Inference Schemes for M Best Solutions for Soft CSPs

Emma Rollon[1], Natalia Flerova[2] and Rina Dechter[2]

[1]Universitat Politecnica de Catalunya, [2]University of California Irvine

erollon@lsi.upc.edu,{nflerova,dechter}@ics.uci.edu

Abstract. The paper present a formalization of the m-best task within the unifying framework of semirings. As a consequence, known inference algorithms are defined and their correctness and completeness for the m-best task are immediately implied. We also describe and analyze a Bucket Elimination algorithm for solving the m-best task, *elim-m-opt*, presented in an earlier workshop[1] and introduce an extension to the mini-bucket framework, yielding a collection of bounds for each of the m-best solutions. Some empirical demonstration of the algorithms and their potential for approximations are provided.

1 Introduction

Given an optimization problem, the objective is typically to find an optimal solution, i.e., a solution that provides the best value of the objective function. However, in many applications it is desirable to obtain not just a single optimal solution but a set (of a given size m) of the best possible solutions. Such a set can be useful, for example, in assessing the sensitivity of the optimal solution to variation of the parameters of the problem, or when a set of diverse assignments with approximately the same cost is wanted.

Lawler [11] provided a general scheme for using any optimization algorithm to solve the m-best task. Its main idea is to compute the m-best solutions by successively computing the best solution, each time using a slightly different reformulation of the original problem. This approach has been extended and improved over the years and is still one of the primary strategies to date for finding the m-best solutions. The approach used in this paper is to develop direct algorithms that avoid the repeated computation inherent in Lawler's scheme. The main idea is to integrate the m-best task into existing optimization schemes. In particular we focus on graphical models. This work is the continuation of our previous efforts [7], [8], where we derived and analyzed algorithm *elim-m-opt* that extends the widely-used Bucket Elimination (BE) to compute the m-best solutions by a relatively simple modification of its underlying combination and marginalization operators [4] and proposed extensions to Mini-Bucket Elimination to compute bounds on each of the m-best solutions, yielding algorithm *mbe-m-opt*.

[1] [7]

The main contribution of this paper is the formalization of the m-best task within the framework of semirings [14, 1, 10, 2]. This unifying formulation ensures the soundness and correctness of inference algorithms applied to any problem that fits into the framework. In particular, we show that *elim-m-opt* solves the m-best optimization task and we provide new empirical analysis for *mbe-m-opt* demonstrating its effectiveness both as an exact and approximation scheme.

2 Background

We consider problems expressed as graphical models. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be an ordered set of variables and $\mathbf{D} = (\mathbf{D}_1, \ldots, \mathbf{D}_n)$ an ordered set of domains. Domain $\mathbf{D}_i$ is a finite set of potential values for X_i. The assignment of variable X_i with $a \in \mathbf{D}_i$ is noted $(X_i = a)$. A tuple is an ordered set of assignments to different variables $(X_{i_1} = a_{i_1}, \ldots, X_{i_k} = a_{i_k})$. A *complete assignment* to all the variables in $\mathbf{X}$ is called a *solution*. Let t and s be two tuples having the same instantiations to the common variables. Their join, noted $t \cdot s$, is a new tuple which contains the assignments of both t and s (this notation is also used for multiplication, so we assume the meaning will be clear from the context). If t is a tuple over a set $T \subseteq X$ and $\mathbf{S}$ is a set of variables, then $t_{[S]}$ is a relational projection of t on $\mathbf{S}$.

We denote by $\mathbf{D_Y}$ the set of tuples over a subset of variables $\mathbf{Y}$, also called the *domain of Y*. Let $f : \mathbf{D_Y} \to \mathbf{A}$ be a function defined over $\mathbf{Y}$. $\mathbf{A}$ is a set of elements called *valuations* . Typical sets of valuations $\mathbf{A}$ are natural, real and booleans. If $f : \mathbf{D_Y} \to \mathbf{A}$ is a function the scope of f, denoted $var(f)$, is $\mathbf{Y}$. In the following, we will use $\mathbf{D}_f$ as a shorthand for $\mathbf{D}_{var(f)}$.

We assume two binary operations over valuations: $\otimes : \mathbf{A} \times \mathbf{A} \to \mathbf{A}$ called combination and $\oplus : \mathbf{A} \times \mathbf{A} \to \mathbf{A}$ called addition. Both operators are associative and commutative. Typical combination operators are sum and product over numbers, and logical and (i.e., $\wedge$) over booleans. Typical addition operators are min, max and sum over numbers and logical or (i.e., $\vee$) over booleans. We extend these operators to operate over functions.

Definition 1 (combination operator, marginalization operator). *Let $f : \boldsymbol{D}_f \to \boldsymbol{A}$ and $g : \boldsymbol{D}_g \to \boldsymbol{A}$ be two functions. Their* combination*, noted $f \bigotimes g$ is a new function with scope $var(f) \cup var(g)$, s.t. $\forall t \in \boldsymbol{D}_{var(f)\cup var(g)}$, $(f \bigotimes g)(t) = f(t) \otimes g(t)$. Let $f : \boldsymbol{D}_f \to \boldsymbol{A}$ be a function and $\boldsymbol{W} \subseteq \boldsymbol{X}$ be a set of variables. The* marginalization *of f over $\boldsymbol{W}$, noted $\Downarrow_{\boldsymbol{W}} f$, is a function whose scope is $var(f) - \boldsymbol{W}$, s.t. $\forall t \in D_{var(f)-\boldsymbol{W}}, (\Downarrow_{\boldsymbol{W}} f)(t) = \oplus_{t' \in D_W} f(t \cdot t')$.*

Definition 2 (graphical model).
A graphical model *is a tuple $\mathcal{M} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes)$, where: $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ is a set of variables; $\boldsymbol{D} = \{D_1, \ldots, D_n\}$ is the set of their finite domains of values; $\boldsymbol{A}$ is a set of valuations; $\boldsymbol{F} = \{f_1, \ldots, f_r\}$ is a set of discrete functions where $var(f_j) \subseteq \boldsymbol{X}$ and $f_j : \boldsymbol{D}_{f_j} \to \boldsymbol{A}$; and $\bigotimes$ is a combination operator over functions as defined in Definition 1. The graphical model $\mathcal{M}$ represents the function $F(\boldsymbol{X}) = \bigotimes_{f \in \boldsymbol{F}} f$.*

Definition 3 (reasoning task).
A reasoning task *is a tuple* $P = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes, \Downarrow)$ *where* $(\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes)$ *is a graphical model and* $\Downarrow$ *is a marginalization operator over functions as defined in Definition 1. The reasoning task is to compute* $F(\boldsymbol{X}) \Downarrow_{\boldsymbol{X}}$.

For a reasoning task $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{A}, \mathbf{F}, \bigotimes, \Downarrow)$ the choice of $(\mathbf{A}, \otimes, \oplus)$ determines the combination $\bigotimes$ and marginalization $\Downarrow$ operators over functions, and thus the nature of the graphical model and its reasoning task. For example, if $\mathbf{A}$ is the set of non-negative reals and $\bigotimes$ is product, the graphical model is a Markov network or a Bayesian network. If $\Downarrow$ is max, the task is to compute the Most Probable Explanation (MPE), while if $\Downarrow$ is sum, the task is to compute the Probability of the Evidence.

The correctness of the algorithmic techniques for computing a given reasoning task relies on the properties of its set of valuations and operators. These properties are axiomatically described by means of an algebraic structure over $(\mathbf{A}, \otimes, \oplus)$. In this paper we consider reasoning tasks $P = (\mathbf{X}, \mathbf{D}, \mathbf{A}, \mathbf{F}, \bigotimes, \Downarrow)$ such that their valuation structure $(\mathbf{A}, \otimes, \oplus)$ is a semiring. Several works [14, 1, 10] showed that the correctness of inference algorithms over a reasoning task P is ensured whenever P is defined over a semiring.

Definition 4 (semiring). *A commutative semiring is a triplet* $(\boldsymbol{A}, \otimes, \oplus)$ *which satisfies the following three axioms:*

A1. *The operation* $\oplus$ *is associative, commutative and idempotent, and there is an additive identity element called* 0 *such that* $a \oplus 0 = a$ *for all* $a \in \boldsymbol{A}$. *In other words,* $(\boldsymbol{A}, \oplus)$ *is a commutative monoid.*
A2. *The operation* $\otimes$ *is also associative and commutative, and there is a multiplicative identity element called* 1 *such that* $a \otimes 1 = a$ *for all* $a \in \boldsymbol{A}$. *In other words,* $(\boldsymbol{A}, \otimes)$ *is also a commutative monoid.*
A3. $\otimes$ *distributes over* $\oplus$, *i.e.,* $(a \otimes b) \oplus (a \otimes c) = a \otimes (b \oplus c)$

Example 1. MPE task is defined over semiring $\mathcal{K} = (\mathtt{R}, \times, \max)$, a CSP is defined over semiring $\mathcal{K} = (\{0, 1\}, \wedge, \vee)$, and a Weighted CSP is defined over semiring $\mathcal{K} = (\mathtt{N} \cup \{\infty\}, +, \min)$. The task of computing the Probability of the Evidence is defined over semiring $\mathcal{K} = (\mathtt{R}, \times, +)$.

Bucket elimination (BE) [4] is a well-known inference algorithm that generalizes dynamic programming for many reasoning tasks.

Definition 5 (bucket elimination). *The input of BE is a reasoning task* $P = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes, \Downarrow)$ *and an ordering* $o = (X_1, X_2, \ldots, X_n)$, *dictating an elimination order for* BE, *from last to first. Each function from* $\boldsymbol{F}$ *is placed in the bucket of its latest variable in* o. *The algorithm processes the buckets from* X_n *to* X_1, *computing for each* $Bucket_{X_i}$, *noted* $\boldsymbol{B}_i$, $\Downarrow_{X_i} \bigotimes_{j=1}^{n} \lambda_j$, *where* λ_j *are the functions in the* $\boldsymbol{B}_i$, *some of which are original* $f_i's$ *and some are earlier computed messages. The result of the computation is a new function, also called* message, *that is placed in the bucket of its latest variable in the ordering* o.

The message passing between buckets follows a bucket-tree structure.

Definition 6 (bucket tree). *Bucket elimination defines a* bucket tree*, where the bucket of each X_i is linked to the destination bucket of its message (called the parent bucket). A node of the bucket is associated with its bucket variable.*

Theorem 1 *[4] Given a reasoning task $\mathcal{P} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes, \Downarrow)$, BE is sound and complete. The time and space complexity of $BE(\mathcal{P})$ is exponential in a structural parameter called* induced width*, which is the largest scope of all the functions computed.*

3 M-best Optimization Task

In this section we formally define the problem of finding a set of best solutions over an optimization task. We consider optimization tasks defined over a set of totally ordered valuations. In other words, we consider reasoning tasks where the marginalization operator $\Downarrow$ is min or max. Without loss of generality, in the following we assume minimization tasks (i.e., $\Downarrow$ is min).

Definition 7 (optimization task). *Given a graphical model $\mathcal{M}$, its optimization task is $P = (\mathcal{M}, min)$. The goal is to find a complete assignment t such that $\forall t' \in D_{\boldsymbol{X}}, F(t) \leq F(t')$. $F(t)$ is called the optimal solution.*

Definition 8 (m-best optimization task). *Given a graphical model $\mathcal{M}$, its m-best optimization task is to find m complete assignments $\boldsymbol{T} = \{t_1, \ldots, t_m\}$ such that $F(t_1) \leq, \cdots, \leq F(t_m)$ and $\forall t' \in \boldsymbol{D_X} \backslash T$ and $\forall t \in \boldsymbol{T}, F(t') \geq F(t)$. The solution is the set of valuations $\{F(t_1), \ldots, F(t_m)\}$, called m-best solutions.*

The main goal of this paper is to phrase the m-best optimization task as a reasoning task over a semiring, so that well known algorithms can be immediately applied to solve this task. Namely, given an optimization task P over a graphical model $\mathcal{M}$, we need to define a reasoning task P^m that corresponds to the set of m-best solutions of $\mathcal{M}$.

We introduce the set of ordered m-best elements of a subset $S \subseteq \mathbf{A}$.

Definition 9 (set of ordered m-best elements, m-space). *Let S be a subset of a set of valuation $\boldsymbol{A}$. The* set of ordered m-best elements *of S is $Sorted^m\{S\} = \{s_1, \ldots, s_j\}$ such that $s_1 \leq s_2 \leq \ldots \leq s_j$ where $j = m$ if $|S| \geq m$ and $j = |S|$ otherwise, and $\forall s' \notin Sorted^m\{S\}, s_j \leq s'$. The* m-space *of $\boldsymbol{A}$, denoted $\boldsymbol{A}^m$, is the set of subsets of ordered m-best elements of $\boldsymbol{A}$. Formally, $\boldsymbol{A}^m = \{S \subseteq A \mid Sorted^m\{S\} = S\}$.*

The combination and addition operators over the m-space $\mathbf{A}^m$, noted $\otimes^m$ and $sort^m$ respectively, are defined as follows.

Definition 10 (combination and addition over the m-space). *Let $\boldsymbol{A}$ be a set of valuations, and $\otimes$ and min be its combination and marginalization operators, respectively. Let $S, T \in A^m$. Their* combination*, noted $S \otimes^m T$, is the set $Sorted^m\{a \otimes b \mid a \in S, b \in T\}$, while their* addition*, noted $sort^m\{S, T\}$, is the set $Sorted^m\{S \cup T\}$.*

Theorem 1. *The valuation structure* $(\boldsymbol{A}^m, \otimes^m, sort^m)$ *is a semiring.*

We will refer to functions over the m-space $\mathbf{A}^m$ $f : \mathbf{D}_f \to \mathbf{A}^m$ as *vector functions*. Abusing notation, we extend the $\otimes^m$ and $sort^m$ operators to operate over vector functions similar to how operators $\otimes$ and $\oplus$ were extended to operate over scalar functions in Definition 1.

Definition 11 (combination and marginalization over vector functions). *Let* $f : \boldsymbol{D}_f \to \boldsymbol{A}^m$ *and* $g : \boldsymbol{D}_g \to \boldsymbol{A}^m$ *be two vector functions. Their combination, noted* $f \overline{\bigotimes} g$*, is a new function with scope* $var(f) \cup var(g)$*, s.t.* $\forall t \in \boldsymbol{D}_{var(f)\cup var(g)}$, $(f \overline{\bigotimes} g)(t) = f(t) \otimes^m g(t)$.

Let $\boldsymbol{W} \subseteq \boldsymbol{X}$ *be a set of variables. The marginalization of* f *over* $\boldsymbol{W}$*, noted* $\underset{\boldsymbol{W}}{sort^m}\{f\}$*, is a new function whose scope is* $var(f) - \boldsymbol{W}$*, s.t.* $\forall t \in D_{var(f)-\boldsymbol{W}}$*,* $\underset{\boldsymbol{W}}{sort^m}\{f\}(t) = sort^m_{t' \in D_W} f(t \cdot t')$.

h_1:

X_1	X_2	
a	a	{2,4}
a	b	{1,3}
b	a	{4}
b	b	{3}

h_2:

X_2	
a	{1,3}
b	{1}

$h_1 \otimes^m h_2$:

X_1	X_2	
a	a	{3,5}
a	b	{2,4}
b	a	{5,7}
b	b	{4}

$sort^m_{X_2}\{h_1\}$:

X_1	
a	$\{1, 2\}$
b	$\{3, 4\}$

Fig. 1: Combination and marginalization over vector functions. For each pair of values of (X_1, X_2) the result of $h_1 \otimes^m h_2$ is an ordered set of size 2 obtained by pair-wise summation of the corresponding elements of h_1 and h_2. The result of $sort^m_{X_2}\{h_1\}$ is an ordered set containing the two lower values of function h_1 for each value of X_1.

Example 2. Figure 1 shows the combination and marginalization over two vector functions h_1 and h_2 for $m = 2$.

The m-best extension of an optimization problem $\mathcal{P}$ is a new reasoning task $\mathcal{P}^m$ that expresses the m-best task over $\mathcal{P}$.

Definition 12 (m-best extension). *Let* $\mathcal{P} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes, \Downarrow)$ *be an optimization problem defined over a semiring* $(\boldsymbol{A}, \otimes, min)$*. Its* m-best extension *is a new reasoning task* $\mathcal{P}^m = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}^m, \boldsymbol{F}^m, \overline{\bigotimes}, sort^m)$ *over semiring* $(\boldsymbol{A}^m, \otimes^m, sort^m)$*. Each function* $f : \boldsymbol{D}_f \to \boldsymbol{A}$ *in* $\boldsymbol{F}$ *is trivially transformed into a new vector function* $f' : \boldsymbol{D}_f \to \boldsymbol{A}^m$ *defined as* $f'(t) = \{f(t)\}$*. In words, function outcomes of* f *are transformed to singleton sets in* f'*. Then, the set* $\boldsymbol{F}^m$ *contains the new* f' *vector functions.*

The following theorem shows that the optimum of P^m corresponds to the set of m-best valuations of P.

Theorem 2 *Consider an optimization problem* $\mathcal{P} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes, \Downarrow)$ *defined over a semiring* $(A, \otimes, min)$*. Let* $\{F(t_1), \ldots, F(t_m)\}$ *be its m-best solutions. Let* P^m *be the m-best extension of* P*. The optimization task* P^m *computes the set of m-best solutions of* P*. Formally,*

$$sort_X{}^m\{\overline{\bigotimes}_{f \in \boldsymbol{F}^m} f\} = \{F(t_1), \ldots, F(t_m)\}$$

It is easy to see how the same extension applies to maximization tasks. The only difference is the set of valuations selected by operator $sort^m$.

4 Algorithm *elim-m-opt*

In this section we extend the bucket elimination algorithm to solving the m-best reasoning task. We subsequently show the derivation of the algorithm through an example.

4.1 The Algorithm Definition

Consider an optimization task $\mathcal{P}$. The bucket-elimination algorithm *elim-m-opt* solving $\mathcal{P}^m$ (i.e., the m-best extension of P) is described in Algorithm 1. First, the algorithm transforms scalar functions in F to their equivalent vector functions as described in Definition 12. Then, the algorithm processes the buckets from last to first as usual, using the two new combination and marginalization operators $\overline{\bigotimes}$ and $sort^m$, respectively. Roughly, the elimination of variable X_i from a vector function will produce a new vector function λ_i such that $\lambda_i(t)$ will contain the m-best extensions of t to the eliminated variables $X_{i+1}, \ldots, X_n$ with respect to the subproblem below the bucket variable in the bucket tree.

Since we are interested in recovering at least one complete assignment for each m-best solution, the algorithm propagates the variable assignments along with the vector messages when processing each bucket. These variable assignments are generated using the $argsort^m$ operator defined as follows.

Definition 13. *Operator* $argsort^m_{X_i} f$ *returns a vector function* $\overline{x_i}(t)$ *such that* $\forall t \in D_{var(f)\setminus X_i}$*, where* $\langle f(t \cdot {x_i}^1), \ldots, f(t \cdot {x_i}^m)\rangle$*, are the m-best valuations extending* t *to* X_i*.*

In words, $\overline{x_i}(t)$ is the vector of assignments to X_i that yields the m-best extensions to t.

The correctness of the algorithm follows from the formulation of the m-best optimization task as a reasoning task over a semiring.

Theorem 3 *Algorithm* elim-m-opt *is sound and complete for finding the m-best solutions over a graphical model.*

The details of how to efficiently compute combination and marginalization are one of the main contributions of our previous work [7]. We recap the main algorithmic issues and demonstrate the intuition behind the method in the following section by deriving *elim-m-opt* through an example. For clarity reasons, we omit the generation of actual m-best solution assignments.

Algorithm 1 elim-m-opt algorithm

Input: An optimization task $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{A}, \mathbf{F}, \bigotimes, min)$; An ordering of variables $o = \{X_1, \ldots, X_n\}$;

Output: A zero-arity function $\lambda_1 : \emptyset \rightarrow \mathbf{A}^m$ containing the solution of the m-best optimization task.

1: **Initialize:** Transform each function $f \in \mathbf{F}$ into a singleton vector function $h(t) = \{f(t)\}$; Generate an ordered partition of vector functions h in buckets $\mathbf{B}_1, \ldots, \mathbf{B}_n$, where $\mathbf{B}_i$ contains all the functions whose highest variable in their scope is X_i.
2: **Backward:**
3: **for** $i \leftarrow n$ down to 1 **do**
4: Generate $\lambda_i = sort^m_{X_i}(\overline{\bigotimes}_{f \in B_i} f)$
5: Generate assignment $\overline{x_i} = argsort^m_{X_i}(\overline{\bigotimes}_{f \in B_i})$, concatenate with relevant elements of the previously generated assignment messages.
6: Place λ_i and corresponding assignments in the bucket of the largest-index variable in $var(\lambda_i)$
7: **end for**
8: **Return:** λ_1

4.2 Deriving the Algorithm Using an Example

Consider a graphical model with three functions $F = \{f_1(z, x), f_2(z, y), f_3(t, z)\}$, and its optimization task over semiring $(\mathbb{N} \cup \{\infty\}, +, min)$ (i.e., the task is to find the minimum cost assignment). Finding the m-best valuations of the function $F(t, z, x, y) = f_3(t, z) + f_1(z, x) + f_2(z, y)$ can be expressed as finding Sol, defined by $Sol = \underset{t,x,z,y}{sort}^m \Big(f_3(t, z) + f_1(z, x) + f_2(z, y)$.

Since operator $sort^m$ is an extention of operator min, it inherits its distributive properties over summation. Due to this distributivity, we can apply symbolic manipulation and migrate each of the functions to the left of the $sort^m$ operator over variables that are not in its scope. In our example we rewrite as:

$$Sol = \underset{t}{sort}^m \underset{z}{sort}^m \left(f_3(t, z) + (\underset{x}{sort}^m f_1(z, x)) + \left(\underset{y}{sort}^m f_2(z, y) \right) \right) \tag{1}$$

The output of $sort^m$ is a set, so in order to make equation 1 well defined, we replace the summation operator by the combination over vector functions as in Definition 11.

$$Sol = \underset{t}{sort}^m \underset{z}{sort}^m (f_3(t, z) \overline{\bigotimes} (\underset{x}{sort}^m f_1(z, x)) \overline{\bigotimes} (\underset{y}{sort}^m f_2(z, y))) \tag{2}$$

BE computes expression 2 from right to left, corresponding to elimination ordering $o = \{T, Z, X, Y\}$. Figure 2 shows the messages passed between buckets and its bucket tree under o. Bucket $\mathbf{B}_Y$ containing function $f_2(z, y)$ is processed first. The algorithm applies operator $\underset{y}{sort}^m$ to $f_2(z, y)$, generating a vector function called a *message* and denoted by $\overline{\lambda_Y}(z)$ which is placed in $\mathbf{B}_Z$. Note that this

message associates each z with the vector of m best valuations of $f_2(z, y)$.Namely,

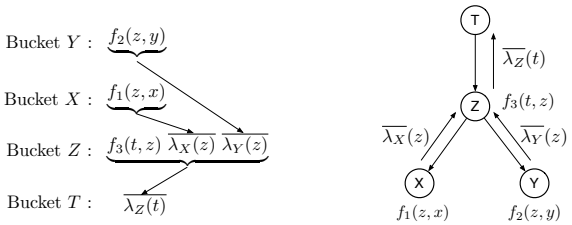
$$\underset{y}{sort}^m f_2(z, y) = (\lambda_Y^1(z), \ldots, \lambda_Y^j(z), \ldots, \lambda_Y^m(z)) = \overline{\lambda_Y}(z) \tag{3}$$

where for z each $\lambda_Y^j(z)$ is the j^{th} best value of $f_2(z, y)$. Similar computation is carried in $\mathbf{B}_X$ yielding $\overline{\lambda_X}(z)$ which is also placed in $\mathbf{B}_Z$.

When processing $\mathbf{B}_Z$, we need to compute, (see expression 2)

$$\overline{\lambda_Z}(t) = \underset{z}{sort}^m f_3(t, z) \overline{\bigotimes} \overline{\lambda_X}(z) \overline{\bigotimes} \overline{\lambda_Y}(z)$$

The result of the combination of the scalar function $f_3(t, z)$ with the two messages $\overline{\lambda_X}(z)$ and $\overline{\lambda_Y}(z)$ is a new vector function that has m^2 elements for each tuple (t, z). Applying $\underset{z}{sort}^m$ to the resulting combination generates the m best elements out of those m^2 yielding message $\overline{\lambda_Z}(t)$. As we show in [7], it is possible apply a more efficient procedure that would calculate at most $2m$ elements per tuple (t, z) instead. Finally, processing the last bucket yields the vector of m best solution costs for the entire problem: $Sol = \overline{\lambda_T} = \underset{t}{sort}^m \overline{\lambda_Z}(t)$ (see Figure 2a).



(a) Bucket messages (b) Bucket-tree

Fig. 2: Example of applying *elim-m-opt*

4.3 Complexity of *elim-m-opt*

Given n buckets, one for each variable X_i, $\mathbf{B}_i$ containing deg_i (i.e., the degree of the respective node in the bucket-tree) functions and at most w^* different variables with largest domain size k, it is possible to efficiently compute a messages between two buckets in $O(k^{w^*} m \cdot deg_i \log m)$, yielding the total time complexity of *elim-m-opt* of $O(\sum_{i=1}^{n} k^{w^*} m \cdot deg_i \log m)$ as we showed in [7]. Assuming $deg_i \leq deg$ and since $\sum_{i=1}^{n} deg_i \leq 2n$, we get the total time complexity of $O(nmk^{w^*} \log m)$. The space complexity is dominated by the size of the messages between buckets, each containing m costs-to-go for each of $O(k^{w^*})$ tuples. Having at most n such message yields the total space complexity of $O(mnk^{w^*})$.

4.4 The Mini-Bucket for the m-best

Mini-bucket Elimination *(MBE)* [5] is an approximation designed to avoid the space and time complexity of BE. Consider a bucket $\mathbf{B}_i$ and an integer bounding parameter z. *MBE* creates a z-partition $Q = \{Q_1, ..., Q_p\}$ of $\mathbf{B}_i$, where each set $Q_j \in Q$, called *mini-bucket*, includes no more than z variables. Then, each mini-bucket is processed separately, thus computing a set of messages $\{\lambda_{ij}\}_{j=1}^{p}$, where $\lambda_{ij} = \Downarrow_{X_i} (\bigotimes_{f \in Q_j} f)$. In general, greater values of z increase the quality of the bound.

Theorem 4 *[5] Given a reasoning task $\mathcal{P}$, MBE computes a bound on $\mathcal{P}$. Given an integer control parameter z, the time and space complexity of MBE is exponential in z.*

Recall that throughout this paper, we are assuming minimization tasks. In this case, MBE computes a lower bound.

Algorithm *mbe-m-opt* (Algorithm 2) is a straightforward extension of MBE to solve the m-best reasoning task, where the combination and marginalization operators are the ones defined over vector functions. The input of the algorithm is an optimization task P, and its output is a *m-best bound* on the m-best solutions of P.

Definition 14 (m-best lower bound). *Let $S = \{a_1, \ldots, a_j\}$ and $T = \{b_1, \ldots, b_k\}$ be two sets of ordered m-best elements (i.e., $S, T \in \boldsymbol{A}^m$). S is a m-best lower bound of T iff: (i) $|S| \geq |T|$, (ii) $b_1, b_2, \ldots, b_{l-1} \in S$ and $b_l, b_{l+1}, \ldots, b_k \notin S$, and (iii) $a_j < b_l$ (where by definition $b_l = 0$ if $l - 1 = |T|$).*

The idea behind this definition is that S contains all elements in T from b_1 up to b_{l-1} plus some other elements, and the maximum element in S (i.e., a_j) is smaller than the first element in T not included in S (i.e., b_l). For example, $S = \{4, 6, 10\}$ is not a 3-best lower bound of $T = \{4, 7, 10\}$, but it is a 3-best lower bound of $R = \{4, 11\}$.

Theorem 5 (*mbe-m-opt* bound and complexity) *Given a minimization task P,* mbe-m-opt *computes an m-best lower bound on the m-best optimization task $\mathcal{P}^m$. Given an integer control parameter z, the time and space complexity of* mbe-m-opt *is $O(mnk^z \log(m))$ and $O(mnk^z)$, respectively, where k is the maximum domain size and n is the number of variables.*

Sketch of proof. mbe-m-opt solves a relaxed version of the original problem. The relaxation is based on adding duplicates of the variables eliminated in different mini-buckets. In the limit (i.e., when m is infinity), the relaxed problem's solution set contains all solutions to the original problem (corresponding to assignments where duplicated variables take on the same domain value), plus a set of other solutions (corresponding to assignments where duplicated variables take on different domain values). When m is different to infinity, and depending on its value, the output of mbe-m-opt will contain all solutions to the original problem, some of them, or none. In all cases, the output satisfies the conditions to be an m-best lower bound of the set of m-best solutions to the original problem.

Algorithm 2 mbe-m-opt algorithm

Input: An optimization task $\mathcal{P} = (\mathbf{X}, \mathbf{D}, \mathbf{A}, \mathbf{F}, \bigotimes, min)$; An ordering of variables $o = \{X_1, \ldots, X_n\}$; parameter z.

Output: bounds on each of the m-best solution costs and the corresponding assignments for the expanded set of variables (i.e., node duplication).

1: **Initialize:** Generate an ordered partition of functions $\overline{f}(t) = \{f(t)\}$ into buckets $\mathbf{B}_1, \ldots, \mathbf{B}_n$, where $\mathbf{B}_i$ along o.
2: **Backward:**
3: **for** $i \leftarrow n$ down to 1 (Processing bucket B_i) **do**
4: Partition functions in bucket B_i into $\{Q_{i_1}, ..., Q_{i_l}\}$, where each Q_{i_j} has no more than z variables.
5: Generate cost messages $\lambda_{i_j} = sort^m_{X_i}(\overline{\bigotimes}_{f \in Q_{i_j}} f)$ and place each in the largest index variable in $var(Q_{i_j})$
6: **end for**
7: **Return:** The set of all buckets, and the vector of m-best costs bounds in the first bucket.

4.5 Using the m-best bound to tighten the first-best bound

Here is a simple, but quite fundamental observation. Recall that whenever upper or lower bounds are generated by solving a relaxed version of a problem, the relaxed problem's solution set contains all the solutions to the original problem. We next discuss the ramification of this observation.

Proposition 1. *Given the m-best solutions generated by* mbe-m-opt *(for clarity we consider minimization problem, the results can be extended for maximization)* $\tilde{C} = \{\tilde{p}_1 \leq \tilde{p}_2 \leq, ..., \leq \tilde{p}_m\}$*, let* p^{opt} *be the optimal value (the minimum cost) and let* j_0 *be the first index such that* $\tilde{p}_{j_0} = p^{opt}$*, or else we assign* $j_0 = m + 1$*. Then, if* $j_0 > m$*,* $\tilde{p}_m$ *is a lower bound on* p^{opt}*, which is as tight or tighter than all other* $\tilde{p}_1, ...\tilde{p}_{m-1}$*. In particular* $\tilde{p}_m$ *is tighter than the bound* $\tilde{p}_1$*.*

Proof. Let $\tilde{C} = \{\tilde{p}_1 \leq \tilde{p}_2 \leq, ..., \leq \tilde{p}_{N_1}$ be an ordered set of costs of all tuples over the relaxed problem (with duplicate variables). By the nature of any relaxation, $\tilde{C}$ must also contain all the cost values associated with solutions of the original problem denoted by $C = \{p_1 \leq \cdots \leq p_{N_2}\}$. Therefore, if j_0 is the first index such that $\tilde{p}_{j_0}$ coincides with p^{opt}, then clearly for all $i < j_0$, $p^{opt} \geq \tilde{p}_i$ with $\tilde{p}_{j-1}$ being the tightest lower-bound. Also, when $j_0 > m$ we have $\tilde{p}_m \leq c^{opt}$

In other words if $j \leq m$, we already have the optimal value, otherwise we can use $\tilde{p}_m$ as our better lower bound. Such tighter bounds would be useful during search algorithm such as A*. It is essential therefore to decide efficiently if a bound coincides with the exact optimal cost. Luckily, the nature of the MBE relaxation supplies us with an efficient decision scheme.

Proposition 2. *Given a m-best lower bound produced by* mbe-m-opt $\tilde{p}_1 \leq \tilde{p}_2 \leq , ... \leq \tilde{p}_m$*, deciding if* $\tilde{p}_j = p^{opt}$ *can be done efficiently.*

Proof. *mbe-m-opt* provides both the bounds on the m-best costs and for each bound a corresponding tuple, where assignments to duplicated variables is maintained. The first assignment from these m-best bounds (going from largest to smallest), that corresponds to a tuple whose duplicate variables are assigned identical value, is optimal. And, if no such tuple is observed, the optimal value is smaller than $\tilde{p}_m$. Since the above tests require just $O(nm)$ steps applied to m-best assignments already obtained in polytime, the claim follows.

5 Related work

Comparing with exact schemes. Lawler's approach, whose complexity is $O(nmT(n))$, where $T(n)$ is the complexity of finding a single best solution, was applied by Nilsson [12] to a join-tree. Nilsson utilizes the results from previous computations, achieving worst case complexity of $O(mT(n))$. If applied to a bucket-tree his algorithm dominates schemes mentioned here, with run time of $O(nk^{w*} + mn\log(mn) + mnk)$. Yanover and Weiss [15] developed a belief propagation approximation scheme for loopy graphs, called BMMF. When applied to juction tree it can function as an exact algorithm with complexity $O(mnk^{w*})$.

Two algorithms that are similar to elim-m-opt, both based on dynamic programming, are [13] and [6]. Seroussi and Golmard algorithm extracts the m solutions directly, by propagating the m best partial solutions along a junction tree that is pre-compiled. Given a junction tree with p cliques, each having at most deg children, the complexity of the algorithm is $O(m^2 p \cdot k^{w^*} deg)$. Elliot [6], explores the representation of Valued And-Or Acyclic Graph, i.e., smooth deterministic decomposable negation normal form (sd-DNNF) [3]. He propagates the m best solutions partial assignments to the problem variables along the DNNF structure which is pre-compiled as well. The complexity of Elliot's algorithm is $O(nk^{w^*} m \log m \cdot deg)$. Clearly our *elim-m-opt* algorithm does not boast the best complexity compared to the related methods. However, it demonstrates the direct applicability of established inference schemes to the generalized formulation of the m best solution problem as the *m-best reasoning problem*. Moreover, the main significance *elim-m-opt* is in the natural extension to an approximation scheme with guarantees on the solution quality that provides flexible trade off between accuracy and complexity.

Comparing with approximation schemes. In addition to BMMF, another extension of Nilsson's and Lawler's idea that yields an approximation scheme is an algorithm called STRIPES by [9]. They focus on m-MAP problem over binary Markov networks, solving each new subproblem by an LP relaxation. The algorithm solves the task exactly if the solutions to all LP relaxations are integral, and provides an upper bound of each m MAP assignments otherwise. In contrast, our algorithm *mbe-m-opt* can compute bounds over any graphical model (not only binary) and over a variety of m-best optimization tasks.

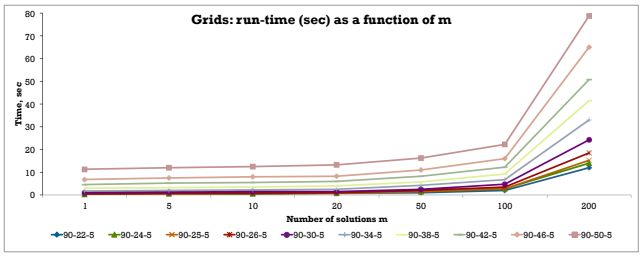


Fig. 3: *mbe-m-opt* run time (sec) as a function of number of solutions m for the grid instances. The z-bound=10, $m = [1, 5, 10, 20, 50, 100, 200]$, $n \in [500, 2500]$, $w^* \in [30, 74]$, $k = 2$.

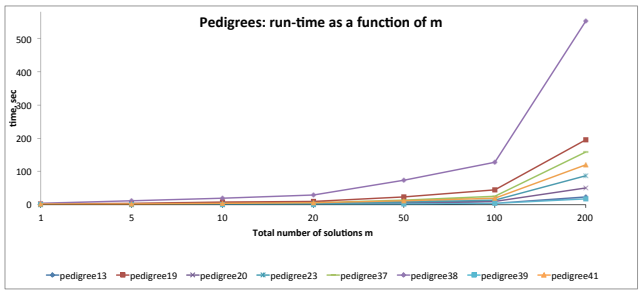


Fig. 4: *mbe-m-opt* run time (sec) as a function of number of solutions m for the pedigree instances. The z-bound=10, $m = \in [1, 5, 10, 20, 50, 100, 200]$, $n \in [400, 1272]$, $w^* \in [20, 30]$, $k = 4$.

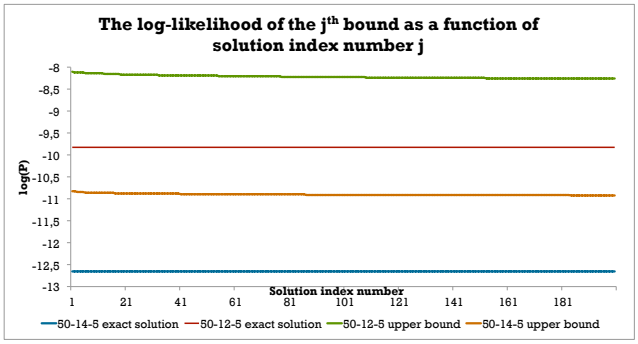


Fig. 5: The change in the cost of the j^{th} solution as j increases from 1 to 200 for two binary grid instances. Instance 50-12-5 has $n = 144$ and $w^* = 15$, 50-14-5 has n=196 and $w^* = 18$. The upper bounds outputted by *mbe-m-opt* with z-bound=10, the exact best solutions found by a search algorithm.

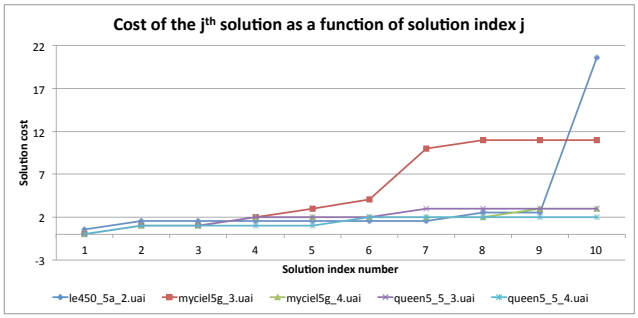


Fig. 6: The change in the cost of the j^{th} solution as j increases for chosen WCSP instances, $n \in [25, 475]$, $w^* \in [18, 293]$, $k \in [2, 4]$. Results obtained by *mbe-m-opt* with z-bound=10.

6 Empirical demonstrations

The first part of our experiments assumes solving m-best MPE task. We evaluated empirically algorithm *mbe-m-opt* with $m = \{1, 5, 10, 20, 50, 100, 200\}$ and with z-bound 10 on two sets of instances. The first set contained grid instances with a hundred to 2.5 thousand variables and tree-width from 12 to 50, the second - pedigree instances with several hundred variables and tree width from 15 to 30. Those instances were taken from the UAI 2008 evaluation. For clarity and space reasons we present only a subset of instances illustrating typical behaviour. Figures 3 and 4 present the dependence of the run-time on m, for a few selected instances.

Figure 5 shows the change in the upper bound as a function of index of the solution j. For these grid instances as j increases, the bound on the cost of the j^{th} solution approaches the exact best solution, but extremely slowly. However, as can be seen in Figure 6, it is not the case for all type of instances. This figure depicts some of the results of the experiments on the set of weighted CSPs from UAI 2008 competition. The instances in question have from 25 to 450 variables, domain size 2-4 and induced width 18-293. We can see considerable differences between the costs of the 1^{st} and 10^{th} for some instances. This demonstrates that there is a potential of improving the bound on the optimal assignment using the m-best bounds as discussed in Section 4.5.

We carried some comparison with BMMF by [15] on randomly generated 10 by 10 grids for MPE task. The run times of the algorithms are not comparable since our algorithm is implemented in C and BMMF in Matlab, which is inherently slower. For most instances that *mbe-m-opt* can solve exactly in under a second, BMMF takes more than 5 minutes. The algorithms also differ in the nature of the outputs: BMMF provides approximate solutions with no guarantees while *mbe-m-opt* generates bounds on all the m-best solutions. Still some information can be learned from viewing the two algorithms side by side as is demonstrated by a typical result in Figure 7. We know that in this case the

solutions obtained with z-bound equal to 1000 are exact, while z-bound equal to 10 yields an upper bound. BMMF outputs significantly less accurate results than *mbe-m-opt* with even a low z-bound. Admittedly, these experiments are quite preliminary and not conclusive.

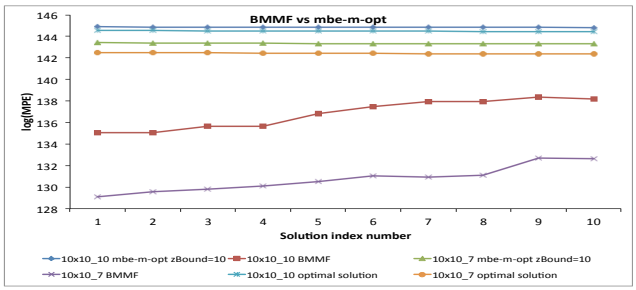


Fig. 7: Comparison of *mbe-m-opt* with z-bounds 10 and BMMF on random 10 by 10 grids. The exact solutions were obtained by *mbe-m-opt* with z-bound$> w^*$. While *mbe-m-opt* provides upper bounds on the solutions, BMMF gives no guarantees whether it outputs an upper or a lower bound. Also, its accuracy on these instances are clearly worse.

7 Conclusions

We presented a formulation of the m-best reasoning task within a framework of c-semiring. Such problem definition make existing inference and search algorithms immediately applicable for the task, as we demonstrated on the example of a new bucket-elimination algorithm for solving the m-best task over a graphical model, analyzed its performance and related it to other approaches in the literature.

The significance of the proposed algorithm is primarily in providing an inference framework for the m-best task that can both suggest approximation schemes and yield heuristic advice. Indeed, optimization tasks that seek a single optimal solution are solved far more effectively by search (e.g., branch and bound and best-first search), than by variable elimination, because they can benefit from the bounding power of the guiding cost function. It is also likely that search will be more effective for m-best task. The promise of the elim-m-opt inference algorithm is in its potential to yield viable lower- and upper-bounds for the m-best solutions via the mini-bucket algorithm, as we discussed.

Furthermore, it could also lead to loopy propagation message-passing schemes that are now the most common way for approximations in graphical models, since those schemes are relaxation of exact message-passing schemes such as bucket-elimination. In particular, our algorithm can be extended into a loopy max-prod for the m-best task. This approach will yield a direct loopy-propagation for the m-best reasoning problem, while the approach by Yanover and Weiss uses loopy

max-prod for solving a sequence of optimization problems in the style of Lawler's approach. Moreover, all such approximation extensions would be applicable to the broad range of graphical models captured by the unifying framework of c-semiring. Future work will focus on such extensions and on empirical evaluations of the emerging schemes.

The empirical analysis we provided is only preliminary. Yet it shows that *mbe-m-opt* scales even better than worst-case predict as a function of m. Comparison with other exact and approximation algorithms is left for future work.

Acknowledgement

This work was supported by NSF grant IIS-1065618.

References

1. Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.
2. S. Bistarelli, H. Faxgier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Basic properties and comparison. *Over-Constrained Systems*, pages 111–150, 1996.
3. A. Darwiche. Decomposable negation normal form. *Journal of the ACM (JACM)*, 48(4):608–647, 2001.
4. R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1):41–85, 1999.
5. R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, 50(2):107–153, 2003.
6. P.H. Elliott. Extracting the K Best Solutions from a Valued And-Or Acyclic Graph. Master's thesis, Massachusetts Institute of Technology, 2007.
7. N. Flerova and R. Dechter. M best solutions over Graphical Models. In *CRAGS10 Workshop*, 2010.
8. N. Flerova, R. Dechter, and E. Rollon. Bucket and mini-bucket schemes for m best solutions over graphical models. In *GKR'11 Workshop*, 2011.
9. M. Fromer and A. Globerson. An LP View of the M-best MAP problem. *Advances in Neural Information Processing Systems*, 22:567–575, 2009.
10. J. Kohlas and N. Wilson. Semiring induced valuation algebras: Exact and approximate local computation algorithms. *Artif. Intell.*, 172(11):1360–1399, 2008.
11. E.L. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18(7):401–405, 1972.
12. D. Nilsson. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing*, 8(2):159–173, 1998.
13. B. Seroussi and JL Golmard. An algorithm directly finding the K most probable configurations in Bayesian networks. *International Journal of Approximate Reasoning*, 11(3):205–233, 1994.
14. G. R. Shafer and P.P. Shenoy. Probability propagation. *Anals of Mathematics and Artificial Intelligence*, 2:327–352, 1990.
15. C. Yanover and Y. Weiss. Finding the M Most Probable Configurations Using Loopy Belief Propagation. In *Advances in Neural Information Processing Systems 16*. The MIT Press, 2004.