The Sum-Product Theorem: A Foundation for Learning Tractable Models (Supplementary Material)

Abram L. Friesen AFRIESEN@CS.WASHINGTON.EDU
Pedro Domingos PEDROD@CS.WASHINGTON.EDU
Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA

A. Decomposable SPF summation complexity

Let $S(\mathbf{X})$ be a decomposable SPF with size $|S|$ on commutative semiring $(R, \oplus, \otimes, 0, 1)$, let $d = |\mathcal{X}_i|$ for all $X_i \in \mathbf{X}$ where $\mathbf{X} = (X_1, \dots, X_n)$, and let the cost of $a \oplus b$ and $a \otimes b$ for any elements $a, b \in R$ be c. Further, let e denote the complexity of evaluating any unary leaf function $\phi_j(X_i)$ in S and let $k = \max_{v \in S_{\text{sum}}, j \in \text{Ch}(v)} |\mathbf{X}_v \backslash \mathbf{X}_j| < n$, where $S_{\text{sum}}, S_{\text{prod}}$, and S_{leaf} are the sum, product, and leaf nodes in S, respectively, and $\text{Ch}(v)$ are the children of v. Then the complexity of computing $\bigoplus_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x})$ is $|S| \cdot c + |S_{\text{leaf}}| \cdot d(e + c) + |S_{\text{sum}}| \cdot (c + kdc)$.

For certain simple SPFs that have very little internal structure and many input variables, the worst case complexity of summing S can be quadratic in $|S|$ and occurs in the rare and restrictive case where $k = O(n) = O(|S|)$, due to the $\bigoplus_{\mathcal{X}_{v \backslash i}} 1$ term at each sum node (see proof of Theorem 1). However, in any semiring with an idempotent sum (i.e., $a \oplus a = a$ for every $a \in R$) such as the min-sum or max-product semirings, this term is always equal to 1 and thus no computation is necessary. Alternatively, if the semiring supports multiplication and division as in the sum-product semiring then this complexity can be reduced by first computing the product over all variables and then dividing out as needed. If the semiring has neither of these properties, these identity summations can still be computed with a single preprocessing pass through the SPF since they are constants and independent of the input variables. For all semirings we've studied, this quadratic cost does not occur, but we include it for completeness.

B. Logical inference (continued)

Model counting. Model counting (#SAT) is the problem of computing the number of satisfying assignments of a Boolean formula. The model count of an NNF F can be obtained by *translating* it from the Boolean semiring to the counting sum-product semiring $\mathcal{P} = (\mathbb{N}, +, \times, 0, 1)$ ($\mathbb{R}_+$ is used instead for weighted #SAT), and then summing it.

Definition 8. Translating *an SPF from semiring* $(R, \oplus, \otimes, 0, 1)$ *to semiring* $(R', \boxplus, \boxtimes, 0', 1')$ *with* $R \subseteq R'$, *involves replacing each* $\oplus$ *node with a* $\boxplus$ *node, each* $\otimes$ *node with a* $\boxtimes$ *node, and each leaf function that returns* 0 *or* 1 *with one that returns* $0'$ *or* $1'$*, respectively.*

However, simply summing the translated function F' may compute an incorrect model count because the same satisfying assignment may be counted multiple times; this occurs when the idempotence (a semiring R is idempotent if $a \oplus a = a$ for $a \in R$) of the two semirings differs, i.e., either semiring R is idempotent and R' is not, or vice versa. If exactly one of the two semirings is idempotent, F must be *deterministic* to ensure that summing F' gives the correct model count.

Definition 9. *An OR node is* deterministic *iff the supports of its children are disjoint. An NNF is deterministic iff all of its OR nodes are deterministic.*

The support of a function $G(\mathbf{X})$ is the set of points $\mathcal{S} \subseteq \mathcal{X}$ such that $G(\mathbf{x}) \neq 0$ for all $\mathbf{x} \in \mathcal{S}$. If F is deterministic and decomposable, then it follows from the sum-product theorem that its model count can be computed efficiently.

Corollary 8 (Darwiche, 2001a). *The model count of a deterministic, decomposable NNF can be computed in time linear in its size.*

Most algorithms for #SAT (e.g., Relsat (Bayardo Jr. & Pehoushek, 2000)), Cachet (Sang et al., 2004), #DPLL (Bacchus et al., 2009)) are also instances of SUMSPF, since they extend DPLL by, at each level of recursion, decomposing the CNF into independent components (i.e., no variable appears in multiple components), solving these separately, and caching the model count of each component. Component decomposition corresponds to a decomposable product node in SUMSPF and component caching corresponds to connecting a sub-SPF to multiple parents. Notice that the sum nodes created by DECOMPOSE are deterministic.

MAX-SAT. MAX-SAT is the problem of computing the maximum number of satisfiable clauses of a CNF, over all assignments. It can be generalized to NNFs as follows.

Definition 10. *Let* $F(\mathbf{X})$ *be an NNF and* $\mathbf{x} \in \mathcal{X}$ *an assignment. The* SAT number *(SN) of a literal* $\phi(X_j) \in F$ *is* 1 *if* $\phi(\mathbf{x}_j)$ *is true and 0 otherwise. The SN of an AND node is the sum of the SNs of its children. The SN of an OR node is the max of the SNs of its children.*

MAX-SAT of an NNF $F(\mathbf{X})$ is the problem of computing the maximum SAT number of the root of F over all assignments $\mathbf{x} \in \mathcal{X}$. If F is a CNF, then this reduces to standard MAX-SAT. MAX-SAT of F can be solved by translating F to the max-sum semiring $\mathcal{M} = (\mathbb{N}_{-\infty}, \max, +, -\infty, 0)$ (where $\mathbb{R}_{+,-\infty}$ is used for weighted MAX-SAT), and then summing it. Clearly, F' is an SPF on $\mathcal{M}$, i.e., a max-sum network. The corollary below follows immediately from the sum-product theorem.

Corollary 9 (Darwiche, 2001b). *MAX-SAT of a decomposable NNF can be computed in time linear in its size.*

MAX-SAT of an arbitrary NNF (or CNF) can be computed by first translating it to $\mathcal{M}$ and then calling SumSPF, which can be extended to perform branch and bound (BnB) (Lawler & Wood, 1966) when traversing the SPF. This allows SumSPF to prune sub-SPFs that are not relevant to the final solution, which can greatly reduce the search space. With this addition, DPLL-based BnB solvers for MAX-SAT (e.g., Heras et al. (2008) and references therein) are instances of SumSPF. Most relevant, however, is the MPE-SAT algorithm of Sang et al. (2007), since both it and SumSPF use decomposition and caching to improve their efficiency.

C. Probabilistic inference (continued)

Marginal inference (continued). Tree-based methods include junction-tree clustering (Lauritzen & Spiegelhalter, 1988) and variable elimination (Dechter, 1999), which correspond (explicitly and implicitly, respectively) to constructing a junction tree and then summing its corresponding tree-like SPN. Conditioning algorithms such as recursive conditioning (Darwiche, 2001c), value elimination (Bacchus et al., 2002), AND/OR search (Dechter & Mateescu, 2007), and #DPLL (Bacchus et al., 2009) traverse the space of partial assignments by recursively conditioning on variables and their values. These algorithms vary in the flexibility of their variable ordering, decomposition, and caching (see Bacchus et al. (2009) for a comparison), but are all instances of SumSPF, which can use a fully-dynamic variable ordering, as value elimination can and #DPLL does, or a fixed ordering, as in variants of recursive conditioning and AND/OR search. Decomposition and caching correspond to decomposable product nodes and connecting sub-SPNs to multiple parents, respectively, in SumSPF. Thirdly, inference in graphical models can be performed by compilation to an arithmetic circuit (AC) (Darwiche, 2003). In discrete domains, Rooshenas & Lowd (2014) showed that SPNs and ACs are equivalent, but that SPNs are always smaller or equal in size. In continuous domains, however, it is unlikely that even this relationship exists, because a AC would require an infinite number of indicator functions. Furthermore, existing compilation methods require first encoding the graphical model in very restrictive languages (such as CNF or SDDs), which can make them exponentially slower than SumSPF. Finally, no tractability properties have been established for ACs so there is no guarantee before compiling that inference will be tractable, nor have they been generalized to other semirings.

MPE. Beyond computing the probability of evidence, another key probabilistic inference problem is finding the most probable or MPE state of the non-evidence variables of $P(\mathbf{X})$ given the evidence, $\arg\max_{\mathcal{X}_{\overline{E}}} P(\mathbf{e}, \mathbf{X}_{\overline{E}})$ for evidence $\mathbf{e} \in \mathcal{X}_E$ where $\mathbf{X}_{\overline{E}} = \mathbf{X} \backslash \mathbf{X}_E$. The MPE value (maximum probability of any state) of an SPN S can be computed by translating S to the non-negative max-product semiring $(\mathbb{R}_+, \max, \times, 0, 1)$ and maximizing the resulting SPF S'. The MPE state can then be recovered by a downward pass in S', recursively selecting the (or a) highest-valued child of each max node and all children of each product node (Poon & Domingos, 2011). As when translating an NNF for model counting, an SPN must be *selective* (Peharz et al., 2014) (the SPN equivalent of deterministic) for summation in the max-product semiring to give the correct MPE.

Corollary 10. *The MPE state of a selective, decomposable SPN can be found in time linear in its size.*

A sum node in an SPN can be viewed as the result of summing out an implicit hidden variable Y_v, whose values $\mathcal{Y}_v = \{y_c\}_{c \in \text{Ch}(v)}$ correspond to $\text{Ch}(v)$, the children of v (Poon & Domingos, 2011). It is often of interest to find the MPE state of both the hidden and observed variables. This can be done in linear time and requires only that the SPN be decomposable, because making each Y_v explicit by multiplying each child c of v by the indicator $[Y_v = y_c]$ makes the resulting SPN $S(\mathbf{X}, \mathbf{Y})$ selective.

D. Integration and optimization

Integration (continued). For non-decomposable SPFs, Decompose must be altered to select only a finite number of values and then use the trapezoidal rule for approximate integration. Values can be chosen using grid search and if S is Lipschitz continuous the grid spacing can be set such that the error incurred by the approximation is bounded by a pre-specified amount. This can significantly reduce the number of values explored in SumSPF if combined with approximate decomposability (Section 4), since SumSPF can treat some non-decomposable product nodes as decomposable, avoiding the expensive call to Decompose while incurring only a bounded integration error.

E. Relational inference

Let $\mathcal{X}$ be a finite set of constants and let $R^k = \mathcal{X}^k$ be the complete relation[1] of arity k on $\mathcal{X}$, i.e., the set of all tuples in $\mathcal{X}^k$, where $\mathcal{X}^k$ is the Cartesian product of $\mathcal{X}$ with itself $k-1$ times. The universe of relations with arity up to m is $\mathbf{U}_m = \mathbf{1}_R \cup \bigcup_{i=1}^m 2^{R^i}$, where 2^{R^k} is the power set of R^k and $\mathbf{1}_R$ is the (identity) relation containing the empty tuple. Since union distributes over join, and both are associative and commutative, $\mathcal{R} = (\mathbf{U}_m, \cup, \bowtie, \varnothing, \mathbf{1}_R)$ is a semiring over relations, where $\bowtie$ is natural join and $\varnothing$ is the empty set (recall that $R = R \bowtie \mathbf{1}_R$ for any relation R). Given an extensional database $\mathbf{R} = \{R_i\}$ containing relations R_i of arity up to m, an SPF on $\mathcal{R}$, referred to as a union-join network (UJN), is a query on $\mathbf{R}$. In a UJN Q, each $R_i \in \mathbf{R}$ is composed of a union of joins of unary tuples, such that $R_i = \bigcup_{\langle c_1,\dots,c_r\rangle \in R_i} \bowtie_{j=1}^r c_j$, where the leaves of Q are the unary tuples c_j. The R_i are then combined with unions and joins to form the full UJN (query). A UJN $Q(\mathbf{X})$ over query variables $\mathbf{X} = (X_1, \dots, X_n)$ defines an intensional output relation $Q_{\text{ans}} = \bigcup_{\mathcal{X}^n} Q(\mathbf{X})$. Clearly, computing Q_{ans} corresponds to summation in $\mathcal{R}$. Let n_J^Q denote the maximum number of variables involved in a particular join over all joins in Q. The corollary below follows immediately, since a decomposable join is a Cartesian product.

Corollary 11. *Q_{ans} of a decomposable UJN Q can be computed in time linear in the size of Q if n_J^Q is bounded.*

Note that n_J^Q can be smaller than the treewidth of Q, since Q composes the final output relation from many small relations (starting with unary tuples) via a relational form of determinism. Since the size of a UJN depends both on the input relations and the query, this is a statement about the combined complexity of queries defined by UJNs.

Regarding expressivity, selection in a UJN can be implemented as a join with the relation $P_\sigma \in \mathbf{U}_m$, which contains all tuples that satisfy the selection predicate σ. Projection is not immediately supported by $\mathcal{R}$, but since union distributes over projection, it is straightforward to extend the results of Theorem 1 to allow UJNs to contain projection nodes. UJNs with projection correspond to non-recursive Datalog queries (i.e., unions of conjunctive queries), for which decomposable UJNs are a tractable sub-class. Thus, SUMSPF defines a recursive algorithm for evaluating non-recursive Datalog queries and the Generic-Join algorithm (Ngo et al., 2014) – a recent join algorithm that achieves worst-case optimal performance by recursing on individual tuples – is an instance of DECOMPOSE.

Another consequence of the sum-product theorem is a much simpler proof of the tractability of tractable Markov logic (Domingos & Webb, 2012).

F. Relational probabilistic models

A tractable probabilistic knowledge base (TPKB) (Niepert & Domingos, 2015; Webb & Domingos, 2013; Domingos & Webb, 2012) is a set of class and object declarations such that the classes form a forest and the objects form a tree of subparts when given the leaf class of each object. A class declaration for a class $\mathtt{C}$ specifies the subparts $\mathtt{Parts(C)} = \{\mathtt{P_i}\}$, (weighted) subclasses $\mathtt{Subs(C)} = \{\mathtt{S_i}\}$, attributes $\mathtt{Atts(C)} = \{\mathtt{A_i}\}$, and (weighted) relations $\mathtt{Rels(C)} = \{\mathtt{R_i}\}$. The subparts of $\mathtt{C}$ are parts that every object of class $\mathtt{C}$ must have and are specified by a name $\mathtt{P_i}$, a class $\mathtt{C_i}$, and a number n_i of unique copies. A class $\mathtt{C}$ with subclasses $\mathtt{S_1}, \dots, \mathtt{S_j}$ must belong to exactly one of these subclasses, where the weights w_i specify the distribution over subclasses. Every attribute has a domain $\mathbf{D_i}$ and a weight function $\mathbf{u_i} : \mathbf{D_i} \to \mathbb{R}$. Each relation $\mathtt{R_i}(\dots)$ has the form $\mathtt{R_i}(\mathtt{P_a}, \dots, \mathtt{P_z})$ where each of $\mathtt{P_a}, \dots, \mathtt{P_z}$ is a part of $\mathtt{C}$. Relations specify what relationships may hold among the subparts. A weight v_i on $\mathtt{R_i}$ defines the probability that the relation is true. A relation can also apply to the object as a whole, instead of to its parts. Object declarations introduce evidence by specifying an object's subclass memberships, attribute values, and relations as well as specifying the names and path of the object from the top object in the part decomposition.

A TPKB $\mathcal{K}$ is a DAG of objects and their properties (classes, attributes, and relations), and a possible world $\mathbf{W}$ is a subtree of the DAG with values for the attributes and relations. The literals are the class membership, attribute, and relation atoms and their negations and thus specify the subclasses of each object, the truth value of each relation, and the value of each attribute. A single (root) top object $(\mathtt{O_0}, \mathtt{C_0})$ has all other objects as descendants. No other objects are of top class $\mathtt{C_0}$. The unnormalized distribution ϕ over possible subworlds $\mathbf{W}$ is defined recursively as $\phi(\mathtt{O}, \mathtt{C}, \mathbf{W}) = 0$ if $\neg\mathtt{Is(O,C)} \in \mathbf{W}$ or if a relation $\mathtt{R}$ of $\mathtt{C}$ is hard and $\neg\mathtt{R}(\mathtt{O}, \dots) \in \mathbf{W}$, and otherwise as

$$\begin{aligned}\phi(\mathtt{O}, \mathtt{C}, \mathbf{W}) = &\left(\sum_{\mathtt{S_i} \in \mathtt{Subs(C)}} e^{w_i} \phi(\mathtt{O}, \mathtt{S_i}, \mathbf{W})\right) \times \\ &\left(\prod_{\mathtt{P_i} \in \mathtt{Parts(C)}} \phi(\mathtt{O.P_i}, \mathtt{C_i}, \mathbf{W})\right) \times \\ &\left(\prod_{\mathtt{A_i} \in \mathtt{Atts(C)}} \alpha(\mathtt{O}, \mathtt{A_i}, \mathbf{W})\right) \times \\ &\left(\prod_{\mathtt{R_i} \in \mathtt{Rels(C)}} \rho(\mathtt{O}, \mathtt{R_i}, \mathbf{W})\right), \end{aligned} \qquad (1)$$

where $\alpha(\mathtt{O}, \mathtt{A_i}, \mathbf{W}) = e^{\mathbf{u_i}(D)}$ if $\mathtt{A_i}(\mathtt{O}, \mathtt{D}) \in \mathbf{W}$ and $\rho(\mathtt{O}, \mathtt{R_i}, \mathbf{W}) = e^{v_i}[\mathtt{R_i}(\dots)] + [\neg\mathtt{R_i}(\dots)]$. Note that

[1] A relation is a set of tuples; see Abiteboul et al. (1995) for details on relational databases.

$\texttt{Parts(C)}$ contains all subparts of $\texttt{C}$, including all duplicated parts. The probability of a possible world $\mathbf{W}$ is $\frac{1}{Z_\mathcal{K}}\phi(\texttt{O}_0, \texttt{C}_0, \mathbf{W})$ where the sub-partition function for $(\texttt{O}, \texttt{C})$ is $Z_{\mathcal{K}_{\texttt{O},\texttt{C}}} = \sum_{\mathbf{W}\in\mathcal{W}} \phi(\texttt{O}, \texttt{C}, \mathbf{W})$ and $Z_\mathcal{K} = Z_{\mathcal{K}_{\texttt{O}_0,\texttt{C}_0}}$.

By construction, $\phi(\texttt{O}_0, \texttt{C}_0, \mathbf{W})$ defines an SPN over the literals. With the sum-product theorem in hand, it is possible to greatly simplify the two-page proof of tractability given in Niepert & Domingos (2015), as we show here. To prove that computing $Z_\mathcal{K}$ is tractable it suffices to show that (1) is decomposable or can be decomposed efficiently. We first note that each of the four factors in (1) is decomposable, since the first is a sum, the second is a product over the subparts of O and therefore its subfunctions have disjoint scopes, and the third and fourth are products over the attributes and relations, respectively, and are decomposable because none of the α or ρ share variables. It only remains to show that the factors can be decomposed with respect to each other without increasing the size of the SPN. Let n_O, n_C, n_r denote the number of object declarations, class declarations, and relation rules, respectively. The SPN corresponding to $\phi(\texttt{O}_0, \texttt{C}_0, \mathbf{W})$ has size $O(n_O(n_C+n_r))$, since for each object $(\texttt{O}, \texttt{C})$ there are a constant number of edges for each of its relations and subclasses. Similarly, $\mathcal{K}$ has size $|\mathcal{K}| = O(n_O(n_C + n_r))$. We can thus prove the following result.

Corollary 12. *The partition function of a TPKB can be computed in time linear in its size.*

G. Experiment details

All experiments were run on the same MacBook Pro with 2.2 GHz Intel Core i7 processor with 16 GB of RAM. Each optimization was limited to a single thread.

The non-separable variant of the Rastrigin function (Torn & Zilinskas, 1989) used on pairs of variables is defined as

$$\begin{aligned} f^R_{x_i,x_j}(Y_i, Y_j) = \; & c_0[(Y_i - x_i)^2 - (Y_j - x_j)^2] + \\ & c_1 - c_1 \cos(Y_i - x_i)\cos(Y_j - x_j), \end{aligned}$$

which has a global minimum at $\mathbf{y}^* = (x_i, x_j)$ with value $f^R_\mathbf{x}(\mathbf{y}^*) = 0$. The constants c_0 and c_1 control the shape of the quadratic basin and the amplitude of the sinusoids, respectively. For our tests, we used $c_0 = 0.1$ and $c_1 = 20$. Figure 1 shows a contour plot of f^R.

Omitting the parameters $\mathbf{x}$ from $f^R_\mathbf{x}$ for simplicity, the full test function for $n = 4m$ variables is defined as

$$F_\mathbf{x}(\mathbf{Y}) = \sum_{i=0}^{m} f^R(Y_{4i}, Y_{4i+k}) + f^R(Y_{4i+3}, Y_{4i+3-k}),$$

where $k = 1$ with probability 0.5 and $k = 2$ otherwise. This creates a function that is non-decomposable between each pair of variables (Y_{4i}, Y_{4i+k}) and (Y_{4i+3}, Y_{4i+3-k}). For the simplest case with $n = 4$ variables, if $k = 1$ then pairs (Y_0, Y_1) and (Y_2, Y_3) are non-decomposable. Alternatively, if $k = 2$ then pairs (Y_0, Y_2) and (Y_1, Y_3) are non-decomposable. The global minimum (x_i, x_j) for each function $f^R_{x_i,x_j}(Y_i, Y_j)$ was sampled uniformly over an interval of length 2 from the line $Y_i = Y_j$ with zero-mean additive Gaussian noise ($\sigma = 0.1$). Thus, each instance of $F_\mathbf{x}$ is highly nonconvex and is decomposable with respect to certain variables and not with respect to others. For a set of instances of $F_\mathbf{x}$, there is structure in the decomposability between variables, but different instances have different decomposability structure, so LEARNSPF must first group those function instances that have similar decomposability structure and then identify that structure in order to learn a min-sum function that is applicable to any instance in the training data.

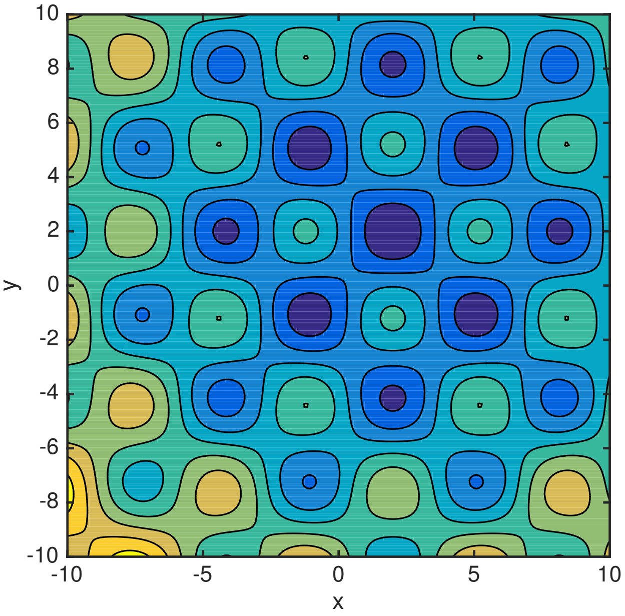

Figure 1. Contour plot of the 2-D nonconvex Rastrigin function.

H. Proofs

Corollary 1. *Every SPF with bounded treewidth can be summed in time linear in the cardinality of its scope.*

Proof. Let $\mathbf{X} = (X_1, \ldots, X_n)$, let $(R, \oplus, \otimes, 0, 1)$ be a commutative semiring, and let $F(\mathbf{X})$ be an SPF with bounded treewidth $tw(F) = a$ for $0 < a < \infty$. Let $S(\mathbf{X})$ be a tree-like SPF that is compatible with F and has junction tree $\mathcal{T} = (T, Q)$ with treewidth $tw(\mathcal{T}) = a$. The size of the largest cluster in $\mathcal{T}$ is $\alpha = a + 1$. Let $m = |Q| \leq n$ and $d = |\mathcal{X}_v|$ for all $X_v \in \mathbf{X}$. Further, other than the root $s \in S$, there is a one-to-one correspondence between separator instantiations $\mathbf{s}_{ij} \in \mathcal{X}_{\mathbf{S}_{ij}}$

and sum nodes $s_{ij} \in S$, and between cluster instantiations $\mathbf{c}_j \in \mathcal{X}_{\mathbf{C}_j}$ and product nodes $c_j \in S$. Now, the number of edges in S can be obtained by counting the edges that correspond to each edge in T and summing over all edges in T, as follows. By construction, each edge $(j,k) \in T$ corresponds to the product nodes $\{c_k\}$; their children, which are the leaf nodes (indicators and constants) and the sum nodes $\{s_{jk}\}$; and the children of $\{s_{jk}\}$, which are the product nodes $\{c_j\}$. By definition, the $\{c_j\}$ have only a single parent, so there are $|\mathcal{X}_{\mathbf{C}_j}| \leq d^\alpha$ edges between $\{s_{jk}\}$ and $\{c_j\}$. Further, each c_k has only $|\mathbf{C}_k| + 1$ leaf node children and $|\mathrm{Ch}(k)|$ sum node children, so there are $|\mathcal{X}_{\mathbf{C}_k}|(|\mathbf{C}_k|+1)(|\mathrm{Ch}(k)|) \leq d^\alpha(\alpha+1)(|\mathrm{Ch}(k)|)$ edges between $\{c_k\}$ and $\{s_{jk}\}$. In addition, there are also $\mathcal{X}_{\mathbf{C}_r} = d^\alpha$ edges between the root $s \in S$ and the product nodes c_r. Thus, since T is a tree with $m-1$ edges, $\mathrm{size}(S) \leq d^\alpha + \sum_{(j,k)\in T} 2d^\alpha(\alpha+1)(|\mathrm{Ch}(k)|) = O(md^\alpha)$, which is $O(n)$. Since S is decomposable and has size $O(n)$, then, from the sum-product theorem, S can be summed in time $O(n)$. Furthermore, S is compatible with F, so F can be summed in time $O(n)$, and the claim follows. □

Corollary 2. *Not every SPF that can be summed in time linear in the cardinality of its scope has bounded treewidth.*

Proof. By counterexample. Let $\mathbf{X} = (X_1, \dots, X_n)$ be a vector of variables, $(R, \oplus, \otimes, 0, 1)$ be a commutative semiring, and $k = |\mathcal{X}_i|$ for all $X_i \in \mathbf{X}$. The SPF $F(\mathbf{X}) = \bigoplus_{j=1}^r \bigotimes_{i=1}^n \psi_{ji}(X_i)$ can be summed in time linear in n because F is decomposable and has size $r(n+1)$. At the same time, $F(\mathbf{X})$ has treewidth $n-1$ (i.e., unbounded) because there are no pairwise-disjoint subsets $\mathbf{A}, \mathbf{B}, \mathbf{C} \subseteq \mathbf{X}$ with domains $\mathcal{X}_A, \mathcal{X}_B, \mathcal{X}_C$ such that $\mathbf{A}$ and $\mathbf{B}$ are conditionally independent in F given $\mathbf{C}$, and thus the smallest junction tree compatible with $F(\mathbf{X})$ is a complete clique over $\mathbf{X}$. This can be seen as follows. Without loss of generality, let $\mathbf{A} \cup \mathbf{B}$ be the first m variables in $\mathbf{X}$, $(X_1, \dots, X_m)$. For any $\mathbf{c} \in \mathcal{X}_C$, $F(\mathbf{A}, \mathbf{B}, \mathbf{c}) \propto \bigoplus_{j=1}^r \bigotimes_{i:X_i \in \mathbf{A}\cup\mathbf{B}} \psi_{ji}(X_i) = (\psi_{11}(X_1) \otimes \cdots \otimes \psi_{1m}(X_m)) \oplus \cdots \oplus (\psi_{r1}(X_1) \otimes \cdots \otimes \psi_{rm}(X_m))$. For $F(\mathbf{A}, \mathbf{B}, \mathbf{c})$ to factor, the terms in the right-hand side must have common factors; however, in general, each ψ_{ji} is different, so there are no such factors. Thus, $F(\mathbf{A}, \mathbf{B}, \mathbf{c}) \neq F(\mathbf{A}, \mathbf{c}) \otimes F(\mathbf{B}, \mathbf{c})$ for all $\mathbf{c} \in \mathcal{X}_C$, and there are no conditional independencies in F. □

Corollary 8. *The model count of a deterministic, decomposable NNF can be computed in time linear in its size.*

Proof. Let $F(\mathbf{X})$ be a deterministic, decomposable NNF and $F'(\mathbf{X})$ be F translated to the sum-product semiring. Clearly, F and F' have equal size and F' is deterministic and decomposable. Thus, from the sum-product theorem, $\sum_{\mathcal{X}} F'(\mathbf{X})$ takes time linear in the size of F. Let v be a node in F and v' its corresponding node in F'. It remains to show that $\sum_{\mathcal{X}} F'_v(\mathbf{X}) = \#\mathrm{SAT}(F_v(\mathbf{X}))$ for all v, v', which we do by induction. The base case with v a leaf node holds trivially. For the induction step, assume that $\sum_{\mathcal{X}_i} F'_i(\mathbf{X}_i) = \#\mathrm{SAT}(F_i(\mathbf{X}_i))$ for each child $c_i \in \mathrm{Ch}(v)$ (resp. $c'_i \in \mathrm{Ch}(v')$). If v is an AND node then v' is a multiplication node and $\#\mathrm{SAT}(F_v(\mathbf{X})) = \#\mathrm{SAT}(\bigwedge_{c_i} F_i(\mathbf{x}_i)) = \prod_{c_i} \#\mathrm{SAT}(F_i(\mathbf{x}_i)) = \sum_{\mathcal{X}} F'_v(\mathbf{X})$, because v and v' are decomposable. If v is an OR node then v' is an addition node and $\#\mathrm{SAT}(F_v(\mathbf{X})) = \#\mathrm{SAT}(\bigvee_{c_i} F_i(\mathbf{x}_i)) = \sum_{c_i} \#\mathrm{SAT}(F_i(\mathbf{x}_i)) = \sum_{\mathcal{X}} F'_v(\mathbf{X})$, because v is deterministic, so its children are logically disjoint. □

Corollary 10. *The MPE state of a selective, decomposable SPN can be found in time linear in its size.*

Proof. Let $S(\mathbf{X})$ be a selective, decomposable SPN and $S'(\mathbf{X})$ its max-product version, which has the same size and is also selective and decomposable. For clarity, we assume no evidence since it is trivial to incorporate. From the sum-product theorem, $\max_{\mathcal{X}} S'(\mathbf{X})$ takes time linear in the size of S. Let v be a node in S and v' its corresponding node in S'. It remains to show that $\max_{\mathcal{X}} S'_v(\mathbf{X}) = \max_{\mathcal{X}} S(\mathbf{X})$ for all v, v', which we do by induction on v. The base case with v a leaf holds trivially because v and v' are identical. For the induction step, assume that $\max_{\mathcal{X}_i} S'_i(\mathbf{X}_i) = \max_{\mathcal{X}_i} S_i(\mathbf{X}_i)$ for each child $c_i \in \mathrm{Ch}(v)$ (resp. $c'_i \in \mathrm{Ch}(v')$). If v is a product node then so is v' and $\max_{\mathcal{X}} S'_v(\mathbf{X}) = \max_{\mathcal{X}} S_v(\mathbf{X})$. If v is a sum node then v' is a max node and $\max_{\mathcal{X}} S(\mathbf{X}) = \max_{\mathbf{x}\in\mathcal{X}} \sum_{c_i} S_i(\mathbf{x}_i) = \max_{\mathbf{x}\in\mathcal{X}}\{\max_{c_i} S_i(\mathbf{x}_i)\} = \max_{c_i}\{\max_{\mathbf{x}_i\in\mathcal{X}_i} S_i(\mathbf{x}_i)\} = \max_{\mathcal{X}} S'(\mathbf{X})$, where the second equality occurs because v is selective. After summing S', the MPE state is recovered by a downward pass in S', which takes linear time. □

Corollary 11. *The partition function of TPKB $\mathcal{K}$ can be computed in time linear in its size.*

Proof. The only sources of non-decomposability in (1) are if an object $(\mathtt{O}, \mathtt{C})$ and one of its subclasses $(\mathtt{O}, \mathtt{S_j})$ both contain (i) the same relation $\mathtt{R_i}(\mathtt{O}, \dots)$ or (ii) the same attribute $\mathtt{A_i}(\mathtt{O}, \mathtt{D})$. Note that they cannot contain the same part since two classes such that one is a descendant of the other in the class hierarchy never have a part with the same name. In each of the above cases, the shared relation (or attribute) can be pushed into each subclass (O, S_j) by distributing $\rho(\mathtt{O}, \mathtt{R_i}, \mathbf{W})$ (or $\alpha(\mathtt{A_i}, \mathtt{C_i}, \mathbf{W})$) over the subclass sum and into each subclass (this can be repeated for multiple levels of the class hierarchy). This makes the product over relations (attributes) in $\phi(\mathtt{O}, \mathtt{S_j}, \mathbf{W})$ non-decomposable, but does not affect the decomposability of any other objects. Now, $\phi(\mathtt{O}, \mathtt{S_j}, \mathbf{W})$ can be decomposed, as follows. For (i), the product over relations in $\phi(\mathtt{O}, \mathtt{S_j}, \mathbf{W})$ now contains the non-decomposable factor

$F(\mathtt{R_i}) = \rho(\mathtt{0}, \mathtt{R_i}, \mathbf{W}) \cdot \rho'(\mathtt{0}, \mathtt{R_i}, \mathbf{W})$, where ρ' was pushed down from $(\mathtt{0}, \mathtt{C})$. However, $F(\mathtt{R_i}) = (e^{w_i}[\mathtt{R_i}] + [\neg\mathtt{R_i}]) \cdot (e^{w'_i}[\mathtt{R_i}] + [\neg\mathtt{R_i}]) = e^{w_i + w'_i}[\mathtt{R_i}] + [\neg\mathtt{R_i}]$ since $[a]^2 = [a]$ and $[a][\neg a] = 0$ for a literal a. Thus, $F(\mathtt{R_i})$ is simply $\rho(\mathtt{0}, \mathtt{R_i}, \mathbf{W})$ with weight $w_i + w'_i$ for $\mathtt{R_i}$, which results in the same decomposable SPN structure with a different weight. For (ii), let the weight function for attribute $\mathtt{A_i}$ of class $\mathtt{C}$ with domain $\mathbf{D}_i$ be $\mathbf{u}_i$ and the weight function from the attribute that was pushed down be $\mathbf{u}'_i$. To render this decomposable, simply replace $\mathbf{u}_i$ with $\mathbf{u}_i \odot \mathbf{u}'_i$, the element-wise product of the two weight functions. Again, decomposability is achieved simply by updating the weight functions. Decomposing (i) and (ii) each adds only a linear number of edges to the original non-decomposable SPN, so the size of the corresponding decomposable SPN is $|\mathcal{K}|$. Thus, from the sum-product theorem, computing the partition function of TPKB $\mathcal{K}$ takes time linear in $|\mathcal{K}|$. □

References

Abiteboul, S., Hull, R., and Vianu, V. *Foundations of Databases*. Addison-Wesley, 1995.

Bacchus, F., Dalmao, S., and Pitassi, T. Value elimination: Bayesian inference via backtracking search. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 20–28, 2002.

Bacchus, F., Dalmao, S., and Pitassi, T. Solving #SAT and Bayesian inference with backtracking search. *Journal of Artificial Intelligence Research*, 34:391–442, 2009.

Bayardo Jr., R. J. and Pehoushek, J. D. Counting models using connected components. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pp. 157–162, 2000.

Darwiche, A. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2): 11–34, 2001a.

Darwiche, A. Decomposable negation normal form. *Journal of the ACM*, 48:608–647, 2001b.

Darwiche, A. Recursive conditioning. *Artificial Intelligence*, 126:5–41, 2001c.

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

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

Dechter, R. and Mateescu, R. AND/OR search spaces for graphical models. *Artificial intelligence*, 171:73–106, 2007.

Domingos, P. and Webb, W. A. A tractable first-order probabilistic logic. In *Proceedings of the 26th Conference on Artificial Intelligence*, pp. 1902–1909, 2012.

Heras, F., Larrosa, J., and Oliveras, A. MiniMaxSAT: An efficient weighted MAX-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.

Lauritzen, S. L. and Spiegelhalter, D. J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50(2):157–224, 1988.

Lawler, E. L. and Wood, D. E. Branch-and-Bound Methods: A Survey. *Operations Research*, 14:699–719, 1966.

Ngo, H. Q., Ré, C., and Rudra, A. Skew strikes back. *ACM SIGMOD Record*, 42(4):5–16, 2014.

Niepert, M. and Domingos, P. Learning and inference in tractable probabilistic knowledge bases. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pp. 632–641, 2015.

Peharz, R., Gens, R., and Domingos, P. Learning selective sum-product networks. In *Proceedings of the ICML-14 Workshop on Learning Tractable Probabilistic Models*, 2014.

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

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

Sang, T., Bacchus, F., Beame, P., Kautz, H., and Pitassi, T. Combining component caching and clause learning for effective model counting. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*, 2004.

Sang, T., Beame, P., and Kautz, H. A dynamic approach to MPE and weighted MAX-SAT. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 173–179, 2007.

Torn, A. and Zilinskas, A. *Global Optimization*. Springer-Verlag, 1989.

Webb, W. A. and Domingos, P. Tractable probabilistic knowledge bases with existence uncertainty. In *Proceedings of the UAI-13 International Workshop on Statistical Relational AI*, 2013.