Bucket and mini-bucket Schemes for M Best Solutions over Graphical Models

Natalia Flerova, Emma Rollon and Rina Dechter

Abstract

The paper focuses on finding the m best solutions of a combinatorial optimization problem defined over a graphical model (e.g., the m most probable explanations for a Bayesian network). We describe *elim-m-opt*, a new bucket elimination algorithm for solving the m-best task, provide efficient implementation of its defining combination and marginalization operators, analyze its worst-case performance, and compare it with that of recent related algorithms. An extension to the mini-bucket framework, yielding a collection of bounds for each of the m-best solutions is discussed and empirically evaluated. We also formulate the m-best task as a regular reasoning task over general graphical models defined axiomatically, which makes all other inference algorithms applicable.

1 Introduction

Given an optimization problem, the objective typically is 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 [Lawler, 1972] 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 such as inference or search. In the paper we focus on graphical models and show how the well-known Bucket Elimination (BE) framework can be extended to compute the m-best solutions by a relatively simple modification of its underlying combination and marginalization operators [Dechter, 1999] yielding algorithm *elim-m-opt*.

We analyze the complexity of *elim-m-opt*, compare it to previously developed schemes, and discuss extensions to Mini-Bucket Elimination to compute bounds on each of the m-best solutions, yielding algorithm *mbe-m-opt*. We then show that the m-best task can be formalized more broadly as a reasoning task over a general graphical model as defined axiomatically by [Shenoy and Shafer, 1990]. As a consequence, any inference algorithm for solving a reasoning task over this representation (including Bucket Elimination) is immediately applicable as a sound and complete algorithm for the m-best task. In particular, elim-m-opt is therefore applicable to a wide range of graphical models.

We also provide empirical analysis for *mbe-m-opt* demonstrating its effectiveness both as an exact scheme as well as for approximation.

2 Background

We consider problems expressed as graphical models. The graphical model framework provides a common formalism to model a broad spectrum of problems, and a collection of general algorithms to efficiently solve them. Examples of graphical models are Markov and Bayesian networks [Pearl, 1988], constraint networks and influence diagrams. We next follow definitions as in [Kask *et al.*, 2005].

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 $\mathbf{S}$.

A valuation set $\mathbf{A}$ admits two binary operations over valuations: $\otimes : \mathbf{A} \times \mathbf{A} \rightarrow \mathbf{A}$ called combination and $\oplus : \mathbf{A} \times \mathbf{A} \rightarrow \mathbf{A}$ 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 denote by $\mathbf{D_Y}$ the set of tuples over a subset of variables $\mathbf{Y}$, also called the *domain of Y*. Functions are defined on subsets of variables of $\mathbf{X}$, called scopes and their range is a set $\mathbf{A}$ whose elements are called *valuations*. If $f : \mathbf{D_Y} \rightarrow \mathbf{A}$ 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 $\mathbf{D}_{var(f)}$.

DEFINITION **1** (combination operator)**.** *Let $f : \boldsymbol{D}_f \rightarrow \boldsymbol{A}$ and $g : \boldsymbol{D}_g \rightarrow \boldsymbol{A}$ be two functions. Their combination, noted $f \bigotimes g$ is a new function with scope $var(f) \cup var(g)$, s.t.*

$$\forall t \in \boldsymbol{D}_{var(f)\cup var(g)},\ (f \bigotimes g)(t) = f(t) \otimes g(t)$$

DEFINITION **2** (marginalization operator)**.** *Let $f : \boldsymbol{D}_f \rightarrow \boldsymbol{A}$ 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.*

$$\forall t \in D_{var(f)-\boldsymbol{W}}, (\Downarrow_{\boldsymbol{W}} f)(t) = \oplus_{t' \in D_W}(t \cdot t')$$

DEFINITION **3** (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 $(\mathbf{A}, \otimes, \oplus)$; $\boldsymbol{F} = \{f_1, \ldots, f_r\}$ is a set of discrete functions, where $var(f_j) \subseteq \boldsymbol{X}$ and $f_j : \boldsymbol{D}_{f_j} \rightarrow \boldsymbol{A}$; and $\bigotimes$ is the combination operator over functions (see Definition 1). The graphical model $\mathcal{M}$ represents the function $C(\boldsymbol{X}) = \bigotimes_{f \in \boldsymbol{F}} f$.*

DEFINITION **4** (**reasoning task**)**.** *A **reasoning task** is a tuple $\mathcal{P} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes, \Downarrow)$, where $(\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes)$ is a graphical model and $\Downarrow$ is a marginalization operator (Definition 2). The reasoning task is to compute $\Downarrow_{\boldsymbol{X}} C(\boldsymbol{X})$.*

For a reasoning task $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{A}, \mathbf{F}, \bigotimes, \Downarrow)$ 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.

It was shown in several works (e.g. [Shenoy and Shafer, 1990]) that, when the operators of a graphical models satisfy certain axioms, inference algorithms, such as variable elimination and join tree schemes, are sound and complete for the reasoning task. In this paper we assume graphical models obeying the Shenoy-Shafer axioms.

DEFINITION **5** (A proper reasoning task)**.** *Given a graphical model $\mathcal{M} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes)$, a reasoning task $\mathcal{P} = (\mathcal{M}, \Downarrow)$ is **proper** if its $\bigotimes$ and $\Downarrow$ operators satisfy the following 3 axioms:*

Axiom A1. Order of marginalization does not matter. Namely, $\Downarrow_{X,Y} f(x,y) = \Downarrow_{Y,X} f(x,y)$.

Axiom A2. $\bigotimes$ is associative and commutative.

Axiom A3. $\Downarrow$ distributes over $\bigotimes$. Namely, $\Downarrow_{X,Y} f(x) \bigotimes f(y) = \Downarrow_X f(x) \bigotimes \Downarrow_Y f(y)$.

An important parameter of a graphical model, characterizing its complexity, is the *induced width*.

DEFINITION **6** (induced graph, induced width)**.** *The **induced graph** of a graphical model relative to ordering o is an undirected graph that has variables as its vertices. The edges of the graph are added by: 1) connecting all variables that are in the scope of the same function, 2) processing nodes from last to first in o, recursively connecting preceding neighbours of each node. The **induced width** $w(o)$ relative to ordering o is the maximum number of preceding neighbours across all variables in the induced graph. The **induced width of the graphical model** w^* is the minimum induced width of all orderings.*

DEFINITION **7** (**bucket elimination**)**.** *Bucket elimination (**BE**) is an algorithmic framework that generalizes dynamic programming for many reasoning tasks [Dechter, 1999]. The input of BE (see Algorithm 1) 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 function 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.*

For example, algorithm *elim-opt*, which solves the optimization task, is obtained by substitution of the operators $\Downarrow_X f = \max_{S-X} f$ and $\bigotimes_j = \prod_j$.

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

DEFINITION **8** (**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.*

Algorithm 1 Bucket elimination

Input: A reasoning task $P = (\mathbf{X}, \mathbf{D}, \mathbf{A}, \mathbf{F}, \bigotimes, \Downarrow)$; An ordering of variables $o = \{X_1, \ldots, X_n\}$;
Output: A zero-arity function $\lambda_1 : \emptyset \rightarrow \mathbf{A}$ containing the solution of the reasoning task.
1: **Initialize:** Generate an ordered partition of functions 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 $\lambda_i = (\bigotimes_{f \in \mathbf{B}_i} f) \Downarrow_{X_i}$
5: Place λ_i in the bucket B_j where j is the largest-index variable in $var(\lambda_i)$
6: **end for**
7: **Return:** λ_1

Theorem 1. *[Dechter, 1999] Given a graphical model and a proper reasoning task $\mathcal{P} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes, \Downarrow)$ BE is sound and complete. Given an ordering o, the* time *and* space complexity *of $BE(\mathcal{P})$ is exponential in the induced width of the ordering.*

The m-best task over a graphical model is defined formally next.

DEFINITION **9** (m-best task)**.** *The m-best task over a graphical model $\mathcal{M}$ is to find m complete assignments $\boldsymbol{T} = \{t_1, \ldots, t_m\}$, such that $\forall t' \notin \boldsymbol{T}\ \forall t \in \boldsymbol{T}, C(t') \leq C(t)$. The solution is the set of valuations $\{C(t_1), \ldots, C(t_m)\}$, called m-best solutions.*

Most proofs are omitted for lack of space.

3 Algorithm *elim-m-opt*

This section provides the main contribution of our paper, presenting BE algorithm for the m-best task. We derive the algorithm through an example, having in mind the MPE (most probable explanation) task in probabilistic networks.

3.1 Deriving the algorithm using an example

Consider a graphical model with four variables $\{X, Y, Z, T\}$ having the following functions (for simplicity we use unnormalizes functions):

x	z	$f_1(z,x)$
0	0	2
0	1	2
1	0	5
1	1	1
2	0	4
2	1	3

y	z	$f_2(z,y)$
0	0	6
0	1	7
1	0	2
1	1	4
2	0	8
2	1	2

z	t	$f_3(t,z)$
0	0	1
0	1	2
1	0	4
1	1	3

Finding the m best solutions to $P(t,z,x,y) = f_3(t,z) \cdot f_1(z,x) \cdot f_2(z,y)$ can be expressed as finding Sol, defined by:

$$Sol = \underset{t,x,z,y}{sort}^{m} \Big(f_3(t,z) \cdot f_1(z,x) \cdot f_2(z,y) \Big) \qquad (1)$$

where operator $\underset{s}{sort}^m f(s)$ returns the first m elements of the set $\{f(s)_{s \in D_S}\}$ or the entire set, if its size is less than m. When the argument of the operator is a subset of the function's scope, like Y in $\underset{y}{sort}^m f(x,y)$, then $\underset{y}{sort}^m$ yields a set of the m best values of $f(x,y)$ for every fixed x (by default kept ordered) $\langle f(x, y^1(x)), \ldots, f(x, y^m(x)) \rangle$ where $f(x, y^j)$ is the j^{th} largest f value relative to D_Y, for a fixed $X = x$.

Since the output of the sort operators are vectors or sets, we will use *vector function* to denote a function whose range is a set (ordered set in our exposition and implementation). The *dimension of a vector function* is the size of its range vector.

Operator $sort^m f(x)$ can be viewed as applying operator $\max f(x)$ m times, while at each step generating and then removing the best element from $\{f(s)_{s \in D_S}\}$, and it inherits, therefore, its distributive properties over multiplication. 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:

$$Sol = \underset{t}{sort}^m \underset{z}{sort}^m \left(f_3(t,z) (\underset{x}{sort}^m f_1(z,x)) \left(\underset{y}{sort}^m f_2(z,y) \right) \right) \qquad (2)$$

However, since the output of $sort^m$ is a set, to make equation 2 well defined, we need to extend the multiplication operation to vector functions as a Cartesian product between all pair-wise elements. This yields a new binary *combination* operator denoted by $\overline{\bigotimes}$ between vector functions (generalizing combination with a scalar function in the obvious way), with which we rewrite expression 2 as

$$Sol = \underset{t}{sort}^m \underset{z}{sort}^m (f_3(t,z) \quad \overline{\bigotimes}(\underset{x}{sort}^m f_1(z,x)) \quad \overline{\bigotimes}(\underset{y}{sort}^m f_2(z,y))) \qquad (3)$$

The right to left computation in the order dictated by expression 3 can be carried out by a BE algorithm (see Figure 1) as follows. We assume that original input functions extends to vector functions e.g., $\overline{f}_i$ as $\overline{f}_i(t) = \{f_i(t)\}$. Then, the algorithm processes bucket $\mathbf{B}_Y$ generating vector function $\overline{\lambda_Y}(z)$ and bucket $\mathbf{B}_X$ generating vector function $\overline{\lambda_X}(z)$:

z	$\overline{\lambda_X}(z)$	$\overline{x}$	$\overline{\lambda_Y}(z)$	$\overline{y}$
0	{5,4,2}	{1,2,0}	{8,6,2}	{2,0,1}
1	{3,2,1}	{2,0,1}	{7,4,2}	{0,1,2}

When processing $\mathbf{B}_Z$, we compute $\overline{\lambda_Z}(t) = sort_z^m[\overline{f}_3(t,z) \overline{\bigotimes} \overline{\lambda_X}(z) \overline{\bigotimes} \overline{\lambda_Y}(z)]$. The result is a new vector function that has m^2 elements for each tuple (t,z) as shown below.

t	z	$f_3(t,z) \overline{\bigotimes} \overline{\lambda_X}(z) \overline{\bigotimes} \overline{\lambda_Y}(z)$
0	0	{40, 32, 30, 16, 24, 12, 10, 8, 4}
0	1	{84, 56, 48, 32, 28, 24, 16, 16, 8}
1	0	{80, 64, 60, 48, 32, 24, 20, 16, 8}
1	1	{63, 42, 36, 24, 21, 18, 12, 12, 6}

Applying $sort_z^m$ to the resulting combination we get vector function $\overline{\lambda_Z}(t)$ along with its variable assignments:

t	$\overline{\lambda_Z}(t)$	$\langle \overline{x}, \overline{y}, \overline{z} \rangle$
0	{84,56,48}	$\{\langle 2,0,1 \rangle, \langle 0,0,1 \rangle, \langle 2,1,1 \rangle\}$
1	{80,64,63}	$\{\langle 1,2,0 \rangle, \langle 2,2,0 \rangle, \langle 2,0,1 \rangle\}$

Finally, processing the last bucket yields the vector of m best solution costs for the entire problem and the corresponding assignments.

$\overline{\lambda_Z}(t)$	$\langle \overline{t}, \overline{z}, \overline{x}, \overline{y}, \rangle$
{84,80,64}	$\{\langle 0,1,2,0 \rangle, \langle 1,0,1,2 \rangle, \langle 1,0,2,2 \rangle\}$

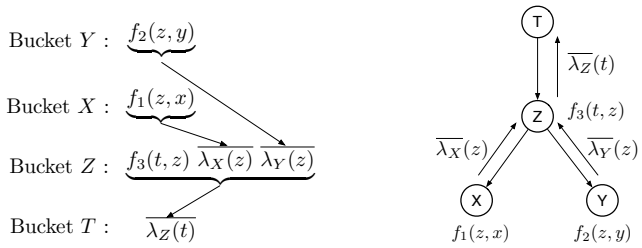
Since we are interested in recovering at least one complete assignment for each m-best solution, the algorithm propagates the variable assignments along with the vector messages when processing each bucket. These variable assignments are generated using the $argsort$ operator defined as follows.

DEFINITION **10.** *Operator $argsort_{X_i}^m f$ 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 costs 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.

3.2 The Algorithm definition

We next more formally define the generalized operators to be used in the bucket-elimination algorithm *elim-m-opt*. Let $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{A}, \mathbf{F}, \bigotimes)$ be a graphical model, over which we want to solve the m-best task. Let $2^{\mathbf{A}}$ be the set of subsets of $\mathbf{A}$, and let us call a function with range in $2^{\mathbf{A}}$ a vector function.



(a) Messages passed between buckets

(b) Bucket-tree

Figure 1: Example of applying *elim-m-opt*

DEFINITION **11** (combination and addition over sets)**.** *Let $S, T \in 2^{\mathbf{A}}$. Their combination, noted $S \otimes T$, is the set $\{a \otimes b \mid a \in S, b \in T\}$. Their addition, noted $sort\{S,T\}$, is the set of the m-best elements in the set $S \cup T$.*

DEFINITION **12** (combination and marginalization over vector functions)**.** *Let $f : \boldsymbol{D}_f \rightarrow 2^{\mathbf{A}}$ and $g : \boldsymbol{D}_g \rightarrow 2^{\mathbf{A}}$ be two vector functions. Their **combination**, noted $f\overline{\bigotimes}g$, is a new vector function defined on scope $var(f) \cup var(g)$ s.t. $\forall t \in D_{var(f)\cup var(g)}$, $f\overline{\bigotimes}g(t) = f(t_{[var(f)]}) \otimes g(t_{[var(g)]})$ The **marginalization** of f over $X_i \in var(f)$, noted $\underset{X_i}{sort^m} f$, is a function over scope $var(f) - \{X_i\}$ such that $\forall t \in D_{var(f)-X_i}$, $(\underset{X_i}{sort^m} f)(t) = sort^m\{\bigcup_{x\in D_{X_i}} f(t \cdot x)\}$*

The bucket-elimination algorithm *elim-m-opt* is described in Algorithm 2 using the two new combination and marginalization operators of $\overline{\bigotimes}$ and $sort^m$. The algorithm processes the buckets from last to first as usual. The bucket computation of a message-function (step 2) is detailed in the next subsection. The message-function associates each tuple in its domain with the m-best *costs-to-go* restricted to the subproblem below the bucket variable in the bucket tree. For clarity we omit the generation of actual m-best solution assignments.

While the correctness of the algorithm can be given directly, we defer to Section 4, where we show that the combination and marginalization operators just defined are an instance of a more general formulation of the m-best task as a *proper reasoning task over a proper graphical model* [Shenoy and Shafer, 1990].

3.3 The bucket processing algorithm

We will next show that the messages computed in a bucket can be obtained more efficiently than through a brute-force application of $\overline{\bigotimes}$ followed by $sort^m$. Consider processing $\mathbf{B}_Z$ (see Figure 1a). A brute-force computation of $\overline{\lambda_Z}(t) = \underset{z}{sort^m}(f_3(z,t)\overline{\bigotimes}\overline{\lambda_Y}(z)\overline{\bigotimes}\overline{\lambda_X}(z))$ for each t combines $f_3(z,t)$, $\overline{\lambda_Y}(z)$ and $\overline{\lambda_X}(z)$ for $\forall z \in D_Z$ first. This results in a vector function with scope $\{T, Z\}$ having m^2 elements that we call *candidate elements* and denote by $E(t,z)$. The second step is to apply $\underset{z}{sort^m} E(t,z)$ yielding the desired m best elements $\overline{\lambda_Z}(t)$.

However, since $\overline{\lambda_Y}(z)$ and $\overline{\lambda_X}(z)$ can be kept sorted, we can generate only a small subset of these m^2 candidates as

Algorithm 2 elim-m-opt algorithm

Input: A set of functions $\mathbf{F} = \{f_1, \cdots, f_n\}$ over scopes $\{S_1, \cdots, S_n\}$; An ordering of variables $o = \{X_1, \cdots, X_n\}$,

Output: A zero-arity function $\lambda_1 : \emptyset \rightarrow 2^{\mathbf{A}}$ 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

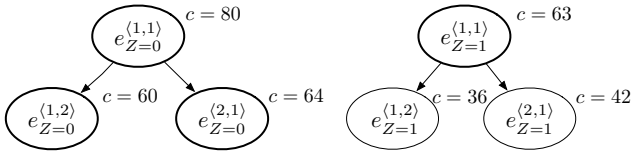


Figure 2: The explored search space for $T = 0$ and $m = 3$. The resulting message is $\overline{\lambda_Z}(1) = \{80, 64, 63\}$.

follows. We denote by $e_z^{\langle i,j\rangle}(t)$ the candidate element obtained by the product of the scalar function value $f_3(t,z)$ with the i^{th} element of $\overline{\lambda}_Y(z)$ and j^{th} element of $\overline{\lambda}_X(z)$, having cost $c_z^{\langle i,j\rangle}(t) = f_3(t,z) \cdot \lambda_Y^i(z) \cdot \lambda_X^j(z)$. We would like to generate the candidates $e_z^{\langle i,j\rangle}$ in decreasing order of their costs while taking their respective indices i and j into account.

The *child elements* of $e_z^{\langle i,j\rangle}(t)$, *children*$(e_z^{\langle i,j\rangle}(t))$ are obtained by replacing in the product either an element $\lambda_Y^i(z)$ with $\lambda_Y^{i+1}(z)$, or $\lambda_X^j(z)$ with $\lambda_X^{j+1}(z)$, but not both.

This leads to a forest-like search graph whose nodes are the candidate elements, where each search subspace corresponds to a different value of z denoted by $G_{Z=z}$ and rooted in $e_{Z=z}^{\langle 1,1\rangle}(t)$. Clearly, the cost along any path from a node to its descendants is non-increasing. It is easy to see that the m best elements $\overline{\lambda_Z}(t)$ can then be generated using a greedy best-first search across the forest search space $G_{Z=0} \cup G_{Z=1}$. It is easy to show that we do not need to keep more than m nodes on the OPEN list (the fringe of the search) at the same time. The general algorithm is described in Algorithm 3. The trace of the search for the elements of cost message $\overline{\lambda}_Z(t=1)$ for our running example is shown in Figure 2.

Proposition 1. *Given a bucket of a variable X over scope S having j functions $\{\lambda_1, ..., \lambda_j\}$ of dimension m, where m is the number of best-solutions sought and k bounds the domain size, the complexity of bucket processing is $O(k^{|S|} \cdot m \cdot j \log m)$, where $|S|$ is the scope size of S.*

Complexity of *elim-m-opt*: Given n buckets, one for each

Algorithm 3 Bucket processing

Input: $\mathbf{B}_X$ of variable X containing a set of ordered m-vector functions $\{\overline{\lambda_1}(S_1, X), \cdots, \overline{\lambda_d}(S_d, X)\}$
Output: m-vector function $\overline{\lambda_X}(S)$, where $S = \cup_{i=1}^{d} S_i - X$.

```
for all t ∈ D_S do
  for all x ∈ D_X do
    OPEN ← e^{⟨1,...,1⟩}_{X=x}(t); Sort OPEN;
  end for
  while j ≤ m, by +1 do
    n ← first element e^{⟨i_1,···,i_d⟩}_{X=x}(t) in OPEN. Remove n from OPEN;
    λ^j_X(s) ← n; {the j^th element is selected}
    C ← children(n) = {e^{⟨i_1,···,i_r+1,···,i_d⟩}_{X=x}(t) | r = 1..d};
    Insert each c ∈ C into OPEN maintaining order based on its computed value. Check for duplicates; Retain the m best nodes in OPEN, discard the rest.
  end while
end for
```

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, the total time complexity of *elim-m-opt* is $\sum_{i=1}^{n} O(k^{w^*} m \cdot deg_i \log m)$. Assuming $deg_i \leq deg$ and since $\sum_{i=1}^{n} deg_i \leq 2n$, we get complexity expression:

$$O(k^{w^*} m \cdot \log m \cdot 2n) = O(nmk^{w^*} \log m)$$

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 $O(mnk^{w^*})$. In summary:

Theorem 2 (complexity of *elim-m-opt*)**.** *Given a graphical model $(\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{F}, \bigotimes)$ having n variables, an ordering o, induced-width of w^*, a bucket-tree degree bounded by deg and the domain size bounded by k, and an operator $\Downarrow = max$, the time complexity of* elim-m-opt *is $O(k^{w^*} nm \cdot \log m)$ and its space complexity is $O(mnk^{w^*})$.*

Note that *elim-m-opt* is superior to applying Lawler's scheme [Lawler, 1972] to BE, which would lead to complexity of $O(mn^2 k^{w^*})$.

4 The M-best reasoning task

The m-best task defined via an optimization reasoning task is not itself in the form of a proper reasoning task over its graphical model. However, by some formal manipulation we can phrase it as a proper optimization reasoning task $\tilde{\mathcal{P}}(m)$ over an associated graphical model $\tilde{\mathcal{M}}(m)$. This formal exercise is worthwhile, because by doing so all inference algorithms, and in particular BE, are immediately applicable. We use definition of combination given in Definition 12.

DEFINITION **13** (m-best **reasoning** task)**.** *Let $\mathcal{M} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes)$ be a graphical model and $\mathcal{P} = (\mathcal{M}, \Downarrow)$ be an optimization reasoning task. The m-best graphical model relative to $\mathcal{M}$ is $\tilde{\mathcal{M}}(m) = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}_m, \overline{\boldsymbol{F}}, \overline{\bigotimes})$, where each function $f : \boldsymbol{D}_f \rightarrow \boldsymbol{A}$ in $\boldsymbol{F}$ is transformed into vector function: $\overline{f}(t) = \{f(t)\}$. The **m-best reasoning task** over $\tilde{\mathcal{M}}(m)$ is $\tilde{\mathcal{P}}(m) = (\tilde{\mathcal{M}}(m), sort^m)$, where $\overline{\bigotimes}$ and $sort^m$ are as in Definition 12.*

We now can show that our original m-best task over a graphical model $\mathcal{M}$ can be captured by $\tilde{\mathcal{P}}(m)$ over $\tilde{\mathcal{M}}(m)$. Namely,

Theorem 3. *Let $\mathcal{M} = (\boldsymbol{X}, \boldsymbol{D}, \boldsymbol{A}, \boldsymbol{F}, \bigotimes)$ be a graphical model and let $P = (M, \max)$ be its proper optimization reasoning task. Then, the reasoning task $\tilde{\mathcal{P}}(m) = (\tilde{\mathcal{M}}(m), sort^m)$ is proper and it coincides with the m-best task over $\mathcal{M}$.*

We can now conclude that the m-best task can be solved via traditional inference algorithms for reasoning tasks, and, in particular, by Bucket Elimination.

Theorem 4. *The m-best task over a graphical model $\mathcal{M}$ can be solved by bucket-elimination as defined by $BE(\tilde{\mathcal{P}}(m))$ and it coincides with* elim-m-opt($\mathcal{M}$)*.*

Proof. By Theorem 3, the m-best reasoning task $\tilde{\mathcal{P}}(m)$ computes the m-best solutions of $\mathcal{M}$ and since $\tilde{\mathcal{P}}_m$ is a proper reasoning task, it can be solved exactly by BE. It is easy to see that *elim-m-opt($\mathcal{M}$)*$= BE(\tilde{\mathcal{P}})$. □

5 Mini-Bucket Elimination for m-best Task

We now demonstrate the power of elim-m-opt in yielding useful bounds via mini-bucket-elimination.

5.1 The Mini-Bucket for the m-best

Mini-bucket Elimination *(MBE)* [Dechter and Rish, 2003] 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 5. *[Dechter and Rish, 2003] 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.*

Algorithm *m-best MBE* (*mbe-m-opt*) (given in Figure 4) is a straightforward extension of MBE to m-best reasoning task, where the combination and marginalization operators are the ones defined over vector functions. Algorithm *mbe-m-opt* solves m-best reasoning task $\tilde{\mathcal{P}}(m)$ and its output is a *m-best bound* on the m-best solutions.

DEFINITION **14** (m-best bound)**.** *Let $S = \{a_1, \ldots, a_j\}$ and $T = \{b_1, \ldots, b_k\}$ be two sets (i.e., $S, T \in 2^{\boldsymbol{A}}$). S is a m-best bound of T iff $\forall 1 \leq i \leq |T|$, $b_i \leq a_i$.*

Theorem 6 (*mbe-m-opt* bound and complexity)**.** *Given an m-best reasoning task $\tilde{\mathcal{P}}(m)$,* mbe-m-opt *computes an m-best bound on $\tilde{\mathcal{P}}(m)$. 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.*

Algorithm 4 MBE-m-opt algorithm

Input: An m-best reasoning task $\tilde{\mathcal{P}}(m) = (\mathbf{X}, \mathbf{D}, \mathbf{A}, \mathbf{F}, \bigotimes, sort^m)$; 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.

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

Here is a simple, but quite fundamental observation: 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 2. *Given the m-best solutions costs generated by* mbe-m-opt *(for clarity we consider MPE problem, the results can be extended for other reasoning tasks)* $\tilde{C} = \{\tilde{p}_1 \geq \tilde{p}_2 \geq , ..., \geq \tilde{p}_m\}$*, let* p^{opt} *be the optimal value (the probability of the most probable explanation) and let* j_0 *be the first index such that* $\tilde{p}_j = p^{opt}$*, or else we assign* $j_0 = m + 1$*. Then, if* $j_0 > m$*,* $\tilde{p}_m$ *is an upper 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 $\tilde{C} = \{\tilde{p}_1 \geq \tilde{p}_2 \geq, ..., \geq \tilde{p}_{N_1}$ be an ordered set of probabilities of all tuples over the relaxed problem (with duplicate variables). By the nature of any relaxation, $\tilde{C}$ must also contain all the probability values associated with solutions of the original problem denoted by $C = \{p_1 \geq \cdots \geq p_{N_2}\}$. Therefore, if j_0 is the first index such that $\tilde{p}_{j_0}$ coincides with p^{opt}, then clearly for all $i < j_0$, $p^{opt} \leq \tilde{p}_i$ with $\tilde{p}_{j-1}$ being the tightest upper-bound. Also, when $j_0 > m$ we have $\tilde{p}_m \geq c^{opt}$ □

In other words if $j \leq m$, we already have optimal value, otherwise we can use $\tilde{p}_m$ as our better upper 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 3. *Given a set of bounds produced by* mbe-m-opt $\tilde{p}_1 \geq \tilde{p}_2 \geq, ... \geq \tilde{p}_m$*, deciding if* $\tilde{p}_j = p^{opt}$ *can be done efficiently.*

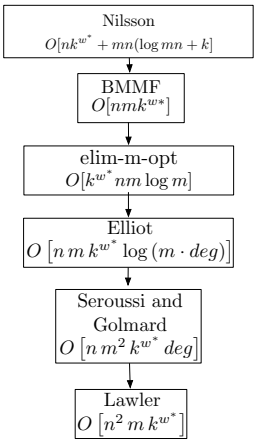


Figure 3: Complexity comparison. A parent node in the graph has a better complexity than its children.

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. □

6 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, 1998] to a join-tree. Instead of solving reformulated problems from scratch in each iteration, as Lawler does, Nilsson utilizes the results from previous computations, achieving worst case complexity of $O(mT(n))$. If applied to a bucket-tree Nilsson's algorithm is superior to all other schemes mentioned here, with run time of $O(nk^{w*} + mn\log(mn) + mnk)$. More recently Yanover and Weiss [Yanover and Weiss, 2004] developed a belief propagation approximation scheme for loopy graphs, called BMMF which also finds solutions iteratively. At each iteration BMMF uses loopy Belief Propagation to solve two new problems obtained by restricting the values of certain variables. When applied to junction tree it can function as an exact algorithm with complexity $O(mnk^{w*})$.

Two algorithms based on dynamic programming, similar to elim-m-opt, are [Seroussi and Golmard, 1994] and [Elliott, 2007]. Seroussi and Golmard extract the m solutions directly, by propagating the m best partial solutions along a junction tree. Given a junction tree with p cliques and branching degree deg, the complexity of the algorithm is $O(m^2 p \cdot k^{w^*} deg)$. Elliot [Elliott, 2007] propagates the m best partial solutions along a representation called Valued And-Or Acyclic Graph, i.e., smooth deterministic decomposable negation normal form (sd-DNNF) [Darwiche, 2001]. The

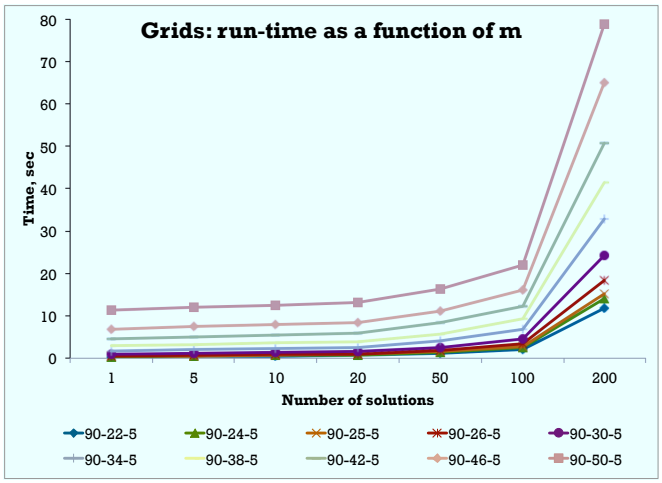


Figure 4: Run time as a function of number of solutions on grid instances

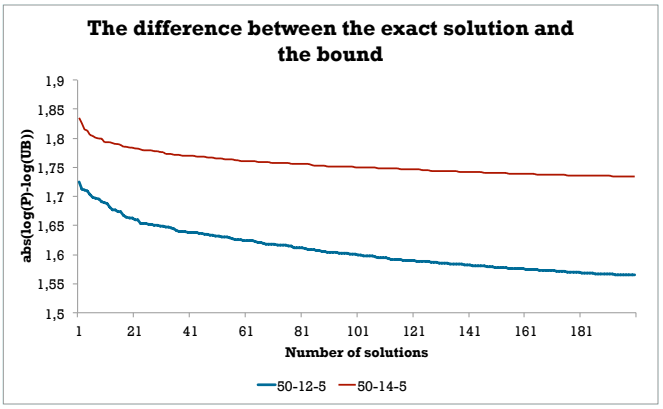


Figure 5: Accuracy of upper bound for different solutions on grid instances

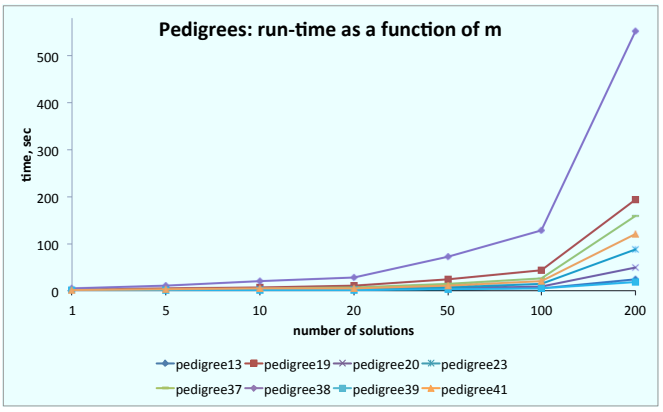


Figure 6: Run-time performance of *mbe-m-opt* on pedigrees

complexity of Elliot's algorithm is $O(nk^{w^*} m \log(mdeg)$.

Figure 3 depicts the dominance relationships between the time complexities of the exact algorithms discussed, when specialized to a bucket tree. Clearly our *elim-m-opt* algorithm does not boast the best complexity, being dominated by the algorithms [Nilsson, 1998; Yanover and Weiss, 2004]. 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 [Fromer and Globerson, 2009]. 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.

7 Empirical demonstrations

All experiments assume 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 4 and 6 present the dependence of the run-time on m, for a few selected instances. Figure 5 shows the change in accuracy as a function of m. Accuracy is measured by the absolute values of the difference between the optimal solution and the bounds on the j^{th} solution up to $m = 200$ solutions. For these grid instances as j increases, the bound on the cost of the j^{th} solution slowly approaches the exact best solution. This demonstrates that there is a potential of improving the bound on the optimal assignment using the m-best bounds as discussed in Section 5.2.

We carried some comparison with BMMF by [Yanover and Weiss, 2004] on randomly generated 10 by 10 grids. 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

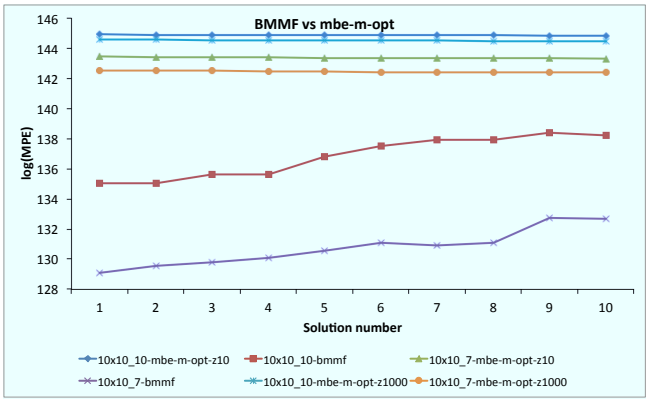


Figure 7: Comparison of *mbe-m-opt* with z-bounds 10 and 1000 vs. BMMF on random 10 by 10 grids

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.

8 Conclusions

We presented 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. We proved the algorithm's correctness by formally showing that the m-best task can be formulated as a reasoning task satisfying the Shenoy-Shafer axioms and, therefore, all inference algorithms and in particular our algorithm, elim-m-opt, are immediately applicable, sound and complete.

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 Shenoy and Shafer. 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.

Acknowledgments

This work was partially supported by NSF grants IIS-0713118 and IIS-1065618 and NIH grant 5R01HG004175-03. Thanks to the reviewers for their helpful feedback.

References

[Darwiche, 2001] A. Darwiche. Decomposable negation normal form. *Journal of the ACM (JACM)*, 48(4):608–647, 2001.

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

[Dechter, 1999] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1):41–85, 1999.

[Elliott, 2007] P.H. Elliott. Extracting the K Best Solutions from a Valued And-Or Acyclic Graph. Master's thesis, Massachusetts Institute of Technology, 2007.

[Fromer and Globerson, 2009] M. Fromer and A. Globerson. An LP View of the M-best MAP problem. *Advances in Neural Information Processing Systems*, 22:567–575, 2009.

[Kask *et al.*, 2005] Kalev Kask, Rina Dechter, Javier Larrosa, and Avi Dechter. Unifying cluster-tree decompositions for automated reasoning. *Artificial Intelligence Journal*, 2005.

[Lawler, 1972] 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.

[Nilsson, 1998] D. Nilsson. An efficient algorithm for finding the M most probable configurations in probabilistic expert systems. *Statistics and Computing*, 8(2):159–173, 1998.

[Pearl, 1988] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[Seroussi and Golmard, 1994] 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.

[Shenoy and Shafer, 1990] P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. *Uncertainty in Artificial Intelligence*, 4:169–198, 1990.

[Yanover and Weiss, 2004] 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.