Faster Exact MPE and Constrained Optimization with Deterministic Finite State Automata

Filippo Bistaffa
IIIA-CSIC
filippo.bistaffa@iiia.csic.es

Abstract

We propose a concise function representation based on deterministic finite state automata for exact *most probable explanation* and *constrained optimization* tasks in graphical models. We then exploit our concise representation within Bucket Elimination (BE). We denote our version of BE as FABE. FABE significantly improves the performance of BE in terms of runtime and memory requirements by minimizing redundancy. Indeed, results on *most probable explanation* and *weighted constraint satisfaction* benchmarks show that FABE often outperforms the state of the art, leading to significant runtime improvements (up to 5 orders of magnitude in our tests).

1 Introduction

Graphical models are a widely used theoretical framework that provides the basis for many reasoning tasks both with probabilistic and deterministic information [Dechter, 2013]. These models employ graphs to concisely represent the structure of the problem and the relations among variables.

One of the most important algorithms for exact reasoning on graphical models is Bucket Elimination (BE) proposed by Dechter [1996; 2013], a general approach based on the concept of *variable elimination* that accommodates many inference and optimization tasks, including *most probable explanation* (MPE) and *constrained optimization*. Solving these tasks is fundamental in real-world applications domains such as genetic linkage analysis [Fishelson and Geiger, 2002], protein side-chain interaction [Yanover *et al.*, 2008], and Earth observation satellite management [Bensana *et al.*, 1999]. BE is also a fundamental component—the approximate version of BE [Dechter, 1997] is used to compute the initial heuristic that guides the search—of all the AND/OR search algorithms by Marinescu *et al*. [2007; 2009; 2014; 2015] that represent the state of the art for exact MPE inference. On the other hand, BE is characterized by memory requirements that grow *exponentially* with respect to the *induced width* of the primal graph associated to the graphical model [Dechter, 2013], severely hindering its applicability to large exact reasoning tasks. As a consequence, several works have tried to mitigate this drawback [Dechter, 1997; Bistaffa *et al.*, 2017; Bistaffa and Farinelli, 2018], but none of these approaches really managed to overcome such a limitation. One of the major reasons for such memory requirements is the fact that the functions employed during BE's execution are usually represented as *tables*, whose size is the product of the domains of the variables in the scope, regardless of the actual values of such functions. This representation can lead to storing many repeated values in the same table, causing a potential waste of memory and other computational resources.

Against this background, in this paper we propose a function representation specifically devised for exact MPE inference and constrained optimization that, instead of the traditional mapping *variable assignment* $\rightarrow$ *value*, adopts a different approach that maps each value v to the minimal *finite state automaton* [Hopcroft and Ullman, 1979] representing all the variable assignments that are associated to v.

Compact function representations have already been investigated by different works [Srinivasan *et al.*, 1990; Bahar *et al.*, 1997; Chavira and Darwiche, 2007; Gogate and Domingos, 2013], which have analyzed the potential benefits of the use of different types of *decision diagrams* from a theoretical point of view. Nonetheless, as noted by Chavira and Darwiche [2007] the effectiveness of these approaches has been limited in practice, because the overhead they incur may very well outweigh any gains. Indeed, no approach based on decision diagrams has ever been shown to outperform the current state of the art, i.e., the AND/OR search algorithms by Marinescu *et al*. [2014; 2015].

To address the above-mentioned limitations, in this paper we propose the novel use of automata theory techniques to efficiently represent and manipulate functions in a compact way within our approach called FABE. By representing each value only once, and by exploiting the well-known capabilities of automata of compactly representing sets of strings (with a reduction that can be *up to exponential* with respect to a full table), we significantly improve the performance of BE in terms of runtime and memory requirements. In more detail, this paper advances the state of the art in the following ways:

- We propose a function representation for exact MPE inference and constrained optimization based on finite state automata, which we exploit within FABE.
- Results on standard benchmark datasets show that FABE often outperforms the current state of the art, with improvements of up to 5 orders of magnitude in our tests.

- Results also show that FABE outperforms the *structured message passing* (SMP) approach by Gogate and Domingos [2013], i.e., the most recent approach in the literature based on decision diagrams.
- Our work paves the way for an improved version of BE as a key component of AND/OR search algorithms, in which the computation of the initial heuristic can represent a bottleneck [Kishimoto *et al.*, 2015].

This paper is structured as follows. Section 2 provides the background on graphical models and deterministic finite state automata. Section 3 positions our approach *wrt* existing literature. Section 4 presents our function representation and how we exploit it within FABE. Section 5 presents our experimental evaluation on standard benchmark datasets, in which we compare FABE against the state of the art. Section 6 concludes the paper and outlines future research directions.

2 Background

2.1 Graphical Models

Graphical models (e.g., Bayesian Networks [Pearl, 1989], Markov Random Fields [Lauritzen, 1996], and Cost Networks [Dechter, 2013]) capture the factorization structure of a distribution over a set of n variables. Formally, a graphical model is a tuple $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, where $\mathbf{X} = \{X_i : i \in V\}$ is a set of variables indexed by set V and $\mathbf{D} = \{D_i : i \in V\}$ is the set of their finite domains of values. $\mathbf{F} = \{\psi_\alpha : \alpha \in F\}$ is a set of discrete local functions defined on subsets of variables, where $F \subseteq 2^V$ is a set of variable subsets. We use $\alpha \subseteq V$ and $\mathbf{X}_\alpha \subseteq \mathbf{X}$ to indicate the *scope* of function ψ_α, i.e., $\mathbf{X}_\alpha = var(\psi_\alpha) = \{X_i : i \in \alpha\}$. The function scopes yield a *primal graph* G whose vertices are the variables and whose edges connect any two variables that appear in the scope of the same function. An important inference task that appears in many real-world applications is MPE. MPE finds a complete assignment to the variables that has the highest probability (i.e., a mode of the joint probability), namely: $\mathbf{x}^* = \arg\max_{\mathbf{x}} \prod_{\alpha \in F} \psi_\alpha(\mathbf{X}_\alpha)$. The task is NP-hard [Pearl, 1989]. Another important task over deterministic graphical models (e.g., Cost Networks) is the optimization task of finding an assignment or a configuration to all the variables that minimizes the sum of the local functions, namely: $\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{\alpha \in F} \psi_\alpha(\mathbf{X}_\alpha)$. This is the task that has to be solved in Weighted Constraint Satisfaction Problems (WCSPs). The task is NP-hard to solve [Dechter, 2013].

To solve the above-mentioned tasks we consider the BE algorithm as discussed by Dechter [2013] (Algorithm 1). BE is a general algorithm that can accommodate several exact inference and optimization tasks over graphical models. In this paper we focus on the version that can optimally solve the above-mentioned MPE and optimization tasks. BE operates on the basis of a *variable ordering* d, which is used to partition the set of functions into sets called *buckets*, each associated with one variable of the graphical model. Each function is placed in the bucket associated with the last bucket that is associated with a variable in its scope. Then, buckets are processed from last to first by means of two fundamental operations, i.e., *combination* ($\otimes \in \{\prod, \sum\}$) and *projection* ($\Downarrow \in \{\max, \min\}$). All the functions in $bucket_p$, i.e., the current bucket, are combined with the $\otimes$ operation, and the result is the input of a $\Downarrow$ operation. Such an operation removes X_p from the scope, producing a new function h_p that does not involve X_p, which is then placed in the last bucket that is associated with a variable appearing in the scope of the new function. To solve the MPE (resp. optimization) task, $\otimes = \prod$ (resp. $\sum$) and $\Downarrow = \max$ (resp. $\min$) operators are used.

Algorithm 1 Bucket Elimination [Dechter, 2013]

Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, an ordering d.
Output: A max probability (resp. min cost) assignment.

```
Partition functions into buckets according to d.
Define ψ_i as the ⊗ of bucket_i associated with X_i.
for p ← n down to 1 do
    for ψ_p and messages h_1, h_2, ..., h_j in bucket_p do
        h_p ← ⇓_{X_p}(ψ_p ⊗ ⊗_{i=1}^{j} h_i).
        Place h_p into the largest index variable in its scope.
Assign maximizing (resp. minimizing) values in ordering
d, consulting functions in each bucket.
return Optimal solution value and assignment.
```

The computational complexity of the BE algorithm is directly determined by the ordering d. Formally, BE's time and space complexity are $\mathcal{O}\big(r \cdot k^{w^*(d)+1}\big)$ and $\mathcal{O}\big(n \cdot k^{w^*(d)}\big)$ respectively, where k bounds the domain size, and $w^*(d)$ is the induced width of its primal graph along d [Dechter, 2013]. Hence, it is of utmost importance to adopt a variable ordering d that minimizes the induced width $w^*(d)$. The task of computing such ordering is NP-complete, and, for this reason, a greedy procedure is usually adopted to compute a variable ordering of acceptable quality. Several $metric(\cdot)$ functions have been proposed in the literature. One of the most widely used is MIN-FILL, such that $metric(X_i)$ is the number of fill edges for X_i's parents in the primal graph [Dechter, 2013].

2.2 Deterministic Finite State Automata

Let Σ denote a finite alphabet of characters and Σ^* denote the set of all strings over Σ. The size $|\Sigma|$ of Σ is the number of characters in Σ. A language over Σ is any subset of Σ^*. A Deterministic Finite State Automaton (DFSA) [Hopcroft and Ullman, 1979] δ is specified by a tuple $\langle Q, \Sigma, t, s, F \rangle$, where Q is a finite set of states, Σ is an input alphabet, $t : Q \times \Sigma \rightarrow 2^Q$ is a transition function, $s \in Q$ is the start state and $F \subseteq Q$ is a set of final states. A string x over Σ is accepted (or recognized) by δ if there is a labeled path from s to a final state in F such that this path spells out the string x. Thus, the language L_δ of a DFSA δ is the set of all strings that are spelled out by paths from s to a final state in F. It is well known that a general DFSA can accept an *infinite* language (i.e., a infinite set of strings) [Hopcroft and Ullman, 1979]. In this paper we focus on Deterministic Acyclic Finite State Automata (DAFSA), i.e., DFSA whose corresponding graph is a directed acyclic graph. In contrast with general DFSA, DAFSA only accept *finite* languages [Daciuk, 2002].

3 Related Work

In recent years, a strand of literature has investigated the use of different algorithms on AND/OR search spaces, progressively showing the effectiveness of these approaches for exact MPE inference and constrained optimization. Specifically, *branch-and-bound* (AOBB) [Marinescu and Dechter, 2009], *best-first* (AOBF) [Marinescu and Dechter, 2007], *recursive best-first* (RBFAOO) [Kishimoto and Marinescu, 2014] and *parallel recursive best-first* (SPRBFAOO) [Kishimoto *et al.*, 2015] algorithms have been proposed. To this day, RBFAOO [Kishimoto and Marinescu, 2014] and SPRBFAOO [Kishimoto *et al.*, 2015] still represent the state of the art for exact MPE inference, since, to the best of our knowledge, no algorithms have been shown to outperform them in terms of runtime. For this reason, we compare our approach against these two algorithms in Section 5. All these approaches use the standard tabular representation to store functions in memory. In the context of constrained optimization, the only notable approach that tries to reduce the size of tables in memory is the one by Bistaffa *et al*. [2017], which avoids representing *unfeasible* assignments during the solution of WCSPs by means of BE.

The task of concisely representing functions for inference has been studied by different works over the years. In the context of this work, the most closely related strand of literature is concerned with the use of *decision diagrams* to compactly represent knowledge. A decision diagram is a directed acyclic graph that maps several decision nodes to two or more terminal nodes. Different types of decision diagrams exist, depending on the domains of the variables and the types of employed operations. Along these lines, several works have proposed the use of Binary Decision Diagrams (BDDs) [Gogate and Domingos, 2013], Algebraic Decision Diagrams (ADDs) [Bahar *et al.*, 1997], affine ADDs [Sanner and McAllester, 2005], Multi-valued Decision Diagrams (MDDs) [Srinivasan *et al.*, 1990; Chavira and Darwiche, 2007] within variable elimination and/or message passing algorithms for inference.

While it is true that DAFSA and decision diagrams are very closely related, to the best of our knowledge the use of automata theory has never been investigated in the context of concise function representation and inference. In this sense, the use of DAFSA cannot be considered entirely novel (indeed, a DAFSA can be considered equivalent to an MDD). On the other hand, our proposed use of different automata theory techniques for automata compilation [Daciuk, 2002], minimization [Bubenzer, 2011], union [Han and Salomaa, 2008], and determinization [Hopcroft and Ullman, 1979] is, as far as we know, entirely novel in the context of inference on graphical models.

Moreover, as noted by Chavira and Darwiche [2007] the effectiveness of approaches based on decision diagrams has been limited in practice, because the overhead they incur may very well outweigh any gains. Indeed, no approach based on any compact representations has ever been shown to outperform the current state of the art, i.e., the AND/OR search algorithms by Marinescu *et al*. [2014; 2015]. Against this background, we believe that our contribution, i.e., to show that a new approach based on well-known theoretical concepts can still be competitive with the current state of the art, can be very useful for the scientific community.

Given the limited amount of available space, in this paper we decided to compare FABE only with the SMP approach by Gogate and Domingos [2013], since it is, to the best of our knowledge, the most recent and relevant one among the above-mentioned works focusing on decision diagrams.[1] We also remark that, despite the importance of considering this strand of literature, the current state of the art is represented by more recent approaches based on completely different techniques [Kishimoto and Marinescu, 2014; Kishimoto *et al.*, 2015]. Hence, our experimental evaluation in Section 5 is mainly devoted to the comparison against these approaches.

The use of MDDs has also been investigated by Mateescu *et al*. [2008] within the above-mentioned AND/OR search algorithms, but this approach has been subsumed and outperformed by more recent and advanced approaches based on AND/OR search trees, such as (SP)RBFAOO. For this reason, in Section 5 we only compare with the most recent ones in such a strand of literature.[2]

Lifted probabilistic inference (LPI) [Kersting, 2012] is also concerned with reducing redundancy within probabilistic inference. Specifically, LPI tackles redundancy *between different factors*, whereas we tackle redundancy *inside the same factor*. Assessing the effectiveness of the combined approach *wrt* to the separate ones is a non-trivial research question, which will be considered in future work. Finally, the approach by Demeulenaere *et al*. [2016] aims at representing *compact tables* by maintaining *generalized arc consistency* (i.e., by removing from domains all values that have no support on a constraint) and not by minimizing redundancy due to repeated values, as we explain in the next section.

4 A Novel DAFSA-based Version of BE

All the datasets commonly used as benchmarks for MPE [Kishimoto *et al.*, 2015] and constrained optimization [Marinescu and Dechter, 2009] are characterized by a very high *redundancy*, i.e., many different variable assignments are associated to the same value in the local functions. Figure 1 shows that the value of redundancy for local functions (defined as $1 - \frac{\text{number of unique values}}{\text{total number of values}}$) for all MPE and WCSP instances is always above 80% (except for smaller `grid` instances).

Furthermore, in probabilistic graphical models, local functions represent probabilities with values in the interval $[0, 1]$, which, in theory, contains *infinite* real values. In practice, such values are represented by *floating point numbers* that can only represent a *finite* amount of values. Thus, while a table ψ has an arbitrarily large size that is the product of the domains of the variables in its scope, in practice the maximum number of unique values in ψ is bounded by a parameter that depends on the numerical representation. These remarks motivate the study of a novel concise representation that exploits such a redundancy to reduce the amount of computation. Notice that state of the art approaches for exact inference [Kishimoto *et al.*, 2015] represent functions as *full tables*, whose size is the product of the domains of the variables in the scope.

[1] Our decision is also motivated by the fact that, to the best of our knowledge, none of these works has an available implementation.

[2] We cannot directly compare with the approach by Mateescu *et al*. [2008] also because its implementation is not publicly available.

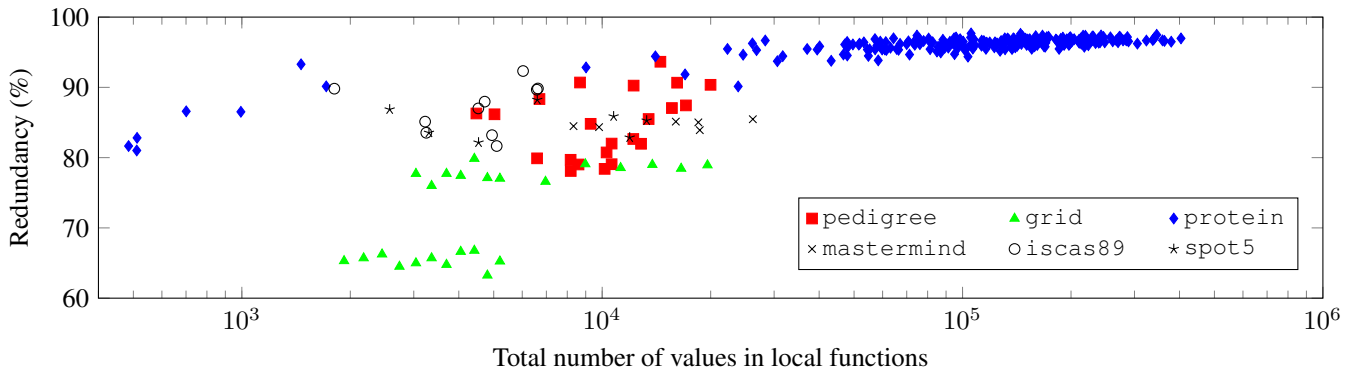

Figure 1: Redundancy in MPE and WCSPs instances. Best viewed in colors.

In this paper we propose a way to represent functions by means of DAFSA, as shown in the example in Figure 2. In the traditional way of representing functions as tables, rows are indexed using variable assignments as *keys* (Figure 2, left).

In contrast, here we propose a novel approach that uses *values* as keys (Figure 2, right). Formally,

Definition 1. *Given a function ψ that maps each possible assignment of the variables in its scope to a value $v \in \mathbb{R} \cup \{\infty\}$,[3] we denote as $D(\psi)$ its corresponding representation in terms of DAFSA. Formally, $D(\psi) = \{(v, \delta)\}$, where v is a value in ψ and δ is the* minimal *DAFSA that accepts all the strings corresponding to the variable assignments that were mapped to v in ψ. For the sake of simplicity, we do not represent the scope of the function in $D(\psi)$, as we assume it is equal to $var(\psi)$. We label a transition that accepts all the values of a variable's domain as $*$. Notice that each δ is acyclic because it accepts a finite language [Daciuk, 2002].*

Remark 1. *Given that values are employed as keys in our function representation, it is crucial to ensure the absence of duplicates in such a set of keys, i.e., we must be able to correctly determine whether two values v_1 and v_2 are equal. While this is a trivial task in theory, in practice it can be very tricky when v_1 and v_2 are floating point numbers representing real values. Indeed, even if v_1 and v_2 are theoretically equal, their floating point representations can differ due to numerical errors implicit in floating point arithmetic, especially if v_1 and v_2 are the result of a series of operations whose numerical errors have accumulated. To mitigate this aspect, we use a well-known technique for comparing floating point numbers known as ϵ-*comparison*, i.e., v_1 and v_2 are considered equal if they differ by a quantity smaller than a small ϵ. While there exist more advanced techniques of tackling numerical issues related to floating point numbers and their arithmetic [Higham, 2002], they are well beyond the scope of this paper. This should* not *be considered as an approximation, rather as a standard method to avoid the propagation of numerical errors.*

[3]We allow ∞ as a possible value, since it can used to represent variable assignments that violate hard constraints in WCSPs.

X_1	X_2	X_4	ψ
0	0	0	v_1
0	0	1	v_1
0	1	0	v_2
0	1	1	v_1
1	0	0	v_3
1	0	1	∞
1	1	0	v_3
1	1	1	∞

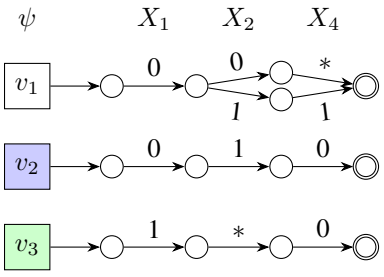

Figure 2: Standard table (left) and corresponding DAFSA-based representation (right). All variables are binary. Best viewed in colors.

A crucial property of DAFSA is that one path can accept multiple strings, or, in our case, represent multiple variable assignments. In the example in Figure 2, the DAFSA corresponding to v_3 contains only one path, but it represents both $\langle 1, 0, 0\rangle$ and $\langle 1, 1, 0\rangle$. By exploiting this property, our representation can reach a reduction in terms of memory that is, in the best case, *up to* exponential *wrt* the traditional table representation. We remark that memory is the main bottleneck that limits the scalability of BE, hence reducing its memory requirements is crucial, leading to significant improvements as shown by our results in the experimental section. Finally, our representation allows one to trivially avoid representing unfeasible assignments, similarly to [Bistaffa *et al.*, 2017]. Predicting the size (e.g., the number of states) of a minimal DAFSA accepting a given set of strings remains, to the best of our knowledge, an open problem, since it depends on the common prefixes/suffixes of the input set.

A minimal DAFSA can be efficiently constructed from a set of assignments by using the algorithm described by Daciuk [2002]. Since all the strings accepted by each DAFSA are of the same length (equal to the cardinality of the scope of the function), so are all the paths in the DAFSA. Thus, there is a mapping between each edge at depth i in each path and the i^{th} variable in the scope (see Figure 2). Without loss of generality, our representation always maintains the variables in the scope ordered *wrt* their natural ordering. We now discuss our DAFSA-based version of BE, i.e., its $\otimes$ and $\Downarrow$ operations.

4.1 A DAFSA-Based Version of $\otimes$

In order to better discuss our DAFSA-based version the $\otimes$ operation, let us first recall how this operation works for traditional tabular functions with an example (Figure 3).

X_1	X_2	X_4	ψ_1
0	0	0	v_1
0	0	1	v_1
0	1	0	v_2
0	1	1	v_1
1	0	0	v_3
1	0	1	∞
1	1	0	v_3
1	1	1	∞

$\otimes$

X_3	X_4	ψ_2
0	0	v_3
0	1	v_7
1	0	v_3
1	1	v_1

$=$

X_1	X_2	X_3	X_4	$\psi_{1\otimes 2}$
0	0	0	0	$v_1 \otimes v_3$
0	0	0	1	$v_1 \otimes v_7$
0	0	1	0	$v_1 \otimes v_3$
0	0	1	1	$v_1 \otimes v_1$
0	1	0	0	$v_2 \otimes v_3$
0	1	0	1	$v_1 \otimes v_7$
0	1	1	0	$v_2 \otimes v_3$
		$\vdots$		$\vdots$

Figure 3: An example of the $\otimes$ operation.

The result of the $\otimes$ operation is a new function whose scope is the union of the scopes of the input functions, and in which the value of each variable assignment is the $\otimes \in \{\cdot, +\}$ of the values of the corresponding assignments (i.e., with the same assignments of the corresponding variables) in the input functions. For example, the assignment $\langle X_1 = 0, X_2 = 1, X_3 = 1, X_4 = 0\rangle$ in the result table corresponds to $\langle X_1 = 0, X_2 = 1, X_4 = 0\rangle$ and $\langle X_3 = 1, X_4 = 0\rangle$ in the input tables, hence its value is $v_2 \otimes v_3$. The $\otimes$ operation is closely related to the *inner join* of relational algebra.

To efficiently implement $D(\psi_1) \otimes D(\psi_2)$ we will make use of the *intersection* operation on automata [Hopcroft and Ullman, 1979]. Intuitively, the intersection of two automata accepting respectively L_1 and L_2 is an automaton that accepts $L_1 \cap L_2$, i.e., all the strings appearing both in L_1 and L_2. In our case, we will exploit the intersection operation to identify all the corresponding variable assignments in $D(\psi_1)$ and $D(\psi_2)$. To make this possible, we first have to make sure that both functions have the same scope, so that corresponding levels in $D(\psi_1)$ and $D(\psi_2)$ correspond to the same variables. We achieve this by means of the ADDLEVELS operation.

Definition 2. *Given two functions $D(\psi_1)$ and $D(\psi_2)$, the* ADDLEVELS *operation inserts (i) one or more levels labeled with $*$ in each DAFSA and (ii) one or more variables in the respective scopes, in a way that the resulting scope is $var(\psi_1) \cup var(\psi_2)$. Each level and variable is added so as to maintain the scope ordered* wrt *the variable ordering.*

Figure 4 shows an example of ADDLEVELS. The operation of adding one level to a DAFSA δ has a linear complexity *wrt* the number of paths in δ. This has to be executed $|D(\psi_1)| \cdot |var(\psi_2) \setminus var(\psi_1)| + |D(\psi_2)| \cdot |var(\psi_1) \setminus var(\psi_2)|$ times.

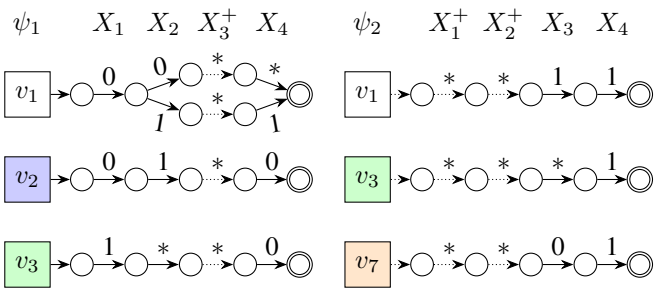

Figure 4: The result of the ADDLEVELS operation on $D(\psi_1)$ and $D(\psi_2)$, where ψ_1 and ψ_2 are the tables in Figure 3. Added levels and variables are denoted with dotted lines and $^+$ superscript.

Algorithm 2 $D(\psi_1) \otimes D(\psi_2)$

1: $(D(\psi_1)', D(\psi_2)') = \text{ADDLEVELS}(D(\psi_1), D(\psi_2))$.
2: **for all** $(v_i, \delta_i) \in D(\psi_1)', (v_j, \delta_j) \in D(\psi_2)'$ **do**
3: **if** $\exists (v_i \otimes v_j, \delta_k) \in D(\psi_1) \otimes D(\psi_2)$ **then**
4: $\delta_k = \delta_k \cup (\delta_i \cap \delta_j)$.
5: **else**
6: Add $\{(v_i \otimes v_j, \delta_i \cap \delta_j)\}$ to $D(\psi_1) \otimes D(\psi_2)$.
7: **return** $D(\psi_1) \otimes D(\psi_2)$.

Our DAFSA-based $\otimes$ operation is implemented by Algorithm 2. Intuitively, for each couple of values (v_i, v_j), where v_i and v_j are values in $D(\psi_1)$ and $D(\psi_2)$ respectively, we compute the variable assignments associated to their $\otimes$ by computing the intersection $\delta_i \cap \delta_j$ between the corresponding DAFSA δ_i and δ_j. The result is then associated to the value $v_i \otimes v_j$ in the output function. We maintain only one entry for each value $v_i \otimes v_j$ (see Remark 1 in this respect) by accumulating (i.e., taking the union of) all the DAFSA that are associated to the same value (Line 4).

Union and intersection over DAFSA have a time complexity of $\mathcal{O}(nm)$ [Han and Salomaa, 2008], where n and m are the number of states of the input automata. Depending on their implementations, such operations may not directly produce a minimal DAFSA. Nonetheless, DAFSA can be minimized in linear time *wrt* the number of states with the algorithm by Bubenzer [2011].

4.2 A DAFSA-Based Version of $\Downarrow$

The $\Downarrow \in \{\max, \min\}$ operation effectively realizes *variable elimination* within the BE algorithm. Specifically, $\Downarrow_{X_i} \psi$ removes X_i from the scope of ψ, and, from all the rows that possibly have equal variable assignments as a result of the elimination of the column associated to X_i, it only maintains the one with the $\max$ (in the case of MPE, or $\min$ in the case of optimization) value. Like $\otimes$, $\Downarrow$ is also related to a relational algebra operation, i.e., the *project* operation. In terms of SQL, $\Downarrow_{X_i} \psi$ is equivalent to `SELECT` $var(\psi) \setminus X_i, \max(\psi(\cdot))$ `FROM` ψ `GROUP BY` $var(\psi) \setminus X_i$, in the case of $\max$.

We realize the elimination of the column associated to X_i with the REMOVELEVEL operation, which can be intuitively thought of as the inverse of ADDLEVELS. REMOVE LEVEL$(D(\psi), X_i)$ removes X_i from the scope of $D(\psi)$ and collapses all the edges at the level associated to X_i from all the DAFSA in $D(\psi)$. REMOVELEVEL has similar computational properties *wrt* ADDLEVELS, so we do not repeat the discussion due to space limitations. Notice that the REMOVELEVEL operation could result in a *non-deterministic* automaton if the removal happens in correspondence of a branching. Our implementation takes this into account by employing a determinization algorithm [Hopcroft and Ullman, 1979].

In general, determinising an automaton could produce a growth (up to exponential, in the worst case) of the number of states. On the other hand, in all our experiments such a worst-case never happens and the growth factor is, on average, only around 10%. Our results confirm that such a small growth does not affect the overall performance of our approach, which is able to outperform the competitors as described in Section 5.

Algorithm 3 $\Downarrow_{X_i} D(\psi)$

1: $D(\psi)' = \text{REMOVELEVEL}(D(\psi), X_i)$.
2: **for all** $(v_i, \delta_i) \in D(\psi)'$ with decr. (resp. incr.) v_i **do**
3: $\quad \delta_i = \delta_i \setminus \delta_{prec}$.
4: $\quad \delta_{prec} = \delta_{prec} \cup \delta_i$.
5: **return** $D(\psi)'$.

We then implement the maximization (resp. minimization) of the values as follows. Without loss of generality, we assume that the values $v_1, \ldots, v_{|D(\psi)|}$ are in decreasing (resp. increasing) order. For each $(v_i, \delta_i) \in D(\psi)$, we subtract from δ_i all δ_j such that v_j precedes v_i in the above-mentioned ordering (i.e., $v_j \geq v_i$, resp. $\leq$). In this way, we remove all duplicate variable assignments and we ensure that each assignment is *only* associated to the maximum (resp. minimum) value, correctly implementing the $\Downarrow$ operation. Subtraction over DAFSA has a time complexity of $\mathcal{O}(nm)$ [Han and Salomaa, 2008], where n and m are the number of states of the input automata. Algorithm 3 details our $\Downarrow$ implementation.

We remark that both our versions of $\otimes$ and $\Downarrow$ entirely operate on our concise representation, never expanding any function to a full table. We directly employ our $\otimes$ and $\Downarrow$ operations within Algorithm 1. We call our DAFSA-based version of BE "Finite state Automata Bucket Elimination" (FABE). Since the results of our $\otimes$ and $\Downarrow$ operations are equivalent to the original ones, it follows that, as BE, FABE is also an exact algorithm. Finally, we remark that our $\otimes$ and $\Downarrow$ operations can directly be used within the approximated version of BE, i.e., Mini-Bucket Elimination (MBE) [Dechter, 1997].

5 Experiments

We evaluate FABE on standard benchmark datasets[4] for exact MPE inference (i.e., `protein`, `pedigree`, `grid`) [Kishimoto and Marinescu, 2014; Kishimoto *et al.*, 2015] and WCSP (i.e., `spot5`, `mastermind`, `iscas89`) [Marinescu and Dechter, 2009], comparing it with several state of the art approaches. For MPE, we consider RBFAOO [Kishimoto and Marinescu, 2014] as our main competitor since it has been empirically shown to be superior to other sequential algorithms for exact MPE inference, namely AOBB [Marinescu and Dechter, 2009] and AOBF [Marinescu and Dechter, 2007] (see Section 3). We cannot directly compare against the *parallel* version of RBFAOO, i.e., SPRBFAOO [Kishimoto *et al.*, 2015], because its implementation has not been made public. We discarded the option of re-implementing SPRBFAOO, as it would have probably led to an unfair comparison due to a sub-optimal implementation. Nonetheless, since RBFAOO is also used as a baseline for speed-up calculation in [Kishimoto *et al.*, 2015], in Table 2 we compare our values of speed-up with the ones reported for SPRBFAOO by its authors. As a second competitor we consider `toulbar` [Hurley *et al.*, 2016], a widely used open-source solver used for exact optimization. We also compare FABE against the SMP approach by Gogate and Domingos [2013] (see associated discussion in Section 3). Since SMP relies on ADDs (which cannot represent non-binary variables natively), we encode non-binary variables using *one-hot* encoding, following a standard practice. We do not show results comparing FABE against the standard version of BE with tabular functions [Dechter, 2013], since the latter runs out of memory on most of the instances. For WCSPs we also compare against CPLEX and GUROBI, two off-the-shelf solvers for constrained optimization.

FABE's reported runtimes also include the compilation of automata. Since both FABE and RBFAOO require an ordering d (see Section 2), we consider this as a pre-processing phase and we do not include its runtime in the results (also because it is negligible *wrt* the solution phase). For each instance, we compute d using a weighted MIN-FILL heuristic [Dechter, 2013], and we use the same d for both algorithms.

We execute RBFAOO with the parameters detailed in authors' previous work [Kishimoto and Marinescu, 2014; Kishimoto *et al.*, 2015], including cache size and i parameter. Every other approach is executed with default parameter values. Following [Kishimoto *et al.*, 2015], we set a time limit of 2 hours. We exclude from our analysis all instances that could not be solved by any algorithm in the considered time limit. FABE and SMP are implemented in C++.[5] We employ the implementations of RBFAOO and `toulbar` provided by the authors. All experiments have been run on a cluster whose computing nodes have 2.50GHz CPUs and 384 GBytes of RAM. As for Remark 1, for FABE we consider $\epsilon = 10^{-10}$.

Given the large number of instances in MPE datasets, in Table 1 we report the runtimes on the 12 largest instances *wrt* the number of variables. In Table 2 we report the aggregated results of the speed-ups achieved by FABE *wrt* other approaches, calculated considering the instances where both algorithms terminate within the time limit.

5.1 Summary of Results

Results confirm that FABE's performance depends on the degree of redundancy. FABE obtains good performance on the `protein` and `pedigree` datasets, achieving speed-ups of $\sim$1–2 orders of magnitude, and solving a total of 34 instances that RBFAOO could not solve. We also observe that `toulbar` is superior on a handful of `protein` instances and on the `grid` dataset, which is characterized by low redundancy. Results also show that, despite not employing parallelism, FABE's speed-up on the `protein` dataset is much higher than the one reported for SPRBFAOO, while it is comparable on the `pedigree` datasets.

As for WCSPs (detailed results in Table 3), FABE outperforms all competitors on the `spot5` dataset, notably achieving a speed-up of 5 orders of magnitude *wrt* `toulbar`. On the `mastermind` dataset, FABE is comparable with `toulbar` (since both compute solutions in tenths of seconds) except for `3·8·5` and `10·8·3` instances, but it is, on average, better than every other competitor. We also observe that `toulbar` is clearly superior on the `iscas89` dataset.

Finally, FABE consistently outperforms SMP using one-hot encoding, confirming that the use of additional encodings (required by the presence of non-binary variables that cannot

[4] Online at: www.ics.uci.edu/~dechter/softwares/benchmarks.

[5] Code available at https://github.com/filippobistaffa/FABE.

protein	1duw	1hcz	1fny	2hft	1ad2	1atg	1qre	1qhv	1pbv	1g3p	2fcb	1euo
FABE	**21.36**	**10.33**	**6.60**	322.33	**25.28**	3.28	**3.47**	**16.54**	234.70	**1.21**	**3.80**	**39.81**
RBFAOO	> 2 h	749.39	> 2 h	1765.22	1654.75	1697.87	734.85	> 2 h	1543.11	5080.64	1677.44	> 2 h
SMP	> 2 h	> 2 h	2036.29	6569.95	> 2 h	4098.89	1721.50	4376.94	> 2 h	580.33	6780.26	1374.51
toulbar	25.89	13.12	8.09	**0.02**	25.75	**0.04**	6.87	23.08	**0.05**	2.05	3.97	57.08
pedigree	25	30	39	18	31	34	51	9	13	7	41	37
FABE	28.82	**7.23**	**3.21**	**7.42**	**910.46**	**8.83**	**132.92**	473.94	**519.08**	**21.79**	**24.06**	**6.73**
RBFAOO	**6.32**	61.34	22.46	20.11	> 2 h	> 2 h	> 2 h	**100.19**	> 2 h	323.77	> 2 h	146.23
SMP	197.89	40.86	17.06	40.89	5881.66	60.78	789.65	2040.32	2011.26	136.11	151.40	27.91
toulbar	5962.05	581.43	17.43	166.29	1505.02	9.59	> 2 h	> 2 h	623.89	24.23	727.16	7.89
grid	90·26·5	90·25·5	90·24·5	75·23·5	90·23·5	75·22·5	90·22·5	75·21·5	90·21·5	50·20·5	75·20·5	90·20·5
FABE	3192.15	> 2 h	5112.09	> 2 h	508.75	> 2 h	4883.60	> 2 h	142.23	> 2 h	831.80	267.50
RBFAOO	925.47	902.19	1758.19	791.70	158.17	816.87	20.37	4.71	8.22	**163.44**	19.01	10.27
SMP	> 2 h	> 2 h	> 2 h	> 2 h	2453.45	> 2 h	> 2 h	> 2 h	802.17	> 2 h	3711.20	946.07
toulbar	**0.04**	**0.07**	**0.02**	**1.68**	**0.03**	**1.44**	**0.02**	**0.20**	**0.02**	1030.17	**0.10**	**0.01**

Table 1: Runtime results (in seconds) on 12 largest MPE instances.

Dataset	protein	pedigree	grid
Average Redundancy	96%	85%	64%
FABE speed-up *wrt* RBFAOO	58.6 (1%, 11%)	5.5 (0%, 32%)	0.1 (43%, 0%)
FABE speed-up *wrt* SMP	1006.5 (1%, 38%)	6.8 (0%, 5%)	4.0 (43%, 70%)
SPRBFAOO speed-up *wrt* RBFAOO	∼7	∼7	∼5
FABE speed-up *wrt* toulbar	1.0 (1%, 0%)	20.6 (0%, 23%)	0.0 (43%, 0%)
Dataset	spot5	mastermind	iscas89
Average Redundancy	85%	85%	87%
FABE speed-up *wrt* RBFAOO	36.9 (0%, 0%)	6.2 (0%, 0%)	0.4 (0%, 0%)
FABE speed-up *wrt* SMP	5.8 (0%, 0%)	2.5 (0%, 0%)	5.1 (0%, 0%)
FABE speed-up *wrt* toulbar	10 615.5 (0%, 50%)	0.4 (0%, 0%)	0.1 (0%, 0%)
FABE speed-up *wrt* CPLEX	2.0 (0%, 0%)	7.2 (0%, 0%)	0.8 (0%, 0%)
FABE speed-up *wrt* GUROBI	1.2 (0%, 0%)	4.4 (0%, 0%)	0.8 (0%, 0%)

Table 2: Average speed-up results for MPE (top) and WCSP (bottom) instances. For SPRBFAOO we report the same speed-up values reported by the authors [Kishimoto *et al.*, 2015]. Values in parentheses indicate the percentages of instances unsolved by first and second approach.

be represented by ADDs) introduces a significant overhead compared to our representation using DAFSA, which can natively represent non-binary variables. Such an impact is more pronounced on datasets with larger variable domains. Indeed, FABE obtains a speed-up of 3 orders of magnitude on the protein dataset, where variables reach a domain of 81.

6 Conclusions

We proposed FABE, an algorithm for exact MPE and constrained optimization that exploits our function representation based on DAFSA. Results obtained following an established methodology confirm the efficacy of our representation.

Future research directions include extending FABE to marginal inference tasks and integrating FABE to compute the initial heuristic for AND/OR search algorithms, which, at the moment, use the table-based implementation of BE. The computation of such an heuristic represents a bottleneck for high values of i, as acknowledged in [Kishimoto *et al.*, 2015]. A faster version of MBE could represent an important contribution, allowing one to employ more precise heuristics and, thus, improving the performance of the AND/OR search.

spot5	42b	505b	408b	29	503	54
FABE	**0.26**	**0.26**	**0.29**	**0.09**	**0.05**	**0.07**
RBFAOO	13.37	10.27	9.97	5.61	1.37	1.36
SMP	1.62	1.80	1.60	0.72	0.27	0.30
CPLEX	0.35	0.34	0.39	0.37	0.10	0.18
GUROBI	0.38	0.40	0.41	0.13	0.25	0.21
toulbar	> 2 h	> 2 h	> 2 h	0.10	1957.80	0.09
master.	3·8·5	10·8·3	4·8·4	3·8·4	4·8·3	3·8·3
FABE	247.27	69.30	0.36	0.22	0.10	**0.06**
RBFAOO	4.93	3.01	2.96	1.96	0.85	0.68
SMP	659.42	185.45	0.95	0.43	0.29	0.11
CPLEX	3.26	0.66	1.26	0.67	1.54	1.28
GUROBI	1.77	0.66	0.72	0.51	0.97	0.75
toulbar	**0.18**	**0.09**	**0.09**	**0.12**	**0.05**	**0.06**
iscas89	s1238	c880	s1196	s953	s1494	s1488
FABE	38.50	25.36	73.54	286.43	1.42	1.12
RBFAOO	1.47	1.17	0.61	0.54	0.41	0.39
SMP	229.64	146.43	410.96	1464.98	9.78	6.02
CPLEX	0.19	0.20	0.09	0.19	0.28	0.23
GUROBI	0.17	0.14	0.09	0.15	0.26	0.17
toulbar	**0.04**	**0.06**	**0.04**	**0.04**	**0.04**	**0.07**

Table 3: Runtime results (in seconds) on WCSP instances.

Ethical Statement

There are no ethical issues.

Acknowledgments

The author was supported by the "YOMA Operational Research" project funded by the Botnar Foundation. The author gratefully acknowledges the computer resources at Artemisa, funded by the European Union ERDF and Comunitat Valenciana as well as the technical support provided by the Instituto de Fisica Corpuscular, IFIC (CSIC-UV).

References

[Bahar *et al.*, 1997] Iris Bahar, Erica Frohm, Charles Gaona, Gary Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebric decision diagrams and their applications. *Formal Methods in System Design*, 10(2):171–206, 1997.

[Bensana *et al.*, 1999] Eric Bensana, Michel Lemaitre, and Gerard Verfaillie. Earth observation satellite management. *Constraints*, 4(3):293–299, 1999.

[Bistaffa and Farinelli, 2018] Filippo Bistaffa and Alessandro Farinelli. A COP model for graph-constrained coalition formation. *Journal of Artificial Intelligence Research*, 62:133–153, 2018.

[Bistaffa *et al.*, 2017] Filippo Bistaffa, Nicola Bombieri, and Alessandro Farinelli. An efficient approach for accelerating bucket elimination on GPUs. *IEEE Transactions on Cybernetics*, 47(11):3967–3979, 2017.

[Bubenzer, 2011] Johannes Bubenzer. Minimization of acyclic DFAs. In *Prague Stringology Conference*, pages 132–146, 2011.

[Chavira and Darwiche, 2007] Mark Chavira and Adnan Darwiche. Compiling Bayesian networks using variable elimination. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2443–2449, 2007.

[Daciuk, 2002] Jan Daciuk. Comparison of construction algorithms for minimal, acyclic, deterministic, finite-state automata from sets of strings. In *International Conference on Implementation and Application of Automata*, pages 255–261, 2002.

[Dechter, 1996] Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211–219, 1996.

[Dechter, 1997] Rina Dechter. Mini-buckets: A general scheme for generating approximations in automated reasoning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1297–1303, 1997.

[Dechter, 2013] Rina Dechter. *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013.

[Demeulenaere *et al.*, 2016] Jordan Demeulenaere, Renaud Hartert, Christophe Lecoutre, Guillaume Perez, Laurent Perron, Jean-Charles Régin, and Pierre Schaus. Compact-table: efficiently filtering table constraints with reversible sparse bit-sets. In *International Conference on Principles and Practice of Constraint Programming (CP)*, pages 207–223. Springer, 2016.

[Fishelson and Geiger, 2002] Maáyan Fishelson and Dan Geiger. Exact genetic linkage computations for general pedigrees. *Bioinformatics*, 18(suppl_1):S189–S198, 2002.

[Gogate and Domingos, 2013] Vibhav Gogate and Pedro Domingos. Structured message passing. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, page 252–261, 2013.

[Han and Salomaa, 2008] Yo-Sub Han and Kai Salomaa. State complexity of union and intersection of finite languages. *International Journal of Foundations of Computer Science*, 19(03):581–595, 2008.

[Higham, 2002] Nicholas Higham. *Accuracy and Stability of Numerical Algorithms*. Siam, 2002.

[Hopcroft and Ullman, 1979] John Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[Hurley *et al.*, 2016] Barry Hurley, Barry O'Sullivan, David Allouche, George Katsirelos, Thomas Schiex, Matthias Zytnicki, and Simon De Givry. Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints*, 21(3):413–434, 2016.

[Kersting, 2012] Kristian Kersting. Lifted probabilistic inference. In *European Conference on Artificial Intelligence (ECAI)*, pages 33–38, 2012.

[Kishimoto and Marinescu, 2014] Akihiro Kishimoto and Radu Marinescu. Recursive best-first AND/OR search for optimization in graphical models. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 400–409, 2014.

[Kishimoto *et al.*, 2015] Akihiro Kishimoto, Radu Marinescu, and Adi Botea. Parallel recursive best-first AND/OR search for exact MAP inference in graphical models. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 928–936, 2015.

[Lauritzen, 1996] Steffen Lauritzen. *Graphical models*. Clarendon Press, 1996.

[Marinescu and Dechter, 2007] Radu Marinescu and Rina Dechter. Best-first AND/OR search for graphical models. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 1171–1176, 2007.

[Marinescu and Dechter, 2009] Radu Marinescu and Rina Dechter. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491, 2009.

[Mateescu *et al.*, 2008] Robert Mateescu, Rina Dechter, and Radu Marinescu. AND/OR multi-valued decision diagrams (AOMDDs) for graphical models. *Journal of Artificial Intelligence Research*, 33:465–519, 2008.

[Pearl, 1989] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1989.

[Sanner and McAllester, 2005] Scott Sanner and David McAllester. Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1384–1390, 2005.

[Srinivasan *et al.*, 1990] Arvind Srinivasan, Timothy Ham, Sharad Malik, and Robert Brayton. Algorithms for discrete function manipulation. In *International Conference on Computer-Aided Design (ICCAD)*, pages 92–93, 1990.

[Yanover *et al.*, 2008] Chen Yanover, Ora Schueler-Furman, and Yair Weiss. Minimizing and learning energy functions for side-chain prediction. *Journal of Computational Biology*, 15(7):899–911, 2008.