initialized to 0 and 1 respectively. When algorithm solves minimization/maximization LP problem, the lower and upper bounds are updated. The bounds are updated repeatedly until they converge. The performance of the scheme was demonstrated in [11] on the example of Alarm network, Ising grid, and regular bi-partite graphs.

The performance of bound propagation is tied to the network structure, namely, the Markov blanket of each variable. Its computation time increases exponentially with Markov blanket size. Hence, it is well suited for problems with regular network structure, having large induced width but bounded Markov blanket size (e.g., grids).

In our work here, we improve the performance of BdP by exploiting the global network properties, namely, restricting the Markov blanket of a node to its relevant subnetwork, resulting in substantial gains in accuracy and speed. Further, since bound propagation yields linear optimization subproblems that fall into a category of fractional packing and covering problems, known to be hard for simplex method, we propose a fast approximate algorithm for solving the LP problems without using simplex method. Although many schemes have been developed for approximately solving linear programming problems [5], they usually solve packing-only or covering-only problems and do not include the additional constraints present in bound propagation. Hence, we propose our own solution obtained by relaxing the original problem until it can be solved exactly using a greedy algorithm. We investigate empirically the trade-offs in bounds interval length and time.

[1] Donald Bren School of Information and Computer Science, University Of California Irvine, CA 92697-3425, USA, email: bbidyuk@ics.uci.edu
[2] email: dechter@ics.uci.edu

2 Background

2.1 Belief Networks

We use upper case letters without subscripts, such as $\overline{X}$, to denote sets of variables and an upper case letter with a subscript, such as X_i, to denote a single variable. We use a lower case letter with a subscript, such as x_i, to denote an instantiated variable. $\mathcal{D}(X_i)$ denotes the domain of the variable X_i. We will use $\overline{x}$ to denote an instantiation of a set of variables $\overline{x} = \{x_1, ..., x_i, ...\}$ and $\overline{x}_{-i} = \overline{x} \backslash x_i$ to denote $\overline{x}$ with element x_i removed.

Definition 1 (belief networks) *Let $\overline{X}$=$\{X_1, ..., X_n\}$ be a set of random variables over multi-valued domains $\mathcal{D}(X_1), ..., \mathcal{D}(X_n)$. A* belief network *(BN) is a pair (G, P) where G is a directed acyclic graph on $\overline{X}$ and P=$\{P(X_i|pa_i)\}$ is the set of conditional probability tables (CPTs) associated with each X_i. The parents of a variable X_i together with its children and parents of its children form a* Markov blanket *ma_i of node X_i. A network is* **singly-connected** *(also called a* **poly-tree***), if its underlying undirected graph has no cycles. The queries over singly-connected network can be processed in time linear in the size of the network [13].*

2.2 Bound Propagation

Bound propagation (BdP) [11] is an iterative algorithm that utilizes the local network structure to formulate a linear optimization problem. For each variable $X_i \in \overline{X}$ the minimum and maximum of the objective function correspond to the upper and lower bounds on posterior marginals $P(x_i|e)$. Let $\overline{Y}$=$\{Y_1, ..., Y_k\}$ denote the Markov blanket of node X_i. The idea is to compute posterior marginals using decomposition:

$$P(x_i|e) = \sum_{\overline{y}} P(x_i|\overline{y}, e)P(\overline{y}|e) \quad (1)$$

Given its Markov blanket, the probability distribution of X_i is independent from the rest of the variables in the network so that $P(x_i|\overline{y}, e)$=$P(x_i|\overline{y})$. Hence, we can rewrite Eq. (1) as:

$$P(x_i|e) = \sum_{\overline{y}} P(x_i|\overline{y})P(\overline{y}|e) \quad (2)$$

Here, $P(x_i|\overline{y})$ is an entry in the probability table of X_i conditioned on the instantiation of variables in its Markov blanket which can be computed as follows [13]:

$$P(x_i|pa_i, \cup_j(ch_j \cup pa_j)) = \alpha P(x_i|pa_i) \prod_j P(ch_j|pa_j)$$

where α is a normalization constant. The probabilities $P(\overline{y}|e)$ are unknown, but we know that the sum of all probabilities equals 1:

$$\sum_{y_1,...,y_k} P(y_1, ..., y_k|e) = 1 \quad (3)$$

Further, $\forall Y_j \in \overline{Y}$, $\forall y_j \in \mathcal{D}(Y_j)$, $\sum_{\overline{y}_j, Y_j=y_j} P(\overline{y}|e) = P(y_j|e)$. Denoting arbitrary lower and upper bounds on $P(y_j|e)$ by $P^L(y_j|e)$ and $P^U(y_j|e)$ respectively, we can write:

$$P^L(y_j|e) \leq \sum_{y \setminus y_j, Y_j=y_j} P(y_1, ..., y_k|e) \leq P^U(y_j|e) \quad (4)$$

Hence, for each variable X_i, we have a linear optimization problem with the objective function $P(x_i|e)$ defined in Eq. (2) and constraints defined in Eq. (3) (sum-to-1 constraint) and Eq. (4). For each $\overline{y} \in \mathcal{D}(\overline{Y})$, the $P(x_i|\overline{y})$ is an objective function coefficient and $P(\overline{y}|e)$ is a variable. The number of variables is exponential in the size of the Markov blanket. The number of constraints equals $1 + \sum_j |D(Y_j)|$ since there are $|\mathcal{D}(Y_j)|$ constraints for each Y_j.

Example 1 *Let ma_i=$\{A, B\}$ where $\mathcal{D}(A)$=$\{0, 1\}$ and $\mathcal{D}(B)$=$\{0, 1, 2\}$. Let $P(x_i|A, B)$ be defined as follows: $P(x_i|0, 0)$=0.1, $P(x_i|0, 1)$=0.2, $P(x_i|0, 2)$=0.3, $P(x_i|1, 0)$=0.4, $P(x_i|1, 1)$=0.5, and $P(x_i|1, 2)$=0.6. Then, denoting P_{pq}=$P(x_i|p, q)$, the objective function of the LP problem for x_i can be defined as follows:*

$$f = 0.1P_{00} + 0.2P_{01} + 0.3P_{02} + 0.4P_{10} + 0.5P_{11} + 0.5P_{12}$$

s.t. $P_{00} + P_{01} + P_{02} + P_{10} + P_{11} + P_{12} = 1$

$$\begin{aligned}
P^L(a=0|e) &\leq P_{00} + P_{01} + P_{02} \leq P^U(a=0|e) \\
P^L(a=1|e) &\leq P_{10} + P_{11} + P_{12} \leq P^U(a=1|e) \\
P^L(b=0|e) &\leq P_{00} + P_{10} \leq P^U(b=0|e) \\
P^L(b=1|e) &\leq P_{01} + P_{11} \leq P^U(b=1|e) \\
P^L(b=2|e) &\leq P_{02} + P_{12} \leq P^U(b=2|e)
\end{aligned}$$

Initializing all bounds $P^L(X_i|e)$ and $P^U(X_i|e)$ to 0 and 1, the algorithm solves the linear minimization and maximization problems for each value $x_i \in \mathcal{D}(X_i)$ of each variable $X_i \in \overline{X}$ and updates the bounds.With every iteration, the bounds get closer to the posterior marginals or do not change. The process is iterated until convergence.The variable processing order does not affect the results although it may affect the number of iterations needed to converge.

Since the number of variables in the LP problems grows exponentially with the size of the Markov blanket, algorithm BdP is feasible only for networks having bounded Markov blanket size e.g. Ising grid an regular two-layer networks explored in [11]. Applied to Alarm network without evidence, BdP obtained small bounds interval for several nodes but could not obtain good bounds for root nodes 11, 12, 13, 14, although their relative subnetworks are singly-connected and, hence, the posteriors equal the priors. The latter shows the weakness of BdP in that it may not compute tight bounds even in a singly-connected network.

3 Improving BdP Performance

In this section we describe how to improve the performance of BdP by exploiting global network structure and how to obtain quick bounds using a simple greedy algorithm instead of a simplex solver.

3.1 Exploiting Network Structure

The performance of bound propagation can be improved also by identifying the irrelevant child nodes and restricting the Markov blanket of X_i to its relevant subnetwork.

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

Removing irrelevant nodes (and their parents) from Markov blanket whenever possible yields a smaller effective Markov blanket and, thus, a smaller LP problem with fewer variables. Also, if the relevant subnetwork of node X_i is singly-connected then its posteriors should be computed exactly and fixed.

We denote as $BdP+$ the bound propagation algorithm that takes advantage of the network structure as described above. Although proposed improvements are straight forward, the gains in accuracy and speed are significant in practice, as we show empirically.

3.2 Managing Resources

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

Since the bounds of variable X_i are used to define constraints of the neighboring variables, fixing the bounds of X_i to their input values will result in a more relaxed LP formulation. Thus, the bounds of neighboring nodes are likely to be less tight as well, affecting, in turn, the bounds of their neighbors. Hence, the effect of fixing bounds of X_i can propagate throughout the network resulting in the overall larger average bounds interval. As k increases, the computation time will increase, but the bounds will become tighter.

3.3 Approximating the LP in Bound Propagation

In this section, we propose an algorithm for solving the linear optimization problem approximately, instead of using a simplex solver.

In large Bayesian networks, we may need to solve linear optimization problems thousands of times. Using the simplex method then becomes impractical time-wise. In general, the linear optimize problems which are formulated in bound propagation fall into a class of linear packing and covering problems which are known to be especially challenging for the simplex method [5]. The standard fractional packing and covering problem can be defined as follows:

$$\min c^T \overline{x} \quad (5)$$

$$A\overline{x} \geq l \quad (6)$$

$$B\overline{x} \leq m \quad (7)$$

$$x \geq 0 \quad (8)$$

Without Eq. (7), it is a **fractional covering** problem. Without Eq. (6), it is a **fractional packing** problem. The BdP (and $BdP+$) linear optimization problems have both packing and covering components with the special properties that A=B and A is a zero-one matrix. Still, the problem remains hard. Existing approximate algorithms solve either packing or covering problem, but not both [5]. The LP formulation in BdP is complicated further by having an additional sum-to-1 constraint. Hence, we resort to solving a relaxed problem.

We considered two relaxations of the LP formulation in BdP. First, we relaxed the problem to an instance of a fractional knapsack packing which can be solved exactly in a greedy fashion [16]. In this case, we maintain the sum-to-1 constraint, but drop the lower bound constraints completely and replace the upper bounds on sums of variables with the derived upper bounds on individual variables. Namely, for each variable $P(\overline{y}|e)$ participating in $|Y|$ constraints, we set:

$$P(\overline{y}|e) \leq UB_{\overline{y}} = \min_j P^U(y_j|e) \tag{9}$$

We obtain an optimal solution to the fractional knapsack packing by first ordering the variables by their coefficient (from maximum to minimum for maximization and from minimum to maximum for minimization) and then assigning each variables its maximum value until the sum of all values equals 1. The complexity of the algorithm is $O(n \log n)$, where n is the number of variables, due to the complexity of sorting.

The second relaxation is more constrained. We maintain the sum-to-1 constraint and and the lower and upper bound constraint pertaining to one variable in the Markov blanket of X_i. We drop the remaining lower bounds and use remaining upper bounds to set upper bounds on individual variables. Consider the example presented previously with a Markov blanket consisting of two nodes A and B. Maintaining the constraints associated with variable A, the resulting relaxed optimization problem can be expressed as:

$$f = 0.1P_{00} + 0.2P_{01} + 0.3P_{02} + 0.4P_{10} + 0.5P_{11} + 0.5P_{12}$$

s.t. $P_{00} + P_{01} + P_{02} + P_{10} + P_{11} + P_{12} = 1$

$$\begin{aligned} P^L(a=0|e) &\leq P_{00} + P_{01} + P_{02} \leq P^U(a=0|e) \\ P^L(a=1|e) &\leq P_{10} + P_{11} + P_{12} \leq P^U(a=1|e) \\ 0 &\leq P_{00}, P_{10} \leq P^U(b=0|e) \\ 0 &\leq P_{01}, P_{11} \leq P^U(b=1|e) \\ 0 &\leq P_{02}, P_{12} \leq P^U(b=2|e) \end{aligned}$$

The domains of the constraints w.r.t. just one Markov variables Y_j are disjoint. Hence, the problem can be treated as an instance of the fractional packing with multiple knapsacks, each having a separate set of packing materials and an individual capacity bound. If it was not for the sum-to-1 constraint, we could solve each knapsack packing problem independently. Nevertheless, we can show that the problem can be solved optimally by a greedy algorithm.

We describe the idea of the algorithm on the example of maximization problem. Similar to fractional packing with 1 knapsack, we first order nodes by their objective function coefficient value from the largest to smallest. We initialize all node values to 0. Then, we make two passes through the list. The first time, we satisfy all lower bound requirements. Namely, we increment each node value until either its upper bound is reached or the lower bound $L(y_j)$ of the equation in which it participates is satisfied. During a second pass, we increment each variable value until either the variables' upper bound or the upper bound $U(y_j)$ of the equation in which it participates is reached or the sum of all variables equals 1. Since we cannot predict which variable $Y_j \in Y$ will yield the LP relaxation with the smallest maximum of the objective function, we repeat computation for each $Y_j \in Y$ and pick the smallest maximum of the objective function.

The solution to the minimization problem is the same except nodes are ordered by their objective function coefficient value from smallest to largest. We prove formally that the algorithm finds an optimal solution in [4]. The total complexity is $O(|Y| \cdot n \log n)$, $n = |\mathcal{D}(Y)|$. We call the bound propagation scheme with an approximate LP algorithm as $ABdP+$.

4 Experiments

We compare empirically the performance of the original bound propagation algorithm BdP, modified $BdP+$ that restricts the Markov blanket of a node to its relevant subnetwork, and a modified bound propagation scheme using the approximate algorithm for solving linear programming subproblems, namely, $ABdP+$.

4.1 Methodology

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

$$\overline{I} = \frac{\sum_i \sum_{D(x_i)} (P^U(x_i|e) - P^L(x_i|e))}{\sum_i |D(x_i)|} \tag{10}$$

We compute 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:

$$\hat{P}(x|e) = \frac{P^U(x|e) + P^L(x|e)}{2} \tag{11}$$

and then measure average absolute error Δ with respect to $\hat{P}(x|e)$:

$$\Delta = \frac{\sum_i \sum_{D(x_i)} |P(x_i|e) - \hat{P}(x_i|e)|}{\sum_i |D(x_i)|} \tag{12}$$

We control the time and memory of bound propagation by restricting the maximum length k of a conditional probability table over the Markov blanket of a variable. The maximum Markov CPT length tested was k=2^{19} (the size of the CPT with 19 bi-valued variables) when the computation demands exceeded available memory.

We report all results for BdP, $BdP+$, and $ABdP+$ schemes "upon convergence" or after 20 iterations, whichever occurs first, so that the computation time is a function of k and number of iterations needed to converge. We implemented bounds propagation algorithm using simplex solver from COIN-OR libraries [1]. The experiments were conducted on 1.8Ghz CPU with 512 MB RAM.

Our benchmarks are 8 networks from Bayesian Network Repository (http://www.cs.huji.ac.il/labs/compbio/ Repository/) and 3 networks for pedigree analysis, gen44, gen50, and gen51 (from http://bioinfo.cs.technion.ac.il/superlink/ExpF.html). Benchmarks' properties are specified in Table 4.1. In gen44, gen50, and gen51 evidence is pre-defined. In all other benchmarks, the results are averaged over 20 instances of each network instantiated with different evidence. In most networks the evidence variables are selected at random among leaf nodes. In cpcs422b, the evidence is selected at random among all variables.

Table 1. Benchmarks' characteristics: N-number of nodes, w^*-induced width, T_{BE}-exact computation time via bucket elimination.

network	N	w^*	T_{BE}
Alarm	37	4	0.01 sec
cpcs54	54	15	1 sec
cpcs179	179	8	2 sec
cpcs360b	360	21	20 min
cpcs422b	422	22	50 min
gen44	873	N/A	47 min
gen50	547	N/A	>6 hrs
gen51	1218	N/A	> 7 hrs

4.2 Results

4.2.1 BdP vs. $BdP+$

In Tables 2 and 3 we report the average error, average bounds interval, and computation times for BdP and $BdP+$ as a function of k=2^m for 14≤m≤19. Each row corresponds to a set of experiments with a single benchmark with a fixed k. Note that, as k increases, the computation time of both BdP and $BdP+$ increases fast, while the bounds interval decreases only a little.

Table 2. Average error Δ, length of the bounds interval $\overline{I}$, and computation time for BdP(k) and $BdP+$(k) in networks without evidence.

		BdP(k)			BdP+(k)		
	k	$\overline{I}$	Δ	**time**	$\overline{I}$	Δ	**time**
Alarm	16384	0.6369	0.1677	14	0.0753	0.0076	0.1
cpcs54	16384	0.4247	0.0229	24	0.0907	0.0049	0.1
	32768	0.4173	0.0224	72	0.0907	0.0049	0.1
	262145	0.4154	0.0221	265	0.0907	0.0049	0.1
cpcs179	16384	0.5759	0.2213	30	0.0006	0.00002	0.3
	32768	0.5759	0.2213	30	0.0006	0.00002	0.3
cpcs360b	16384	0.1505	0.0649	64	0.0006	0.0002	1.2
	32768	0.1485	0.0641	80	0.0006	0.0002	1.2
cpcs422b	16384	0.2339	0.0756	79	0.0082	0.0008	8
	32768	0.2329	0.0751	88	0.0082	0.0008	8

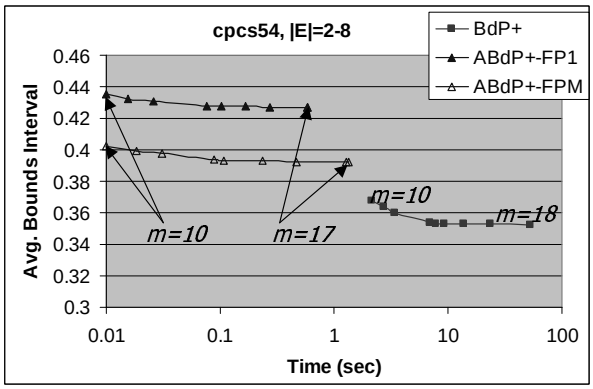
Table 3. Average error Δ, length of the bounds interval $\overline{I}$, and computation time for BdP(k) and $BdP+$(k) in networks with evidence.

		BdP(k)			BdP+(k)		
	k	$\overline{I}$	Δ	**time**	$\overline{I}$	Δ	**time**
Alarm	16384	0.8276	0.2661	13	0.6376	0.2084	5.3
$\|E\|$=3-6	65536	0.8276	0.2661	13	0.6376	0.2084	5.3
cpcs54	16384	0.6021	0.0448	46	0.2638	0.0138	6.6
$\|E\|$=2-6	32768	0.5986	0.0445	64	0.2637	0.0138	7.4
	65536	0.5957	0.0440	98	0.2637	0.0138	10
	131072	0.5954	0.0439	116	0.2635	0.0137	16
cpcs179	16384	0.6034	0.2227	30	0.1525	0.0456	20
$\|E\|$=12-24	32768	0.6034	0.2227	30	0.1502	0.0450	24
	65536	0.5983	0.2214	90	0.1237	0.0365	130
cpcs360b	16384	0.3375	0.1423	68	0.0637	0.0247	15
$\|E\|$=11-23	32768	0.3370	0.1419	85	0.0554	0.0215	24
	65536	0.1430	0.3367	120	0.0500	0.0192	36
	131072	0.1430	0.3366	128	0.0429	0.0160	80
	262144	0.1428	0.3364	190	0.0377	0.0137	130
cpcs422b	16384	0.3373	0.1200	34	0.2140	0.0665	24
$\|E\|$=6-11	32768	0.3287	0.1081	80	0.2034	0.0617	74
	65536	0.3171	0.1023	317	0.1815	0.0522	256

$BdP+$ always computes tighter bounds and requires less computation time than BdP. The performance gap is wider in the networks without evidence (Table 2), where the Markov blanket of each node, restricted to its relevant subnetwork, contains node's parents only and $BdP+$ converges after one iteration when processing nodes in topological order. For the largest benchmark, cpcs422b, with 422 nodes and w^*=21, the average bounds interval length is 0.23 for BdP and 0.008 for $BdP+$. At the same time, BdP computations take 190 sec while $BdP+$ only takes 16 sec.

In benchmarks with evidence, reported in Table 3, $BdP+$ computation time increases but its bounds remain superior to BdP. Consider the results for cpcs360b network with 360 nodes and $|E|$ ranging from 11 to 23. For k=16384, $BdP+$ computes the average bounds interval of length 0.06 within 15 seconds. BdP average bounds interval equals 0.34 and requires 68 seconds. We observed similar results for other benchmarks.

4.2.2 Approximating LP



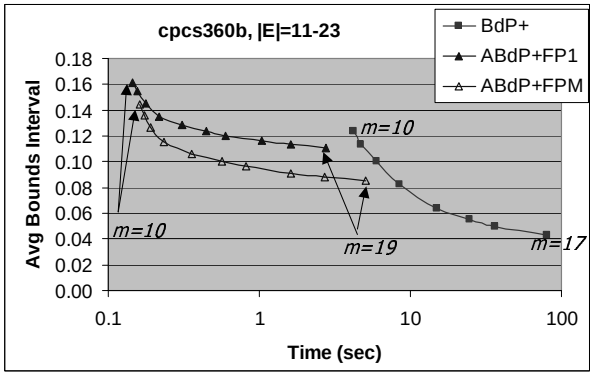


Figure 1. Bounds interval for $BdP+$ and $ABdP+$ optimizing LP by fractional packing with 1 ($ABdP$+$FP1$) and many ($ABdP$+FPM) knapsacks. Each data point corresponds to a value of input parameter m that bounds the maximum Markov table length k=2^m.

We compare the performance of $BdP+$ and $ABdP+$ with two different approximation schemes, $FP1$ and FPM. $FP1$ solves the fractional packing with 1 knapsack; FPM solves the fractional packing and covering problem with multiple knapsacks.

The results shown in Figure 1 for cpcs54 and cpcs360b are typical. The average bounds interval is shown as a function of time which, in turn, depends on the input parameter k=2^m. Hence, each point on the graph corresponds to a particular value of m. First, we observe that $ABdP+$ with FPM (denoted $ABdP$+FPM) substantially outperforms $ABdP+$ with $FP1$ (denoted $ABdP$+$FP1$) in

both benchmarks. FPM incurs negligible amount of computation overhead but computes considerably tighter bounds. $BdP+$ outperforms both $ABdP+$ with enough time, but $ABdP+$ is more efficient at the start. For the same k, $BdP+$ requires an order of magnitude more time than $ABdP+$. In cpcs54, for k=2^{10}, $ABdP+FPM$ computes bounds in <0.01 seconds while $BdP+$ requires a few seconds. We observe similar result in case of cpcs360b. Roughly, $BdP+$ requires as much time to compute bounds for k=2^{10} as $ABdP+FPM$ for k=2^{17}-2^{19}. As a result, $BdP+$ begins to outperform $ABdP+FPM$ only after ≈2 seconds in cpcs54 and only after 10 seconds in cpcs360b. The overall improvement in $BdP+$ bounds over $ABdP+FPM$ in cpcs54 is minor; the difference between their average bounds intervals is only about 0.04 which is ≈1% of the interval length. In cpcs360b, the $BdP+$ bounds interval after 100 seconds is a factor of 2 smaller, but the computation time is an order of magnitude greater.

Table 4. The # of probabilities in each pedigree network (gen44, gen50, gen51) with bounds interval $\overline{I}$ in different ranges, BdP+ vs. ABdP+FPM.

	gen44		gen50		gen51	
	BdP+	ABdP+	BdP+	ABdP+	BdP+	ABdP+
time	54	0.8	66	8.4	140	0.7
$0.5 \leq \overline{I} < 1$	1312	1324	1114	1114	1428	1432
$0.4 \leq \overline{I} < 0.5$	14	6	6	6	36	36
$0.3 \leq \overline{I} < 0.4$	6	2	8	10	2	0
$0.2 \leq \overline{I} < 0.3$	2	2	6	4	4	2
$0.1 \leq \overline{I} < 0.2$	10	10	6	6	6	6
$0.01 \leq \overline{I} < 0.1$	0	0	10	10	12	12
$0.001 \leq \overline{I} < 0.01$	0	0	2	2	5	5
$0 \leq \overline{I} < 0.001$	14	14	18	18	174	174

In pedigree networks, both schemes performed poorly computing $\overline{I}$<0.5 only for 5-15% of posteriors using maximum k=2^{19}. Yet, importantly, $ABdP+$ obtained similar results to $BdP+$ in a fraction of time. Table 4 specifies the number of posterior marginals with bounds in different ranges obtained by both algorithms. In gen44, both $BdP+$ and $ABdP+FPM$ computed $\overline{I} \in$[0,0.001] for 14 values and $\overline{I} \in$[0.1,0.2] for another 10 values. However, $BdP+$ requires 54 sec whereas $ABdP+FPM$ takes 0.8 sec. In gen50, the results only differ in that 2 values with $BdP+$ bounds interval in range [0.2,0.3] have $ABdP+FPM$ bounds interval in range [0.3,0.4], but $BdP+$ takes 66 sec while $ABdP+FPM$ computes bounds in 8.4 sec. Similarly, in gen51, algorithm $ABdP+FPM$ computes only 4 values less with the bounds interval in range [0.2,0.4], but completes computation in 0.7 sec while $BdP+$ requires 140 sec.

5 Conclusions and Future Work

In this paper, we proposed two improvements to the bound propagation algorithm [11]. First improvement exploits the directionality of Bayesian networks, restricting the Markov blanket of a variable to its relevant subnetwork. Although the idea is straight forward, it results in substantial improvement in the accuracy of bounds while the computation time is reduced which we demonstrated empirically.

Second improvement defines an approximation algorithm for the linear optimization problems in bound propagation scheme. Namely, we proposed a relaxation of the LP problem and a greedy algorithm that can solve it exactly. Although the bounds obtained by approximate LP solver are less accurate, we reduce computation time by an order of magnitude or more. Hence, bound propagation with approximate LP solver is more practical for real-time on-line applications where speed is of critical importance while a small loss in accuracy may be acceptable. An efficient practical strategy may be also to select a threshold value k^* and use the simplex method to solve small LP problems (Markov blanket size $<k^*$) and use an approximate method to solve large LP problems.

We used the proposed bound propagation algorithm with approximate LP solver as a plug-in in the any-time framework for computing bounds on posterior marginals [3], where using $BdP+$ with simplex solver was not feasible time-wise. The loss of bounds accuracy was not significant since the framework focuses on enumerating high-probability cutset-tuples and only uses $ABdP+FPM$ to bound the remaining probability mass of $P(e)$. We showed in [3] that any-time framework with $ABdP+FPM$ outperformed $BdP+$ after exploring a few hundred to a few thousand cutset tuples.

Performance of bound propagation depends on the efficiency of the linear optimization algorithm. We have looked at only two approximation schemes. Other approximation algorithms, offering different time/accuracy trade-offs, need to be investigated.

6 Acknowledgments

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

REFERENCES

[1] *COmputational INfrastructure for Operations Research*, http://www.coin-or.org.
[2] A. M. Abdelbar and S. M. Hedetniemi, 'Approximating maps for belief networks is NP-hard and other theorems', *Artificial Intelligence*, **102**, 21–38, (1998).
[3] B. Bidyuk and R. Dechter, 'An anytime scheme for bounding posterior beliefs', in *In Proc. of 21st National Conf. on AI (AAAI)*, (2006).
[4] B. Bidyuk and R. Dechter, 'Improving bound propagation', Technical report, UCI, http://www.ics.uci.edu/~bbidyuk/bp.html, (2006).
[5] D. Bienstock, *Potential function methods for approximately solving linear programming problems: theory and practice*, Kluwer Academic Publishers, 2002.
[6] P. Dagum and M. Luby, 'Approximating probabilistic inference in Bayesian belief networks is NP-hard', *Artificial Intelligence*, **60**(1), 141–153, (1993).
[7] E.J. Horvitz, H.J. Suermondt, and G.F. Cooper, 'Bounded conditioning: Flexible inference for decisions under scarce resources', in *Workshop on Uncertainty in Artificial Intelligence*, pp. 181–193, (1989).
[8] M. Kearns and L. Saul, 'Large deviation methods for approximate probabilistic inference, with rates of convergence', in *Proc. of Uncertainty in AI*, pp. 311–319. Morgan Kaufmann, (1998).
[9] M. Kearns and L. Saul, 'Inference in multilayer networks via large deviation bounds', *Advances in Neural Information Processing Systems*, **11**, 260–266, (1999).
[10] D. Larkin, 'Approximate decomposition: A method for bounding and estimating probabilistic and deterministic queries', in *Proc. of UAI*, pp. 346–353, (2003).
[11] M. A. R. Leisink and H. J. Kappen, 'Bound propagation', *Journal of Artificial Intelligence Research*, **19**, 139–154, (2003).
[12] M. V. Mannino and V. S. Mookerjee, 'Probability bounds for goal directed queries in Bayesian networks', *IEEE Transactions on Knowledge and Data Engineering*, **14**(5), 1196–1200, (September/October 2002).
[13] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, 1988.
[14] D. Poole, 'Probabilistic conflicts in a search algorithm for estimating posterior probabilities in Bayesian networks', *Artificial Intelligence*, **88**(1–2), 69–100, (1996).
[15] David Poole, 'Context-specific approximation in probabilistic inference', in *Proc. of Uncertainty in AI*, pp. 447–454, (1998).
[16] R. L. Rivest T. H. Cormen, C. E. Leiserson, *Introduction to Algorithms*, MIT Press, 1990.