Semiring-Based Mini-Bucket Partitioning Schemes

Emma Rollon and **Javier Larrosa**
{erollon, larrosa}@lsi.upc.edu
Universitat Politècnica de Catalunya, Barcelona, Spain

Rina Dechter
dechter@ics.uci.edu
University of California, Irvine, USA

Abstract

Graphical models are one of the most prominent frameworks to model complex systems and efficiently query them. Their underlying algebraic properties are captured by a valuation structure that, most usually, is a *semiring*. Depending on the semiring of choice, we can capture *probabilistic models, constraint networks, cost networks, etc*. In this paper we address the *partitioning problem* which occurs in many approximation techniques such as *mini-bucket elimination* and *join-graph propagation* algorithms. Roghly speaking, subject to complexity bounds, the algorithm needs to find a partition of a set of factors such that best approximates the whole set. While this problem has been addressed in the past in a particular case, we present here a general description. Furthermore, we also propose a general partitioning scheme. Our proposal is general in the sense that it is presented in terms of a generic semiring with the only additional requirements of a *division* operation and a *refinement* of its order. The proposed algorithm instantiates to the particular task of computing the probability of evidence, but also applies directly to other important reasoning tasks. We demonstrate its good empirical behaviour on the problem of computing the most probable explanation.

1 Introduction

The *graphical model* framework provides a common formalism to model complex systems such as probabilistic models, which includes *Markov* and *Bayesian networks* [Pearl, 1988], and deterministic models, which includes *constraint networks* [Bistarelli *et al.*, 1999] and *decision diagrams* [Dechter, 2003]. In general, a graphical model is defined by a collection of functions or *factors* $\mathcal{F}$ over a set of variables $\mathcal{X}$. Factors return values from a valuation set A. Depending on each particular case, functions may express probabilistic, deterministic or preferential information. Given a graphical model, one can compute different *reasoning tasks*. A reasoning task is defined by two operators $\oplus$ and $\otimes$, where the triplet $(A, \oplus, \otimes)$ constitutes a semiring.

Since the exact computation of reasoning tasks is in general intractable, several approximation methods exist. Some of them need to solve internally an optimization problem over the set of partitions of a set of factors. Although it is known that the quality of the approximation depends greatly on the quality of the partitions, little research has been done on it.

This paper builds on top of the recent work of [Rollon and Dechter, 2010], where a greedy scheme is proposed for solving the partitioning problem of the very specific task of computing the probability of certain evidence. Our paper generalizes the partitioning problem and the greedy scheme to general tasks on graphical models. We show that the generalization applies as long as the semiring admits a *division* operator and a *refinement* of its order, which is the most usual case. Furthermore, we show the potential of this general partitioning scheme on the task of finding the most probable explanation of probabilistic networks.

2 Preliminaries

2.1 Semirings

A *commutative semiring* [Kohlas and Wilson, 2008] is a triplet $(A, \oplus, \otimes)$, where A is a set, and $\oplus, \otimes$ are binary operations. Both operatios are associative and commutative. Additionally, $\otimes$ distributes over $\oplus$ (i.e, $(a \otimes b) \oplus (a \otimes c) = a \otimes (b \oplus c)$). Commutative semirings have a unique **0** element such that $\mathbf{0} \otimes a = \mathbf{0}$. Additionally, they implicitely define a pre-order relation $\leq$ as $a \leq b$ (i.e., b is *better* than a) iff $a = b$ or there exists $c \in A$ such that $a \oplus c = b$. In this paper we will restrict ourselves to semirings whose pre-order is a partial order.

Proposition 1 *For any semiring $(A, \oplus, \otimes)$, its associated relation $\leq$ satisfies:*

1. *$a \leq b$ and $c \leq d$ implies $a \otimes c \leq b \otimes d$.*
2. *$a \otimes b \leq c \otimes b$ implies $a \leq c$.*

In this paper we will consider *invertible* semirings [Kohlas and Wilson, 2008; Bistarelli and Gadducci, 2006; Cooper and Schiex, 2004; Lauritzen and Jensen, 1997], for which a division operation $a \oplus b$ exists. Division satisfies that for all $a, b \in A$ such that $a \leq b$ and $a \neq \mathbf{0}$, $(a \oplus b) \otimes b = a$. When $a \leq b$ and $a = \mathbf{0}$, we follow the approach in [Cooper and Schiex, 2004] and define $\mathbf{0} \oplus b = \mathbf{0}$.

2.2 Factors

Let $\mathcal{X} = (x_1, \ldots, x_n)$ be an ordered set of variables and $\mathcal{D} = (D_1, \ldots, D_n)$ an ordered set of domains, where D_i is the finite set of potential values for x_i. $\mathcal{D}_\mathcal{X}$ is the set of possible assignments of $\mathcal{X}$. Tuples are assignments of domain values to some or all the variables. The join of two tuples t and s is noted $t \cdot s$.

A *factor* [Darwiche, 2009; Kask *et al.*, 2005] f with *scope* $\mathcal{Y} \subseteq \mathcal{X}$ is a function $f : \mathcal{D}_\mathcal{Y} \to A$, where A is a semiring. The evaluation of factor f on tuple t will be noted $f(t)$. If t assigns more variables than needed, they will be ignored. The scope of factor f will be denoted $var(f)$.

The semiring order can also be extended to factors: $f \leq h$ iff $\forall t \in \mathcal{D}_{var(f)\cup var(h)}$, $f(t) \leq h(t)$. Note that this is a very coarse partial ordering. It requires the outcome of *every tuple* to be ordered. It may be the case of a function being *almost always* smaller than another and yet the partial order will not be able to discriminate between them.

Operations over valuations can be extended to functions:

- The *combination* of two functions f and g, noted $f \bigotimes g$, is a new function with scope $var(f) \cup var(g)$ such that, $\forall t \in \mathcal{D}_{var(f)\cup var(g)}, (f \bigotimes g)(t) = f(t) \otimes g(t)$.
- The *division* of two functions f and g such that $\forall t \in \mathcal{D}_{var(f)\cup var(g)},\ f(t) \leq g(t)$, noted $f \bigoplus\!\!\!\!\!\div\, g$, is a new function with scope $var(f) \cup var(g)$ such that, $\forall t \in \mathcal{D}_{var(f)\cup var(g)}, (f \bigoplus\!\!\!\!\!\div\, g)(t) = f(t) \oplus\!\!\!\!\div\, g(t)$.
- The *marginalization* of f over $x \in var(f)$, noted $f \Downarrow_x$, is a function whose scope is $var(f) - \{x\}$ such that, $\forall t \in \mathcal{D}_{var(f)-\{x\}}, (f \Downarrow_x)(t) = \oplus_{v\in D_x}(t \cdot v)$.

2.3 Graphical Models and Reasoning Tasks

A *graphical model* is a set of factors $\mathcal{F}$ over a set of variables $\mathcal{X}$ with domains $\mathcal{D}$. A *reasoning task* is defined by $P = (\mathcal{X}, \mathcal{D}, A, \mathcal{F}, \bigoplus, \bigotimes)$ where $(\mathcal{X}, \mathcal{D}, \mathcal{F})$ is a *graphical model* and $(A, \bigoplus, \bigotimes)$ is a semiring. Computing the reasoning task means computing $(\bigotimes_{f\in\mathcal{F}} f) \Downarrow_{x_1,x_2,\ldots,x_n}$.

Example 1 *In* probabilistic graphical *models valuations are probabilities (i.e, $A = [0, 1]$), the $\otimes$ operation is the product and the $\oplus\!\!\!\!\div$ operation is the division. For the reasoning task of finding the probability of evidence, the $\oplus$ operation is the sum. For the reasoning task of finding the most probable explanation, the $\oplus$ operation is the maximum.*

In standard constraint networks *we have boolean valuations (i.e, $A = \{true, false\}$), the $\otimes$ operation is the conjunction $\wedge$ and the $\oplus\!\!\!\!\div$ operation is also the conjunction $\wedge$. For the reasoning task of finding solutions, the $\oplus$ operation is the disjunction $\vee$. For the reasoning task of counting solutions, the $\oplus$ operation is the sum.*

In weighted constraint networks *valuations are natural numbers with infinity (i.e., $A = \mathbb{N} \cup \{\infty\}$), the $\otimes$ operation is the sum and the $\oplus\!\!\!\!\div$ is the substraction. For the reasoning task of finding optimal solutions, the $\oplus$ operation is the minimum. For the reasoning task of counting weighted solutions, the $\oplus$ operation is the sum.*

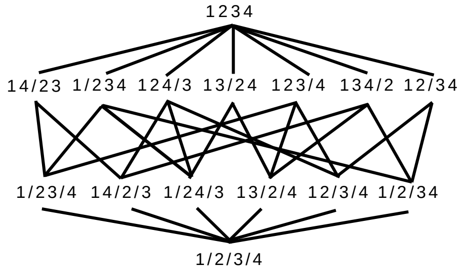

Figure 1: Partitioning lattice of bucket $\mathcal{B} = \{f_1, f_2, f_3, f_4\}$. We specify each function by its subindex.

3 The Partitioning Problem

Computing reasoning tasks is in general intractable. Thus, several approximation methods have been proposed. Some of them (such as mini-bucket elimination [Dechter and Rish, 2003] or join-graph propagation algorithms [Mateescu *et al.*, 2010]) require the computation of a *good* partition out of a set of factors, as described in the following.

A *bucket* $\mathcal{B}$ is a set of factors, all of which have a certain variable x in their scope. The *scope of the bucket* is the set of all variables in the scopes of its factors. The *bucket function* is,

$$\mu = (\bigotimes_{f\in\mathcal{B}} f) \Downarrow_x$$

Let $Q = \{Q_1, Q_2, \ldots, Q_k\}$ be a partition of bucket $\mathcal{B}$. Each partition element is called a *mini-bucket*. We say that Q is a z-partition if the scope size of all its mini-buckets is smaller than or equal to z. The *function of partition* Q is,

$$\mu^Q = \bigotimes_{j=1}^{k}((\bigotimes_{f\in Q_j} f) \Downarrow_x)$$

The rationale of the approximation is that μ^Q is likely to resemble μ, while being computationally simpler. More precisely, if Q is a z-partition, the cost of computing μ^Q is, at most, exponential in z. Approximation algorithms replace the bucket function by a function of one partition, for a fixed parameter z. Thus, it is of utmost importance finding the z-partition whose function resembles μ as much as possible.

3.1 The Partitions Lattice

Given a bucket $\mathcal{B}$, the set of all its partitions can be arranged as a lattice [Rollon and Dechter, 2010]. 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, noted $Q^\perp$, is the partition where every mini-bucket consists of a single function, while the *top* partition, noted $Q^\top$, is the partition with one mini-bucket containing all functions. Note that $Q^\top$ is equivalent to the whole bucket.

Example 2 *Figure 1 depicts the partitioning lattice of bucket $\mathcal{B} = \{f_1, f_2, f_3, f_4\}$. Its bottom partition $Q^\perp$ is $\{\{f_1\}, \{f_2\}, \{f_3\}, \{f_4\}\}$, while its top partition $Q^\top$ is $\{\{f_1, f_2, f_3, f_4\}\}$. Partition $Q = \{\{f_1, f_2\}, \{f_3, f_4\}\}$ is a* *child of partition $Q' = \{\{f_1\}, \{f_2\}, \{f_3, f_4\}\}$ because Q merges mini-buckets $\{f_1\}$ and $\{f_2\}$ in Q'. However, Q is not a child of partition $\{\{f_1\}, \{f_3\}, \{f_2, f_4\}\}$.*

Clearly, the set of z-partitions, for a given z, divides the lattice in two regions: the bottom region contains the z-partitions whose implicit function can be efficiently computed and the top bottom contains the rest of partitions whose implicit function is expensive.

There is a clear relation between lattice edges and the partial order of the partition's implicit functions.

Theorem 1 *[Dechter and Rish, 2003; Bistarelli* et al.*, 1997] Given two partitions Q and Q' of bucket $\mathcal{B}$, if Q' is a descendent of Q then $\mu^{Q'} \leq \mu^{Q}$.*

The previous theorem indicates that following any bottom-up path the implicit functions decrease monotonically. Thus, as we follow the path, we obtain better approximations of the bucket function μ. Thus, given z, the low region of the lattice corresponds to more dissimilar functions, while the high region corresponds to more similar functions.

It is worth to mention that the lattice edges does not explicit all the orders among implicit functions. Some functions from different paths may also be ordered by the partial order although their partitions are not upward connected in the lattice.

3.2 Similarity Functions

The division allows us to capture how similar two functions are. Given two partitions Q, Q' such that $\mu^{Q'} \leq \mu^{Q}$, we define *the similarity function of Q and Q'*, noted $\delta^{Q \to Q'}$, as

$$\delta^{Q\to Q'} = \mu^{Q'} \,\bigcirc\!\!\!\!\div\, \mu^{Q}$$

Moreover, it can be shown that it is more efficiently computed as,

$$\delta^{Q\to Q'} = \mu^{Q'\setminus I} \,\bigcirc\!\!\!\!\div\, \mu^{Q\setminus I}$$

where $I = Q \cap Q'$ is the set of common subsets.

There is a relation between the order among functions of partitions and their similarity delta functions.

Theorem 2 *Let Q, Q', Q'' be three partitions. Then,*

$$\mu^{Q} \leq \mu^{Q'} \leq \mu^{Q''} \Leftrightarrow \delta^{Q'\to Q} \geq \delta^{Q''\to Q}$$

and

$$\mu^{Q} \leq \mu^{Q'} \leq \mu^{Q''} \Leftrightarrow \delta^{Q''\to Q'} \geq \delta^{Q''\to Q}$$

As a consequence, there is a relation among any partition and the top and bottom partitions.

Corollary 1 *Let Q', Q'' be two partitions. Then,*

$$\mu^{Q'} \leq \mu^{Q''} \Leftrightarrow \delta^{Q'\to Q^{\top}} \geq \delta^{Q''\to Q^{\top}}$$

and

$$\mu^{Q'} \leq \mu^{Q''} \Leftrightarrow \delta^{Q^{\perp}\to Q''} \geq \delta^{Q^{\perp}\to Q'}$$

3.3 Formal Definition

We are now in the position of defining and discussing the *partitioning problem*. Given a bucket $\mathcal{B}$ and a complexity parameter z, find a z-partition Q^* that maximally resembles $Q^{\top}$. That is,

$$Q^* = \arg\max_{Q}\{\delta^{Q\to Q_{\top}}\}$$

where $\max$ uses the order among functions, and Q is a z-partition.

A close look at the problem definition shows that the objective function may not be sufficiently discriminative. The reason is that the objective function is partially ordered with very strong requirements for one partition being better than another. As an example, consider two partitions Q and Q' such that $\delta^{Q\to Q^{\top}}(t) \leq \delta^{Q'\to Q^{\top}}(t)$ for every tuple t except one. Both partitions would be consider as equally good in the problem formulation, while commonsense clearly dictates that Q' should be preferred.

One way to overcome this limitation is to refine the partial order $\leq$ among functions. A *refinement* is a partial order $\leq_d$ such that if $f \leq g$ then $f \leq_d g$. To be useful in practice, the refinement should also order pairs of functions where one of them *mainly dominates* the other. We introduce this idea in a *refined* version of the *partitioning problem*.

Given a bucket $\mathcal{B}$, a complexity parameter z and a refinement of the partial order over the functions $\leq_d$, the goal is to find a z-partition Q^* that maximally resembles $Q^{\top}$ according to $\leq_d$. Formally,

$$Q^* = \arg\max_{Q}^{d}\{\delta^{Q\to Q^{\top}}\}$$

where $\max^d$ uses the $\leq_d$ refinement, and Q is a z-partition.

Note that any optimal solution of the refined partitioning problem is also an optimal solution of the original partitioning problem, while the opposite does not hold.

4 A Greedy Algorithm for the Partitioning Problem

There are two difficulties associated with solving the (refined) partitioning problem. On the one hand, the size of the search space may be too large to be traversed (larger than exponential in the number of factors in the bucket). On the other hand, evaluating $\delta^{Q\to Q^{\top}}$ may be too expensive (exponential in the scope of the full bucket).

In the following, we propose solutions to overcome these difficulties. There are several well-known ways to deal with the first issue. Following [Rollon and Dechter, 2010], we take a simple approach and use a greedy procedure that only expands the most promising path. For the second issue we propose an incremental way to compute the objective function of a partition from its parent.

4.1 The Greedy Algorithm

Algorithm 1 shows the pseudo-code of the greedy scheme. Starting at the bottom partition $Q^{\perp}$ of bucket $\mathcal{B}$, the algorithm iteratively selects and moves to the best child until a maximal z-partition is found. At each step, the algorithm selects the maximal child Q' of Q according to $\leq_d$ and the similarity function between Q' and the top partition $Q^{\top}$ (i.e., $\delta^{Q'\to Q^{\top}}$).

Algorithm 1: Greedy Partitioning Scheme

Input : A bucket $\mathcal{B}$; A natural number z; A refinement $\leq_d$.

Output: A partition Q of bucket $\mathcal{B}$ based on a greedy traversal of the partitioning lattice according to $\leq_d$.

1 $Q \leftarrow$ bottom partition of $\mathcal{B}$;
2 **while** $\exists Q' \in ch(Q)$ *which is a z-partition* **do**
3 | $Q \leftarrow \arg\max^d_{Q'}\{\delta^{Q' \to Q^\top}\}$;
4 **end**
5 **return** Q;

4.2 Incremental computation of the objective function

An additional problem of the greedy algorithm is that computing $\delta^{Q \to Q^\top}$ is too expensive in practice. Note that it may be exponential in the scope of the bucket. This is not acceptable in the context of mini-buckets or other bounded complexity algorithms, because every computation should be less than exponential on bounding parameter z.

However, we can take advange of the similarity between a partition and its children, since they only differ on two partition elements. Let Q^{jk} be a child of Q in which mini-buckets Q_j and Q_k have been merged. The only difference between μ^Q and $\mu^{Q^{jk}}$ is that $\mu^{Q_j} \bigotimes \mu^{Q_k}$ is replaced by $\mu^{\{Q_j \cup Q_k\}}$. Therefore, the similarity function is

$$\delta^{Q \to Q^{jk}} = \mu^{\{Q_j \cup Q_k\}} \bigodot\!\!\!\!\div (\mu^{Q_j} \bigotimes \mu^{Q_k})$$

Note that this function captures somehow the *decrement ratio* caused by the transition.

When the greedy algorithm visits partition Q and considers which child to move to, it would be good to evaluate the different alternatives by comparing the different *decrements* that the movements would cause. From Theorem 2, we know that given three partitions Q, Q', Q'' such that $Q', Q'' \in ch(Q)$, then

$$\delta^{Q' \to Q^\top} \geq \delta^{Q'' \to Q^\top} \iff \delta^{Q \to Q'} \leq \delta^{Q \to Q''}$$

However, the previous property does not hold in general when $\leq$ is replaced by $\leq_d$. When a refinement d preserves this property, we say that it is *greedily optimal*. In that case line 3 of Algorithm 1 can be replaced by,

$$Q \leftarrow \arg\min_{Q'}^{d}\{\delta^{Q \to Q'}\}$$

without affecting its behaviour.

The obvious advantage of this new formulation is that the optimization criterion is much cheaper to compute. In particular, it is at most exponential in z, because, by definition, the algorithm only considers successors which are z-partitions. Therefore, it is consistent with the mini-buckets time complexity bounds.

5 Empirical Evaluation

We evaluate the performance of the semiring-based partitioning scheme on the task of computing the Most Probable Explanation (MPE). We apply the well-known logarithmic transformation with which the problem becomes an additive minimization problem over the naturals (equivalent to a *weighted constraint satisfaction problem* [Park, 2002]).

5.1 Refinements d for the MPE task

We consider two refinements for the partial order among functions that already showed good behaviour in the problem of computing the probability of evidence [Rollon and Dechter, 2010]:

1. $\leq_{\texttt{avg-}L^1}$, called *average* 1*-norm* order, defined as:

$$f \leq_{\texttt{avg-}L^1} g \iff \frac{1}{|\mathcal{D}_f|}\sum_t f(t) \geq \frac{1}{|\mathcal{D}_g|}\sum_t g(t)$$

2. $\leq_{L^\infty}$, called ∞*-norm* order, defined as:

$$f \leq_{L^\infty} g \iff \max_t\{f(t)\} \geq \max_t\{g(t)\}$$

It is easy to see that both $\leq_{\texttt{avg-}L^1}$ and $\leq_{L^\infty}$ are refinements of the order among functions. Moreover, both are computed in time proportional to the size of f and g. It is also worth mentioning that $\leq_{\texttt{avg-}L^1}$ is greedily optimal, while $\leq_{L^\infty}$ is not.

Finally, it is important to observe that when the problem has ∞ valuations (i.e, zero probabilities in the original probabilistic model), there may exist some tuples for which their evaluation in a delta function is ∞. Both average 1-norm and ∞-norm return ∞ for those functions. If more than one child of Q is ranked as ∞, the selection among them would be uninformed. When using the average 1-norm we replace the infinities by very high numbers. When using ∞-norm we discriminate by counting the number of occurrences of infinities. In both cases, the goal is to let the infinity be very influential, but not absorving.

5.2 Algorithms and Benchmarks

We compare three partitioning schemes: (i) the scope-based scheme (SCP) described in [Rollon and Dechter, 2010; Dechter and Rish, 1997]; (ii) our ∞-norm refinement (L^∞); and, (iii) our average 1-norm refinement ($\texttt{avg-}L^1$). Roughly, SCP 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.

We report the results for mini-bucket elimination (MBE) [Dechter and Rish, 2003] and for the recently proposed mini-bucket elimination with max-marginal matching (MBE-MM) [Ihler *et al.*, 2012]. Briefly, MBE-MM introduces a cost propagation phase once the partition is built, and it was shown to obtain accurate bounds for a number of benchmarks. Both algorithms use the variable elimination ordering established by the *min-fill* heuristic after instantiating evidence variables (if any).

We conduct our empirical evaluation on three benchmarks: *coding networks*, two sets of *linkage analysis* (denoted *pedigree* and *Type_4*), and *noisy-or bayesian networks*. All instances are included in the UAI08 evaluation[1]. Table 1 reports

[1]http://graphmod.ics.uci.edu/uai08/Software

$-\log$(upper bound) (i.e., a lower bound on the log scale) and runtime (in seconds) for the different algorithms and partitioning schemes as a function of the value of the control parameter z.

5.3 Experimental Results

Coding Networks. For MBE, L^∞ and avg-L^1 outperforms SCP on five instances each when $z = 20$, and on six and four instances, respectively, when $z = 22$. When they are better, the increment of the bound is usually of more than one order of magnitude. For MBE-MM, L^∞ outperforms SCP on four and seven instances when $z = 20$ and $z = 22$, respectively, while avg-L^1 does so on three and four instances. The improvement is not as dramatic as with standard MBE, but for some instances it is still of orders of magnitude.

As observed in [Ihler *et al.*, 2012], MBE-MM using SCP is always superior to MBE using SCP. In this benchmark, we also see that: (i) for any fixed partitioning scheme MBE-MM is superior to MBE; (ii) MBE-MM using SCP is always superior to MBE using any partitioning scheme; and (iii) MBE-MM benefits from the semiring-based partitioning scheme (in particular, from L^∞).

As expected, all semiring-based partitioning schemes are slower than SCP. The reason is that during the traversal of the partitioning lattice semiring-based heristics have to compute intermediate functions that the greedy algorithm will eventually discard.

Linkage Analysis. For MBE, we see that semiring-based schemes generally outperform SCP. For pedigree instances and $z = 17$, the increasement is very often of orders of magnitude. When $z = 19$ we observe the same improvement very often. For Type_4 instances, the increment is in general of more than one order of magnitude for both values of the control parameter z.

For MBE-MM, each of the semiring-based schemes also outperforms in general SCP. Again, the improvement margin is reduced with respect to standard MBE. For pedigree instances, the improvement is in some cases of orders of magnitude, while for Type_4 instances, the increase is still in general of orders of magnitude for both values of z. It is also important to note that, in some cases, the effect of the cost propagation leads all partitioning schemes to obtain the same bound on pedigree instances (i.e., pedigree-18 and pedigree-25).

As for the previous benchmark, MBE-MM using SCP is always superior to MBE using any partitioning scheme. The only exceptions are instances pedigree-20 and pedigree-33 and $z = 17$. Again, running MBE-MM with one of the semiring-based schemes seems a better choice than running MBE.

The cpu time of all partitioning schemes is relatively close. The only exceptions are four instances on pedigree instances (i.e., pedigree-31, pedigree-34, pedigree-37 and pedigree-41) and two on Type_4 instances (i.e., Type_4-140-19 and Type_4-140-19), where semiring-based partitioning schemes are 2 to 3 times slower than SCP.

Noisy-or Bayesian Networks. For space reasons, we only report results on *bn2o-30-20-200* instances. Results for *bn2o-30-15-150* and *bn2o-30-25-250* instances are similar.

For MBE, each semiring-based partitioning scheme is always superior to SCP for both values of z. The only exception is instance bn2o-30-20-200-3b, for which L^∞ is inferior to SCP when $z = 17$. For MBE-MM, each semiring-based scheme outperforms SCP in general, although the improvement margin is less notable. In some cases, the effect of cost propagation yields all heuristics to obtain the same bound. Yet, running MBE-MM using one semiring-based partitioning scheme seems the best choice for this benchmark.

6 Conclusions and Future Work

This paper generalizes the partitioning problem proposed in [Rollon and Dechter, 2010] to any task defined as a graphical model. The generalization is possible under a semiring with an additional division operation and a refinement of its order. These requirements can be considered as *mild* because they are satisfied by the usual tasks such as counting and optimization. We propose a general greedy scheme to solve this problem efficiently. Finally, we propose two particular order refinements for optimization tasks. These refinements are based on two well-known metrics as 1-norm and ∞-norm.

Our experimental results show that the semiring-based partitioning schemes improve significantly in many cases the accuracy of the standard MBE. When this algorithm is enhanced with a cost propagation phase (i.e., MBE-MM), the impact of the partitioning schemes is reduced, but still quite remarkable. Overall, the empirical evaluation suggests that the best bounds are obtained with MBE-MM using a semiring-based partitioning scheme at the only cost of a constant increase in time.

In our future work we want to investigate the impact of the semiring-based partitioning schemes on other partition-based algorithms as join-graph propagation algorithms [Mateescu *et al.*, 2010], and as heuristic generator. We also want to explore the impact of alternative refinements and if the accuracy of the refinements depends on the task at hand. Finally, we want to study the effectiveness of more sophisticated algorithms beyond our greedy approach.

Acknowledgments

This work was supported by project TIN2009-13591-C02-01 and NSF grant IIS-1065618.

References

[Bistarelli and Gadducci, 2006] S. Bistarelli and F. Gadducci. Enhancing constraints manipulation in semiring-based formalisms. In *ECAI*, pages 63–67, 2006.

[Bistarelli *et al.*, 1997] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, March 1997.

[Bistarelli *et al.*, 1999] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based

Id.	z	SCP		L^∞		avg-L^1		z	SCP		L^∞		avg-L^1	
		Time	LB	Time	LB	Time	LB		Time	LB	Time	LB	Time	LB
CODING NETWORKS														
MBE														
126	20	2.71	44.1649	15.16	**47.9546**	14.78	44.2676	22	8.07	44.5750	69.13	**47.6332**	61.9	45.3310
127	20	2.64	**49.5839**	16.77	48.5490	15.37	45.2983	22	10.75	48.1252	59.2	**48.7722**	52.79	47.5072
128	20	2.81	41.4837	17.92	41.0880	17.33	**43.2502**	22	10.74	**44.6335**	67.46	41.6413	57.95	41.6335
129	20	2.31	47.3691	13.63	**47.3930**	13.31	44.4312	22	7.59	**46.4928**	45.8	44.5064	46.29	45.1959
130	20	2.61	46.9032	14.11	47.3609	14.01	**47.8376**	22	8.57	47.8710	44.36	**49.0464**	46.27	46.4622
131	20	2.59	**47.0599**	16.05	46.6705	12.66	46.8777	22	7.58	47.8448	49.64	**48.2524**	41.05	47.0263
132	20	2.67	46.0854	14.48	49.3534	14.76	**49.6561**	22	10.26	50.5320	42.88	50.8409	49.75	**51.3809**
133	20	2.69	**46.6227**	12.52	43.5029	14.21	44.4477	22	10.95	43.9615	57.92	44.0481	49.96	**46.3188**
134	20	2.53	43.4042	15.43	44.1869	15.42	**46.8288**	22	10.81	46.9455	52.58	43.9870	57.24	**50.0214**
MBE-MM														
126	20	6.16	50.9868	19.08	51.4298	16.95	**51.7849**	22	28.83	**52.1866**	70.77	51.9130	69.1	52.0769
127	20	8.17	**54.2311**	21.15	53.8390	18.54	54.1132	22	30.35	**54.9843**	73.46	54.9352	60.39	53.8129
128	20	8.1	**46.6324**	24.22	46.0965	19.71	46.0335	22	29.67	46.3810	72	**46.6970**	68.01	46.2075
129	20	6.89	52.8272	16.13	**52.9273**	17.8	52.2187	22	26.94	54.1139	68.46	54.8979	62.74	**55.2956**
130	20	7.25	53.9811	16.69	**55.3593**	16.83	55.2183	22	26.74	54.3547	62.86	**55.1318**	49.67	54.4864
131	20	7.29	**53.2935**	18.86	52.8953	15.16	52.5563	22	25.85	53.1382	60.72	**53.4991**	48.9	52.7956
132	20	7.61	56.6294	19.6	**56.6458**	17.54	56.5919	22	22.89	57.4683	55.91	**57.7120**	58.96	57.3692
133	20	7.29	50.8308	19.5	50.1530	17.79	**51.1713**	22	24.93	50.1155	56.97	50.1969	61.15	**50.4601**
134	20	7.84	**52.0498**	18.76	51.0830	19.71	51.7879	22	29.1	52.1059	67.52	**53.8257**	66.12	52.9524
PEDIGREE NETWORKS														
MBE														
7	17	2.72	108.8927	3.76	**109.4564**	4.84	109.2850	19	18.11	109.1999	21.84	109.4359	21.17	**109.4937**
9	17	1.31	116.0396	1.85	115.7635	1.86	**116.2614**	19	4.61	116.9488	6.94	**118.9390**	6.86	**118.9390**
13	17	1.38	69.6829	1.95	70.9686	1.86	**71.3244**	19	4.87	70.3736	7.47	70.6534	7.03	**70.8203**
18	17	0.64	**121.3239**	0.67	**121.3239**	0.67	**121.3239**	19	2.04	123.2094	2.05	123.2094	2.05	**123.2841**
20	17	9.99	51.1976	9.16	**52.7681**	9.34	51.1475	19	36.18	**51.7526**	37.44	51.3947	37.41	51.3947
25	17	0.54	**156.7323**	0.45	155.7781	0.48	155.7781	19	1.02	**159.2994**	1.07	**159.2994**	1.07	**159.2994**
30	17	1.49	132.7058	1.21	**133.2865**	1.21	**133.2865**	19	4.4	**135.9630**	4.66	**135.9630**	4.63	**135.9630**
31	17	7.52	125.9962	8.26	**126.7028**	7.77	126.3257	19	26.96	126.3103	57.86	**126.7808**	57.62	**126.7808**
33	17	3.36	67.4128	5.62	70.0187	5.1	**70.9729**	19	10.3	65.5044	10.97	**68.1102**	13.52	68.0679
34	17	22	105.5951	34.62	**107.8021**	33.21	**107.8021**	19	117.25	106.1329	233.77	**107.8579**	219	107.5615
37	17	62.42	138.8355	166.75	**140.7067**	228.84	139.8428	19	163.43	**142.6193**	1356.08	**142.6193**	350.84	**142.6193**
41	17	44.15	114.1528	72.19	113.8273	69.82	**115.0162**	19	128.53	**114.9441**	261.63	114.2727	246.4	114.0889
44	17	1.45	89.5737	2.37	**91.2718**	1.97	90.0481	19	5.1	90.3476	9.45	90.2808	9.1	**90.7143**
51	17	2.12	100.9149	3.52	**102.4860**	2.45	101.6225	19	8.65	101.0238	10.15	101.0238	8.79	**101.3729**
100_16	17	30.81	1145.5791	43.26	1151.3618	42.01	**1157.1399**	19	97.31	1158.0012	139.7	**1161.3181**	135.6	1160.654785
100_19	17	11.41	1067.8678	14.3	**1074.1741**	15.46	1070.5029	19	31.63	1082.7845	43.28	**1085.1501**	44.49	1080.159302
120_17	17	7.43	1296.9375	8.77	**1298.1321**	8.85	1297.2715	19	15.43	1306.7068	17.68	**1314.4250**	17.93	1311.921631
130_21	17	10.27	1300.8636	12.6	**1310.1495**	12.46	1310.1292	19	22.07	1311.9829	29.25	**1322.6984**	28.89	1321.91272
140_19	17	19.37	1386.5961	28.46	1398.6418	27.27	**1401.7791**	19	44.39	1413.8478	79.08	1420.3602	71.52	**1422.321899**
140_20	17	35.53	1295.2239	44.69	**1296.2660**	44.22	1292.2687	19	123.87	1315.7791	194.54	**1316.6406**	487.25	1313.216064
150_14	17	57.01	1497.8391	66.27	1504.8148	113.19	**1513.5554**	19	107.49	1505.3149	139.74	1509.2795	140.49	**1515.777954**
150_15	17	77.39	1228.0110	73.2	1229.6445	26.32	**1232.9501**	19	46.81	1239.6547	54.41	**1247.2201**	54.62	1246.114502
160_14	17	23.8	1879.8701	34.88	1887.1932	30.36	**1889.0613**	19	54.66	1899.8004	72.05	**1907.7247**	77.73	1903.588379
160_15	17	21	1468.4277	26.24	**1471.2092**	25.31	1457.9683	19	48.96	**1485.6819**	66.2	1484.6494	62.5	1480.59375
160_23	17	17.87	1881.1091	20.04	**1905.0249**	20.08	1894.3540	19	27.58	1900.0710	34.63	**1915.3242**	34.51	1914.324585
170_23	17	8.37	1889.8179	8.97	**1892.2351**	9.03	1891.9114	19	12.61	1905.4634	13.09	**1905.8533**	13.27	1902.946899
190_20	17	23.54	2436.5767	28.69	2439.2964	28.71	**2441.3315**	19	43.92	2440.3169	56.18	2445.7246	57.61	**2445.96875**
MBE-MM														
7	17	4.57	110.1427	5.65	110.3001	9.49	**110.3677**	19	28.42	110.8623	30.59	110.8220	28.13	**110.9526**
9	17	2.22	120.3932	2.82	121.0717	2.7	**121.3174**	19	7.87	121.5204	10.61	**121.6989**	10.64	**121.6989**
13	17	2.32	71.0492	3.15	71.1748	3.4	**71.5114**	19	7.99	**71.4750**	10.96	71.2046	10.68	71.2046
18	17	0.8	**124.1096**	0.86	**124.1096**	0.86	**124.1096**	19	2.25	**124.4249**	2.27	**124.4249**	2.27	**124.4249**
20	17	11.84	51.4184	10.14	**51.4343**	10.59	**51.4343**	19	41.45	**52.7168**	44.87	**52.7168**	44.72	**52.7168**
25	17	0.6	**159.6288**	0.72	**159.6288**	0.67	**159.6288**	19	1.44	**159.9930**	1.48	**159.9930**	1.47	**159.9930**
30	17	1.39	135.8177	1.52	135.8178	1.56	**135.8454**	19	4.82	136.5649	5.23	136.5649	4.88	**136.6445**
31	17	10.27	128.5108	11.08	**129.0052**	11.42	128.8895	19	38.65	**128.6116**	96.26	128.5891	95.44	128.5891
33	17	6.17	70.0013	9.35	70.7769	9.48	**70.8993**	19	17.05	69.8644	16.26	71.0661	19.03	**71.4836**
34	17	38.11	109.0189	46.93	**109.2000**	45.49	108.7519	19	199.63	109.4744	333.38	109.4890	423.74	**109.5095**
37	17	135.73	**142.8687**	293.24	**142.8687**	294.51	**142.8687**	19	331.82	144.0392	706.72	**144.0657**	431.93	**144.0657**
41	17	74.62	115.4667	89.12	**116.1025**	97.32	115.4144	19	204.35	116.2645	335.51	**116.2781**	324.52	116.1317
44	17	2.47	94.1250	4.7	94.5632	2.89	**94.6676**	19	8.73	94.3481	9.11	**94.7630**	9.6	94.7034
51	17	3.62	104.9397	5.22	**106.1441**	3.86	105.4351	19	12.02	**106.1931**	13.6	106.1323	12.12	106.1849
100_16	17	42.78	1176.6797	55.11	**1180.5432**	180.65	1178.8740	19	139.21	1181.8257	182	1185.1167	175.57	**1185.293701**
100_19	17	15.11	1104.0924	18.46	1105.6055	17.8	**1107.0393**	19	44.15	1109.6017	58.05	**1110.4194**	52.72	1110.032104
120_17	17	8.7	1320.8333	9.83	1321.9402	9.98	**1322.0950**	19	17.54	1324.0256	19.34	1323.8835	19.58	**1324.308594**
130_21	17	12.69	1346.7722	14.29	**1349.8878**	14.77	1349.2976	19	29.06	1356.0505	32.91	**1356.8892**	32.68	1355.352783
140_19	17	28.9	1445.3862	36.63	**1447.8936**	35.65	1446.7283	19	74.12	**1459.3081**	109.79	1455.9856	92.23	1454.226685
140_20	17	50.99	1345.8759	58.19	**1348.2992**	62.5	1347.0886	19	185.39	1356.7883	255.03	1357.7189	275.23	**1359.001221**
150_14	17	22.72	1581.7888	37.63	**1583.0146**	23.79	1582.6594	19	44.02	**1592.5331**	50.24	1591.8013	51.45	1592.232178
150_15	17	28.92	**1319.2913**	32	1318.8572	32.14	1317.1138	19	60.03	1323.8816	74.59	1325.3538	66.15	**1325.453491**
160_14	17	33.56	1932.0858	44.23	1936.6246	35.39	**1936.8888**	19	73.35	1942.7789	89.91	1943.5851	90.61	**1943.680542**
160_15	17	29.38	1568.8596	33.86	**1569.1587**	33.05	1566.0881	19	70.33	1580.8682	89.27	1583.6843	86.66	**1584.233154**
160_23	17	20.33	1990.3468	22.37	**1991.9583**	21.8	1991.4231	19	35.28	**2001.8970**	41.72	2001.5148	42.86	2001.546631
170_23	17	9.2	**1920.2833**	9.68	1919.1329	9.78	1920.0374	19	14.16	1924.2651	14.75	1924.2798	14.86	**1924.574219**
190_20	17	27.96	2512.3047	32.6	**2513.1763**	32.56	2509.6494	19	57.77	2520.0928	66.88	2521.9863	68.62	**2522.884277**
NOISY-OR NETWORKS														
MBE														
1a	10	0.01	6.3609	0.21	7.8882	0.21	**7.9994**	15	0.05	7.2272	0.36	8.1921	0.35	**8.2927**
1b	10	0.01	4.5187	0.21	**4.6424**	0.21	4.6042	15	0.05	4.5971	0.36	4.6831	0.35	**4.7275**
2a	10	0.01	6.4033	0.21	**8.8711**	0.21	8.8120	15	0.05	7.5348	0.36	9.3413	0.38	**9.3682**
2b	10	0.01	3.8327	0.21	**3.9598**	0.21	3.9178	15	0.05	3.9277	0.37	**4.0311**	0.36	4.0199
3a	10	0.01	7.3648	0.21	10.2783	0.22	**10.3138**	15	0.05	8.5514	0.37	10.5725	0.42	**10.6037**
3b	10	0.01	3.9869	0.21	3.9815	0.21	**4.0029**	15	0.05	3.9977	0.35	4.0490	0.36	**4.0712**
MBE-MM														
1a	10	0.01	**9.1105**	0.22	9.1035	0.22	9.1039	15	0.1	9.2036	0.42	**9.2099**	0.4	**9.2099**
1b	10	0.01	4.9684	0.22	4.9684	0.22	**4.9687**	15	0.1	4.9869	0.41	4.9869	0.39	**4.9920**
2a	10	0.01	9.5315	0.22	**9.5533**	0.22	**9.5533**	15	0.1	**9.8632**	0.4	9.7863	0.41	9.7515
2b	10	0.01	**4.1164**	0.22	**4.1164**	0.22	**4.1164**	15	0.1	**4.1277**	0.42	**4.1277**	0.41	**4.1277**
3a	10	0.01	**10.9827**	0.21	**10.9827**	0.22	**10.9827**	15	0.1	11.1249	0.47	**11.1278**	0.42	11.0819
3b	10	0.01	**4.1198**	0.21	**4.1198**	0.22	**4.1198**	15	0.1	**4.1247**	0.4	**4.1247**	0.4	**4.1247**

Table 1: Empirical results on coding, linkage analysis, and noisy-or Bayesian networks for the task of computing the MPE. The table reports $-\log$(upper bound) (i.e., a lower bound on the log scale) obtained by MBE and MBE-MM for different values of the control parameter z and different partitioning schemes (i.e., scope-based (SCP), ∞-norm (L^∞) and average 1-norm (avg-L^1)). The first column (Id.) shows the name of the instance: for coding networks the name is *BN_*Id; for pedigree networks the name is *pedigree*-Id. and *Typle4_*Id. for the first and second set of instances, respectively; and, for noisy-or Bayesian networks the name is *bn2o-30-20-200*-Id. We highlight in bold face the best lower bound for each instance and value of z.

CSPs and valued CSPs: Frameworks, properties and comparison. *Constraints*, 4:199–240, 1999.

[Cooper and Schiex, 2004] M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154(1-2):199–227, 2004.

[Darwiche, 2009] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, San Francisco, 2009.

[Dechter and Rish, 1997] R. Dechter and I. Rish. A scheme for approximating probabilistic inference. In *Proceedings of the 13th UAI-97*, pages 132–141, 1997.

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

[Dechter, 2003] R. Dechter. *Constraint Processing*. Morgan Kaufmann, San Francisco, 2003.

[Ihler *et al.*, 2012] A. T. Ihler, N. Flerova, R. Dechter, and L. Otten. Join-graph based cost-shifting schemes. In *UAI*, pages 397–406, 2012.

[Kask *et al.*, 2005] K. Kask, R. Dechter, J. Larrosa, and A. Dechter. Unifying tree decompositions for reasoning in graphical models. *Artif. Intell.*, 166(1-2):165–193, 2005.

[Kohlas and Wilson, 2008] J. Kohlas and N. Wilson. Semiring induced valuation algebras: Exact and approximate local computation algorithms. *Artif. Intell.*, 172(11):1360–1399, 2008.

[Lauritzen and Jensen, 1997] S. L. Lauritzen and F. V. Jensen. Local computation with valuations from a commutative semigroup. *Ann. Math. Artif. Intell.*, 21(1):51–69, 1997.

[Mateescu *et al.*, 2010] R. Mateescu, K. Kask, V. Gogate, and R. Dechter. Join-graph propagation algorithms. *J. Artif. Intell. Res. (JAIR)*, 37:279–328, 2010.

[Park, 2002] J. D. Park. Using weighted max-sat engines to solve mpe. In *Proc. of the 18th AAAI*, pages 682–687, Edmonton, Alberta, Canada, 2002.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[Rollon and Dechter, 2010] E. Rollon and R. Dechter. New mini-bucket partitioning heuristics for bounding the probability of evidence. In *Proc. of the* 24^{th} *AAAI, Atlanta, Georgia, USA*, 2010.