Approximation Complexity of Maximum A Posteriori Inference in Sum-Product Networks

Diarmaid Conaty
Queen's University Belfast
Belfast, United Kingdom

Denis D. Mauá
Universidade de São Paulo
São Paulo, Brazil

Cassio P. de Campos
Queen's University Belfast
Belfast, United Kingdom

Abstract

We discuss the computational complexity of approximating maximum a posteriori inference in sum-product networks. We first show NP-hardness in trees of height two by a reduction from maximum independent set; this implies non-approximability within a sublinear factor. We show that this is a tight bound, as we can find an approximation within a linear factor in networks of height two. We then show that, in trees of height three, it is NP-hard to approximate the problem within a factor $2^{f(n)}$ for any sublinear function f of the size of the input n. Again, this bound is tight, as we prove that the usual max-product algorithm finds (in any network) approximations within factor $2^{c \cdot n}$ for some constant $c < 1$. Last, we present a simple algorithm, and show that it provably produces solutions at least as good as, and potentially much better than, the max-product algorithm. We empirically analyze the proposed algorithm against max-product using synthetic and real-world data.

1 INTRODUCTION

Finding the mode of a probability distribution is a key step of many solutions to problems in artificial intelligence such as image segmentation (Geman and Geman, 1984), 3D image reconstruction (Boykov et al., 1998), natural language processing (Koo et al., 2010), speech recognition (Peharz et al., 2014), sentiment analysis (Zirn et al., 2011), protein design (Szeliski et al., 2008) and multicomponent fault diagnosis (Steinder and Sethi, 2004), to name but a few. This problem is often called (full) maximum a posteriori (MAP) inference, or most likely explanation (MPE).

Sum-Product Networks (SPNs) are a relatively new class of graphical models that allow marginal inference in linear time in their size (Poon and Domingos, 2011). This is therefore in sharp difference with other graphical models such as Bayesian networks and Markov Random Fields that require #P-hard effort to produce marginal inferences (Darwiche, 2009). Intuitively, an SPN encodes an arithmetic circuit whose evaluation produces a marginal inference (Darwiche, 2003). SPNs have received increasing popularity in applications of machine learning due to their ability to represent complex and highly multidimensional distributions (Poon and Domingos, 2011; Amer, 2012; Peharz et al., 2014; Cheng et al., 2014; Nath and Domingos, 2016; Amer and Todorovic, 2016).

In his PhD thesis, Peharz showed a direct proof of NP-hardness of MAP in SPNs by a reduction from maximum satisfiability; his proof however is not correct as it encodes clauses as products (Peharz, 2015, Theorem 5.3).[1] Later, Peharz et al. (2016) noted that NP-hardness can be proved by transforming a Bayesian network with a naive Bayes structure into a distribution-equivalent SPN of height two (this is done by adding a sum node to represent the latent root variable and its marginal distribution, and product nodes as children to represent the conditional distributions). As MAP inference in the former is NP-hard (de Campos, 2011), the result follows.

In this paper, we show a direct proof of NP-hardness of MAP inference by a reduction from maximum independent set, the problem of deciding whether there is a subset of vertices of a certain size in an undirected graph such that no two vertices in the set are connected. This new proof is quite simple, and (as with the reduction from naive Bayesian networks) uses a sum-product network of height two. An advantage of the new proof is that, as a corollary, we obtain the non-approximability of MAP inference within a sublinear factor in networks of height two. This is a tight bound, as we show that there

[1] The proof has been recently rectified in an Erratum note.

HEIGHT	LOWER BOUND	UPPER BOUND
1	1	1
2	$(m-1)^{\varepsilon}$	$m-1$
≥ 3	$2^{s^{\varepsilon}}$	2^{s}

Table 1: Lower and upper bounds on the approximation threshold for a polynomial-time algorithm: s denotes the size of the instance, m is the number of internal nodes, ε is a nonnegative number less than 1.

is a polynomial-time algorithm that produces approximations within a linear factor in networks of height two. For networks of height three or more we prove that it is NP-hard to approximate the problem within any factor $2^{f(n)}$ for any sublinear function f of the input size n, even if the SPN is a tree. This bound is tight, as we show that the usual max-product algorithm by Poon and Domingos (2011), which replaces sums with maximizations, finds an approximation within a factor $2^{c \cdot n}$ for some constant $c < 1$. Table 1 summarizes these results. As far as we are concerned, these are the first results about the complexity of approximating MAP in SPNs.

We also show that a simple modification to the max-product algorithm leads to an algorithm that produces solutions which are never worse and potentially significantly better than the solutions produced by max-product. We compare the performance of the proposed algorithm against max-product in several structured prediction tasks using both synthetic networks and SPNs learned from real-world data. The synthetic networks encode instances of maximum independent set problems. The purpose of these networks is to evaluate the quality of solutions produced by both algorithms on shallow SPNs which (possibly) encode hard to approximate MAP problems. Deeper networks are learned from UCI datasets using the LEARNSPN algorithm by Gens and Domingos (2013). The purpose of these experiments is to assess the relative quality of the algorithms on SPNs from realistic datasets, and their sensitivity to evidence. The empirical results show that the proposed algorithm often finds significantly better solutions than max-product does, but that this improvement is less pronounced in networks learned from real data. We expect these results to foster research in new approximation algorithms for MAP in SPNs.

Before presenting the complexity results in Section 3, we first review the definition of sum-product networks, and comment on a few selected results from the literature in Section 2. The experiments with the proposed modified algorithm and max-product appear in Section 4. We conclude the paper with a review of the main results in Section 5.

2 SUM-PRODUCT NETWORKS

We use capital letters without subscripts to denote random vectors (e.g. X), and capital letters with subscripts to denote random variables (e.g., X_1). If X is a random vector, we call the set $\mathcal{X}$ composed of the random variables X_i in X its *scope*. The scope of a function of a random vector is the scope of the respective random vector. In this work, we constrain our discussion to random variables with finite domains.

Poon and Domingos (2011) originally defined SPNs as multilinear functions of indicator variables that allow for space and time efficient representation and marginal inference. In its original definition SPNs were not constrained to represent valid distributions; this was achieved by imposing properties of consistency and completeness. This definition more closely resembles Darwiche's arithmetic circuits which represent the *network polynomial* of a Bayesian network (Darwiche, 2003), and also allow inference in the size of the circuit.

Later, Gens and Domingos (2013) re-stated SPNs as complex mixture distributions as follows.

- Any univariate distribution is an SPN.
- Any weighted sum of SPNs with the same scope and nonnegative weights is an SPN.
- Any product of SPNs with disjoint scopes is an SPN.

This alternative definition (called generalized SPNs by Peharz (2015)) implies *decomposability*, a stricter requirement than consistency. Peharz et al. (2015) showed that any consistent SPN over discrete random variables can be transformed in an equivalent decomposable SPN with a polynomial increase in size, and that weighted sums can be restricted to the probability simplex without loss of expressivity. Hence, we assume in the following that SPNs are *normalized*: the weights of a weighted sum of SPNs add up to one. This implies that SPNs specify (normalized) distributions. A similar result was obtained by Zhao et al. (2015). We note that the base of the inductive definition can also be extended to accommodate any class of tractable distributions (e.g., Chow-Liu trees) (Rooshenas and Lowd, 2014; Vergari et al., 2015). For the purposes of this work, however, it suffices to consider only univariate distributions.

An SPN is usually represented graphically as a weighted rooted graph where each internal node is associated with an operation $+$ or $\times$, and leaves are associated with variables and distributions. The arcs from a sum node to its children are weighted according to the corresponding convex combinations. The remaining arcs have implicitly weight 1. The height of an SPN is defined as the

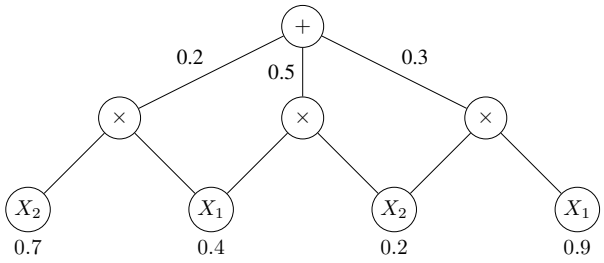

Figure 1: A sum-product network over binary variables X_1 and X_2. Only the probabilities $\mathbb{P}(X_i = 1)$ are shown.

maximum distance, counted as the number of arcs, from the root to a leaf of its graphical representation. Figure 1 shows an example of an SPN with scope $\{X_1, X_2\}$ and height two. Unit weights are omitted in the figure. Note that by definition every node represents an SPN (hence a distribution) on its own; we refer to nodes and their corresponding SPNs interchangeably.

Consider an SPN $S(X)$ over a random vector $X = (X_1, \ldots, X_n)$. The value of S at a point $x = (x_1, \ldots, x_n)$ in its domain is denoted by $S(x)$ and defined recursively as follows. The value of a leaf node is the value of its corresponding distribution at the point obtained by projecting x onto the scope of the node. The value of a product node is the product of the values of its children at x. Finally, the value of a sum node is the weighted average of its children's values at x. For example, the value of the SPN $S(X_1, X_2)$ in Figure 1 at the point $(1, 0)$ is $S(1, 0) = 0.2 \cdot 0.3 \cdot 0.4 + 0.5 \cdot 0.4 \cdot 0.8 + 0.3 \cdot 0.8 \cdot 0.9 = 0.4$. Note that since we assumed SPNs to be normalized, we have that $\sum_x S(x) = 1$.

Let $\mathcal{E} \subseteq \{1, \ldots, n\}$ and consider a random vector $X_\mathcal{E}$ with scope $\{X_i : i \in \mathcal{E}\}$, and an assignment $e = \{X_i = e_i : i \in \mathcal{E}\}$. We write $x \sim e$ to denote a value of X consistent with e (i.e., the projection of x on $\mathcal{E}$ is e). Given an SPN $S(X)$ representing a distribution $\mathbb{P}(X)$, we denote the marginal probability $\mathbb{P}(e) = \sum_{x \sim e} S(x)$ by $S(e)$. This value can be computed by first marginalizing the variables $\{X_j : j \notin \mathcal{E}\}$ from every (distribution in a) leaf and then propagating values as before. Thus marginal probabilities can be computed in time linear in the network size (assuming that univariate distributions are represented as tables). The marginal probability $\mathbb{P}(X_2 = 0) = 0.7$ induced by the SPN in Figure 1 can be obtained by first marginalizing leaves without $\{X_2\}$ (thus producing the value 1 at the respective leaves), and then propagating values as before.

In this work, we are interested in the following computational problem with SPNs:

Definition 1 (Functional MAP inference problem)**.** *Given an* SPN S *specified with rational weights and an assignment* e*, find* x^* *such that* $S(x^*) = \max_{x \sim e} S(x)$.

A more general version of the problem would be to allow some of the variables to be summed out, while others are maximized. However, the marginalization (i.e., summing out) of variables can performed in polynomial time as a preprocessing step, the result of which is a MAP problem as stated above. We stick with the above definition for simplicity (bearing in mind that complexity is not changed).

To prove NP-completeness, we use the decision variant of the problem:

Definition 2 (Decision MAP inference problem)**.** *Given an* SPN S *specified with rational weights, an assignment* e *and a rational* γ*, decide whether* $\max_{x \sim e} S(x) \geq \gamma$.

We denote both problems by MAP, as the distinction to which particular (functional or decision) version we refer should be clear from context. Clearly, NP-completeness of the decision version establishes NP-hardness of the functional version. Also, approximation complexity always refers to the functional version.

The support of an SPN S is the set of configurations of its domain with positive values: $\text{supp}(S) = \{x : S(x) > 0\}$. An SPN is *selective* if for every sub-SPN T corresponding to a sum node in S it follows that the supports of any two children are disjoint. Peharz et al. (2016) recently showed that MAP is tractable in selective SPNs.

Here, we discuss the complexity of (approximately) solving MAP in general SPNs. We assume that instances of the MAP problem are represented as bitstrings $\langle S, e \rangle$ using a reasonable encoding; for instance, weights and probabilities are rational values represented by two integers in binary notation, and graphs are represented by (a binary encoding of their) adjacency lists.

3 COMPLEXITY RESULTS

As we show in this section, there is a strong connection between the height of an SPN and the complexity of MAP inferences. First, note that an SPN of height 0 is just a univariate distribution (where MAP is trivial). So consider an SPN of height 1. If the root is a sum node, then the network encodes a sum of univariate distributions over the same variable, and MAP can be solved trivially by enumerating all values of that variable. If on the other hand the root is a product node, then the network encodes a distribution of fully independent variables. Also in this case, we can solve MAP easily by optimizing independently for each variable. So MAP in networks of height 1 or less is solvable in polynomial time.

Let us now consider SPNs of height 2. As already discussed in the introduction, Peharz et al. (2016) briefly observed that the MAP problem is NP-hard even for tree-shaped networks of height 2. Here, we give the following alternative, direct proof of NP-hardness of MAP in SPNs, that allows us to obtain results on non-approximability.

Theorem 1. MAP *in sum-product networks is* NP*-complete even if there is no evidence, and the underlying graph is a tree of height 2.*

Proof. Membership is straightforward as we can evaluate the probability of a configuration in polynomial time.

We show hardness by reduction from the NP-hard problem maximum independent set (see e.g. (Zuckerman, 2007)): Given an undirected graph $G = (V, E)$ with vertices $\{1, \ldots, n\}$ and an integer v, decide whether there is an independent set of size v. An independent set is a subset $V' \subseteq V$ such that no two vertices are connected by an edge in E.

Let N_i denote the neighbors of i in V. For each $i \in V$, build a product node S_i whose children are leaf nodes $S_{i1}, \ldots, S_{in}$ with scopes $X_1, \ldots, X_n$, respectively. If $j \in N_i$ then associate S_{ij} with distribution $\mathbb{P}(X_i = 1) = 0$; if $j \notin N_i \cup \{i\}$ associate S_{ij} with $\mathbb{P}(X_j = 1) = 1/2$; finally, associate S_{ii} with distribution $\mathbb{P}(X_i = 1) = 1$. See Figure 2 for an example. Let $n_i = |N_i|$ be the number of neighbors of i. Then $S_i(x) = 1/2^{n-n_i-1}$ if $x_i = 1$ and $x_j = 0$ for all $j \in N_i$; and $S_i(x) = 0$ otherwise. That is, $S_i(x) > 0$ if there is a set V' which contains i and does not contain any of its neighbors. Now connect all product nodes S_i with a root sum node parent S; specify the weight from S to S_i as $w_i = 2^{n-n_i-1}/c$, where $c = \sum_i 2^{n-n_i-1}$. Suppose there is an independent set I of size v. Take x such that $x_i = 1$ if $i \in I$ and $x_i = 0$ otherwise. Then $S(x) = v/c$. For any configuration x of the variables, let $I(x) = \{i : S_i(x) > 0\}$. Then $I(x)$ is an independent set of size $c \cdot S(x)$. So suppose that there is no independent set of size v. Then $\max_x S(x) < v/c$. Thus, there is an independent set if and only if $\max_x S(x) \geq v/c$. □

Consider a real-valued function $f(\langle S, e\rangle)$ of the encoded network S and evidence e. An algorithm for MAP in SPNs is a $f(\langle S, e\rangle)$-approximation if it runs in time polynomial in the size of its input (which specifies the graph, the weights, the distributions, the evidence) and outputs a configuration $\tilde{x}$ such that $S(\tilde{x}) \cdot f(\langle S, e\rangle) \geq \max_{x \sim e} S(x)$. That is, a $f(\langle S, e\rangle)$-approximation algorithm provides, for every instance $\langle S, e\rangle$ of the MAP problem, a solution whose value is at most a factor $f(\langle S, e\rangle)$ from the optimum value. The value $f(\langle S, e\rangle)$ is called the *approximation factor*. We have the following consequence of Theorem 1:

Corollary 1. *Unless* P *equals* NP*, there is no* $(m-1)^\varepsilon$*-approximation algorithm for* MAP *in* SPN*s for any* $0 \leq \varepsilon < 1$*, where* m *is the number of internal nodes of the* SPN*, even if there is no evidence and the underlying graph is a tree of height 2.*

Proof. The proof of Theorem 1 encodes a maximum independent set problem and the reduction is a weighted reduction (see Definition 1 in (Bulatov et al., 2012)), which suffices to prove the result. To dispense with weighted reductions, we now give a direct proof. So suppose that there is a $(m-1)^\varepsilon$-approximation algorithm for MAP with $0 \leq \varepsilon < 1$. Let $\tilde{x}$ be the configuration returned by this algorithm when applied to the SPN S created in the proof of Theorem 1 for a graph G given as input of the maximum independent set problem. We have that

$$S(\tilde{x}) \cdot c \cdot (m-1)^\varepsilon \geq c \cdot \max_x S(x) = \max_{I \in \mathcal{I}(G)} |I|\,,$$

where $c = \sum_i 2^{n-n_i-1}$, $\mathcal{I}(G)$ is the collection of independent sets of G, n is the number of vertices in G and n_i is the number of neighbors of vertex i in G. Consequently, this algorithm is a n^ε-approximation for maximum independent set (note that $n = m - 1$ by construction). We know that there is no n^ε-approximation for maximum independent set with $0 \leq \varepsilon < 1$ unless P equals NP (Zuckerman, 2007), so the result follows. □

Corollary 1 shows that there is *probably* no approximation algorithm for MAP with sublinear approximation factor in the size of the input. The following result shows that this lower bound is tight:

Theorem 2. *There exists a* $(m-1)$*-approximation algorithm for* MAP *in sum-product networks whose underlying graph has height at most 2, where* m *is the number of internal nodes.*

Proof. Consider a sum-product network of height 2. If the root is a product node then the problem decomposes into independent MAP problems in SPNs of height 1; each of those problems can be solved exactly. So assume that the root S is a sum node connected either to leaf nodes or to nodes which are connected to leaf nodes. Solve the respective MAP problem for each child S_i independently (which is exact, as the corresponding SPN has height at most 1); denote by x^i the corresponding solution. Note that $S_i(x^i)$ is an upper bound on the value $S_i(x^*)$, where x^* is a (global) MAP configuration. Let $w_1, \ldots, w_{m-1}$ denote the weights from the root to children $S_1, \ldots, S_{m-1}$. Return $\tilde{x} = \arg\max_i w_i \cdot S_i(x^i)$. It follows that $(m-1)S(\tilde{x}) \geq \max_{x \sim e} S(x)$. Note that this is the same value returned by the max-product algorithm by Poon and Domingos (2011). □

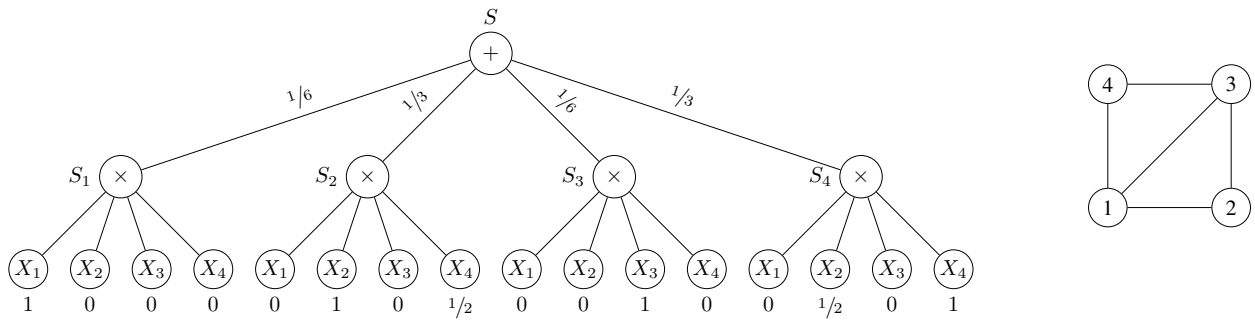

Figure 2: A sum-product network encoding the maximum independent set problem for the graph on the right. Only the values for $\mathbb{P}(X_i = 1)$ are shown.

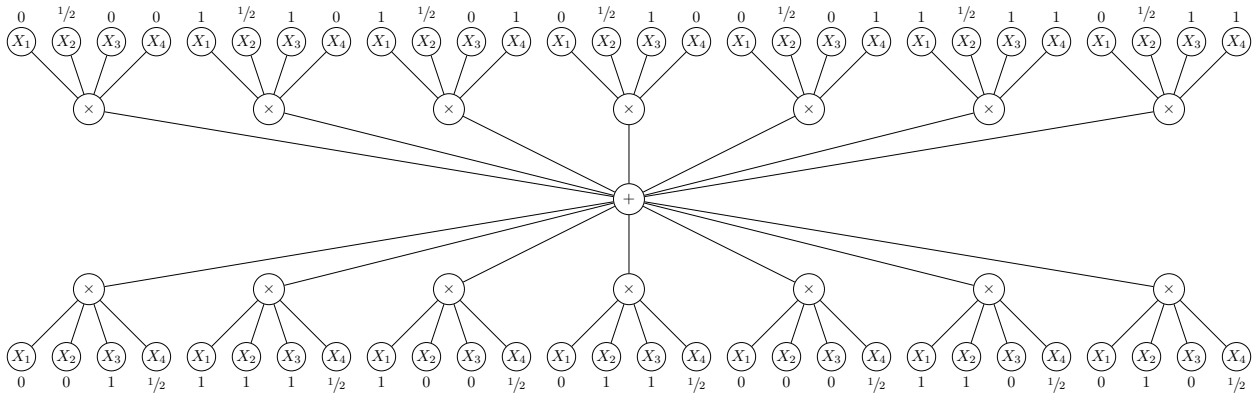

Figure 3: A sum-product network encoding the Boolean formula $(\neg X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_3 \vee X_4)$. We represent only the probabilities $\mathbb{P}(X_i = 1)$, and omit the uniform weights $1/14$ associated with the root sum node.

Thus, for networks of height 2, we have a clear divide: there is an approximation algorithm with linear approximation factor in the number of internal nodes, and no approximation algorithm with sublinear approximation factor in the number of internal nodes. Allowing an additional level of nodes reduces drastically the quality of the approximations in the worst case:

Theorem 3. *Unless* P *equals* NP*, there is no* 2^{s^ε}*-approximation algorithm for* MAP *in* SPN*s for any* $0 \leq \varepsilon < 1$*, where* s *is the size of the input, even if there is no evidence and the underlying graph is a tree of height 3.*

Proof. First, we show how to build an SPN for deciding satisfiability: Given a Boolean formula ϕ in conjunctive normal form, decide if there is a satisfying truth-value assignment. We assume that each clause contains exactly 3 distinct variables (NP-completeness is not altered by this assumption, but if one would like to drop it, then the weights of the sum node we define below could be easily adjusted to account for clauses with less than 3 variables).

Let $X_1, \ldots, X_n$ denote the Boolean variables and $\phi_1, \ldots, \phi_m$ denote the clauses in the formula. For $i = 1, \ldots, m$, consider the conjunctions $\phi_{i1}, \ldots, \phi_{i7}$ over the variables of clause ϕ_i, representing all the satisfying assignments of that clause. For each such assignment, introduce a product node S_{ij} encoding the respective assignment: there is a leaf node with scope X_k whose distribution assigns all mass to value 1 (resp., 0) if and only if X_k appears nonnegated (resp., negated) in ϕ_{ij}; and there is a leaf node with uniform distribution over X_k if and only if X_k does not appear on ϕ_{ij}. See Figure 3 for an example. For a fixed configuration of the random variables, the clause ϕ_i is true if and only if one of the product nodes $S_{i1}, \ldots, S_{i7}$ evaluates to $1/2^{n-3}$. And since these products encode disjoint assignments, at most one such product is nonzero for each configuration. We thus have that $\sum_{ij} S_{ij}(x) = m/2^{n-3}$ if $\phi(x)$ is true, and $\sum_{ij} S_{ij}(x) < m/2^{n-3}$ if $\phi(x)$ is false. So introduce a sum node S with all product nodes as children and with uniform weights $1/7m$. There is a satisfying assignment for ϕ if and only if $\max_x S(x) \geq 2^{3-n}/7$.

We "amplify" the approximation error by multiplying independent copies of the SPN built so far. So take

the SPN above and make q copies of it with disjoint scopes: each copy contains different random variables X_k^t, $t = 1, \ldots, q$, at the leaves, but otherwise represents the very same distribution/satisfiability problem. Name each copy S_t, $t = 1, \ldots q$, and let its size be s_t. Denote by $s' = \max_t s_t$ (note that, since they are copies, their size is the same, apart from possible indexing, etc). Connect these copies using a product node S with networks $S_1, \ldots, S_q$ as children, so that $S(x) = \prod_{t=1}^{q} S_t(x)$. Note that $\max_x S_t(x) \geq 2^{3-n}/7$ if there is a satisfying assignment to the Boolean formula, and $\max_x S_t(x) \leq \frac{m-1}{m} \cdot 2^{3-n}/7$ if there is no satisfying assignment. Hence, $\max_x S(x) \geq (2^{3-n}/7)^q$ if there is a satisfying assignment and $\max_x S(x) \leq ((m-1)2^{3-n}/(7m))^q$ if there is no satisfying assignment. Specify

$$q = 1 + \left\lfloor (\ln(2) \cdot m \cdot (s'+2)^{\varepsilon})^{\frac{1}{1-\varepsilon}} \right\rfloor ,$$

which is polynomial in s', so the SPN S can be constructed in polynomial time and space and has size $s < q(s'+2)$. From the definition of q, we have that

$$q > (\ln(2) \cdot m \cdot (s'+2)^{\varepsilon})^{\frac{1}{1-\varepsilon}} .$$

Raising both sides to $1-\varepsilon$ yields

$$q > q^{\varepsilon} \ln(2) \cdot m \cdot (s+2)^{\varepsilon} = m \ln(2^{(q(s'+2))^{\varepsilon}}) > m \ln 2^{s^{\varepsilon}} .$$

Since $\frac{1}{m} \leq \ln \frac{m}{m-1}$ for any integer $m > 1$, it follows that

$$q \ln \left(\frac{m}{m-1} \right) > \ln 2^{s^{\varepsilon}} .$$

By exponentiating both sides, we arrive at

$$\left(\frac{m}{m-1} \right)^q > 2^{s^{\varepsilon}} \text{ hence } 2^{s^{\varepsilon}} \left(\frac{m-1}{m} \right)^q < 1 ,$$

Finally, by multiplying both sides by $(2^{3-n}/7)^q$, we obtain

$$2^{s^{\varepsilon}} \left(\frac{2^{3-n}(m-1)}{7m} \right)^q < \left(\frac{2^{3-n}}{7} \right)^q .$$

Hence, if we can obtain an $2^{s^{\varepsilon}}$-approximation for $\max_x S(x)$, then we can decide satisfiability: there is a satisfying assignment to the Boolean formula if and only if the approximation returns a value strictly greater than $(2^{3-n}(m-1)/(7m))^q$. □

According to Theorem 3, there is no $2^{f(s)}$-approximation (unless P equals NP) for any sublinear function f of the input size s. The following result is used to show that this lower bound is tight.

Theorem 4. *Let S^+ denote the sum nodes in* SPN *S, and d_i be the number of children of sum node $S_i \in S^+$. Then there exists a $(\prod_{S_i \in S^+} d_i)$-approximation algorithm for* MAP *with input S and evidence e.*

Proof. There are two cases to consider, based on the value of $S(e)$, which can be checked in polynomial time. If $S(e) = 0$, then we can return any assignment consistent with e, as the result will be exact (and equal to zero). If $S(e) > 0$, then take the max-product algorithm by Poon and Domingos (2011), which consists of an upward pass where sums are replaced by maximizations in the evaluation of an SPN, and a downward pass which selects the maximizers of the previous step. Define $\text{pd}(S, e)$ recursively as follows. If S is a leaf then $\text{pd}(S, e) = \max_{x \sim e} S(x)$. If S is a sum node, then $\text{pd}(S, e) = \max_{j=1,\ldots,t} w_j \cdot \text{pd}(S_j, e)$, where $S_1, \ldots, S_t$ are the children of S. Finally, if S is a product node with children $S_1, \ldots, S_t$, then $\text{pd}(S, e) = \prod_{j=1}^{t} \text{pd}(S_j, e)$. Note that $\text{pd}(S, e)$ corresponds to the upward pass of the max-product algorithm; hence it is a lower bound on the value of the configuration obtained by such algorithm. We prove that the max-product algorithm is a $(\prod_{S_i \in S^+} d_i)$-approximation by proving by induction in the height of the SPN that

$$\text{pd}(S, e) \geq \left(\prod_{S_i \in S^+} \frac{1}{d_i} \right) \max_{x \sim e} S(x) .$$

To show the base of the induction, take a network S of height 0 (i.e., containing a single node). Then $\text{pd}(S, e) = \max_{x \sim e} S(x)$ trivially. So take a network S with children $S_1, \ldots, S_t$, and suppose (by inductive hypothesis) that $\text{pd}(S_j, e) \geq (\prod_{S_i \in S_j^+} \frac{1}{d_i}) \max_{x \sim e} S_j(x)$ for every child S_j. If S is a product node, then

$$\begin{aligned}
\text{pd}(S, e) &= \prod_{j=1}^{t} \text{pd}(S_j, e) \\
&\geq \prod_{j=1}^{t} \left(\prod_{S_i \in S_j^+} \frac{1}{d_i} \right) \max_{x \sim e} S_j(x) \\
&= \left(\prod_{S_i \in S^+} \frac{1}{d_i} \right) \prod_{j=1}^{t} \max_{x \sim e} S_j(x) \\
&= \left(\prod_{S_i \in S^+} \frac{1}{d_i} \right) \max_{x \sim e} S(x) ,
\end{aligned}$$

where the last two equalities follow as the scopes of products are disjoints, which implies that the children do not share any node. If S is a sum node, then

$$\begin{aligned}
\text{pd}(S, e) &= \max_{j=1,\ldots,t} w_j \cdot \text{pd}(S_j, e) \\
&\geq \max_{j=1,\ldots,t} \left(\prod_{S_i \in S_j^+} \frac{1}{d_i} \right) w_j \cdot \max_{x \sim e} S_j(x)
\end{aligned}$$

$$
\begin{aligned}
&= \max_{j=1,\ldots,t} \left(\frac{t}{t \prod_{S_i \in S_j^+} d_i} \right) w_j \cdot \max_{x \sim e} S_j(x) \\
&\geq \max_{j=1,\ldots,t} \frac{t}{\prod_{S_i \in S^+} d_i} \cdot w_j \cdot \max_{x \sim e} S_j(x) \\
&= \frac{t \cdot \max_{j=1,\ldots,t} w_j \max_{x \sim e} S_j(x)}{\prod_{S_i \in S^+} d_i} \\
&\geq \left(\prod_{S_i \in S^+} \frac{1}{d_i} \right) \max_{x \sim e} S(x) \,.
\end{aligned}
$$

The first inequality uses the induction hypothesis. The second inequality follows since $1/(t \cdot \prod_{S_i \in S_j^+} d_i) \geq 1/(t \cdot \prod_{S_i \in S^+, S_i \neq S} d_i) = 1/\prod_{S_i \in S^+} d_i$. The last inequality follows as $\max_j w_j \cdot \max_{x \sim e} S_j(x)$ is an upper bound on the value of any child of S. This concludes the proof. □

We have the following immediate consequence, showing the tightness of Theorem 3.

Corollary 2. *There exists a $2^{\varepsilon \cdot s}$-approximation algorithm for* MAP *for some $0 < \varepsilon < 1$, where s is the size of the* SPN.

Proof. Assume the network has at least one sum node (otherwise we can find an exact solution in polynomial time). Given the result of Theorem 4, we only need to show that there is $\varepsilon < 1$ such that $\prod_{S_i \in S^+} d_i < 2^{\varepsilon \cdot s}$, with S^+ the sum nodes in SPN S and d_i be the number of children of sum node $S_i \in S^+$. Because s is strictly greater than the number of nodes and arcs in the network (as we must somehow encode the graph of S), we know that $s > \sum_{S_i \in S^+} d_i$. One can show that $3^{x/3} > x$ for any positive integer. Hence,

$$
\begin{aligned}
\prod_{S_i \in S^+} d_i &\leq \prod_{S_i \in S^+} 3^{d_i/3} = \prod_{S_i \in S^+} 2^{d_i \log_2(3)/3} \\
&= 2^{\log_2(3)/3 \cdot \sum_{S_i \in S^+} d_i} < 2^{s \log_2(3)/3} < 2^{\varepsilon \cdot s} \,,
\end{aligned}
$$

for some $\varepsilon < 0.5284$. □

The previous result shows that the max-product algorithm by Poon and Domingos (2011) achieves tight upper bounds on the approximation factor. This however does not rule out the existence of approximation algorithms that achieve the same (worst-case) upper bound but perform significantly better on average. For instance, consider the following algorithm that takes an SPN S and evidence e, and returns amap(S, e) as follows, where amap is short for *argmax-product* algorithm. If S is a sum node with children $S_1, \ldots, S_t$, then compute

$$
\text{amap}(S, e) = \arg \max_{x \in \{x^1, \ldots, x^t\}} \sum_{j=1}^{t} w_j \cdot S_j(x) \,,
$$

where $x^k = \text{amap}(S_k, e)$, that is, x^k is the solution of the MAP problem obtained by argmax-product for network S_k (argmax-product is run bottom-up). If S is a product node with children $S_1, \ldots, S_t$, then amap(S, e) is the concatenation of amap$(S_1, e), \ldots,$ amap(S_t, e). Else, S is a leaf, so return amap$(S, e) = \arg\max_{x \sim e} S(x)$.

The argmax-product has a worst-case time complexity quadratic in the size of the network; that is because the evaluation of all the children of a sum node with the argument which maximizes each of the children takes linear time (with a smart implementation, it might be possible to achieve subquadratic time). For comparison, the max-product (with a smart implementation to keep track of solutions and evaluations) takes linear time. While this is a drawback of the argmax-product algorithm, worst-case quadratic time is still quite efficient. More importantly, argmax-product always produces an approximation at least as good as that of max-product, and possibly exponentially better:

Theorem 5. *For any* SPN S *and evidence e, we have that $S(\text{amap}(S, e)) \geq S(\text{PD}(S, e))$, where PD$(S, e)$ is the configuration returned by the max-product algorithm. Moreover, there exists S and e such that $S(\text{amap}(S, e)) > 2^m S(\text{PD}(S, e))$, where m is the number of sum nodes in S.*

Proof. It is not difficult to see that $S(\text{amap}(S, e)) \geq S(\text{PD}(S, e))$, because the configuration that is selected by max-product at each sum node is one of the configurations that are tried by the maximization of argmax-product (and both algorithms perform the same operation on leaves and product nodes). To see that this improvement can be exponentially better, consider the SPN S_i in Figure 4. Let pd(S_i, e) be defined as in the proof of Theorem 4. One can verify that pd$(S_i, e) = 5/16$, while

$$
S_i(\text{amap}(S_i, e)) = 3 \cdot 11/48 = 11/16 > 2 \cdot 5/16 \,.
$$

Now, create an SPN S with a product root node connected to children $S_1, \ldots, S_m$ as described (note that the scope of S is $X_1, \ldots, X_m$). Then,

$$
\begin{aligned}
S(\text{amap}(S, e)) &= (11/16)^m \\
&> 2^m (5/16)^m = 2^m \cdot \text{pd}(S, e) \,.
\end{aligned}
$$

The result follows as (for this network) pd$(S, e) = S(\text{PD}(S, e))$. □

As an immediate result, the solutions produced by argmax-product achieve the upper bound on the complexity of approximating MAP. We hope that this simple result motivates researchers to seek for more sophisticated algorithms that exhibit the time performance of max-product while achieving the accuracy of argmax-product.

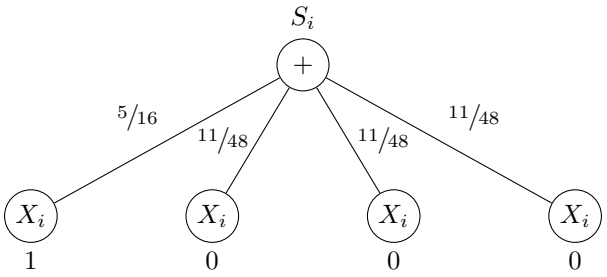

Figure 4: Fragment of the sum-product network used to prove Theorem 5.

4 EXPERIMENTS

We perform two sets of experiments to verify empirically the difference between argmax-product and max-product. We emphasize that our main motivation is to understand the complexity of the problem and how much it can be approximated efficiently in practice. In the light of Theorem 5, one may suggest that a MAP problem instance is easy to approximate when argmax-product and max-product produce similar approximations.

In the first set of evaluations, we build SPNs from random instances for the maximum independent set problem, that is, we generate random undirected graphs with number of vertices given in the first column of Table 2 and number of edges presented as percentage of the maximum number of edges (that is, the number of edges in a complete graph). For each graph, we apply the transformation presented in the proof of Theorem 1 to obtain an SPN. The number of nodes of such SPN is given in the third column of the table. The fourth column shows the ratio of the values of the configurations found by argmax-product and max-product (that is, $S(\text{amap}(S, e))/S(\text{PD}(S, e))$), averaged over 100 random repetitions. Take the first row: On average, the result of argmax-product is 1.58 times better than the result of max-product in SPNs encoding maximum independent set problems with graphs of 5 vertices and 10% of edges (which creates SPNs of 31 nodes: $5 \cdot 5 = 25$ leaves, 5 product nodes and one sum node, same structure as exemplified in Figure 2). The standard deviation for these ratios are also presented (last column in the table). It is clear that argmax-product obtains results that are significantly better than max-product, often surpassing the $2^m = 2$ ratio lower bound in Theorem 5. While in theory argmax-product can be significantly slower than max-product, we did not observe significant differences in running time for these SPNs with up to 6481 nodes (either algorithm terminated almost instantaneously).

In the second set of evaluations, we use realistic SPNs that model datasets from the UCI Repository.[2] The SPNs are learned using the GoSPN library,[3] a freely available implementation of the ideas presented in (Gens and Domingos, 2013). For each dataset, we use a random sample of 90% of the dataset for learning the model and save the remaining 10% to be used as test. Then we perform two types of queries. The first type of query consists in using the learned network to compute the mode of the corresponding distribution, that is, running both algorithms with no evidence. In the second type, we split the set of variables in half; for each instance (row) in the test set, we set the respective evidence for the variables in the first half, and compute the MAP configuration/value for the other half. This allows us to assess the effect of evidence in the approximation complexity. The results are summarized in Table 3. The first three columns display information about the dataset (name, number of variables and number of samples); the middle four columns display information about the learned SPN (total number of nodes, number of sum nodes, number of product nodes and height); the last three columns show the approximation results: the ratio of probabilities of solutions found by argmax-product and max-product

Vertices	% Edges	Nodes	Ratio	StDev
5	10	31	1.58	0.76
5	20	31	1.72	0.78
5	40	31	1.61	0.75
5	60	31	1.57	0.60
10	10	111	1.89	0.95
10	20	111	2.16	1.09
10	40	111	2.12	1.01
10	60	111	2.04	0.89
20	10	421	1.94	0.88
20	20	421	2.89	1.72
20	40	421	3.02	1.24
20	60	421	2.60	1.04
40	10	1641	2.64	1.42
40	20	1641	3.37	1.45
40	40	1641	2.33	0.73
40	60	1641	2.27	0.76
80	10	6481	3.96	1.81
80	20	6481	2.10	0.49
80	40	6481	1.07	0.26
80	60	6481	1.04	0.20

Table 2: Empirical comparison of the quality of approximations produced by argmax-product and max-product in SPNs encoding randomly generated instances of the maximum independent set problem.

[2] Obtained at `http://archive.ics.uci.edu/ml/`.

[3] `http://www.github.com/RenatoGeh/gospn`

Dataset	No. of variables	No. of samples	No. of nodes	No. of nodes $+$	No. of nodes $\times$	Height	No Evidence	50% Evidence	
							Ratio	Ratio	StDev
audiology	70	204	13513	28	27	12	1.0000	1.0029	0.0133
breast-cancer	10	258	1357	23	24	24	1.1572	1.1977	0.1923
car	7	1556	16	2	3	3	1.1028	1.0514	0.0514
cylinder-bands	33	487	3129	61	62	62	1.1154	1.1185	0.0220
flags	29	175	787	25	26	26	1.3568	1.3654	0.0363
ionosphere	34	316	603	43	44	42	1.1176	1.1109	0.0273
nursery	9	11665	20	2	3	3	1.6225	1.2060	0.2926
primary-tumor	18	306	804	113	114	114	1.0882	1.0828	0.0210
sonar	61	188	1057	101	102	26	1.2380	1.2314	0.0261
vowel	14	892	23	2	3	3	1.0751	1.0666	0.0229

Table 3: Empirical comparison of the quality of approximations produced by argmax-product and max-product in SPNs learned from UCI datasets with height at least 2.

in the test cases with no evidence, the same ratio in test cases with 50% of variables given as evidence, and the standard deviation for the latter (as it is run over different evidences corresponding to 10% of the data). We only show datasets where the learned SPN has at least one sum node, since in the other cases the MAP can be trivially found and a comparison would be pointless. The results suggest that in these SPNs learned from real data, the difference between argmax-product and max-product is less prominent, yet non negligible. We also see that the complexity of approximation is not considerably affected by the presence of evidence.

5 CONCLUSION

We analyzed the complexity of maximum a posteriori inference in sum-product networks and showed that it relates with the height of the underlying graph. We first provided an alternative (and more direct) proof of NP-hardness of maximum a posteriori inference in sum-product networks. Our proof uses a reduction from maximum independent set in undirected graphs, from which we obtain the non-approximability for any sublinear factor in the size of input, even in networks of height 2 and no evidence. We then showed that this limit is tight, that is, that there is a polynomial-time algorithm that produces solutions which are at most a linear factor for networks of height 2. We also showed that in networks of height 3 or more, complexity of approximation increases considerably: there is no approximation within a factor $2^{f(n)}$, for any sublinear function f of the input size n. This is also a tight bound, as we showed that the usual max-product algorithm by Poon and Domingos (2011) finds an approximation within factor $2^{c \cdot n}$ for some constant $c < 1$. Last, we showed that a simple modification to max-product results in an algorithm that is at least as good, and possibly greatly superior to max-product. We compared both algorithms in two different types of networks: shallow sum-product networks that encode random instances of the maximum independent set problem and deeper sum-product networks learned from real-world datasets. The empirical results show that while the proposed algorithm produces better solutions than max-product does, this improvement is less pronounced in the deeper realistic networks than in the shallower synthetic networks. This suggests that characteristics other than the height of the network might be equally important in determining the hardness of approximating maximum a posteriori inference, and that further (theoretical and empirical) investigations are required. We hope that these results foster research on approximation algorithms for maximum a posteriori inference in sum-product networks.

Acknowledgements

The second author received financial support from the São Paulo Research Foundation (FAPESP) grant #2016/01055-1 and the CNPq grants #303920/2016-5 and #420669/2016-7. We thank Renato Geh for making his source code freely available and promptly answering our questions.

References

Amer, M. R. (2012). Sum-product networks for modeling activities with stochastic structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1314–1321.

Amer, M. R. and Todorovic, S. (2016). Sum product networks for activity recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):800–813.

Boykov, Y., Veksler, O., and Zabih, R. (1998). Markov random fields with efficient approximations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 648–655.

Bulatov, A., Dyer, M., Goldberg, L. A., Jalsenius, M., Jerrum, M., and Richerby, D. (2012). The complexity of weighted and unweighted #CSP. *Journal of Computer and System Sciences*, 78(2):681–688.

Cheng, W.-C., Kok, S., Pham, H. V., Chieu, H. L., and Chai, K. M. A. (2014). Language modeling with sum-product networks. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association*.

Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

de Campos, C. P. (2011). New complexity results for MAP in Bayesian networks. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 2100–2106.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741.

Gens, R. and Domingos, P. (2013). Learning the structure of sum-product networks. In *Proceedings of 30th International Conference on Machine Learning*, pages 873–880.

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298.

Nath, A. and Domingos, P. (2016). Learning tractable probabilistic models for fault localization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1294–1301.

Peharz, R. (2015). *Foundations of sum-product networks for probabilistic modeling*. PhD thesis.

Peharz, R., Gens, R., Pernkopf, F., and Domingos, P. (2016). On the latent variable interpretation in sum-product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14.

Peharz, R., Kapeller, G., Mowlaee, P., and Pernkopf, F. (2014). Modeling speech with sum-product networks: Application to bandwidth extension. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3699–3703.

Peharz, R., Tschiatschek, S., Pernkopf, F., and Domingos, P. (2015). On theoretical properties of sum-product networks. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 744–752.

Poon, H. and Domingos, P. (2011). Sum-product networks: A new deep architecture. In *Proceedings of 27th Conference on Uncertainty in Artificial Intelligence*, pages 337–346.

Rooshenas, A. and Lowd, D. (2014). Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of the 31st International Conference on Machine Learning*, pages 710–718.

Steinder, M. and Sethi, A. (2004). Probabilistic fault localization in communication systems using belief networks. *IEEE/ACM Transactions on Networking*, 12(5):809–822.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2008). A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080.

Vergari, A., Mauro, N. D., and Esposito, F. (2015). Simplifying, regularizing and strengthening sum-product network structure learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 343–358.

Zhao, H., Melibari, M., and Poupart, P. (2015). On the relationship between sum-product networks and bayesian networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 116–124.

Zirn, C., Niepert, M., and Strube, Heiner Stuckenschmidt, M. (2011). Fine-grained sentiment analysis with structural features. *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 336–344.

Zuckerman, D. (2007). Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128.