New Mini-Bucket Partitioning Heuristics for Bounding the Probability of Evidence

Emma Rollon and **Rina Dechter**
Department of Information and Computer Science
University of California, Irvine
{erollon, dechter}@ics.uci.edu

Abstract

Mini-Bucket Elimination (MBE) is a well-known approximation algorithm deriving lower and upper bounds on quantities of interest over *graphical models*. It relies on a procedure that partitions a set of functions, called *bucket*, into smaller subsets, called *mini-buckets*. The method has been used with a single partitioning heuristic throughout, so the impact of the partitioning algorithm on the quality of the generated bound has never been investigated. This paper addresses this issue by presenting a framework within which partitioning strategies can be described, analyzed and compared. We derive a new class of partitioning heuristics from first-principles geared for likelihood queries, demonstrate their impact on a number of benchmarks for probabilistic reasoning and show that the results are competitive (often superior) to state-of-the-art bounding schemes.

Introduction

Mini-Bucket Elimination (MBE) (Dechter and Rish 2003) is one of the most popular bounding techniques for reasoning tasks defined over *graphical models*. The power of MBE has been demonstrated for optimization tasks such as finding the most likely tuple of a probabilistic network, or finding the optimal solution for a weighted csp (Dechter and Rish 2003; Kask and Dechter 2001; Marinescu and Dechter 2007; Choi, Chavira, and Darwiche 2007) showing its power in producing lower-bounding heuristics to guide search. In this paper, however, we focus on the more challenging task of weighted counting which includes finding the probability of evidence over Bayesian networks, determining the partition function over Markov networks, and counting solutions of a constraint network. These tasks are $\#P$-complete and are central to both probabilistic and deterministic reasoning.

MBE derives bounds by applying exact Bucket Elimination (BE) algorithm (Dechter 1999; Bertele and Brioschi 1972) on a simplified version of the problem. In BE all the functions in the *bucket* are processed together, yielding a single *bucket's function* defined over the union of their arguments. Since this processing can be computationally expensive, MBE partitions the bucket into smaller *mini-buckets*, such that the number of variables in each mini-bucket is bounded by $z + 1$, for a given constant z. Then, MBE processes each mini-bucket independently, yielding a set of mini-bucket functions defined over smaller subsets of variables which together bound the bucket's function. The mini-bucket scheme can be interpreted as the application of exact algorithm (e.g., BE) to a relaxation of the problem which is obtained via node duplication (Kask and Dechter 2001; Choi, Chavira, and Darwiche 2007).

The partitioning of a bucket into z bounded mini-buckets can be carried out in many ways, each resulting in a different impact on the overall accuracy. In all the earlier work, the partitioning heuristic, was *scope-based*, relying solely on the functions arguments. It aimed at minimizing the number of mini-buckets in the partitioning. Its effectiveness compared against random partitioning heuristics was sporadically demonstrated, but no systematic study was ever carried out.

Early attempts to evaluate the bounding power of the mini-bucket scheme for likelihood queries were abandoned due to very discouraging initial results. Focus was shifted instead to treating the mini-bucket scheme as mere approximation with no guarantees (Mateescu, Dechter, and Kask 2002).

In this paper we develop a new class of *content-based* mini-bucket partitioning heuristics for counting-based queries that consult the function's contents in addition to their scopes. We show that the resulting set of heuristics provides an ensemble of bounds that taken together yield a more effective bounding scheme across a varied set of benchmarks. We demonstrate that our MBE based bounding scheme is often far more accurate than recent competing schemes (e.g., Tree-Reweighted Belief Propagation (Wainwright, Jaakkola, and Willsky 2003), Box-Prop (Mooij and Kappen 2008), and the Any-Time Bounding scheme (Bidyuk and Dechter 2006)) over benchmarks such as *noisy-or bayesian networks*, *coding networks* and *genetic linkage analysis*.

Background

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be an ordered set of variables and $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ an ordered set of domains, where D_{x_i} is the finite set of potential values for x_i. The assignment of variable x_i with $a \in D_{x_i}$ is noted $(x_i = a)$. A *tuple* t is an ordered set of assignments to different variables $(x_{i_1} = a_{i_1}, \ldots, x_{i_k} = a_{i_k})$. The *scope* of t, noted $var(t)$, is the set

of variables that it assigns.

Belief Networks

A *Bayesian network* (Pearl 1988) is a quadruple $(\mathcal{X}, \mathcal{D}, \mathcal{G}, \mathcal{P})$ where $\mathcal{G}$ is a directed acyclic graph over $\mathcal{X}$ and $\mathcal{P} = \{p_1, \ldots, p_n\}$, where $p_i = P(x_i|pa_i)$ denotes the conditional probability tables (CPTs). The set pa_i is the set of parents of the variable x_i in $\mathcal{G}$. A Bayesian network represents a probability distribution $P(\mathcal{X}) = \prod_{i=1}^{n} P(x_i|pa_i)$. Given a Bayesian network and an evidence tuple e over a subset of variables $E \subseteq \mathcal{X}$, the *probability of evidence* $P(e)$ is defined as: $P(e) = \sum_{\mathcal{X}-var(e)} \prod_{i=1}^{n} P(x_i|pa_i)_{|e}$ where $f(X)_{|e}$ is a new function h defined over $\mathcal{X} - var(e)$ such that the variables E are assigned to e.

Bucket and Mini-Bucket Elimination

Bucket elimination (BE) (Dechter 1999; Bertele and Brioschi 1972) is an exact algorithm for answering a variety of queries over graphical models. In particular, given a Bayesian network $(\mathcal{X}, \mathcal{D}, \mathcal{G}, \mathcal{P})$, BE computes the probability of evidence e as shown in the following pseudo-code:

function BE$((\mathcal{X}, \mathcal{D}, \mathcal{G}, \mathcal{P}), e)$
1. $\mathcal{S} := \{f_{|e} \mid f \in \mathcal{P}\}$; $\mathcal{X} := \mathcal{X} - var(e)$;
2. **while** $\mathcal{X} \neq \emptyset$ **do**
3. $\quad x := select\_variable(\mathcal{X})$;
4. $\quad \mathcal{B}_x := \{f \in \mathcal{S} \mid x \in var(f)\}$;
5. $\quad g_x := \sum_x (\prod_{f \in \mathcal{B}_x} f)$;
6. $\quad \mathcal{S} := \mathcal{S} - \mathcal{B}_x \cup \{g_x\}$;
7. $\quad \mathcal{X} := \mathcal{X} - \{x\}$;
8. **endwhile**
9. **return** $(P(e) = \prod_{f \in \mathcal{S}} f())$;

endfunction

After incorporating the evidence in the network (line 1), BE eliminates the remaining variables $\mathcal{X} - var(e)$ one at a time. The elimination of variable x is as follows. First, BE computes the so called *bucket* of variable x, noted $\mathcal{B}_x$, which contains all the functions in $\mathcal{S}$ having x in their scope (line 4). Next, it computes the function g_x of bucket $\mathcal{B}_x$ (line 5),

Definition 1 (the function of a bucket) *Given a bucket* $\mathcal{B}_x = \{f_1, \ldots, f_r\}$, *the* function represented by the bucket $\mathcal{B}_x$ *is* $g_x = \sum_x \prod_{f \in \mathcal{B}_x} f$.

When all variables have been eliminated, $\mathcal{S}$ contains a set of empty-scope functions (*i.e.*, a set of constants). The multiplication of those functions is the probability of evidence $P(e)$. The time and space complexity of the algorithm is exponential in a graph parameter called *induced width*, which equals the largest scope of all the functions computed.

Mini-bucket elimination (MBE) (Dechter and Rish 2003) is an approximation of full bucket elimination that bounds the exact solution when the induced width is too large. Given a bucket $\mathcal{B}_x = \{f_1 \ldots, f_m\}$, MBE generates a partition $Q = \{Q_1, \ldots, Q_p\}$ of $\mathcal{B}_x$, where each subset $Q_j \in Q$ is called *mini-bucket*. Abusing notation, the scope of a set of functions $\mathcal{F}$, noted $var(\mathcal{F})$, is the union of scopes of the functions it contains. Given an integer control parameter z, MBE restricts the arity of each of its mini-buckets to $z + 1$. We say that Q is a *z-partition*. Then, each mini-bucket is processed independently. The pseudo-code of MBE is obtained by replacing lines 5 and 6 in BE by,

5. $\{Q_1, \ldots, Q_p\} :=$ `Partition`$(\mathcal{B}_x, z)$;
5b. **for each** $j = 1 \ldots p$ **do** $g_{x,j} := \sum_x (\prod_{f \in Q_j} f)$;
6. $\mathcal{S} := (\mathcal{S} \cup \{g_{x,1}, \ldots, g_{x,p}\}) - \mathcal{B}_x$;

Definition 2 (the function of a bucket partition) *Given a partition* $Q = \{Q_1, \ldots, Q_p\}$ *of a bucket* $\mathcal{B}_x$, *the* function represented by the partition Q *is* $g_x^Q = \prod_{j=1}^{p} \sum_x \prod_{f \in Q_j} f$.

The time and space complexity of MBE is $O(exp(z+1))$ and $O(exp(z))$, respectively. The parameter z allows trading time and space for accuracy: as the partitions are more coarced, both the complexity and accuracy of the algorithm increase.

Definition 3 (refinement relation) *A partition Q is a* refinement *of a partition Q', noted $Q \sqsubset Q'$, iff $Q \neq Q'$ and every element of Q is a subset of some element of Q'. The finest partition, noted $Q^{\perp}$, has one mini-bucket for each function, and the coarcest partition, noted $Q^{\top}$, has only one mini-bucket containing all functions.*

Theorem 1 *(Dechter and Rish 2003) Given two partitions Q and Q' of bucket $\mathcal{B}_x$, $Q \sqsubset Q' \Rightarrow \forall t,\ g_x^{Q'}(t) \leq g_x^{Q}(t)$*

In particular, given any partition Q of $\mathcal{B}_x$, $\forall t, g_x^{Q^{\top}}(t) = g_x(t) \leq g_x^{Q}(t)$. Therefore, the upper bound computed in each bucket accumulates and yields an upper bound of $P(e)$.

For the sake of readability, in the following we fix the bucket to be $\mathcal{B}_x$ and drop subindex's referring to variable x.

Scope-based Partitioning Heuristic

The *scope-based* partition heuristic (SCP) proposed in (Dechter and Rish 1997) and used since, aims at minimizing the number of mini-buckets in the partition by including in each mini-bucket as many functions as possible as long as the z bound is satisfied. First, single function mini-buckets are decreasingly ordered according to their arity. Then, each mini-bucket is absorved into the left-most mini-bucket with whom it can be merged. The time and space complexity of `Partition`$(\mathcal{B}, z)$ using the SCP heuristic is $O(|\mathcal{B}| \log(|\mathcal{B}|) + |\mathcal{B}|^2)$ and $O(exp(z))$, respectively.

One virtue of the scope-based heuristic is its small overhead. Its shortcoming is that it does not consider the actual information contained in each function.

Partitioning Framework

Given a bucket $\mathcal{B}$, the goal of the partition process is to find a z-partition Q of $\mathcal{B}$ such that g^Q is the *closest* to the bucket function g, where closeness can be defined using any distance measure d. Therefore, the partition task is to find a z-partition Q^* of $\mathcal{B}$ such that $Q^* = \arg\min_Q \{d(g^Q, g)\}$.

We considered four distance measures: *log relative error*, *maximum log relative error*, *KL divergence* and *absolute error*. For space reasons however, here we will report on the first two[1]:

- Log relative error:

$$RE(f, h) = \sum_t (\log(f(t)) - \log(h(t)))$$

[1]Full details are available on-line in a technical report at http: www.ics.uci.edu/∼dechter/publications.html

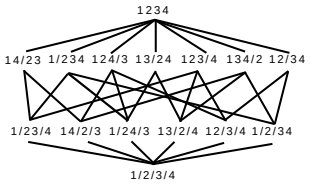


Figure 1: Partitioning lattice of bucket $\{f_1, f_2, f_3, f_4\}$. We specify each function by its subindex.

- *Max log relative error*:

$$MRE(f,h) = \max_t\{\log(f(t)) - \log(h(t))\}$$

We can organize the space of partitions in a lattice using the *refinement* relation since it yields a partial order.

Definition 4 (partitioning lattice of a bucket) *Each partition Q of $\mathcal{B}$ is a vertex in the lattice. There is an upward edge from Q to Q' if Q' results from merging two mini-buckets of Q in which case Q' is a* child *of Q. The set of all children of Q is denoted by $ch(Q)$. The* bottom *partition in the lattice is $Q^{\perp}$ while the* top *partition is $Q^{\top}$.*

Example 1 *Figure 1 shows the partitioning lattice of bucket $\mathcal{B} = \{f_1, f_2, f_3, f_4\}$.*

For any two partitions Q and Q', if Q' is a descendent of Q then $g^{Q'}$ is clearly tighter than g^{Q} (see Theorem 1). Namely,

Corollary 1 *Given two partitions Q and Q',*

$$Q \sqsubset Q' \Rightarrow d(g^{Q'}, g) \leq d(g^{Q}, g)$$

for any distance measure d defined earlier.

Since the distance to the top partition is always non-increasing along any upward path in the lattice, any optimal z-partition Q^* is *maximal* (i.e., all its children in the lattice are l-partitions where $z < l$). Therefore, we can view any partition-seeking algorithm as a traversal algorithm over the partition lattice seeking for maximal z-partitions.

Greedy Heuristic Partitioning

Since an optimal partition-seeking algorithm may need to traverse the partitioning lattice bottom-up along all paths, a computationally hard task, we will focus on depth-first greedy traversals only, as defined below. The traversal is guided by a heuristic function h defined on a partition Q and its child partition Q', denoted $Q \rightarrow Q'$.

function `GreedyPartition`($\mathcal{B}, z, h$)
1. **Initialize** Q as the bottom partition of $\mathcal{B}$;
2. **while** $\exists Q' \in ch(Q)$ which is a z-partition **select**
   $Q \leftarrow \arg\min_{Q'}\{h(Q \rightarrow Q')\}$ among child z-partitions of Q;
3. **return** Q;

endfunction

At each step, the algorithm ranks each child Q' of the current partition Q according to h. Clearly, each iteration is guaranteed to tighten the resulting bound.

Proposition 1 *The time complexity of* `GreedyPartition` *is $O(|\mathcal{B}| \times T)$ where $O(T)$ is the time complexity of selecting the min child partition according to h.*

It is natural to use $h(Q \rightarrow Q') \triangleq d(g^{Q'}, g)$.

Definition 5 (greedily optimal partitioning heuristic) *Given a bucket $\mathcal{B}$ and its bucket function g, and a partition Q of $\mathcal{B}$, a partitioning heuristic h is* greedily optimal *relative to d iff $\forall Q', Q'' \in ch(Q)$:*

$$h(Q \rightarrow Q') \leq h(Q \rightarrow Q'') \Leftrightarrow d(g^{Q'}, g) \leq d(g^{Q''}, g)$$

By definition, when using $h(Q \rightarrow Q') \triangleq d(g^{Q'}, g)$, h is greedily optimal. However, computing such an h is time exponential in the arity of g (i.e., $T = O(exp(var(\mathcal{B})))$), and therefore impractical.

We therefore focus on local heuristics. Let $Q^{jk} \in ch(Q)$ denote the child z-partition of Q that results from merging mini-buckets $Q_j, Q_k \in Q$.

Definition 6 (local partitioning heuristic) *A partitioning heuristic h is* local *iff for any $Q^{jk} \in ch(Q)$, $h(Q \rightarrow Q^{jk})$ depends on the merged mini-buckets $Q_j, Q_k \in Q$ only.*

The virtue of local heuristics is that they can be computed in time exponential in z only (i.e., T in Proposition 1 satisfies $T = O(exp(z))$).

Content-based heuristics. We will next show that we can derive a partition heuristic which is both greedily optimal relative to the log relative error and local. We first define a *local* log relative error distance measure and then show that, when used as a guiding heuristic, it is greedily optimal relative to the log relative error.

Let us denote the number of tuples over a set of variables $\mathcal{Y}$ by $\mathcal{W}(\mathcal{Y})$ (i.e., $\mathcal{W}(\mathcal{Y}) = \prod_{y \in \mathcal{Y}} |D_y|$). Then,

Definition 7 (local RE) *The* local RE *(LRE) between a partition Q and its child Q^{jk} is*

$$LRE(g^{Q}, g^{Q^{jk}}) = \frac{1}{\mathcal{W}(var(Q_j \cup Q_k))} RE(g^{\{Q_j, Q_k\}}, g^{Q_j \cup Q_k})$$

Note that LRE is only defined by the two merged mini-buckets in Q^{jk}. This local function captures the gain due to merging Q_j and Q_k, independent of other mini-buckets in Q.

We next provide the main theorem.

Theorem 2 *Given a bucket $\mathcal{B}$ and its bucket function g, and given a partition $Q = \{Q_1, \ldots, Q_p\}$ of $\mathcal{B}$ and two child z-partitions Q^{jk} and Q^{lm} of Q,*

$$RE(g^{Q^{jk}}, g) \leq RE(g^{Q^{lm}}, g) \Longleftrightarrow$$
$$-LRE(g^{Q}, g^{Q^{jk}}) \leq -LRE(g^{Q}, g^{Q^{lm}})$$

Sketch of proof. First, let us suposse that $(Q_j \cup Q_k) \cap (Q_l \cup Q_m) = \emptyset$. $RE(g^{Q^{jk}}, g) \leq RE(g^{Q^{lm}}, g)$, iff

$$\sum_t \log\left[g^{Q_1}(t) \times \ldots \times g^{Q_j \cup Q_k}(t) \times \ldots \times g^{Q_p}(t)\right] \leq$$
$$\sum_t \log\left[g^{Q_1}(t) \times \ldots \times g^{Q_l \cup Q_m}(t) \times \ldots \times g^{Q_p}(t)\right]$$

where $var(t) = var(\mathcal{B})$. Using properties of log function, reordering and cancelling, the previous expression yields:

$$\sum_t (\log\left[g^{Q_l}(t) \times g^{Q_m}(t)\right] - \log\left[g^{Q_l \cup Q_m}(t)\right]) \leq$$
$$\sum_t (\log\left[g^{Q_j}(t) \times g^{Q_k}(t)\right] - \log\left[g^{Q_j \cup Q_k}(t)\right])$$

Instead of summing over all tuples in the bucket's scope, we can sum over the tuples in the scopes of the mini-buckets involved in each side of the inequality and weigh each side by its number of extensions to the full scope. Note that the number of extension of a tuple t' to the full scope of $\mathcal{B}$ is $\frac{\mathcal{W}(var(\mathcal{B}))}{\mathcal{W}(var(t'))}$. Then, the previous expression can be rewritten as,

$$\frac{1}{\mathcal{W}(var(t'))} \sum_{t'} (\log\left[g^{Q_l}(t') \times g^{Q_m}(t')\right] - \log\left[g^{Q_l \cup Q_m}(t')\right]) \leq$$
$$\frac{1}{\mathcal{W}(var(t''))} \sum_{t''} (\log\left[g^{Q_j}(t'') \times g^{Q_k}(t'')\right] - \log\left[g^{Q_j \cup Q_k}(t'')\right])$$

where $var(t') = var(Q_l \cup Q_m)$ and $var(t'') = var(Q_j \cup Q_k)$, which by definition is

$$-LRE(g^Q, g^{Q^{jk}}) \leq -LRE(g^Q, g^{Q^{lm}})$$

The derivation when $(Q_j \cup Q_k) \cap (Q_l \cup Q_m) \neq \emptyset$ is similar. Therefore, we can conclude that the theorem holds. □

In words, the theorem states that when using $h(Q \rightarrow Q') \triangleq -LRE(g^Q, g^{Q'})$ as the greedy partitioning heuristic it yields a greedily optimal heuristic relative to distance RE. The virtue of LRE is that it is local and, therefore, computationally feasible.

Proposition 2 *Given a bucket $\mathcal{B}$ and a value of z, using $h(Q \rightarrow Q') \triangleq -LRE(g^Q, g^{Q'})$ the time and space complexity of* `GreedyPartition` *is $O(|\mathcal{B}|^3 \times exp(z))$ and $O(exp(z))$, respectively.*

We investigated all other distance measures, but unfortunately none lead to a local partitioning heuristic which is greedily optimal. Namely, greedily optimal heuristics relative to those distance measures seem to be inherently exponential in the bucket arity.

Nevertheless, we define two distance measures derived based on approximating an MRE-based greedily optimal heuristic that yield two local heuristics[2]. The first one, denoted $MRE1$, is defined by

$$MRE1(g^Q, g^{Q^{jk}}) = \max_t \{\log\left[g^{\{Q_j, Q_k\}}\right]\} + - \max_t \{\log\left[g^{Q_j \cup Q_k}\right]\}$$

and the second one, denoted $LMRE$, is defined by

$$LMRE(g^Q, g^{Q^{jk}}) = MRE(g^{\{Q_j, Q_k\}}, g^{Q_j \cup Q_k})$$

The time and space complexity of `GreedyPartition` with any of these two heuristics is captured by Proposition 2.

We also developed local partitioning heuristics using the KL divergence and the absolute error which, for space reasons, we do not report here.

[2]Full details are available on-line in a technical report at http: www.ics.uci.edu/∼dechter/publications.html

Empirical Evaluation

We evaluated the performance of the mini-bucket partitioning heuristics and compare with state of the art bounding schemes. We also considered the impact of a combined approach, taking the minimum upper bound over all partitioning heuristics. This combined MBE approach is more time consuming: linear in the number of participating partition heuristics.

We compare against three recently introduced bounding schemes: (i) the any-time bounding scheme ATB (Bidyuk and Dechter 2006) which depends on a control parameter H and uses an improved version of Bound-Propagation (Leisink and Kappen 2003) as plug-in algorithm; (ii) BoxProp (Mooij and Kappen 2008), which was derived for bounding posterior probabilities (and therefore P(e) is obtained by applying the chain rule to individual bounds on posteriors); and (iii) Tree-Reweighted Belief Propagation (Wainwright, Jaakkola, and Willsky 2003).

We conduct our empirical evaluation on three benchmarks [3]. We report log upper bound on P(e) and cpu time in seconds. The value of z reported is the highest feasible value given the available memory (2GB ram). The value of H reported is indicated in each benchmark. MBE uses the variable ordering established by the *min-fill* heuristic after instantiating evidence variables. We will index each run of MBE using `GreedyPartition` by the type of heuristic function h used.

Coding Networks. Figure 2 (top row) reports the results. The exact P(e) is too hard to compute. Looking at the different partitioning heuristics we see that for each instance there is a huge range of upper bounds. The improvement of the best partitioning heuristic over the second best is always higher than 50% and of orders of magnitude on some instances (e.g., 3 on *BN_130* and 1 on *BN_132*). SCP and LMRE heuristics obtain the best upper bounds on 4 instances each, while LRE does on one. The least accurate heuristic is MRE1. All content-based heuristics are 2 to 3 times slower than the scope-based heuristic (SCP computes bounds in around 30 seconds, while content-based heuristics in around $60 - 80$ seconds). The reason is that during the traversal of the partitioning lattice content-based heristics have to compute intermediate functions.

Another, more detailed view of the relative performance of the different heuristics as a function of their bound z is given in the bottom right graph. It reinforces our analysis above, and also shows the changes as a function of z. Similar behaviour was observed in other benchmarks.

Comparing MBE with alternative approaches, we see that BoxProp is clearly the worst algorithm on this benchmark. Otherwise, all the partitioning heuristics outperform ATB on most of all instances. All the partitioning heuristics are superior to TRW (SCP and LRE heuristics outperform TRW on 6 instances, LMRE on 7, and MRE1 on 5). When we consider the combination of all partitioning heuristics, the resulting combined MBE approach outperforms ATB and TRW on all instances but *BN_129*. In general, the improve-

[3]All instances are included in the UAI08 evaluation: http://graphmod.ics.uci.edu/uai08/Software

ment of the combined approach with respect to ATB and TRW is of orders of magnitude. In terms of cpu time, BoxProp computes upper bounds is around 60 seconds, the combined MBE scheme is around 190 seconds, TRW in around 500 seconds, and ATB in around 1300 seconds.

Noisy-or Bayesian Networks. Figure 2 (second row) reports the results. Looking at the different partitioning heuristics we see that the SCP heuristic computes upper bounds greater than the trivial upper bound of 1 on 8 instances. Content-based heuristics however compute informative upper bounds in all cases. For each instance, the range of upper bounds depends on the value of P(e): small when the evidence is very probable (i.e., P(e) near 1), while of orders of magnitude when the evidence is small (i.e., P(e) smaller than 10^{-4}). The improvement of the best over the second best upper bound is typically higher than 25% on instances '$*$a', while higher than 10% on instances '$*$b'. $LMRE$ is the most accurate heuristic in this benchmark, obtaining the best upper bound on 11 instances, while SCP is the least accurate. As expected, content-based heuristics are slower than SCP heuristic (from 2 to 6 times slower).

Comparing with competing approaches, we see again that BoxProp is the worst, obtaining upper bounds around 0.9 for all instances, even when P(e) is smaller than 10^{-4}. TRW is the most accurate approach on this benchmark. The gap between the upper bounds computed by TRW and any of the MBE schemes seems to depend on the value of P(e). For instances with high P(e), the gap is very small, and it increases for instances with relatively small P(e). Disregarding uninformative upper bounds computed by SCP, all content-based heuristics outperform ATB.

Linkage Analysis. Figure 2 (third row) reports the results. Comparison with ATB and BoxProp was not possible. Both algorithms require a Bayesian network and an independent set of evidence. However, the pedigree instances we have already incorporate the evidence into the network.

Looking at the different partitioning heuristics we see that the improvement of the best over the second best upper bound is typically higher than 30% and, in some case, up to orders of magnitude (e.g., see *ped31*, *ped33*, *ped34*, *ped38* and *ped41*). SCP obtains the best upper bound on 7 instances, while each content-based heuristic is best on 4 instances (all heuristics obtain the same upper bound on *ped20*). The content-based heuristics are typically 2 to 3 times slower than the SCP heuristic.

TRW is clearly not good on this benchmark. For some instances, TRW computes upper bounds greater than the trivial upper bound of 1. Moreover, TRW requires in almost all cases 1 to 2 orders of magnitude more computation time. The combined MBE scheme increases further the gap on accuracy compared with TRW.

Conclusions

The paper investigates new heuristic schemes for mini-bucket partitioning that consider the actual function's content for generating upper bounds on the probability of evidence over Bayesian networks and partition function over Markov networks. Its contribution is (i) introducing the first systematic investigation of partition heuristics for the mini-bucket scheme and, (ii) showing that MBE is competitive with other schemes for upper-bounding counting type tasks.

We consider several guiding distance measures and we show that in general a greedily optimal heuristic is computationally too expensive. However, for the log relative error we derive a greedily optimal heuristic that is time and space exponential in the control parameter z only. We also derive a set of local heristics based on other distance measures that approximate the greedy optimallity goal.

Our experimental results show that the mini-bucket scheme is competitive with recent state of the art bounding schemes and, often, far more accurate. Most remarkably, the combination of all heuristics yields a far more effective bounding scheme at the cost of only a linear increase in time. Note that one way to improve the mini-bucket accuracy is by increasing the value of the control parameter z. However, the largest feasible value of z is bounded by memory and not by its computation time, which is typically a few seconds only. Therefore, the availability of various heuristics for the same value of z may be the only practical enhancement of MBE.

References

Bertele, U., and Brioschi, F. 1972. *Nonserial Dynamic Programming*. Academic Press.

Bidyuk, B., and Dechter, R. 2006. An anytime scheme for bounding posterior beliefs. In *AAAI*, 1095–1100.

Choi, A.; Chavira, M.; and Darwiche, A. 2007. Node splitting: A scheme for generating upper bounds in bayesian networks. In *UAI*, 57–66.

Dechter, R., and Rish, I. 1997. A scheme for approximating probabilistic inference. In *UAI*, 132–141.

Dechter, R., and Rish, I. 2003. Mini-buckets: A general scheme for bounded inference. *J. ACM* 50(2):107–153.

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

Kask, K., and Dechter, R. 2001. A general scheme for automatic generation of search heuristics from specification dependencies. *Artificial Intelligence* 129:91–131.

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

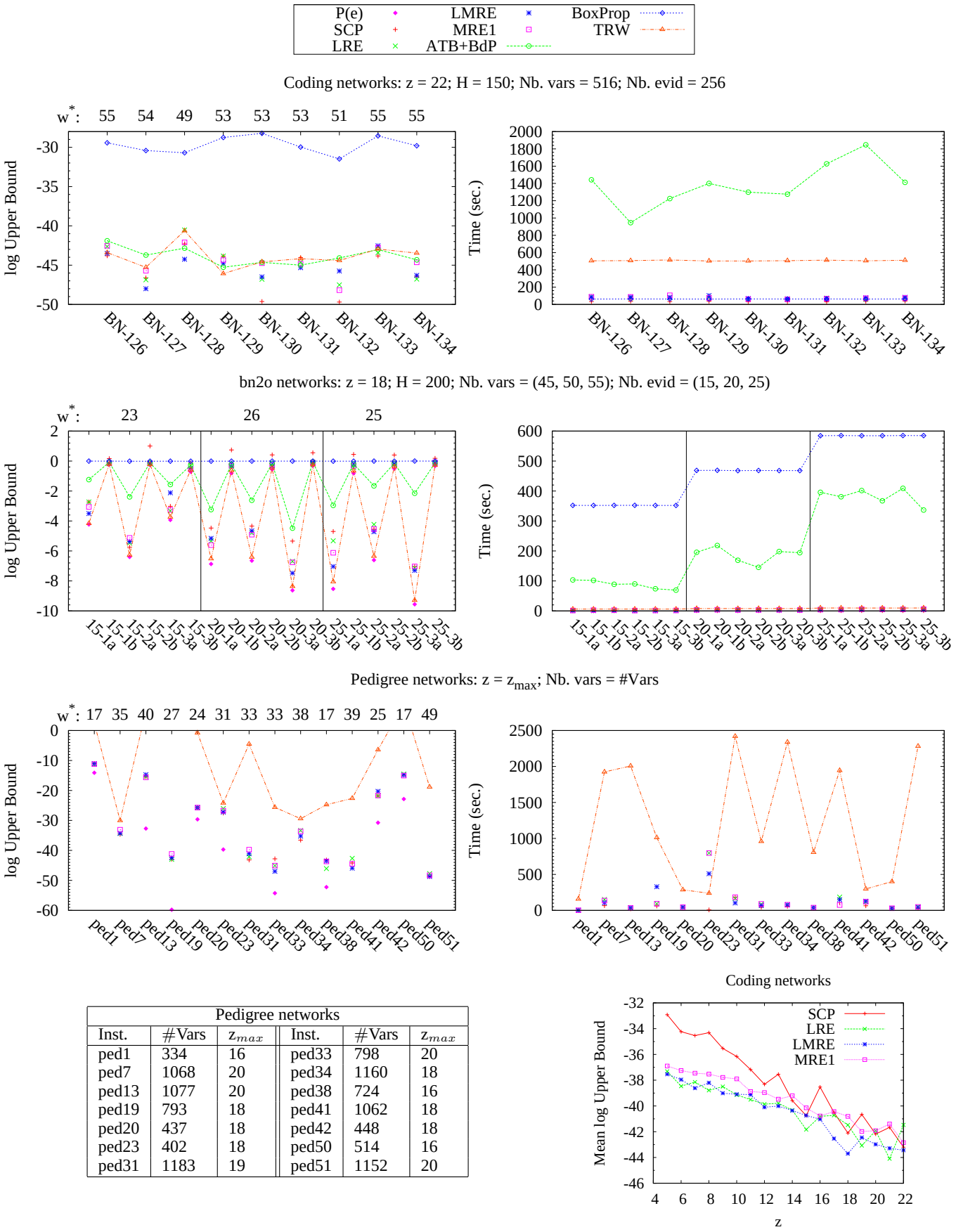
Marinescu, R., and Dechter, R. 2007. Best-first and/or search for graphical models. In *AAAI*, 1171–1176.

Mateescu, R.; Dechter, R.; and Kask, K. 2002. Tree approximation for belief updating. In *AAAI/IAAI*, 553–559.

Mooij, J. M., and Kappen, H. J. 2008. Bounds on marginal probability distributions. In *NIPS*, 1105–1112.

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

Wainwright, M. J.; Jaakkola, T. S.; and Willsky, A. S. 2003. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In *AISTATS*.



Pedigree networks					
Inst.	#Vars	z_{max}	Inst.	#Vars	z_{max}
ped1	334	16	ped33	798	20
ped7	1068	20	ped34	1160	18
ped13	1077	20	ped38	724	16
ped19	793	18	ped41	1062	18
ped20	437	18	ped42	448	18
ped23	402	18	ped50	514	16
ped31	1183	19	ped51	1152	20

Figure 2: Experimental results on *coding networks* (first row), *two-layer noisy-or networks* (second row), and *linkage analysis* (third row). Left column shows log upper bound, along with the induced width w^* of each instance on the top x-axis, and right column shows cpu time in seconds. The bottom left table indicates the number of varibles ($\#Vars$ column) and the highest feasible value of the control parameter z (z_{max} column) for each pedigree instance. The bottom right graph shows the mean log upper bound among all coding instances as a function of the control parameter z. For pedigree networks, only on instance *ped41* the $P(e)$ is not available.