Tree Approximation for Belief Updating

Robert Mateescu, Rina Dechter and Kalev Kask

Department of Information and Computer Science
University of California, Irvine, CA 92697-3425
{mateescu,dechter,kkask}@ics.uci.edu

Abstract

The paper presents a parameterized approximation scheme for probabilistic inference. The scheme, called *Mini-Clustering (MC)*, extends the partition-based approximation offered by mini-bucket elimination, to tree decompositions. The benefit of this extension is that all single-variable beliefs are computed (approximately) at once, using a two-phase message-passing process along the cluster tree. The resulting approximation scheme allows adjustable levels of accuracy and efficiency, in anytime style. Empirical evaluation against competing algorithms such as iterative belief propagation and Gibbs sampling demonstrates the potential of the MC approximation scheme for several classes of problems.

Introduction and related work

Probabilistic reasoning using Belief networks, computing the probability of one or more events given some evidence, is known to be NP-hard (Cooper 1990). However most commonly used exact algorithms for probabilistic inference such as join-tree clustering (Lauritzen & Spiegelhalter 1988; Jensen, Lauritzen, & Olesen 1990) or variable-elimination (Dechter 1996; Zhang & Poole 1994), exploit the network structure. These algorithms are time and space exponential in a graph parameter capturing the density of the network called tree-width. Yet, for large belief networks, the tree-width is often large, making exact inference impractical and therefore approximation methods must be pursued. Although approximation within given error bounds is also NP-hard (Dagum & Luby 1993; Roth 1996), some approximation strategies work well in practice.

The paper presents an anytime approximation scheme for probabilistic inference called mini-clustering (MC), which allows a flexible tradeoff between accuracy and efficiency. MC extends the partition-based approximation offered by mini-bucket elimination (Dechter & Rish 1997) to tree decompositions. The benefit of this extension is that all single-variable beliefs are computed (approximately) at once, using a two-phase message-passing process along the cluster tree. We present new empirical evaluation[1] against competing algorithms such as iterative belief propagation and Gibbs sampling. The experiments demonstrate the anytime behavior of this methodology and its overall potential: On several classes of problems (e.g. random noisy-or, grid networks and CPCS networks) mini-clustering exhibited superior performance.

Related work. Approximation algorithms proposed for belief updating fall into two main categories: 1. Monte Carlo sampling algorithms. Those algorithms sample the probability distribution and compute the probability required based on the obtained sample. Algorithms in this class are logic sampling (Henrion 1986), Gibbs sampling (Pearl 1988), likelihood weighting (Shachter & Peot 1989) and importance sampling (Cheng & Druzdzel 2000). 2. Algorithms that weaken or ignore some of the network's dependencies, forcing the generated dependencies of the network to be bounded. The mini-bucket scheme and the currently presented mini-cluster scheme fall into this second category.

Another algorithm in this class is *Iterative belief propagation (IBP)*, also called loopy belief propagation, that applies Pearl's belief propagation algorithm for singly connected networks to loopy-networks (Pearl 1988). It was recently observed that this algorithm works extremely well for coding applications (McEliece, MacKay, & Cheng 1997; Weiss 1997; Welling & Teh 2001). A related algorithm that removes *weak dependencies* in a join-tree clustering is given in (Kjæaerulff 1994) and another, for stochastic processes is in (Boyen & Koller 1998).

Hybrid methods exploiting both structure and Monte Carlo sampling were also introduced (Koller, Lerner, & Angelov 1998). Finally, an orthogonal collection of approximation algorithms using variational methods (Jaakkola & Jordan 1996) does not fall exactly in these two categories.

Preliminaries

Belief networks provide a formalism for reasoning about partial beliefs under conditions of uncertainty. A belief network is defined by a directed acyclic graph over nodes representing random variables of interest.

Belief networks. A *belief network* is a quadruple $BN = < X, D, G, P >$[2] where $X = \{X_1, \ldots, X_n\}$ is a set of random variables, $D = \{D_1, ..., D_n\}$ is the set of the corresponding domains, G is a directed acyclic graph over X

[1] Note that empirical evaluation of mini-bucket to belief updating was never carried out.

[2] Also abbreviated $< G, P >$ when X and D are clear.

and $P = \{p_1, ..., p_n\}$, where $p_i = P(X_i|pa_i)$ (pa_i are the parents of X_i in G) denote conditional probability tables (CPTs). Given a function f, we denote by $scope(f)$ the set of arguments of function f. The *moral graph* of a directed graph is the undirected graph obtained by connecting the parent nodes of each variable and eliminating direction.

Belief updating. The *belief updating* problem defined over a belief network (also referred to as *probabilistic inference*) is the task of computing the posterior probability $P(Y|e)$ of *query* nodes $Y \subseteq X$ given evidence e. We will focus on the basic case when Y consists of a single variable X_i. Namely, computing $Bel(X_i) = P(X_i = x|e)$, $\forall X_i \in X$, $\forall x \in D_i$.

Tree-decomposition schemes

We will describe our algorithms relative to a unifying tree-decomposition framework based on (Gottlob, Leone, & Scarello 1999). It generalizes tree-decompositions to include join-trees, bucket-trees and other variants applicable to both constraint processing and probabilistic inference.

DEFINITION **1 (tree-decomposition, cluster tree)** *Let $BN =< X, D, G, P >$ be a belief network. A* tree-decomposition *for BN is a triple $< T, \chi, \psi >$, where $T = (V, E)$ is a tree, and χ and ψ are labeling functions which associate with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq P$ satisfying:*
1. For each function $p_i \in P$, there is exactly *one vertex $v \in V$ such that $p_i \in \psi(v)$, and $scope(p_i) \subseteq \chi(v)$.*
2. For each variable $X_i \in X$, the set $\{v \in V | X_i \in \chi(v)\}$ induces a connected subtree of T. This is also called the running intersection property.
We will often refer to a node and its functions as a cluster *and use the term* tree-decomposition *and* cluster tree *interchangeably.*

DEFINITION **2 (tree-width, separator, eliminator)** *The* tree-width *(Arnborg 1985) of a tree-decomposition $< T, \chi, \psi >$ is $max_{v \in V} |\chi(v)|$. Given two adjacent vertices u and v of a tree-decomposition, the* separator *of u and v is defined as $sep(u, v) = \chi(u) \cap \chi(v)$, and the* eliminator *of u with respect to v is $elim(u, v) = \chi(u) - \chi(v)$.*

Join-trees and cluster-tree-elimination

The most used tree decomposition method is called join-tree decomposition (Lauritzen & Spiegelhalter 1988). Such decompositions can be generated by embedding the network's moral graph, G, in a chordal graph, often using a triangulation algorithm and using its maximal cliques as nodes in the join-tree. Such a join-tree satisfies the properties of tree-decomposition.

There are a few variants for processing join-trees for belief updating (Jensen, Lauritzen, & Olesen 1990; Shafer & Shenoy 1990). The variant which we use here, called cluster-tree-elimination (CTE) is applicable to tree-decompositions in general and is geared toward space savings. It is a message passing algorithm (either two-phase message passing, or in asynchronous mode). Algorithm CTE for belief updating denoted CTE-BU is given in Figure 1. The algorithm pays a special attention to the processing of observed variables since the presence of evidence is a central component in belief updating. When a cluster sends a message to a neighbor, the algorithm operates on all the functions in the cluster except the message from that particular neighbor. The message contains a single *combined* function and *individual* functions that do not share variables with the relevant eliminator. All the non-individual functions are *combined* in a product and summed over the eliminator.

Algorithm CTE for Belief-Updating (CTE-BU)
Input: A tree decomposition $< T, \chi, \psi >$, $T = (V, E)$ for $BN =< X, D, G, P >$. Evidence variables $var(e)$.
Output: An augmented tree whose nodes are clusters containing the original CPTs and the messages received from neighbors. $P(X_i, e)$, $\forall X_i \in X$.

Denote by $H_{(u,v)}$ the message from vertex u to v, $ne_v(u)$ the neighbors of u in T excluding v.
$cluster(u) = \psi(u) \cup \{H_{(v,u)} | (v, u) \in E\}$.
$cluster_v(u) = cluster(u)$ excluding message from v to u.

• **Compute messages:**
For every node u in T, once u has received messages from all $ne_v(u)$, compute message to node v:

1. **process observed variables**
   Assign relevant evidence to all $p_i \in \psi(u)$
2. **Compute the combined function:**

$$h_{(u,v)} = \sum_{elim(u,v)} \prod_{f \in A} f.$$

   where A is the set of functions in $cluster_v(u)$ whose scope intersects $elim(u, v)$.
   Add $h_{(u,v)}$ to $H_{(u,v)}$ and add all the individual functions in $cluster_v(u) - A$
   Send $H_{(u,v)}$ to node v.

• **Compute** $P(X_i, e)$**:**
For every $X_i \in X$ let u be a vertex in T such that $X_i \in \chi(u)$.
Compute $P(X_i, e) = \sum_{\chi(u)-\{X_i\}} (\prod_{f \in cluster(u)} f)$

Figure 1: Algorithm Cluster-Tree-Elimination for Belief Updating (CTE-BU)

Example 1 *Figure 2 describes a belief network and a join-tree decomposition for it. Figure 3 shows the trace of running CTE-BU. In this case no individual functions appear between any of the clusters. To keep the figure simple, we only show the combined functions $h_{(u,v)}$ (each of them being in fact the only element of the set $H_{(u,v)}$ that represents the corresponding message between clusters u and v).*

Similar to (Dechter, Kask, & Larossa 2001) we can show that:

THEOREM **1 (Complexity of CTE-BU)** *The time complexity of CTE-BU is $O(deg \cdot (n + N) \cdot d^{w^*+1})$ and the space complexity is $O(N \cdot d^{sep})$, where deg is the maximum degree of a node in the tree, n is the number of variables, N is the number of nodes in the tree decomposition, d is the maximum domain size of a variable, w^* is the tree-width and sep is the maximum separator size.*

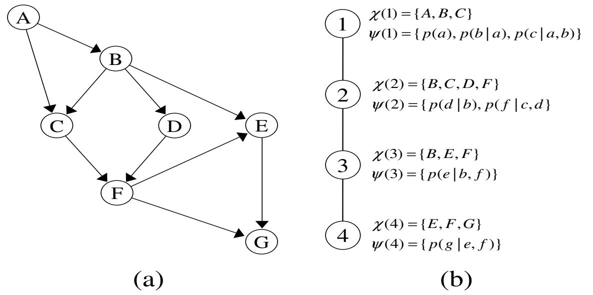

Figure 2: $a)$ A belief network; $b)$ A join-tree decomposition;

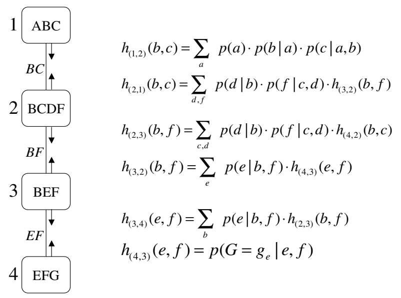

Figure 3: Execution of CTE-BU; no individual functions appear in this case.

Mini-Clustering for belief updating

The time, and especially the space complexity of CTE-BU renders the algorithm infeasible for problems with high tree-width. In this section we introduce the mini-clustering, a partition-based anytime algorithm which computes approximate values or bounds on $P(X_i, e)$ for every variable X_i in the network. It is a natural extension of the mini-bucket idea to tree-decompositions. Rather than computing the mini-bucket approximation n times, one for each variable as would be required by the mini-bucket approach, the algorithm performs an equivalent computation with just two message passings along each arc of the cluster tree. The idea is to partition each cluster into mini-clusters having at most i variables, where i is an accuracy parameter. Node u partitions its cluster into p mini-clusters $mc(1), \dots, mc(p)$. Instead of computing $h_{(u,v)} = \sum_{elim(u,v)} \prod_{k=1}^{p} \prod_{f \in mc(k)} f$ as in CTE-BU, we can compute an upper bound by migrating the summation operator into each mini-cluster. However, this would give $\prod_{k=1}^{p} \sum_{elim(u,v)} \prod_{f \in mc(k)} f$ which is an unnecessarily large upper bound on $h_{(u,v)}$ in which each $\prod_{f \in mc(k)} f$ is bounded by its sum over $elim(u,v)$. Instead, we rewrite $h_{(u,v)} = \sum_{elim(u,v)} (\prod_{f \in mc(1)} f) \cdot (\prod_{i=2}^{p} \prod_{f \in mc(i)} f)$. Subsequently, instead of bounding $\prod_{f \in mc(i)} f, (i \geq 2)$ by summation over the eliminator, we bound it by its maximum over the eliminator, which yields $(\sum_{elim(u,v)} \prod_{f \in mc(1)} f) \cdot \prod_{k=2}^{p} (\max_{elim(u,v)} \prod_{f \in mc(k)} f)$. Therefore, if we are interested in an upper bound, we marginalize one mini-cluster by summation and the others by maximization. Note that the summation in the first mini-cluster must be over *all* variables in the eliminator, even if some of them might not appear in the scope of functions in $mc(1)$.

Procedure MC for Belief Updating (MC-BU(i))

2. **Compute the combined mini-functions:**
   Make an (i)-size mini-clusters partitioning of $cluster_v(u)$, $\{mc(1), \dots, mc(p)\}$;
   $h^1_{(u,v)} = \sum_{elim(u,v)} \prod_{f \in mc(1)} f$
   $h^i_{(u,v)} = \max_{elim(u,v)} \prod_{f \in mc(i)} f \quad i = 2, \dots, p$
   add $\{h^i_{(u,v)} | i = 1, \dots, p\}$ to $H_{(u,v)}$. Send $H_{(u,v)}$ to v.

Compute upper bounds on $P(X_i, e)$:
For every $X_i \in X$ let $u \in V$ be a cluster such that $X_i \in \chi(u)$. Make (i) mini-clusters from $cluster(u)$, $\{mc(1), \dots, mc(p)\}$; Compute
$(\sum_{\chi(u)-X_i} \prod_{f \in mc(1)} f) \cdot (\prod_{k=2}^{p} \max_{\chi(u)-X_i} \prod_{f \in mc(k)} f)$.

Figure 4: Procedure Mini-Clustering for Belief Updating (MC-BU)

Consequently, the combined functions are approximated via mini-clusters, as follows. Suppose $u \in V$ has received messages from all its neighbors other than v (the message from v is ignored even if received). The functions in $cluster_v(u)$ that are to be combined are partitioned into mini-clusters $\{mc(1), \dots, mc(p)\}$, each one containing at most i variables. One of the mini-clusters is processed by summation and the others by maximization over the eliminator, and the resulting combined functions as well as all the individual functions are sent to v.

Lower-bounds and mean approximations. We can also derive a lower-bound on beliefs by replacing the max operator with min operator (see above derivation for rationale). This allows, in principle, computing both an upper bound and a lower bound on the joint beliefs. Alternatively, if we yield the idea of deriving a bound (and indeed the empirical evaluation encourages that) we can replace max by a $mean$ operator (taking the sum and dividing by the number of elements in the sum), deriving an approximation of the joint belief.

Algorithm MC-BU for upper bounds can be obtained from CTE-BU by replacing step 2 of the main loop and the final part of computing the upper bounds on the joint belief by the procedure given in Figure 4.

Partitioning strategies. In our current implementation, the partitioning is done in an arbitrary brute-force manner and the choice of the first mini-cluster for upper bound computation is random. Clearly, a more informed approach may improve the accuracy significantly but this exploration is outside the scope of the current paper.

Example 2 *Figure 5 shows the trace of running MC-BU(3) on the problem in Figure 2. First, evidence $G = g_e$ is assigned in all CPTs. There are no individual functions to be sent from cluster 1 to cluster 2. Cluster 1 contains only 3 variables, $\chi(1) = \{A, B, C\}$, therefore it is not partitioned. The combined function $h^1_{(1,2)}(b,c) = \sum_a p(a) \cdot$*

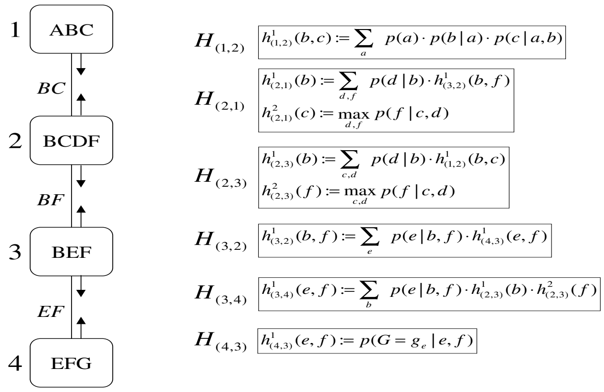

Figure 5: Execution of MC-BU for $i = 3$

$p(b|a) \cdot p(c|a,b)$ is computed and the message $H_{(1,2)} = \{h^1_{(1,2)}(b,c)\}$ is sent to node 2. Now, node 2 can send its message to node 3. Again, there are no individual functions. Cluster 2 contains 4 variables, $\chi(2) = \{B, C, D, F\}$, and a partitioning is necessary: MC-BU(3) can choose $mc(1) = \{p(d|b), h_{(1,2)}(b,c)\}$ and $mc(2) = \{p(f|c,d)\}$. The combined functions $h^1_{(2,3)}(b) = \sum_{c,d} p(d|b) \cdot h_{(1,2)}(b,c)$ and $h^2_{(2,3)}(f) = \max_{c,d} p(f|c,d)$ are computed and the message $H_{(4,3)} = \{h^1_{(2,3)}(b), h^2_{(2,3)}(f)\}$ is sent to node 3. The algorithm continues until every node has received messages from all its neighbors. An upper bound on $p(a, G = g_e)$ can now be computed by choosing cluster 1, which contains variable A. It doesn't need partitioning, so the algorithm just computes $\sum_{b,c} p(a) \cdot p(b|a) \cdot p(c|a,b) \cdot h^1_{(2,1)}(b) \cdot h^2_{(2,1)}(c)$. Notice that unlike CTE-BU which processes 4 variables in cluster 2, MC-BU(3) never processes more than 3 variables at a time.

Properties of Mini-Clustering

THEOREM **2** *Algorithm MC-BU(i) with max (respectively min) computes an upper (respectively lower) bound on the joint probability $P(X, e)$ of each variable and each of its values.*

A similar mini-clustering scheme for combinatorial optimization was developed in (Dechter, Kask, & Larossa 2001) having similar performance properties as MC-BU.

THEOREM **3 (Complexity of MC-BU(i))** *(Dechter, Kask, & Larossa 2001) The time and space complexity of MC-BU(i) is $O(n \cdot hw^* \cdot d^i)$ where n is the number of variables, d is the maximum domain size of a variable and $hw^* = max_u |\{f | scope(f) \cap \chi(u) \neq \phi\}|$, which bounds the number of functions that may travel to a neighboring cluster via message-passing.*

Accuracy. For a given i, the accuracy of MC-BU(i) can be shown to be not worse than that of executing the mini-bucket algorithm MB(i) n times, once for each variable (an algorithm that we call nMB(i)). Given a specific execution of MC-BU(i), we can show that for every variable X_i, there exists an ordering of the variables and a corresponding partitioning such that MB(i) computes the same approximation value for $P(X_i, e)$ as does $MC - BU(i)$. In empirical analysis (Kask 2001) it is shown that MC-BU has an up to linear speed-up over nMB(i).

Normalization

The MC-BU algorithm using max operator computes an upper bound $\overline{P}(X_i, e)$ on the joint probability $P(X_i, e)$. However, deriving a bound on the conditional probability $P(X_i|e)$ is not easy when the exact value of $P(e)$ is not available. If we just try to divide (multiply) $\overline{P}(X_i, e)$ by a constant, the result is not necessarily an upper bound on $P(X_i|e)$. In principle, if we can derive a lower bound $\underline{P}(e)$ on $P(e)$, then we can compute $\overline{P}(X_i, e)/\underline{P}(e)$ as an upper bound on $P(X_i|e)$. However, due to compound error, it is likely to be ineffective. In our empirical evaluation we experimented with normalizing the upper bound as $\overline{P}(X_i, e)/\sum_{X_i} \overline{P}(X_i, e)$ over the values of X_i. The result is not necessarily an upper bound on P($X_i|e$). Similarly, we can also normalize the values when using $mean$ or min operators. It is easy to show that normalization with the $mean$ operator is identical to normalization of MC-BU output when applying the summation operator in all the mini-clusters.

Empirical evaluation

We tested the performance of our scheme on random noisy-or networks, random coding networks, general random networks, grid networks, and three benchmark CPCS files with 54, 360 and 422 variables respectively (these are belief networks for medicine, derived from the Computer based Patient Case Simulation system, known to be hard for belief updating). On each type of network we ran Iterative Belief Propagation (IBP) - set to run at most 30 iterations, Gibbs Sampling (GS) and MC-BU(i), with i from 2 to the treewidth w^* to capture the anytime behavior of MC-BU.

We immediately observed that the quality of MC-BU in providing upper or lower bounds on the joint $P(X_i, e)$ was ineffective. Although the upper bound decreases as the accuracy parameter i increases, it still is in most cases greater than 1. Therefore, following the ideas explained in the previous section we report the results with normalizing the upper bounds (called max) and normalizing the mean (called $mean$). We notice that MC-BU using the $mean$ operator is doing consistently better.

For noisy-or networks, general random networks, grid networks and for the CPCS networks we computed the exact solution and used three different measures of accuracy: 1. Normalized Hamming Distance (NHD) - We picked the most likely value for each variable for the approximate and for the exact, took the ratio between the number of disagreements and the total number of variables, and averaged over the number of problems that we ran for each class. 2. Absolute Error (Abs. Error) - is the absolute value of the difference between the approximate and the exact, averaged over all values (for each variable), all variables and all problems. 3. Relative Error (Rel. Error) - is the absolute value of the difference between the approximate and the exact, divided by the exact, averaged over all values (for each variable),

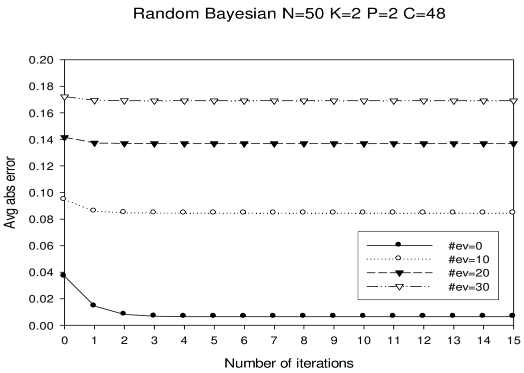

Figure 6: Convergence of IBP

N=50, P=3, 25 instances

	$\|e\|$	NHD max	NHD mean	Abs. Error max	Abs. Error mean	Rel. Error max	Rel. Error mean	Time max	Time mean
IBP	10		0		1.3E-04		7.9E-01		0.242
	20		0		3.6E-04		2.2E+00		0.184
	30		0		6.8E-04		4.2E+00		0.121
MC-BU(2)	10	0	0	1.3E-03	9.6E-04	8.2E+00	5.8E+00	0.107	0.108
	20	0	0	5.3E-04	4.0E-04	3.1E+00	2.4E+00	0.077	0.077
	30	0	0	2.3E-04	1.9E-04	1.4E+00	1.2E+00	0.064	0.064
MC-BU(5)	10	0	0	1.0E-03	8.3E-04	6.4E+00	5.1E+00	0.133	0.133
	20	0	0	4.6E-04	4.1E-04	2.7E+00	2.4E+00	0.104	0.105
	30	0	0	2.0E-04	1.9E-04	1.2E+00	1.2E+00	0.098	0.095
MC-BU(8)	10	0	0	6.6E-04	5.7E-04	4.0E+00	3.5E+00	0.498	0.509
	20	0	0	1.8E-04	1.8E-04	1.1E+00	1.0E+00	0.394	0.406
	30	0	0	3.4E-05	3.4E-05	2.1E-01	2.1E-01	0.300	0.308
MC-BU(11)	10	0	0	2.6E-04	2.4E-04	1.6E+00	1.5E+00	2.339	2.378
	20	0	0	3.8E-05	3.8E-05	2.3E-01	2.3E-01	1.421	1.439
	30	0	0	6.4E-07	6.4E-07	4.0E-03	4.0E-03	0.613	0.624
MC-BU(14)	10	0	0	4.2E-05	4.1E-05	2.5E-01	2.4E-01	7.805	7.875
	20	0	0	0	0	0	0	2.075	2.093
	30	0	0	0	0	0	0	0.630	0.638

Table 1: Performance on noisy-or networks, w*=16

all variables and all problems. For coding networks, we report only one measure, Bit Error Rate (BER). In terms of the measures defined above, BER is the normalized Hamming distance between the approximate (computed by an algorithm) and the actual input (which in the case of coding networks may be different from the solution given by exact algorithms), so we denote them differently to make this semantic distinction. We also show the time taken by each algorithm.

In Figure 6 we show that IBP converges after about 5 iterations. So, while in our experiments we report its time for 30 iterations, its time is even better when sophisticated termination is used. These results are typical of all runs.

The random noisy-or networks and the random networks were generated using parameters (N,K,C,P), where N is the number of variables (a square integer for grid networks), K is their domain size (we used only K=2), C is the number of conditional probability matrices and P is the number of parents in each conditional probability matrix. The grid networks have the structure of a square, with edges directed to form a diagonal flow (all parallel edges have the same direction). They were generated by specifying N (a square integer) and K (we used K=2). We also varied the number of evidence nodes, denoted by $|e|$ in the tables. The parameter

N=50, P=2, 50 instances

	$\|e\|$	NHD max	NHD mean	Abs. Error max	Abs. Error mean	Rel. Error max	Rel. Error mean	Time max	Time mean
IBP	0		0.01840		0.00696		0.01505		0.100
	10		0.19550		0.09022		0.34608		0.080
	20		0.27467		0.13588		3.13327		0.062
GS	0		0.50400		0.10715		0.26621		13.023
	10		0.51400		0.15216		0.57262		12.978
	20		0.51267		0.18066		4.71805		13.321
MC-BU(2)	0	0.11400	0.08080	0.03598	0.02564	0.07950	0.05628	0.055	0.055
	10	0.10600	0.08800	0.04897	0.03957	0.12919	0.10579	0.047	0.048
	20	0.08667	0.07333	0.04443	0.03639	0.13096	0.10694	0.041	0.042
MC-BU(5)	0	0.10120	0.06480	0.03392	0.02242	0.07493	0.04937	0.071	0.072
	10	0.06950	0.05850	0.03254	0.02723	0.08613	0.07313	0.063	0.065
	20	0.03933	0.03400	0.02022	0.01831	0.05533	0.04984	0.059	0.060
MC-BU(8)	0	0.05080	0.02680	0.01872	0.01030	0.04103	0.02262	0.216	0.221
	10	0.01550	0.01450	0.00743	0.00587	0.01945	0.01547	0.178	0.180
	20	0.00600	0.00400	0.00228	0.00200	0.00597	0.00542	0.129	0.134

Table 2: Performance on random networks, w*=10

BER	$\sigma = .22$ max	$\sigma = .22$ mean	$\sigma = .26$ max	$\sigma = .26$ mean	$\sigma = .32$ max	$\sigma = .32$ mean	$\sigma = .40$ max	$\sigma = .40$ mean	$\sigma = .51$ max	$\sigma = .51$ mean	Time
N=100, P=3, 50 instances, w*=7											
IBP	0.000	0.000	0.000	0.000	0.002	0.002	0.022	0.022	0.088	0.088	0.00
GS	0.483	0.483	0.483	0.483	0.483	0.483	0.483	0.483	0.483	0.483	31.36
MC-BU(2)	0.002	0.002	0.004	0.004	0.024	0.024	0.068	0.068	0.132	0.131	0.08
MC-BU(4)	0.001	0.001	0.002	0.002	0.018	0.018	0.046	0.045	0.110	0.110	0.08
MC-BU(6)	0.000	0.000	0.000	0.000	0.004	0.004	0.038	0.038	0.106	0.106	0.12
MC-BU(8)	0.000	0.000	0.000	0.000	0.002	0.002	0.023	0.023	0.091	0.091	0.19
N=100, P=4, 50 instances, w*=11											
IBP	0.000	0.000	0.000	0.000	0.002	0.002	0.013	0.013	0.075	0.075	0.00
GS	0.506	0.506	0.506	0.506	0.506	0.506	0.506	0.506	0.506	0.506	39.85
MC-BU(2)	0.006	0.006	0.015	0.015	0.043	0.043	0.093	0.094	0.157	0.157	0.19
MC-BU(4)	0.006	0.006	0.017	0.017	0.049	0.049	0.104	0.102	0.158	0.158	0.19
MC-BU(6)	0.005	0.005	0.011	0.011	0.035	0.034	0.071	0.074	0.151	0.150	0.29
MC-BU(8)	0.002	0.002	0.004	0.004	0.022	0.022	0.059	0.059	0.121	0.122	0.71
MC-BU(10)	0.001	0.001	0.001	0.001	0.008	0.008	0.033	0.032	0.101	0.102	1.87

Table 3: BER for coding networks

values are reported in each table.

Comment: Note that since our evaluation measures are based on comparing against exact figures, we had to restrict the instances to be relatively small or sparse enough to be managed by exact algorithms.

For all the problems, Gibbs sampling performed consistently poorly so we only include part of the results in the following tables.

Random noisy-or networks results are summarized in Table 1. For NHD, both IBP and MC-BU gave perfect results. For the other measures, we noticed that IBP is more accurate for no evidence by about an order of magnitude. However, as evidence is added, IBP's accuracy decreases, while MC-BU's increases and they give similar results. We also notice that MC-BU gets better as the accuracy parameter i increases, which shows its anytime behavior. We also observed a similar pattern of behavior when experimenting with smaller noisy-or networks, generated with P=2 (w*=10).

General random networks results are summarized in Table 2. They are in general similar to those for random noisy-or networks. NHD is non-zero in this case. Again, IBP has the best result only for few evidence variables. It is remarkable how quickly MC-BU surpasses the performance of IBP as evidence is added. We also experimented with larger networks generated with P=3 (w*=16) and observed a similar behavior.

N=225, 10 instances, $mean$ operator				
$\|e\|$ = 0, 10, 20, 30	NHD	Abs. Error	Rel. Error	Time
	0.0094	0.0037	0.0080	0.071
IBP	0.0665	0.0665	0.0761	0.070
	0.1205	0.0463	0.1894	0.068
	0.1462	0.0632	0.1976	0.062
	0.5178	0.1096	0.2688	9.339
GS	0.5047	0.5047	0.3200	9.392
	0.4849	0.1232	0.4009	9.524
	0.4692	0.1335	0.4156	9.220
	0.1256	0.0474	0.1071	0.049
MC-BU(2)	0.1312	0.1312	0.1070	0.041
	0.1371	0.0523	0.1205	0.042
	0.1287	0.0512	0.1201	0.053
	0.1050	0.0356	0.0775	0.217
MC-BU(6)	0.0944	0.0944	0.0720	0.064
	0.0844	0.0313	0.0701	0.059
	0.0759	0.0286	0.0652	0.120
	0.0406	0.0146	0.0313	0.500
MC-BU(10)	0.0358	0.0358	0.0288	0.368
	0.0337	0.0122	0.0272	0.484
	0.0256	0.0116	0.0265	0.468
	0.0233	0.0081	0.0173	2.315
MC-BU(14)	0.0209	0.0209	0.0152	2.342
	0.0146	0.0055	0.0126	2.225
	0.0118	0.0046	0.0105	2.350
	0.0089	0.0031	0.0065	10.990
MC-BU(17)	0.0116	0.0116	0.0069	10.105
	0.0063	0.0022	0.0048	9.381
	0.0036	0.0017	0.0038	9.573

Table 4: Performance on 15x15 grid, w*=22

N=54, 50 instances								
0 $\|e\|$ 10 20	NHD		Abs. Error		Rel. Error		Time	
	max	mean	max	mean	max	mean	max	mean
	0.01852		0.00032		0.00064		2.450	
IBP	0.15727		0.03307		0.07349		2.191	
	0.20765		0.05934		0.14202		1.561	
	0.49444		0.07797		0.18034		17.247	
GS	0.51409		0.09002		0.21298		17.208	
	0.48706		0.10608		0.26853		17.335	
	0.16667	0.07407	0.02722	0.01221	0.05648	0.02520	0.154	0.153
MC-BU(2)	0.11636	0.07636	0.02623	0.01843	0.05581	0.03943	0.096	0.095
	0.10529	0.07941	0.02876	0.02196	0.06357	0.04878	0.067	0.067
	0.18519	0.09259	0.02488	0.01183	0.05128	0.02454	0.157	0.155
MC-BU(5)	0.10727	0.07682	0.02464	0.01703	0.05239	0.03628	0.112	0.112
	0.08059	0.05941	0.02174	0.01705	0.04790	0.03778	0.090	0.087
	0.12963	0.07407	0.01487	0.00619	0.03047	0.01273	0.438	0.446
MC-BU(8)	0.06591	0.05000	0.01590	0.01040	0.03394	0.02227	0.369	0.370
	0.03235	0.02588	0.00977	0.00770	0.02165	0.01707	0.292	0.294
	0.11111	0.07407	0.01133	0.00688	0.02369	0.01434	2.038	2.032
MC-BU(11)	0.02818	0.01500	0.00600	0.00398	0.01295	0.00869	1.567	1.571
	0.00353	0.00353	0.00124	0.00101	0.00285	0.00236	0.867	0.869

Table 5: Performance on cpcs54.erg, w*=15

Random coding networks results are given in Tables 3. The instances fall within the class of linear block codes, (σ is the channel noise level). It is known that IBP is very accurate for this class. Indeed, these are the only problems that we experimented with where IBP outperformed MC-BU throughout. The anytime behavior of MC-BU can again be seen in the variation of numbers in each column.

Grid networks results are given in Table 4. We only report results with $mean$ operator for a 15x15 grid for which the induced width is w*=22. We notice that IBP is more accurate for no evidence and MC is better as more evidence is added. The same behavior was consistently manifested for smaller grid networks that we experimented with (from 7x7 up to 14x14).

CPCS networks results. We also tested on three CPCS benchmark files. The results are given in Tables 5, 6 and 7. It is interesting to notice that the MC scheme scales up

N=360, 5 instances, $mean$ operator				
$\|e\|$ = 0, 20, 40	NHD	Abs. Error	Rel. Error	Time
	0.0000	0.0027	0.0054	82
IBP	0.0112	0.0256	3.4427	76
	0.0363	0.0629	736.1080	60
	0.0056	0.0125	0.0861	16
MC-BU(8)	0.0041	0.0079	0.0785	14
	0.0113	0.0109	0.2997	9
	0.0000	0.0072	0.0562	71
MC-BU(12)	0.0006	0.0052	0.0525	71
	0.0063	0.0067	0.0796	60
	0.0000	0.0015	0.0123	775
MC-BU(16)	0.0000	0.0023	0.0155	784
	0.0000	0.0009	0.0080	548

Table 6: Performance on cpcs360.erg, w*=20

N=422, 1 instance, $mean$ operator				
$\|e\|$ = 0, 20, 40	NHD	Abs. Error	Rel. Error	Time
	0.0024	0.0062	0.0150	2838
IBP	0.0721	0.0562	7.5626	2367
	0.0654	0.0744	37.5096	2150
	0.0687	0.0455	1.4341	161
MC-BU(3)	0.0373	0.0379	0.9792	85
	0.0366	0.0233	2.8384	48
	0.0545	0.0354	0.1531	146
MC-BU(7)	0.0249	0.0253	0.3112	77
	0.0262	0.0164	0.5781	45
	0.0166	0.0175	0.0738	152
MC-BU(11)	0.0448	0.0352	0.6113	95
	0.0340	0.0237	0.6978	63
	0.0024	0.0039	0.0145	526
MC-BU(15)	0.0398	0.0278	0.5338	564
	0.0183	0.0113	0.5248	547

Table 7: Performance on cpcs422.erg, w*=23

even to fairly large networks, like the real life example of CPCS422 (induced width 23). IBP is again slightly better for no evidence, but is quickly surpassed by MC when evidence is added.

Discussion and conclusion

The paper presents an approximation scheme for probabilistic inference, the single most important task over belief networks. The scheme, called mini-clustering, is governed by a controlling parameter that allows adjustable levels of accuracy and efficiency in an anytime style.

We presented empirical evaluation of mini-cluster approximation on several classes of networks, comparing its anytime performance with competing algorithms such as Gibbs Sampling and Iterative Belief Propagation, over benchmarks of noisy-or random networks, general random networks, grid networks, coding networks and CPCS type networks. Our results show that, as expected, IBP is superior to all other approximations for coding networks. However, for random noisy-or, general random networks, grid networks and the CPCS networks, in the presence of evidence, the mini-clustering scheme is often superior even in its weakest form. Gibbs sampling was particularly bad and we believe that enhanced variants of Monte Carlo approach, such as likelihood weighting and importance sampling, should be compared with (Cheng & Druzdzel 2000). The empirical results are particularly encouraging as we use an unoptimized scheme that exploits a universal principle applicable to many reasoning tasks.

The contribution of the current paper beyond recent works in this area (Dechter & Rish 1997; Dechter, Kask, & Larossa 2001) is in: 1. Extending the partition-based approximation for belief updating from mini-buckets to general tree-decompositions, thus allowing the computation of the updated beliefs for all the variables at once. This extension is similar to the one proposed in (Dechter, Kask, & Larossa 2001) but replaces optimization with probabilistic inference. 2. Providing for the first time empirical evaluation demonstrating the effectiveness of the partition-based idea for belief updating.

There are many potential ways for improving the MC scheme. Among the most important, the partitioning step can be further elaborated. In our present work we used only a brute-force approach for partitioning. It remains to be seen if more refined schemes can make the upper/lower bounds on the joint belief tighter, and if normalizing such results gives a better estimate of exact belief.

One extension we recently pursued (Dechter, Kask, & Mateescu 2002) is an iterative version of MC called Iterative Join-Graph Propagation (IJGP), which is both anytime and iterative and belongs to the class of generalized belief propagation methods (Yedidia, Freeman, & Weiss 2001). Rather than assuming an underlying join-tree, IJGP works on a join-graph that may contain loops. IJGP is related to MC in a similar way as IBP is related to BP (Pearl's belief propagation). Experimental work shows that in most cases iterating improves the quality of the MC approximation even further, especially for low i-bounds.

Acknowledgments

This work was supported in part by NSF grant IIS-0086529 and by MURI ONR award N00014-00-1-0617.

References

Arnborg, S. A. 1985. Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey. *BIT* 25:2–23.

Boyen, X., and Koller, D. 1998. Tractable inference in complex stochastic processes. In *Artificial Intelligence (UAI'98)*, 33–42.

Cheng, J., and Druzdzel, M. 2000. AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *Journal of Artificial Intelligence Research* 13:155–188.

Cooper, G. 1990. The computational complexity of probabistic inferences. *Artificial Intelligence* 393–405.

Dagum, P., and Luby, M. 1993. Approximating probabilistic inference in bayesian belief networks is np-hard. In *National Conference on Artificial Intelligence (AAAI-93)*.

Dechter, R., and Rish, I. 1997. A scheme for approximating probabilistic inference. In *Artificial Intelligence (UAI'97)*, 132–141.

Dechter, R.; Kask, K.; and Larossa, J. 2001. A general scheme for multiple lower bound computation in constraint optimization. In *Constraint Programming*.

Dechter, R.; Kask, K.; and Mateescu, R. 2002. Iterative join-graph propagation. Technical report, UCI.

Dechter, R. 1996. Bucket elimination: A unifying framework for probabilistic inference algorithms. In *Uncertainty in Artificial Intelligence (UAI'96)*, 211–219.

Gottlob, G.; Leone, N.; and Scarello, F. 1999. A comparison of structural CSP decomposition methods. *IJCAI-99*.

Henrion, M. 1986. Propagating uncertainty by logic sampling. In *Technical report, Department of Engineering and Public Policy, Carnegie Melon University*.

Jaakkola, T. S., and Jordan, M. I. 1996. Recursive algorithms for approximating probabilities in graphical models. *Advances in Neural Information Processing Systems* 9.

Jensen, F.; Lauritzen, S.; and Olesen, K. 1990. Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly* 4:269–282.

Kask, K. 2001. Approximation algorithms for graphical models. Technical report, Ph.D. thesis, Information and Computer Science, University of California, Irvine.

Kjæaerulff, U. 1994. Reduction of computational complexity in bayesian networks through removal of weak dependencies. In *Uncertainty in Artificial Intelligence (UAI'94)*.

Koller, D.; Lerner, U.; and Angelov, D. 1998. A general algorithm for approximate inference and its applciation to hybrid bayes nets. In *Uncertainty in Artificial Intelligence (UAI'98)*, 324–333.

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

McEliece, R.; MacKay, D.; and Cheng, J.-F. 1997. Turbo decoding as an instance of pearl's belief propagation algorithm. *IEEE J. Selected Areas in Communication*.

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

Roth, D. 1996. On the hardness of approximate reasoning. *AI* 82(1-2):273–302.

Shachter, R. D., and Peot, M. A. 1989. Simulation approaches to general probabilistic inference on belief networks. In *Uncertainty in Artificial Intelligene (UAI'89)*.

Shafer, G. R., and Shenoy, P. 1990. Probability propagation. *Anals of Mathematics and Artificial Intelligence* 2:327–352.

Weiss, Y. 1997. Belief propagation and revision in networks with loops. In *NIPS*.

Welling, M., and Teh, Y. 2001. Belief optimization for binary networks: A stable alternative to loopy belief propagation. In *UAI'01*, 554–561.

Yedidia, J. S.; Freeman, W.; and Weiss, Y. 2001. Generalized belief propagation. In *Advances in Neural Information Processing Systems 13*.

Zhang, N., and Poole, D. 1994. A simple algorithm for bayesian network computations. In *Proc of the tenth Canadian Conference on Artificial Intelligence*, 171–178.