Solving Marginal MAP Exactly by Probabilistic Circuit Transformations

YooJung Choi
Computer Science Department
UCLA
yjchoi@cs.ucla.edu

Tal Friedman
Computer Science Department
UCLA
tal@cs.ucla.edu

Guy Van den Broeck
Computer Science Department
UCLA
guyvdb@cs.ucla.edu

Abstract

Probabilistic circuits (PCs) are a class of tractable probabilistic models that allow efficient, often linear-time, inference of queries such as marginals and most probable explanations (MPE). However, marginal MAP, which is central to many decision-making problems, remains a hard query for PCs unless they satisfy highly restrictive structural constraints. In this paper, we develop a pruning algorithm that removes parts of the PC that are irrelevant to a marginal MAP query, shrinking the PC while maintaining the correct solution. This pruning technique is so effective that we are able to build a marginal MAP solver based solely on iteratively transforming the circuit—no search is required. We empirically demonstrate the efficacy of our approach on real-world datasets.

1 INTRODUCTION

Probabilistic circuits (PCs) refer to a family of tractable probabilistic models that are known to be able to closely capture the probability space in density estimation tasks (Dang et al., 2020; Liu and Van den Broeck, 2021; Peharz et al., 2020; Rooshenas and Lowd, 2014), while allowing tractable probabilistic inference of many useful queries (Li et al., 2021; Yu et al., 2021; Vergari et al., 2021). Perhaps the most widely supported queries for tractable inference by different kinds of PCs are: marginal inference, which computes the probability of a partial assignment; and the most probable explanations (MPE),[1] which computes for a given partial assignment (or evidence) the most likely state of all the remaining variables.

However, many related inference tasks remain hard even on those PCs tractable for marginals and MPE (Rahman et al., 2021; Rouhani et al., 2018). In particular, marginal MAP (maximum a posteriori hypothesis) is a closely related problem that still appears to be hard for most probabilistic circuits, despite being used in many applications including image segmentation, planning, and diagnosis, among others (Lee et al., 2014; Kiselev and Poupart, 2014; Bioucas-Dias and Figueiredo, 2016). A marginal MAP (MMAP) problem, unlike MPE, computes the most likely state of a subset of variables, while marginalizing out the others. Although these queries appear closely related, a PC that can tractably solve both marginals and MPE queries does not necessarily solve the marginal MAP tractably. In fact, exactly solving marginal MAP is known to be NP-hard, even for tractable PCs (de Campos, 2011). This remains to be the case when solving it approximately (Conaty et al., 2017; Mei et al., 2018).

Most existing marginal MAP solvers on PCs, especially exact solvers, are based on variations of branch-and-bound search (Mei et al., 2018; Huang et al., 2006), as has been the case for exact marginal MAP solvers for probabilistic graphical models (Park and Darwiche, 2002; Marinescu et al., 2014). In this paper, we propose a novel approach to marginal MAP inference: probabilistic circuit transformations.

In particular, we show that large parts of the circuit may be irrelevant to the marginal MAP problem at hand, and thus can be pruned away without affecting the solution. This in a sense "specializes" the PC to a particular MMAP instance and makes it more amenable to solving. We then develop an efficient algorithm to determine which parts of the cir-

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

[1] MPE is sometimes referred to as MAP (maximum a posteriori hypothesis). To avoid confusion, in this paper we will use the terms MPE and marginal MAP.

cuit can be safely pruned, using a novel edge bound. Lastly, we propose an exact MMAP solver that leverages this pruning algorithm and iteratively transforms the PC structure until the MMAP solution can be easily read from it. We show empirically on real-world benchmark datasets that our method can solve more marginal MAP instances with faster run time than existing solvers.

2 BACKGROUND

We use uppercase letters (X) to denote random variables and lowercase letters (x) for their assignments. Sets of variables are denoted by bold uppercase letters ($\mathbf{X}$) and their joint assignments by bold lowercase letters ($\boldsymbol{x}$). For a binary random variable X, we use logical negation $\neg X$ to denote $X = 0$. Lastly, we write the set of all values for $\mathbf{X}$ as $\mathsf{val}(\mathbf{X})$.

2.1 Marginal MAP

Suppose $p(\mathbf{X})$ is a probability distribution over a set of variables $\mathbf{X}$ which is partitioned into three subsets $\mathbf{Q}$, $\mathbf{E}$, and $\mathbf{H}$, referred to as the query, evidence, and hidden variables, respectively. Given some evidence $\boldsymbol{e} \in \mathsf{val}(\mathbf{E})$, the marginal MAP problem $\mathsf{MMAP}(\mathbf{Q}, \boldsymbol{e})$ is defined as follows:

$$\arg\max_{\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})} p(\boldsymbol{q}, \boldsymbol{e}) = \arg\max_{\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})} \sum_{\boldsymbol{h} \in \mathsf{val}(\mathbf{H})} p(\boldsymbol{q}, \boldsymbol{h}, \boldsymbol{e}).$$

Note that if $\mathbf{H}$ is empty, this corresponds to an MPE (most probable explanations) problem.

2.2 Probabilistic Circuits

A large family of tractable probabilistic models—including arithmetic circuits (Darwiche, 2003), and-or search spaces (Marinescu and Dechter, 2005), probabilistic sentential decision diagrams (Kisa et al., 2014), cutset networks (Rahman et al., 2014), and sum-product networks (Poon and Domingos, 2011)—are collectively referred to as *probabilistic circuits* (PCs) (Vergari et al., 2020).

A probabilistic circuit $\mathcal{C}$ over variables $\mathbf{X}$ is a directed acyclic graph (DAG) structure with parameters that defines a (possibly unnormalized) probability distribution over $\mathbf{X}$ in a recursive manner. Specifically, the DAG structure consists of leaf, product, and sum nodes. A leaf node is associated with a univariate function, denoted f_n, such as the indicator function $[X = 1]$. Every input edge (n, c) to a sum unit n is also associated with a parameter $\theta_{n,c} > 0$. Let $\mathsf{ch}(n)$ denote the set of children, or inputs, of an inner node n. A PC node then recursively defines a distribution as the following:

$$n(\boldsymbol{x}) = \begin{cases} f_n(\boldsymbol{x}) & \text{if } n \text{ is a leaf node} \\ \prod_{c \in \mathsf{ch}(n)} c(\boldsymbol{x}) & \text{if } n \text{ is a product node} \\ \sum_{c \in \mathsf{ch}(n)} \theta_{n,c} \cdot c(\boldsymbol{x}) & \text{if } n \text{ is a sum node} \end{cases}$$

We write $\mathcal{C}(\boldsymbol{x})$ to refer to $n(\boldsymbol{x})$ where n is the root of the PC $\mathcal{C}$.

A key strength of probabilistic circuits is that they support tractable inference, enabled by certain structural constraints. In particular, *smooth* and *decomposable* PCs allow efficient computation of marginal probabilities.

Definition 1. A PC $\mathcal{C}$ is *smooth* if for every sum node, its children depend on the same set of variables. A PC $\mathcal{C}$ is *decomposable* if for every product node, its children depend on disjoint sets of variables.

For a smooth and decomposable PC over variables $\mathbf{X}$, computing the marginal probability of some partial assignment $\boldsymbol{q} \in \mathsf{val}(\mathbf{Q}), \mathbf{Q} \subseteq \mathbf{X}$ amounts to the following procedure. A leaf node n is evaluated as 1 if it does not depend on a variable in $\mathbf{Q}$, and as $f_n(\boldsymbol{q})$ otherwise. Then we simply evaluate the circuit, taking (weighted) sums and products accordingly. For instance, consider the smooth and decomposable PC $\mathcal{C}$ in Figure 1a and a partial assignment $\boldsymbol{q} = \{X_1 = 1, X_2 = 0\}$. Then to compute the marginal $\mathcal{C}(\boldsymbol{q})$, we first set the leaf nodes labeled $\neg X_1$ and X_2 as 0, and all others as 1. Evaluating the circuit bottom up, we get the marginal probability $\mathcal{C}(\boldsymbol{q}) = 0.222$.

In addition, probabilistic circuits satisfying more restrictive structural constraints even support efficient inference of marginal MAP and related queries (Oztok et al., 2016; Choi et al., 2017). These structural constraints can be generalized into the notion of $\mathbf{Q}$-*determinism* (Choi et al., 2020).

Definition 2. Suppose $\mathcal{C}$ is a PC over variables $\mathbf{X}$ and let $\mathbf{Q} \subseteq \mathbf{X}$ be a subset. A sum node in $\mathcal{C}$ is $\mathbf{Q}$-deterministic if computing the marginal probability for any partial assignment $\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})$ makes at most one of its children evaluate to a nonzero output. A PC $\mathcal{C}$ is $\mathbf{Q}$-deterministic if all sum nodes containing variables in $\mathbf{Q}$ are $\mathbf{Q}$-deterministic.

Then, solving a marginal MAP problem $\mathsf{MMAP}(\mathbf{Q}, \boldsymbol{e})$ of a $\mathbf{Q}$-deterministic PC simply amounts to evaluating the circuit bottom-up similar to computing a marginal, except that every sum node that contains a variable in $\mathbf{Q}$ takes the weighted maximum of its inputs, instead of the weighted sum.

As one may intuit from the complexity of marginal MAP, enforcing this structural constraint on an arbitrary PC is an intractable task, as we also later demonstrate empirically. Furthermore, even if one somehow

learns or constructs a PC that satisfies $\mathbf{Q}$-determinism, this would support tractable marginal MAP only for this specific $\mathbf{Q}$. This is clearly infeasible in applications where one wishes to answer different marginal MAP queries using the probabilistic model.

In the following sections, we assume a PC that satisfies smoothness and decomposability. Moreover, for simplicity of exposition, we consider only the marginal MAP problems without any evidence. This is because a given evidence can be incorporated into the PC by setting the leaf nodes (just like for computing marginals), and then we can equivalently solve the marginal MAP problem with no evidence on the resulting PC.

3 CIRCUIT PRUNING FOR MARGINAL MAP

We now describe the main contribution behind our proposed marginal MAP solver: pruning parts of a probabilistic circuit without affecting its MMAP solution. This is motivated by two key observations.

3.1 Motivation

Consider the following two observations.

(i): Computing the marginal probability of any partial assignment $\boldsymbol{q}$ is equivalent to evaluating a sub-circuit in which every $\mathbf{Q}$-deterministic sum node has one input. In other words, the sub-circuit for $\boldsymbol{q}$ includes the parts of the PC that are used or "activated" when computing the marginal of $\boldsymbol{q}$. Let us call this the $\boldsymbol{q}$-subcircuit and denote it by $\mathcal{C}'_{\boldsymbol{q}}$. We illustrate this with the example PC in Figure 1a. Suppose $\mathbf{Q} = \{X_1, X_2\}$ and we wish to compute the marginal probability of $\boldsymbol{q} = \{X_1 = 1, X_2 = 0\}$. Recall from Section 2 that this corresponds to setting the input units for $\neg X_1$ and X_2 to 0 and all others to 1, then evaluating the circuit in a bottom-up fashion. We can quickly check that the output is $0.6 \cdot (0.7 \cdot 0.1 + 0.3) = 0.222$, which is equivalent to simply evaluating the sub-circuit highlighted in blue with its input units set to 1. Moreover, observe that every $\mathbf{Q}$-deterministic sum node (highlighted in orange) that is included in this sub-circuit has exactly one input.

(ii): If we remove an edge that does not appear in the sub-circuit for any assignment $\boldsymbol{q}$, then the (unnormalized) probability of $\boldsymbol{q}$ is unchanged in the resulting PC. This directly follows from observation (i). For example, removing any non-colored edge from the PC in Figure 1a does not affect the marginal for $\boldsymbol{q}$, as defined previously, in the resulting circuit. Moreover, if an edge in the sub-circuit for $\boldsymbol{q}$ is removed, then the probability of $\boldsymbol{q}$ decreases in the resulting PC. Again visiting Figure 1a, removing the edge represented by the dashed line will drop the probability of $\boldsymbol{q} = \{X_1 = 1, X_2 = 0\}$ from 0.222 to $0.6 \cdot 0.3 = 0.18$.

We can apply observations (i) and (ii) to the marginal MAP state, denoted by $\boldsymbol{q}^\star$, to conclude that any edge that does not appear in the $\boldsymbol{q}^\star$-subcircuit (namely the "solution sub-circuit") can be pruned away while keeping the MMAP problem equivalent. That is, removing an edge that is not in the solution sub-circuit will not affect the probability of $\boldsymbol{q}^\star$ but may decrease the probabilities of other assignments to $\mathbf{Q}$; hence, $\boldsymbol{q}^\star$ remains as the solution for marginal MAP problem in the pruned circuit. Solving a MMAP instance by solving the equivalent problem on a pruned circuit can have the following important benefits. First, the complexity of inference algorithms on PCs generally depends on the size of the circuit, and thus reducing the size by pruning edges is desirable. In addition, because pruning as described above keeps the marginal MAP probability while potentially decreasing other marginal probabilities, it effectively increases the gap between the solution and other states. This can not only lead to more iterations of pruning, further specializing the circuit to the MMAP problem, but also arguably make the problem easier to solve. For example, in the extreme case that all edges other than the solution sub-circuit are pruned, the resulting MMAP problem becomes trivial to solve.

Given these benefits, we naturally raise the following question: *can we efficiently determine which edges do not appear in the solution sub-circuit (i.e. $\boldsymbol{q}^\star$-subcircuit)?* The challenge is to do this without knowing a priori the marginal MAP state $\boldsymbol{q}^\star$. In the following section, we propose an algorithm that efficiently computes, for every edge, an upper bound on the output of any sub-circuit that includes the edge, which gives a positive answer to the previous question.

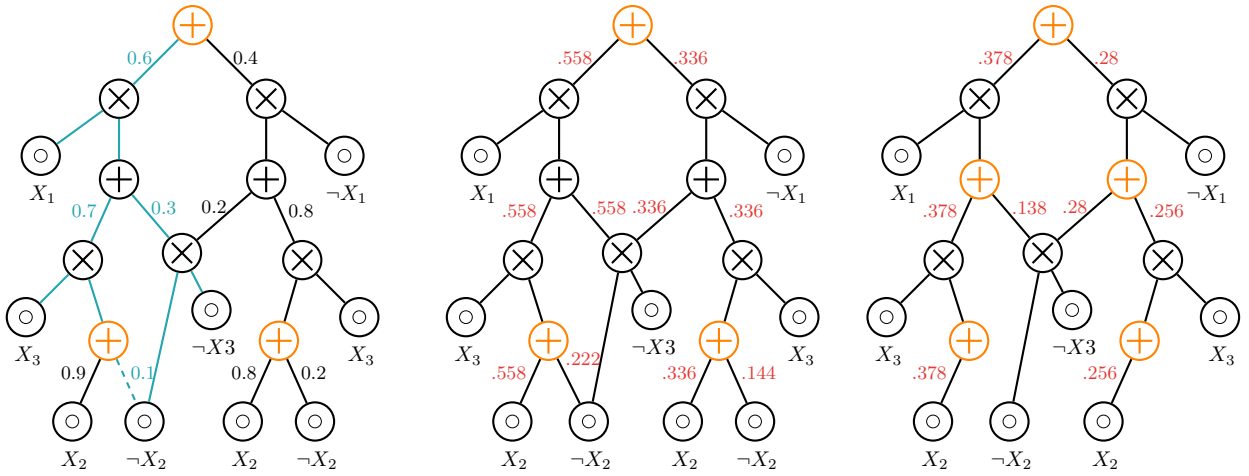
3.2 Edge Bounds

We will now define more formally our edge bounds and the algorithm to efficiently compute them.

Definition Abusing notation, let us denote by $\mathsf{MMAP}(\mathbf{Q}|_{(n,c)})$ the largest marginal probability obtainable by an assignment $\boldsymbol{q}$ whose $\boldsymbol{q}$-subcircuit includes the edge (n, c). Formally,

$$\mathsf{MMAP}(\mathbf{Q}|_{(n,c)}) := \max_{\boldsymbol{q}:(n,c)\in\mathcal{C}'_{\boldsymbol{q}}} \mathcal{C}(\boldsymbol{q}). \tag{1}$$

Intuitively, this corresponds to a marginal MAP problem where the possible states have been reduced from $\mathsf{val}(\mathbf{Q})$ to those that "activate" the edge (n, c) when computing their marginal probability. Moreover, suppose we define a hypothetical edge from the root to



(a) Example PC

(b) PC before and after pruning. Sum edges are labeled with edge bounds.

Figure 1: A smooth and decomposable PC over variables $\{X_1, X_2, X_3\}$. Orange sum nodes are $\mathbf{Q}$-deterministic for $\mathbf{Q} = \{X_1, X_2\}$; blue edges form the sub-circuit for joint assignment $\boldsymbol{q} = \{X_1 = 1, X_2 = 0\}$.

output, denoted $(\cdot, \text{root})$. Then by definition, the MMAP reduced to this edge, i.e. $\mathsf{MMAP}(\mathbf{Q}|_{(\cdot,\text{root})})$ is simply the MMAP problem $\mathsf{MMAP}(\mathbf{Q})$.

For each edge (n, c), we wish to obtain an edge bound $\mathsf{EB}(n, c)$ that satisfies the following:

$$\mathsf{MMAP}(\mathbf{Q}|_{(n,c)}) \leq \mathsf{EB}(n, c).$$

Let us also introduce EB for each node n, which may be useful as intermediate quantities as will be apparent later.

$$\mathsf{MMAP}(\mathbf{Q}|_{n}) \leq \mathsf{EB}(n).$$

It is important to note that the edge bound $\mathsf{EB}(n, c)$ is not a bound on some output from the edge or either of the nodes connected by it. Rather, it bounds from above the output of the PC at the root, using the edge to limit the state space. Suppose we are given such edge bound; clearly, if we also have a lower bound on the marginal MAP probability, we can safely prune any edge whose EB is smaller than the given lower bound.

Computing the Edge Bound To develop an edge bound with the properties described above, we first observe that every $\boldsymbol{q}$-subcircuit that includes an edge (n, c) must also include the node n. Then, we can conclude that $\mathsf{MMAP}(\mathbf{Q}|_{(n,c)}) \leq \mathsf{MMAP}(\mathbf{Q}|_{n})$. Suppose we have an upper bound on the MMAP reduced to node n. Such bound will also be at least as large as the MMAP reduced to edge (n, c), and can be used as edge bound $\mathsf{EB}(n, c)$. However, $\mathsf{EB}(n, c)$ need not be as large as $\mathsf{MMAP}(\mathbf{Q}|_{n})$, so there may be some opportunity to tighten the bound going from n to (n, c).

As a base case of the top-down recursion, we need an upper-bound of MMAP at the root. For this, we use the algorithm by Huang et al. (2006), shown in Algorithm 1, which not only computes the upper-bound on marginal MAP at the root node but also bounds the output of every node, via a single feedforward pass on the PC. Formally, for every node $n \in \mathcal{C}$ it computes an upper bound on:

$$\max_{\boldsymbol{q}: n \in \mathcal{C}'_{\boldsymbol{q}}} n(\boldsymbol{q}) = \max_{\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})} n(\boldsymbol{q}), \tag{2}$$

and stores it in m_n. Let us denote the upper-bound at the root by $\mathsf{m}_{\mathcal{C}}$.

Intuitively, our proposed edge bound $\mathsf{EB}(n, c)$ aims to upper-bound the largest value returned by Algorithm 1 on a $\boldsymbol{q}$-subcircuit that includes the edge (n, c). In other words, for each edge (n, c), we bound from above the following:

$$\max_{\boldsymbol{q}: (n,c) \in \mathcal{C}'_{\boldsymbol{q}}} \mathsf{m}_{\mathcal{C}'_{\boldsymbol{q}}} \geq \mathsf{MMAP}(\mathbf{Q}|_{(n,c)}).$$

$\mathsf{EB}(n)$ then similarly upper-bounds $\mathsf{MMAP}(\mathbf{Q}|_{n})$. It is worth pointing out that this bounds the output *at the root* for states $\boldsymbol{q}$ that includes n in their sub-circuits, whereas m_n by Algorithm 1 upper-bounds the output *at each node*.

Let us now describe the recursive steps. First, suppose we want to compute $\mathsf{EB}(n)$ where $\mathsf{EB}(p, n)$ for every parent p of n (i.e. $n \in \mathsf{ch}(p)$) has been computed by the recursion. In order to make sure that $\mathsf{EB}(n)$ upper-bounds the marginal MAP reduced to n, we observe that if $\boldsymbol{q}$ is the solution to $\mathsf{MMAP}(\mathbf{Q}|_{n})$ then the $\boldsymbol{q}$-subcircuit must also include one of the edges (p, n).

Algorithm 1 OUTPUT-BOUNDS($\mathcal{C}, \mathbf{Q}$)

Input: a smooth & decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a set of query variables $\mathbf{Q} \subset \mathbf{X}$
Output: m_n storing output bounds for each node n

$\mathsf{N} \leftarrow$ FEEDFORWARDORDER($\mathcal{C}$)
for each $n \in \mathsf{N}$ **do**
  if n is an input unit **then**
    $\mathsf{m}_n \leftarrow \mathcal{C}_n^{\max}(\boldsymbol{x}_{\phi(n)})$
  else if n is a product unit **then**
    $\mathsf{m}_n \leftarrow \prod_{c \in \mathsf{ch}(n)} \mathsf{m}_c$
  else if n is $\mathbf{Q}$-deterministic **then**
    $\mathsf{m}_n \leftarrow \max_{c \in \mathsf{ch}(n)} \theta_{n,c} \mathsf{m}_c$
  else
    $\mathsf{m}_n \leftarrow \sum_{c \in \mathsf{ch}(n)} \theta_{n,c} \mathsf{m}_c$

Thus, $\mathsf{EB}(n) = \max_p(\mathsf{EB}(p, n))$ is a valid edge bound:

$$\max_{p: n \in \mathsf{ch}(p)} \mathsf{EB}(p, n) \geq \max_{p: n \in \mathsf{ch}(p)} \mathsf{MMAP}(\mathbf{Q}|_{(p,n)}) = \mathsf{MMAP}(\mathbf{Q}|_n)$$

Next, suppose we wish to compute $\mathsf{EB}(n, c)$ from a given $\mathsf{EB}(n)$. We consider the three possible cases of n being a $\mathbf{Q}$-deterministic sum node, a non $\mathbf{Q}$-deterministic sum node, and a product node. For the latter two cases, the edge bounds are simply propagated from the node. This is because any sub-circuit that includes such node will also include both of its input edges, and thus their bounds will be the same.

Finally, we consider the edge bound $\mathsf{EB}(n, c)$ for an input edge to a $\mathbf{Q}$-deterministic sum node. To illustrate the intuition, we use the example PC in Figure 1a. Suppose we want the edge bound between the root and its right input, denoted $\mathsf{EB}(\text{root}, r)$. Running Algorithm 1, we get the upper bound $\mathsf{m}_{\text{root}} = 0.558$ at the root and $\mathsf{m}_l = 0.93$ and $\mathsf{m}_r = 0.84$ for its left and right input, respectively. Note that for every $\boldsymbol{q}$ that includes this edge in its sub-circuit,[2] the marginal $\mathcal{C}(\boldsymbol{q})$ must be $0.4 \cdot r(\boldsymbol{q})$, leading to:

$$\max_{\boldsymbol{q}: (\text{root}, r) \in \mathcal{C}'_{\boldsymbol{q}}} \mathcal{C}(\boldsymbol{q}) = 0.4 \cdot \max_{\boldsymbol{q}: (\text{root}, r) \in \mathcal{C}'_{\boldsymbol{q}}} r(\boldsymbol{q}) \leq 0.4 \cdot \mathsf{m}_r.$$

Thus, we can use $0.4 \cdot \mathsf{m}_r = 0.336$ as the edge bound for (root, r). Similarly, we can derive the edge bound for (root, l) as $0.6 \cdot \mathsf{m}_l = 0.558$. This can be expressed as:

$$\mathsf{EB}(\text{root}, c) = \mathsf{EB}(\text{root}) - \mathsf{m}_{\text{root}} + \theta_{\text{root},c} \mathsf{m}_c \qquad (3)$$

for any $c \in \mathsf{ch}(\text{root})$. Note that this holds trivially because $\mathsf{EB}(\text{root}) = \mathsf{m}_{\text{root}}$ as the base case. However, we can generalize this to derive the expression for edge bound from an inner $\mathbf{Q}$-deterministic node.

[2] This corresponds to $\{X_1 = 0, X_2 = 0\}$ and $\{X_1 = 0, X_2 = 1\}$.

Algorithm 2 EDGE-BOUNDS($\mathcal{C}, \mathbf{Q}$)

Input: a smooth & decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a set of query variables $\mathbf{Q} \subset \mathbf{X}$
Output: $\mathsf{r}_{n,c}$ storing edge bounds for each edge (n, c)

$\mathsf{m} \leftarrow$ OUTPUT-BOUNDS($\mathcal{C}, \mathbf{Q}$)
$\mathsf{t}_{\text{root}} \leftarrow 1$
$\mathsf{r}_{\text{root}} \leftarrow \mathsf{m}_{\text{root}}$
$\mathsf{N} \leftarrow$ BACKWARDORDER($\mathcal{C}$)
for each $n \in \mathsf{N}$ s.t. $\mathsf{t}_n > 0$, $c \in \mathsf{ch}(n)$ **do**
  if n is a product unit **then**
    $\mathsf{r}_{n,c} \leftarrow \mathsf{r}_n$
    $\mathsf{r}_c \leftarrow \max(\mathsf{r}_c, \mathsf{r}_{n,c})$
    $\mathsf{t}_c \leftarrow \min(\mathsf{t}_c, \mathsf{t}_n)$
  else if n is a sum unit **then**
    if n is $\mathbf{Q}$-deterministic **then**
      $\mathsf{r}_{n,c} \leftarrow \mathsf{r}_n + \mathsf{t}_n(\theta_{n,c}\mathsf{m}_c - \mathsf{m}_n)$
    else
      $\mathsf{r}_{n,c} \leftarrow \mathsf{r}_n$
    $\mathsf{r}_c \leftarrow \max(\mathsf{r}_c, \mathsf{r}_{n,c})$
    $\mathsf{t}_c \leftarrow \min(\mathsf{t}_c, \theta_{n,c}\mathsf{t}_n)$

Let us again use Figure 1a as an example; this time we consider the blue dashed edge, denoting it (n, c). Recall that $\mathsf{EB}(n, c)$ aims to upper-bound what Algorithm 1 would return at the root of a $\boldsymbol{q}$-subciruit that includes edge (n, c). In such sub-circuit, (n, c) would be the only input edge to node n, and thus the algorithm would propagate up $\theta_{n,c}\mathsf{m}_c = 0.1$ instead of m_n. This hints at a similar expression as Equation (3) where we subtract the contribution of m_n and add $\theta_{n,c}\mathsf{m}_c$. However, a key observation is that m bounds from Algorithm 1 concern the output of each node, whereas the edge bounds concern the output of the root node. Thus, we need to consider how the contribution of m_n gets scaled when it is propagated up to the root node. In this instance, it would be multiplied by $0.7 \cdot 0.6$, which is the product of edge parameters that lie in the path from n to the root. In other words, we get the following expression:

$$\mathsf{EB}(n, c) = \mathsf{EB}(n) + 0.7 \cdot 0.6(-\mathsf{m}_n + \theta_{n,c}\mathsf{m}_c)$$

The pseudocode for this recursive algorithm is described in Algorithm 2.

Proposition 1. Given a smooth and decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a subset $\mathbf{Q} \subset \mathbf{X}$, Algorithm 2 computes an upper bound on Equation (1) for every edge in $\mathcal{C}$.

Pruning example We refer to the Appendix for a formal proof of the above proposition, and instead conclude this section with an example round of pruning.

Suppose we wish to prune edges from the PC in Figure 1a, for an MMAP problem with $\mathbf{Q} = \{X_1, X_2\}$. First, we compute the edge bounds as shown in the left circuit in Figure 1b. To perform pruning, we need a lower bound on the marginal MAP probability to compare against. The probability of any $\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})$ state suffices; suppose we use $\boldsymbol{q} = \{X_1 = 0, X_2 = 1\}$ with $p(\boldsymbol{q}) = 0.256$. Then we can prune two edges, resulting in the circuit on the right in Figure 1b. More notably, all sum nodes in the resulting circuit become $\mathbf{Q}$-deterministic (highlighted in orange). In particular, as we will discuss more in the next section, this allows us to answer the marginal MAP query via a single feedforward pass. Running Algorithm 1 on this PC, the output at the root is 0.378 which exactly corresponds to the marginal MAP solution $p(X_1 = 1, X_2 = 1) = 0.378$.

Thus, pruning not only has the immediate effect of decreasing the circuit size, but also changes the PC and its distribution in such a way that can make it easier to solve the marginal MAP problem.

4 ITERATIVE MARGINAL MAP SOLVER

We are now ready to show how the pruning algorithm from the previous section can be leveraged to solve marginal MAP exactly.

As discussed briefly in Section 2, we can tractably answer a marginal MAP query for a $\mathbf{Q}$-deterministic PC. Thus, a naive solver may try to transform the input PC into a $\mathbf{Q}$-deterministic one to solve a marginal MAP instance. For example, one could apply the *split* operation (Liang et al., 2017; Dang et al., 2020) on the root for each variable in $\mathbf{Q}$. Splitting on a variable $Q \in \mathbf{Q}$ effectively turns the root of the PC into a Q-deterministic sum node while maintaining the distribution represented by it; thus, splitting on every variable in $\mathbf{Q}$ would result in a $\mathbf{Q}$-deterministic circuit. However, this would be highly intractable as each split operation could at most double the size of the PC.

Instead, we propose to prune the circuit as well as split on a query variable in each iteration. While the circuit could grow exponentially in the worst case, we show empirically in the next section that pruning plays a crucial role in indeed keeping the circuit size from growing too much. In fact, in many instances, it decreases the circuit size over the iterations.

A pseudocode of our approach is shown in Algorithm 3.[3] The solver maintains an upper and lower bound on marginal MAP and updates it after every prune and split. The upper bound is computed using Algorithm 1 as discussed in Section 3.2. The marginal probability of any instantiation of $\mathbf{Q}$ can be used as a lower bound on the MMAP probability. In particular, we use the solution to a different MMAP instance whose query variables include $\mathbf{Q}$ and can be solved efficiently; more details can be found in the Appendix. In each iteration, we first prune all edges whose edge bound, computed by Algorithm 2, does not exceed the current lower bound. Then we split on a variable chosen according to some heuristic (discussed further in the next section). The solver is guaranteed to converge after at most $|\mathbf{Q}|$ iterations, at which point the PC must be $\mathbf{Q}$-deterministic, allowing exact computation of MMAP. Furthermore, each prune and split improves the bounds, and thus the solver may also terminate before splitting on all query variables. That is, pruning can decrease the upper bound as we saw in Figure 1b, and a split operation also improves the bounds by adding a new $\mathbf{Q}$-deterministic node at the root. Lastly, we again emphasize that our marginal MAP solver only assumes smoothness and decomposability; determinism is not required. For example, this implies that we can also exactly solve MPE for non-deterministic PCs.

Algorithm 3 ITER-SOLVE($\mathcal{C}, \mathbf{Q}$)

Input: a smooth & decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a set of query variables $\mathbf{Q} \subset \mathbf{X}$
Output: $\mathsf{MMAP}(\mathbf{Q})$

$u \leftarrow$ OUTPUT-BOUNDS($\mathcal{C}, \mathbf{Q}$)
$l \leftarrow$ LOWER-BOUND($\mathcal{C}, \mathbf{Q}$)
$\mathbf{V} \leftarrow \mathbf{Q}$
while $u > l$ **do**
  $\mathsf{r} \leftarrow$ EDGE-BOUNDS($\mathcal{C}, \mathbf{Q}$)
  for all $(n, c) \in \mathcal{C}$ s.t. $\mathsf{r}_{n,c} \leq l$ **do**
    $\mathcal{C} \leftarrow$ PRUNE-EDGE($\mathcal{C}, (n, c)$)
  $X \leftarrow$ PICK-VAR($\mathbf{V}$); $\mathbf{V} \leftarrow \mathbf{V} \setminus \{X\}$
  $\mathcal{C} \leftarrow$ SPLIT($\mathcal{C}, X$)
  $u \leftarrow \min(u,$ OUTPUT-BOUNDS($\mathcal{C}, \mathbf{Q}$))
  $l \leftarrow \max(l,$ LOWER-BOUND($\mathcal{C}, \mathbf{Q}$))
return u

5 EXPERIMENTS

We evaluated the iterative solver on probabilistic circuits learned from twenty widely-used benchmark datasets. The number of variables ranges from 16 to 1,556, and the size of PCs, learned using Strudel (Dang et al., 2020), ranges from 3,177 to 745,815. We generated marginal MAP instances with two different proportions of query, evidence, and hidden variables—30%, 30%, 40% and 50%, 20%, 30%, respectively—

[3] Our marginal MAP solver is implemented in https://github.com/Juice-jl/ProbabilisticCircuits.jl.

Table 1: Average run time in seconds (with 1-hour time limit for each instance) and the number of instances solved for different proportions of (query, evidence, hidden) variables.

	(30%, 30%, 40%)			(50%, 20%, 30%)		
Dataset	**MaxSPN**	**(Pruned)**	**(UB)**	**MaxSPN**	**(Pruned)**	**(UB)**
NLTCS	**0.004** (10)	0.35 (10)	0.54 (10)	**0.01** (10)	0.39 (10)	0.63 (10)
MSNBC	**0.01** (10)	0.29 (10)	0.50 (10)	**0.03** (10)	0.43 (10)	0.73 (10)
KDD	**0.02** (10)	0.42 (10)	0.64 (10)	**0.04** (10)	0.49 (10)	0.68 (10)
Plants	**0.27** (10)	0.99 (10)	1.36 (10)	2.95 (10)	**2.61** (10)	2.72 (10)
Audio	188.59 (10)	16.57 (10)	**2.87** (10)	2041.33 (6)	15.61 (10)	**13.70 (10)**
Jester	265.50 (10)	16.16 (10)	**6.17** (10)	2913.04 (2)	44.16 (10)	**14.74 (10)**
Netflix	344.71 (10)	22.23 (10)	**5.61** (10)	– (0)	936.83 (10)	**47.18 (10)**
Accidents	**0.54** (10)	2.00 (10)	2.00 (10)	109.56 (10)	19.81 (10)	**15.86** (10)
Retail	**0.03** (10)	0.47 (10)	0.61 (10)	**0.06** (10)	0.67 (10)	0.81 (10)
Pumsb-star	273.70 (10)	106.04 (10)	**6.04** (10)	2208.27 (7)	54.32 (10)	**20.88 (10)**
DNA	2809.44 (4)	65.27 (10)	**9.16 (10)**	– (0)	2634.41 (3)	**505.75 (9)**
Kosarek	1.60 (10)	**0.81** (10)	0.98 (10)	48.74 (10)	**2.65** (10)	3.41 (10)
MSWeb	25.70 (10)	3.63 (10)	**0.96** (10)	1543.49 (10)	48.89 (10)	**1.28** (10)
Book	– (0)	56.47 (10)	**7.25 (10)**	– (0)	907.51 (9)	**46.50 (10)**
EachMovie	– (0)	2563.02 (3)	**93.66 (10)**	– (0)	3293.78 (1)	**1216.89 (8)**
WebKB	– (0)	3378.03 (2)	**102.37 (10)**	– (0)	– (0)	**575.68 (10)**
Reuters-52	– (0)	1238.10 (7)	**22.91 (10)**	– (0)	3107.57 (3)	**120.58 (10)**
20 NewsGrp.	– (0)	2882.95 (3)	**88.13 (10)**	– (0)	– (0)	**504.52 (9)**
BBC	– (0)	– (0)	**766.93 (9)**	– (0)	– (0)	**2757.18 (3)**
Ad	– (0)	– (0)	**344.81 (10)**	– (0)	– (0)	**1254.37 (8)**
Total Solved	124	155	**199**	105	146	**187**

randomly dividing the variables and generating evidence while ensuring its probability is nonzero. We generated 10 instances for each dataset and each proportion.

On each instance, we run our iterative solver with two different variable split heuristics. (Pruned) selects variables based on the number of pruned edges associated with the variable; (UB) selects variables by the expected change in upper bound after splitting on a variable, which can be computed efficiently via a single pass on the circuit. We refer to the Appendix for a more detailed description of split heuristics and algorithms to compute them. For comparison, we also solved the marginal MAP problems using MaxSPN[4] which is a search-based exact solver for (marginal) MAP on sum-product networks (Mei et al., 2018). All experiments were ran on a Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz.

Table 1 summarizes the results. First, we compare the two heuristics. (Pruned) is comparable or faster than (UB) on relatively easy datasets, but is significantly slower on most of the datasets. Moreover, (Pruned) failed to solve any instance on BBC and Ad datasets, whereas (UB) was able to solve at least one instance in all datasets. In fact, it was able to solve all 20 instances (10 for each proportion) on 15 out of the 20 datasets. This clearly demonstrates the importance of variable split heuristics and the benefit of explicitly choosing splits that lead to better bounds.

Next, we compare our iterative solver to the search-based approach of MaxSPN. We observe that MaxSPN is faster than our algorithm on easy instances (sub-1 second average run time). This is likely because there is a minimum overhead of performing circuit transformations. On the other hand, our iterative approach clearly outperforms MaxSPN on all other datasets, both in terms of average run time and the number of instances solved.

Lastly, we examine more closely an example run of our solver to empirically demonstrate the benefits of pruning a PC for a specific marginal MAP problem; see Figure 2. As we expected, iterative prune and split improve the upper and lower bounds until they converge. The next plot on circuit size clearly illustrates the importance of pruning the circuit. Even though split operations can increase the circuit size, we are very effective at pruning away irrelevant parts of the circuit for MMAP that the circuit size actually

[4]Specifically, we use the forward checking technique with ordering and stage, which was shown to be the best performing among the exact solvers by Mei et al. (2018).

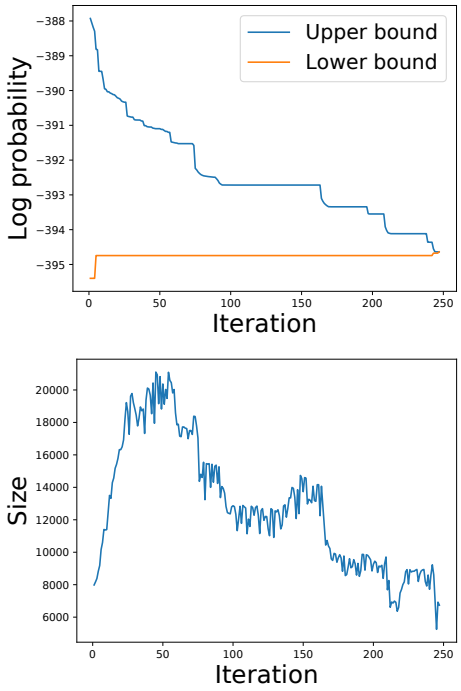


Figure 2: Upper and lower bounds (top) and circuit size (bottom) in each iteration of the solver on an example instance on EachMovie dataset.

decreases over time. Indeed, the size at the point of convergence is smaller than the initial size. Judging by the rate of increase in the early iterations, it is not hard to imagine that without pruning, the circuit would quickly grow too large to run any inference.

6 CONCLUSION

We have introduced a novel approach to marginal MAP inference on probabilistic circuits. It is fundamentally distinct from existing solvers, which are based on a branch and bound search (Mauá et al., 2020; Mei et al., 2018; Huang et al., 2006) using the tractable circuit to prune the search. Instead, we showed that the circuit can be pruned by keeping edges that are relevant to the marginal MAP state. Furthermore, our edge bounds algorithm can effectively find such edges to prune. What remains to solve marginal MAP is to perform simple splits on the circuit, tightening the bounds, and providing more opportunity to prune edges, until a marginal MAP solution is found. Our experiments empirically show that this novel approach to marginal MAP outperforms the search-based approach on a large number of real-world learned probabilistic circuits.

Acknowledgements

This work is partially supported by a DARPA PTG grant, NSF grants #IIS-1943641, #IIS-1956441, #CCF-1837129, Samsung, CISCO, and a Sloan Fellowship.

References

José M. Bioucas-Dias and Mário A. T. Figueiredo. Bayesian image segmentation using hidden fields: Supervised, unsupervised, and semi-supervised formulations. In *24th European Signal Processing Conference (EUSIPCO)*, pages 523–527, 2016.

YooJung Choi, Adnan Darwiche, and Guy Van den Broeck. Optimal feature selection for decision robustness in bayesian networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2017.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. Oct 2020.

Diarmaid Conaty, Denis D Maua, and Casio P de Campos. Approximation complexity of maximum a posteriori inference in sum-product networks. In *The 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI, 2017.

Meihua Dang, Antonio Vergari, and Guy Van den Broeck. Strudel: Learning structured-decomposable probabilistic circuits. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models (PGM)*, sep 2020.

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

Cassio P de Campos. New complexity results for map in bayesian networks. In *IJCAI*, volume 11, pages 2100–2106, 2011.

Jinbo Huang, Mark Chavira, and Adnan Darwiche. Solving MAP exactly by searching on compiled arithmetic circuits. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 143–148, 2006.

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2014.

Igor Kiselev and Pascal Poupart. Policy optimization by marginal-map probabilistic inference in generative models. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1611–1612, 2014.

Junkyu Lee, Radu Marinescu, and Rina Dechter. Applying marginal map search to probabilistic conformant planning: Initial results. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Wenzhe Li, Zhe Zeng, Antonio Vergari, and Guy Van den Broeck. Tractable computation of expected kernels. In *Proceedings of the 37th Conference on Uncertainty in Aritifical Intelligence (UAI)*, 2021.

Yitao Liang, Jessa Bekker, and Guy Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, August 2017.

Anji Liu and Guy Van den Broeck. Tractable regularization of probabilistic circuits. In *Proceedings of the UAI Workshop on Tractable Probabilistic Modeling (TPM)*, 2021.

Radu Marinescu and Rina Dechter. And/or branch-and-bound for graphical models. In *IJCAI*, pages 224–229, 2005.

Radu Marinescu, Rina Dechter, and Alexander T Ihler. And/or search for marginal map. In *UAI*, pages 563–572. Citeseer, 2014.

Denis Deratani Mauá, Heitor Ribeiro Reis, Gustavo Perez Katague, and Alessandro Antonucci. Two reformulation approaches to maximum-a-posteriori inference in sum-product networks. In *International Conference on Probabilistic Graphical Models*, pages 293–304. PMLR, 2020.

Jun Mei, Yong Jiang, and Kewei Tu. Maximum a posteriori inference in sum-product networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Umut Oztok, Arthur Choi, and Adnan Darwiche. Solving PP^{PP}-complete problems using knowledge compilation. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 94–103, 2016.

James D Park and Adnan Darwiche. Solving map exactly using systematic search. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 459–468, 2002.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011.

Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 630–645. Springer, 2014.

Tahrima Rahman, Sara Rouhani, and Vibhav Gogate. Novel upper bounds for the constrained most probable explanation task. *Advances in Neural Information Processing Systems*, 34, 2021.

Amirmohammad Rooshenas and Daniel Lowd. Learning sum-product networks with direct and indirect variable interactions. In *International Conference on Machine Learning*, pages 710–718. PMLR, 2014.

Sara Rouhani, Tahrima Rahman, and Vibhav Gogate. Algorithms for the nearest assignment problem. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5096–5102, 2018.

Antonio Vergari, YooJung Choi, Robert Peharz, and Guy Van den Broeck. Probabilistic circuits: Representations, inference, learning and applications. *AAAI Tutorial*, 2020.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, dec 2021.

Zhongjie Yu, Mingye Zhu, Martin Trapp, Arseny Skryagin, and Kristian Kersting. Leveraging probabilistic circuits for nonparametric multi-output regression. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 2008–2018. PMLR, 27–30 Jul 2021.

Supplementary Material: Solving Marginal MAP Exactly by Probabilistic Circuit Transformations

A In-depth Look at Circuit Pruning

A.1 q-subcircuit

We first formally define the notion of $\boldsymbol{q}$-subcircuit used throughout the paper. This is expressed through the notion of contexts.

Definition 3 (Context)**.** Let $\mathcal{C}$ be a PC over variables $\mathbf{X}$ and n be one of its nodes. The *context* γ_n of node n denotes all joint assignments that return a nonzero value for all nodes in a path between the root of $\mathcal{C}$ and n.

$$\gamma_n := \bigcup_{p \in \mathsf{pa}(n)} \gamma_p \cap \mathsf{supp}(n)$$

where $\mathsf{pa}(n)$ refers to the parent nodes of n and $\mathsf{supp}(n) := \{\boldsymbol{x} : \mathcal{C}_n(\boldsymbol{x}) > 0\}$ is the support of node n. The context $\gamma_{(n,c)}$ of an edge (n, c) is defined as $\gamma_{(n,c)} := \gamma_n \cap \gamma_c$.

Then for any $\boldsymbol{q}$, an edge (n, c) is said to be in the $\boldsymbol{q}$-subcircuit if $\boldsymbol{q} \in \gamma_{(n,c)}\big|_{\mathbf{Q}}$; i.e., the context of (n, c) reduced to variables in Q contains the assignment $\boldsymbol{q}$.

A.2 Proof of Proposition 1

Proposition 1. Given a smooth and decomposable PC $\mathcal{C}$ over variables $\mathbf{X}$ and a subset $\mathbf{Q} \subset \mathbf{X}$, Algorithm 2 computes an upper bound on Equation (1) for every edge in $\mathcal{C}$.

To prove above proposition, let us define some auxiliary circuit structures. First, running Algorithm 1 to compute m can be interpreted as a feedforward evaluation on a circuit obtained from $\mathcal{C}$ by replacing every $\mathbf{Q}$-deterministic sum node n with a node that simply returns the output of child node $c = \arg\max_{c \in \mathsf{ch}(n)} \theta_{n,c} \mathsf{m}_c$ (i.e. they are "fixed" to select the same branch as Line 7 in Algorithm 1). Suppose we unroll such circuit into a tree structure: i.e. create copies of any node with multiple parents and recurse down. We denote this circuit by $\mathcal{M}$. Then we have $\mathsf{m}_{\text{root}} = \mathcal{M}(\emptyset)$, where $\mathcal{M}(\emptyset)$ represents the circuit evaluation for marginal with no evidence. Moreover, for any node n' in $\mathcal{M}$ that corresponds to node n in $\mathcal{C}$, written as $n' \in \text{copy}(n)$, we have $\mathsf{m}_n = \mathcal{M}_{n'}(\emptyset)$.

In addition, for every node n' in $\mathcal{M}$, we define a circuit denoted $\mathcal{M}^{(n')}$ obtained from $\mathcal{M}$ by "fixing" the $\mathbf{Q}$-deterministic nodes that appear in the path from root to n' such that they select the branch that reaches n'. In other words, let $\mathbf{Q}'$ be $\mathbf{Q} \setminus \phi(n')$ and $\boldsymbol{q}' = \gamma_{n'}|_{\mathbf{Q}'}$. Note that because $\mathcal{M}$ is a tree structure, every assignment in the context of n' has the same value for variables in $\mathbf{Q}'$; this is given by the $\mathbf{Q}$-deterministic sum nodes in the path from n' to the root which is unique. Then $\mathcal{M}^{(n')}$ is identical to $\mathcal{M}$, except for the $\mathbf{Q}$-deterministic nodes that are ancestors of n', which output the child node whose context agrees with $\boldsymbol{q}'$.

Lemma 1. Let $\mathcal{C}$ be a PC over variables $\mathbf{X}$ and $\mathcal{M}$ be its tree-unrolled max-sum circuit (as described above) for a set of query variables $\mathbf{Q}$. For any $\mathcal{M}^{(n')}$ constructed from $\mathcal{M}$ as above, the following statements hold:

1. $\mathcal{M}^{(n')}(\emptyset) = \sum_{\boldsymbol{y} \in \mathsf{val}(\mathbf{X} \setminus \mathbf{Q})} \mathcal{M}^{(n')}(\boldsymbol{y})$.
2. For any $\boldsymbol{q} \in \gamma_{n'}|_{\mathbf{Q}}$, $\mathcal{M}^{(n')}(\emptyset) \geq \mathcal{C}(\boldsymbol{q})$.

Note that above statements also apply to $\mathcal{M} = \mathcal{M}^{(\text{root})}$. We now provide a proof of Proposition 1 using above lemma, which we will prove at the end of this section.

Proof. We will show that for every node n in $\mathcal{C}$, Algorithm 2 returns

$$\mathsf{r}_n \geq \max_{n' \in \text{copy}(n)} \mathcal{M}^{(n')}(\emptyset), \tag{4}$$

and for every edge (n, c) it returns

$$\mathsf{r}_{n,c} \geq \max_{(n',c') \in \text{copy}((n,c))} \mathcal{M}^{(c')}(\emptyset). \tag{5}$$

Note that Equation 5 implies that $\mathsf{r}_{n,c}$ upper-bounds the quantity $\mathsf{MMAP}(\mathbf{Q}|_{(n,c)})$ given by Equation 1:

$$\begin{aligned}
\max_{(n',c') \in \text{copy}((n,c))} \mathcal{M}^{(c')}(\emptyset) &\geq \max_{(n',c') \in \text{copy}((n,c))} \max_{\boldsymbol{q} \in \gamma_{c'}|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) = \max_{(n',c') \in \text{copy}((n,c))} \max_{\boldsymbol{q} \in (\gamma_{n'} \cap \gamma_{c'})|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) \\
&= \max_{\boldsymbol{q} \in \bigcup_{(n',c') \in \text{copy}((n,c))} (\gamma_{n'} \cap \gamma_{c'})|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) = \max_{\boldsymbol{q} \in (\gamma_n \cap \gamma_c)|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) = \max_{\boldsymbol{q} \in \gamma_{(n,c)}\big|_{\mathbf{Q}}} \mathcal{C}(\boldsymbol{q}) \\
&= \max_{\boldsymbol{q}: (n,c) \in \mathcal{C}'_{\boldsymbol{q}}} \mathcal{C}(\boldsymbol{q}) = \mathsf{MMAP}(\mathbf{Q}|_{(n,c)})
\end{aligned}$$

We will now prove that Equations 4 and 5 hold by induction. For the base case, r_{root} is set as m_{root}, which is exactly $\mathcal{M}(\emptyset) = \mathcal{M}^{(\text{root})}(\emptyset)$.

Next, assume Equation 4 holds for a node n in $\mathcal{C}$, and we want to show that Equation 5 holds for any of its input edges (n, c). If n is a product unit or a sum unit that is not $\mathbf{Q}$-deterministic, for any edge (n, c) and its copy (n', c') the circuits $\mathcal{M}^{(n')}$ and $\mathcal{M}^{(c')}$ are identical by definition. Then Equation 5 holds as follows:

$$\max_{(n',c') \in \text{copy}((n,c))} \mathcal{M}^{(c')}(\emptyset) = \max_{(n',c') \in \text{copy}((n,c))} \mathcal{M}^{(n')}(\emptyset) = \max_{n' \in \text{copy}(n)} \mathcal{M}^{(n')}(\emptyset) \leq \mathsf{r}_n = \mathsf{r}_{n,c}.$$

If n is a $\mathbf{Q}$-deterministic sum node, the circuits $\mathcal{M}^{(n')}$ and $\mathcal{M}^{(c')}$ can differ only by whether node n' is fixed to take c'. Thus, for any $\boldsymbol{y} \notin \gamma_{n'}|_{\mathbf{Y}}$ where $\mathbf{Y} = \mathbf{X} \setminus \mathbf{Q}$, $\mathcal{M}^{(n')}(\boldsymbol{y}) = \mathcal{M}^{(c')}(\boldsymbol{y})$. For $\boldsymbol{y} \in \gamma_{n'}|_{\mathbf{Y}}$, we have

$$\mathcal{M}^{(n')}(\boldsymbol{y}) - \mathcal{M}^{(c')}(\boldsymbol{y}) = \Big(\prod_{\theta \in \mathsf{path}(n')} \theta \Big) \cdot \mathcal{M}^{(n')}_{n'}(\boldsymbol{y}) - \Big(\prod_{\theta \in \mathsf{path}(n')} \theta \Big) \cdot \theta_{n',c'} \cdot \mathcal{M}^{(c')}_{c'}(\boldsymbol{y})$$

where $\mathsf{path}(n')$ denotes the set of all edge parameters that appear in the path from root to node n'. Note that $\mathcal{M}^{(n')}_{n'}$, i.e. the subcircuit of $\mathcal{M}^{(n')}$ rooted at n', is identical to $\mathcal{M}_{n'}$ as the two max-sum circuits differ only in the ancestors of n'. Similarly, $\mathcal{M}^{(c')}_{c'}$ is equal to $\mathcal{M}_{c'}$. Then we can express the circuit evaluation of $\mathcal{M}^{(c')}$ as

$$\begin{aligned}
\mathcal{M}^{(c')}(\emptyset) &= \sum_{\boldsymbol{y} \in \mathsf{val}(\mathbf{Y})} \mathcal{M}^{(c')}(\boldsymbol{y}) = \sum_{\boldsymbol{y} \notin \gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(c')}(\boldsymbol{y}) + \sum_{\boldsymbol{y} \in \gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(c')}(\boldsymbol{y}) \\
&= \sum_{\boldsymbol{y} \notin \gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(n')}(\boldsymbol{y}) + \sum_{\boldsymbol{y} \in \gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(c')}(\boldsymbol{y}) = \mathcal{M}^{(n')}(\emptyset) - \sum_{\boldsymbol{y} \in \gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(n')}(\boldsymbol{y}) + \sum_{\boldsymbol{y} \in \gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}^{(c')}(\boldsymbol{y}) \\
&= \mathcal{M}^{(n')}(\emptyset) + \left(\prod_{\theta \in \mathsf{path}(n')} \theta \right) \left(\theta_{n',c'} \sum_{\boldsymbol{y} \in \gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}_{c'}(\boldsymbol{y}) - \sum_{\boldsymbol{y} \in \gamma_{n'}|_{\mathbf{Y}}} \mathcal{M}_{n'}(\boldsymbol{y}) \right) \\
&= \mathcal{M}^{(n')}(\emptyset) + \Big(\prod_{\theta \in \mathsf{path}(n')} \theta \Big) \left(\theta_{n',c'} \mathcal{M}_{c'}(\emptyset) - \mathcal{M}_{n'}(\emptyset) \right) = \mathcal{M}^{(n')}(\emptyset) + \Big(\prod_{\theta \in \mathsf{path}(n')} \theta \Big) \left(\theta_{n,c} \mathsf{m}_c - \mathsf{m}_n \right)
\end{aligned}$$

Because $\mathsf{m}_n \geq \theta_{n,c} \mathsf{m}_c$, above equation among copies of (n, c) can be bounded from above by:

$$\max_{(n',c') \in \text{copy}((n,c))} \mathcal{M}^{(c')}(\emptyset) \leq \max_{n' \in \text{copy}(n)} \mathcal{M}^{(n')}(\emptyset) + \Big(\min_{n' \in \text{copy}(n)} \prod_{\theta \in \mathsf{path}(n')} \theta \Big) \left(\theta_{n,c} \mathsf{m}_c - \mathsf{m}_n \right).$$

We will show that $\mathsf{r}_{n,c} = \mathsf{r}_n + \mathsf{t}_n \left(\theta_{n,c} \mathsf{m}_c - \mathsf{m}_n \right)$ (Line 12 in Algorithm 2) is at most the right-hand side quantity of above inequality, thereby satisfying Equation 5. First, we have $\mathsf{r}_n \geq \max_{(n') \in \text{copy}(n)} \mathcal{M}^{(n')}(\emptyset)$ by the inductive

Algorithm 4 LOWER-BOUND($\mathcal{C}, \mathbf{Q}$)

Input: a PC $\mathcal{C}$ over variables $\mathbf{X}$ and a set of query variables $\mathbf{Q} \subset \mathbf{X}$
Output: an assignment $\boldsymbol{q} \in \mathsf{val}(\mathbf{Q})$

$\mathsf{N} \leftarrow$ FEEDFORWARDORDER($\mathcal{C}$)
for each $n \in \mathsf{N}$ **do**
  if n is an input unit **then** $\mathsf{m}_n \leftarrow 1.0$
  else if n is a product unit **then** $\mathsf{m}_n \leftarrow \prod_{c \in \mathsf{ch}(n)} \mathsf{m}_c$
  else if n or its descendant is $\mathbf{Q}$-deterministic **then** $\mathsf{m}_n \leftarrow \max_{c \in \mathsf{ch}(n)} \theta_{n,c} \mathsf{m}_c$
  else $\mathsf{m}_n \leftarrow \sum_{c \in \mathsf{ch}(n)} \theta_{n,c} \mathsf{m}_c$
return EXTRACT-STATE($\mathcal{C}, \mathbf{Q}, \mathsf{m}$)

procedure EXTRACT-STATE(n,$\mathbf{Q}$,m)
  if n is an input unit **then**
    if Variable(n) $\in \mathbf{Q}$ **then return** {Literal(n)} **else return** {}
  else if n is a product unit **then**
    return $\bigcup_{c \in \mathsf{ch}(n)}$ EXTRACT-STATE($c, \mathbf{Q}, \mathsf{m}$)
  else
    return EXTRACT-STATE($\arg\max_{c \in \mathsf{ch}(n)} \theta_{n,c} \mathsf{m}_c, \mathbf{Q}, \mathsf{m}$)

hypothesis. Next, we want to show that $\mathsf{t}_n \leq \min_{n' \in \mathrm{copy}(n)} \prod_{\theta \in \mathsf{path}(n')} \theta$. For a given node c, suppose this holds for t_n of every parent node $n \in \mathsf{pa}(c)$. Then we have

$$\mathsf{t}_c = \min_{n \in \mathsf{pa}(c)} \theta_{n,c} \mathsf{t}_n \leq \min_{n \in \mathsf{pa}(c)} \theta_{n,c} \Big(\min_{n' \in \mathrm{copy}(n)} \prod_{\theta \in \mathsf{path}(n')} \theta \Big) = \min_{c' \in \mathrm{copy}(c)} \prod_{\theta \in \mathsf{path}(c')} \theta$$

For simplicity, we say $\theta_{n,c} = 1$ for a product node n.

Finally, assume that Equation 5 holds for edges (p, n) where $p \in \mathsf{pa}(n)$, and we will show that Equation 4 must hold then for node n. r_n, which is set to $\max_{p \in \mathsf{pa}(n)} \mathsf{r}_{p,n}$ in Algorithm 2, satisfies Equation 4 as follows:

$$\max_{p \in \mathsf{pa}(n)} \mathsf{r}_{p,n} \geq \max_{p \in \mathsf{pa}(n)} \max_{(p', n') \in \mathrm{copy}((p,n))} \mathcal{M}^{(n')}(\emptyset) = \max_{n' \in \mathrm{copy}(n)} \mathcal{M}^{(n')}(\emptyset).$$

This concludes the proof of Proposition 1. □

Proof of Lemma 1. To show property (1) $\mathcal{M}^{(n')}(\emptyset) = \sum_{\boldsymbol{y} \in \mathsf{val}(\mathbf{X} \setminus \mathbf{Q})} \mathcal{M}^{(n')}(\boldsymbol{y})$, first observe that $\mathcal{M}^{(n')}$ fixes every $\mathbf{Q}$-deterministic node to always return the value of one of its children and thus can be simplified by removing those nodes and directly connecting its parent to the appropriate child node. This results in a smooth and decomposable PC with the normal types of sum and product nodes. Then (1) simply holds by the fact that smooth and decomposable PCs allow marginal inference by feedforward evaluation.

Property (2) $\mathcal{M}^{(n')}(\emptyset) \geq \mathcal{C}(\boldsymbol{q})$ holds for any $\boldsymbol{q} \in \gamma_{n'}|_{\mathbf{Q}}$ if and only if $\mathcal{M}^{(n')}(\emptyset) \geq \mathcal{C}'_{\boldsymbol{q}}(\emptyset)$, as computing the marginal probability of $\boldsymbol{q}$ is equivalent to evaluating the $\boldsymbol{q}$-subcircuit. Note that for any $\boldsymbol{q} \in \gamma_{n'}|_{\mathbf{Q}}$, the ancestor nodes of n' in $\mathcal{M}^{(n')}$ are equivalent to those in the $\boldsymbol{q}$-subcircuit. On the other hand, $\mathcal{M}^{(n')}_{n'}(\emptyset) = \mathcal{M}_{n'}(\emptyset)$ upper bounds $\max_{\boldsymbol{q} \in \gamma_{n'}|_{\mathbf{Q}}} n(\boldsymbol{q})$ (recall Equation (2)), hence must be at least $n(\boldsymbol{q})$. Therefore, at the root nodes, $\mathcal{M}^{(n')}$ must evaluate to at least $\mathcal{C}'_{\boldsymbol{q}}$. □

A.3 MMAP Lower Bound

As mentioned in Section 4, the solver maintains a lower bound on marginal MAP to be used for pruning. We now describe the algorithm to compute the lower bound used in our iterative solver. First, note that the probability of any assignment to query variables can be used as a lower bound for marginal MAP by definition. A simple and common approach to approximate the marginal MAP state is to solve MPE instead and reduce the MPE state to the query variables. We use a similar approach but with a key additional guarantee: after splitting on all query variables, it exactly solves the marginal MAP problem. A pseudocode of our method is shown in Algorithm 4.

Note the similarity of its feedforward pass to Algorithm 1: they both evaluate the circuit while replacing some sum nodes to take the weighted maximum. However, our algorithm not only replaces the $\mathbf{Q}$-deterministic sum nodes but all of their ancestors as well. This is so that we can extract a state $\boldsymbol{q}$ by a backward pass, following the edges that were selected by the weighted maximum. Moreover, if the input PC $\mathcal{C}$ is $\mathbf{Q}$-deterministic, this algorithm behaves the same as *Algorithm* 1 and exactly solves the MMAP problem.

B Split Heuristics

This section describes the two variable split heuristics that were evaluated in Section 5.

Using the (Pruned) heuristic, at every iteration we split on the query variable that had the most number of associated edges pruned. In other words, for each query variable $Q \in \mathbf{Q}$ that is yet to be split on, we count how many edges of a Q-deterministic sum node have been pruned (this value can be cached to minimize redundant calculations) and choose the variable with the highest count. Intuitively, using this heuristic would tend to minimize a size blow-up by each split.

On the other hand, (UB) aims to maximize opportunities for pruning in the iteration following each split. To compute the heuristic, we first compute for each query variable $Q \in \mathbf{Q}$ the MMAP upper-bounds as described in Algorithm 1, one setting $Q = 0$ as evidence and the other $Q = 1$. Because splitting the root on Q would introduce a deterministic sum node whose children set Q to 0 and 1, these bounds equal the edge bounds on the two input edges to the root after splitting. Let us denote these bounds $B_{Q=0}$ and $B_{Q=1}$ respectively, the lower bound in the current iteration as lb, and the candidate query variables by $\mathbf{Q}' \subseteq \mathbf{Q}$ (i.e. query variables that have not been split on in the previous iterations). Then the (UB) heuristic selects a variable as follows:

$$\begin{cases} \arg\min_{Q:\min(B_{Q=0},B_{Q=1})<lb} \max(B_{Q=0}, B_{Q=1}) & \text{if } \exists Q \in \mathbf{Q}' \text{ s.t. } \min(B_{Q=0}, B_{Q=1}) < lb, \\ \arg\min_{Q \in \mathbf{Q}'} B_{Q=0} + B_{Q=1} & \text{otherwise.} \end{cases}$$

In other words, if any variable would have a corresponding edge bound drop below the lower bound, we prioritize selecting from those variables as this guarantees a large part of the circuit is pruned in the next iteration. Then we choose the variable that would decrease the upper bound the most, which would, intuitively, result in more edges being pruned in the next iteration. Note that computing this heuristic requires additional passes through the PC, but as we showed empirically in Section 5, it makes pruning much more effective and the resulting solver more efficient, despite the added time to compute the heuristic.