Exploring the Space of Probabilistic Sentential Decision Diagrams

Lilith Mattei[*1] **Décio L. Soares**[*2] **Alessandro Antonucci**[1] **Denis D. Mauá**[2] **Alessandro Facchini**[1]

Abstract

Probabilistic sentential decision diagrams (PSDDs) are annotated circuits providing a possibly compact specification of joint probability mass functions consistent with a formula over a set of propositional variables. PSDD inference is *tractable* in the sense that marginal queries can be achieved in linear time with respect to the circuit size by traversal algorithms. Unlike other probabilistic graphical models such as Bayesian networks, the problem of learning the structure for PSDDs received relatively little attention. We discuss some preliminary ideas related to the development of pure likelihood-score-based search methods for the learning of PSDD structures fitting a formula and a data set of consistent observations. A sampling algorithm for these models is also provided.

1. Introduction

Probabilistic graphical models (e.g., Bayesian networks) allow for the compact specification of joint probability mass functions by exploiting conditional independences between variables (Pearl, 1988; Darwiche, 2009; Koller & Friedman, 2009). The space-efficient representation does not imply efficient inference (Cooper, 1990), which lead researchers to consider novel classes of *tractable* models, which allow for polynomial-time inferences at the cost of a decrease in expressiveness or interpretability. (Meila & Jordan, 2000; Darwiche, 2003; Chechetka & Guestrin, 2007; Lowd & Domingos, 2008; Elidan & Gould, 2008; Roth & Samdani, 2009; Rahman et al., 2014).

Sum-product networks (SPNs) (Poon & Domingos, 2011) are currently the most popular example of tractable models. A SPN specifies a joint mass function by means of an arithmetic circuit whose edges are annotated with probabilities. By enforcing certain structural constraints over the circuit, one can ensure that the probability of any event can be computed in linear time in the circuit size, without sacrificing representational power. SPNs have been used in a varied set of machine learning tasks (Poon & Domingos, 2011; Amer & Todorovic, 2012; Sguerra & Cozman, 2016; Llerena & Mauá, 2017), with performances often comparable to those of deep neural networks (Peharz et al., 2018), with the advantage of a clear probabilistic interpretation not provided by the latter models.

Probabilistic sentential decision diagrams (PSDDs) (Kisa et al., 2014) are similar to SPNs, but unlike SPNs can natively embed logical constraints. Given a formula over a set of propositional variables, PSDDs allow for the specification of a joint probability mass function consistent with the formula, i.e., assigning zero probability to the universes that are impossible for the formula. This enables learning in highly structured spaces such as in the case of preference learning (Choi et al., 2015) and routing problems (Shen et al., 2018).

Algorithms for structure learning of standard graphical models such as Bayesian networks are usually categorized into constraint-based (Spirtes & Meek, 1995), if they build a model by identifying independences in data, search-based (Teyssier & Koller, 2005), if they build a model by optimizing a score function, or hybrid (Tsamardinos et al., 2006), if they combine both approaches. Despite the early success of constraint-based approaches, search-based methods have come to dominate the literature (Scanagatta et al., 2018), most likely due to their ability to trade off accuracy and computational cost, and robustness to learning parameters.

Compared to standard graphical models, structure learning of tractable models has received less attention. For SPNs, constraint-based approaches are largely more popular (Gens & Domingos, 2013; Vergari et al., 2015; Rooshenas & Lowd, 2014), possibly due to the cost of obtaining score-maximizing parameters, unless additional constraints are imposed (Peharz et al., 2016). Unlike SPNs, PSDDs allow for closed-form maximum likelihood estimators of the parameters (Kisa et al., 2014). This feature, shared with other tractable models (e.g., cutset networks, thin junction trees and tree Bayesian networks), might be used to imple-

[*]Equal contribution [1]Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland [2]University of São Paulo (USP), São Paulo, Brazil. Correspondence to: Alessandro Antonucci <alessandro@idsia.ch>.

3rd Tractable Probabilistic Modeling Workshop, 36^{th} International Conference on Machine Learning, Long Beach, California, 2019.

ment search-based approaches to structure learning using (penalized) log-likelihood score. Yet, to our knowledge, the only approach to PSDD structure learning is *LearnPSDD*, an algorithm mixing constraint-based methods with local score-improving transformations (Liang et al., 2017), for which a discriminative counterpart of that algorithm has been also proposed (Liang & Van den Broeck, 2019).

In this work, we give the very first steps towards learning PSDD structures by a pure search-based algorithm guided by (penalized) likelihood scores. Parallel to work on order-based Bayesian network structure learning (Teyssier & Koller, 2005), we propose to search for PSDD structures by sampling in the space of structures constrained to an ordering of the variables. While in Bayesian networks an ordering is represented by a permutation of the variables, in PSDDs the ordering is decided by a *vtree* (Choi & Darwiche, 2013), a full binary tree whose leaves are identified with the variables. This introduces extra complexity to the procedure that needs to be addressed. There is yet another difficulty. There are many PSDDs consistent with a given vtree, unless strong assumptions are adopted (such as compression), which drastically reduce their representation power. Thus, we propose to search for ordered PSDDs by a second sampling of *maximally uncompressed* structures followed by random partial compressions.

The paper is organised as follows. We review some necessary background material about vtrees and PSDDs in Section 2. The role of compression and the structure of the space of uncompressed PSDDs is discussed in Section 3. Our sampling procedure is sketched in Section 4. A demonstrative example and some preliminary tests are reported in Section 5. Conclusion and outlooks are finally discussed in Section 6.

2. Background

We first review some basic concepts leading to the definition of sentential decision diagram and its probabilistic extension.

Let us introduce a notion of *decomposition* of a formula.

Definition 1. *Given a Boolean formula ϕ over $\boldsymbol{X}$, consider a set of tuples $\{(p_i, s_i)\}_{i=1}^k$ such that p_i is a formula over $\boldsymbol{X}'$ and s_i a formula over $\boldsymbol{X}''$, with $\boldsymbol{X}' \cap \boldsymbol{X}'' = \emptyset$ and $\boldsymbol{X}' \cup \boldsymbol{X}'' = \boldsymbol{X}$. We say that the tuples are a $(\boldsymbol{X}', \boldsymbol{X}'')$-*decomposition *of ϕ if and only if:*

$$\phi(\boldsymbol{x}', \boldsymbol{x}'') = \bigvee_{i=1}^{k} p_i(\boldsymbol{x}') \wedge s_i(\boldsymbol{x}'')\,, \tag{1}$$

for each $\boldsymbol{x}' \in \{\top, \bot\}^{|\boldsymbol{X}'|}$ and $\boldsymbol{x}'' \in \{\top, \bot\}^{|\boldsymbol{X}''|}$.

The decomposition is *strongly deterministic* if and only if $p_i \wedge p_j = \bot$ for each $i \neq j$. If this is the case, the tuple (p_i, s_i) is called an *element* of the decomposition and, in particular, p_i is a *prime* and s_i a *sub*, for each $i = 1, \ldots, k$, where k is called the *size* of the decomposition. If the primes of a strongly deterministic $(\boldsymbol{X}', \boldsymbol{X}'')$-decomposition of ϕ are also consistent and exhaustive, we call the decomposition a $\boldsymbol{X}'$-partition. Note that in a $\boldsymbol{X}'$-partition $\bot$ cannot be a prime, while $\top$ is a prime if and only if the partition has size one.

The following definition provides a generalisation of the notion of order over a set of variables $\boldsymbol{X} := (X_1, \ldots, X_n)$.

Definition 2. *A* vtree *over $\boldsymbol{X}$ is a full binary tree whose leaves are in one-to-one correspondence with $\boldsymbol{X}$.*

We call by the same name a node v and the subtree rooted at v. Notation $v^{\leftarrow}$ (resp. $v^{\rightarrow}$) is used to denote the left (resp. the right) child of node v as well as the corresponding subtree. A vtree induces a total order over its variables, but two distinct vtrees can induce the same order.

Vtrees and partitions define sentential decision diagrams.

Definition 3. *A* sentential decision diagram *(SDD) α for a vtree v over variables $\boldsymbol{X}$ is either a terminal node or a decomposition node for v. It is defined as follows, where $\langle\alpha\rangle$ - its* interpretation *- denotes the Boolean formula represented by α:*

- *v is a leaf with variable X and α is either a constant, $\alpha \in \{\bot, \top\}$, or a literal, $\alpha \in \{X, \neg X\}$. In this case α is a* terminal *node and its interpretation is respectively $\langle\alpha\rangle = \bot$, $\langle\alpha\rangle = \top$ and $\langle\alpha\rangle = X$, $\langle\alpha\rangle = \neg X$.*
- *v is an internal node and $\alpha = \{(p_i, s_i)\}_{i=1}^k$, where: the p_i's and the s_i's are SDDs for $v^{\leftarrow}$ and $v^{\rightarrow}$ respectively, such that $\{(\langle p_i\rangle, \langle s_i\rangle)\}_{i=1}^k$ is an $\boldsymbol{X}'$-partition of $\langle\alpha\rangle$, where $\boldsymbol{X}'$ are the variables of $v^{\leftarrow}$. In this case α is called a* decomposition *(or decision) node, and its interpretation is*

$$\langle\alpha\rangle = \bigvee_{i=1}^{k} \langle p_i\rangle \wedge \langle s_i\rangle\,. \tag{2}$$

As a decomposition node α is interpreted as a disjunctive formula and each of its elements as a conjunctive one, we can intend a SDD α as a logical circuit (hence a directed acyclic graph) providing a representation of its interpretation $\langle\alpha\rangle$. Observe that by definition of $\boldsymbol{X}'$-partition, the OR gates associated to decision nodes act as XOR gates.

The next notion, characterizing paths in SDDs, is needed for a global interpretation of parameters when extending SDDs with probabilities. For the sake of simplicity we assume the SDD singly connected, while the extension to the general case is straightforward.

Definition 4. *Let q be a node of a SDD. Denote as $(p_1, s_1) \dots (p_l, s_l)$ the path from the root to node q. Then the conjunction of the interpretations of the primes encountered in this path, i.e., $\langle p_1 \rangle \wedge \dots \wedge \langle p_l \rangle$, is called the* context *of q and denoted as γ_q. The context γ_q is* feasible *if and only if $s_i \neq \bot$ for each $i = 1, \dots, l$.*

The interpretation of a node is implied by its context and by the interpretation of the SDD to which it belongs, i.e., for each node q of a SDD r, $\langle r \rangle \wedge \gamma_q \models \langle q \rangle$. Moreover, nodes for the same vtree node have mutually exclusive and exhaustive contexts.

We are now in the condition of providing a formal definition of PSDDs as parametrised SDDs inducing a probability mass function over the variables of the vtree.

Definition 5. *A* probabilistic sentential decision diagram *(PSDD) is a SDD with the following parameters assigned:*

- *For each terminal node $\top$, a positive parameter θ is provided such that $0 \leq \theta \leq 1$.*[1] *The notation for such a terminal node is $X : \theta$, where X is the variable of its correspondent leaf vtree node. Terminal nodes other than $\top$ appear as they are;*
- *For each decision node $\{(p_i, s_i)\}_{i=1}^k$, each prime p_i is provided with a positive parameter θ_i, such that $\theta_1 + \dots + \theta_k = 1$ while $\theta_i = 0$ if and only if $s_i = \bot$. Notation $\{(p_i, s_i, \theta_i)\}_{i=1}^k$ is used.*

In other words, PSDDs are just SDDs with probability mass functions associated to the terminal and decision nodes.

Each node q of a PSDD α induces a probability mass function $\Pr_q$ over the variables of the vtree node q is defined for. According to the *Base Theorem* for PSDDs [Theorem 1, (Kisa et al., 2014)], $\Pr_q$ assigns zero probability to events which do not respect the propositional sentence associated to the SDD q, more precisely, for any instantiation $\boldsymbol{x}$ of variables $\boldsymbol{X}$ of the vtree q is defined for, $\Pr_q(\boldsymbol{x}) > 0$ if and only if $\boldsymbol{x} \models \langle q \rangle$.

The joint mass function induced by a PSDD r satisfies the following independence relations: for each node q in r defined for vtree node v, for each feasible context γ_q of node q, variables inside v are independent of those outside v given γ_q with respect to $\Pr_r$ [Theorem 5, (Kisa et al., 2014)]. Moreover, the probabilities $\Pr_r(\langle p_i \rangle | \gamma_q)$ are the parameters θ_i of $q = \{(p_i, s_i, \theta_i)\}_{i=1}^k$ [Corollary 1, (Kisa et al., 2014)]. Such a semantics allow to learn in a closed form the maximum likelihood estimators of the parameters of a PSDD from a dataset $\mathcal{D}$ of complete consistent observations.

[1] We consider PSDDs assigning zero probability only to logically impossible configurations of its variables.

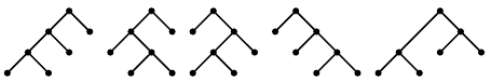

Figure 1. Pre-vtrees with four leaves.

The *size* of a PSDD is the number of *free parameters* in the model. Each decision node with k elements contributes $k - 1$ free parameters, while each terminal node $\top$ contributes with one free parameter. We remove from this count parameters whose values are defined by the semantics of PSDDs; for example, parameters θ_i associated with elements such that $\langle p_i \rangle$ is either implied or contradicted by its context γ_q, as well as decision nodes with infeasible contexts. Note that the size of the underlying SDD (given by the number of nodes) can be much larger than the size of the PSDD, as many elements have their value fixed by the semantics. For example, the PSDD in Figure 3 has size 15, while the underlying SDD has size 81.

3. Coping with Uncompressed PSDDs

Given a formula ϕ over the n propositional variables in $\boldsymbol{X}$ and a dataset $\mathcal{D}$ of consistent observations let us denote as $\Pi(n, \phi)$ the set of *all* the PSDDs we can learn from $\mathcal{D}$.

Counting vtrees and PSDDs. To characterize $\Pi(n, \phi)$, let us first consider all the vtrees we can define over $\boldsymbol{X}$. We call *pre-vtree* a vtree as in Definition 2 before the specification of the one-to-one correspondence between the elements of $\boldsymbol{X}$ and the n leaves. Figure 1 shows all pre-vtrees with four leaves. Each pre-vtree corresponds to $n!$ vtrees, and the number of pre-vtrees with n leaves is $(n-1)$-th Catalan number, i.e., $C_{n-1} := \frac{(2n-2)!}{n!(n-1)!}$ (Choi & Darwiche, 2013). C_{n-1} is exponential with respect to n. This makes a potential difference between the space of the directed acyclic graphs to be considered for the structural learning of a Bayesian network and that of PSDDs. Given an ordering over $\boldsymbol{X}$, the best Bayesian network structure with topology consistent with the ordering can be found efficiently (Teyssier & Koller, 2005), while, due to the exponential number of vtrees consistent with an ordering, a similar result cannot be easily derived for PSDDs.

The role of compression. The situation is indeed even more twisted. Given vtree v, let $\Pi(n, \phi, v)$ denote the set of PSDDs based on SDDs for v implementing the formula ϕ. To discuss the cardinality of $\Pi(n, \phi, v)$ we better need to invoke the basic notion of compression of a partition.

A $\boldsymbol{X'}$-partition is *compressed* if and only if its subs are distinct. Uncompressed partitions can be compressed by merging the elements with the same sub in a single element

with the same sub and whose prime is the disjunction of the primes. E.g., (p, s) and (p', s) becomes $(p \vee p', s)$. An inverse operation of *decompression* might be also considered. Compression and decompression clearly preserve partitions.

The notion of compression can be extended to SDDs: an SDD is compressed if and only if the partitions of its decision nodes are compressed. We say that a PSDD is compressed if and only if its underlying SDD is compressed. A less strict notion of compression for PSDDs is provided in [Theorem 10, (Kisa et al., 2014)].

Compressed elements of $\Pi(n, \phi, v)$ are, by construction, the elements of smallest size in this class. Coping with compressed PSDDs might be therefore not appealing from a structural learning perspective. Consider for instance the following case.

Example 1. *Given $n = 3$, $\phi = \top$ and the vtree v in the left side of Figure 2. A compressed SDD for v implementing the formula ϕ is the one in right side of Figure 2. The PSDD associated to this model defines a joint mass function such that* $\Pr(A, B, C) = \Pr(A) \cdot \Pr(B) \cdot \Pr(C)$.

Compressed PSDDs have been used in (Shen et al., 2017) and (Shen et al., 2019). Yet, following the discussion in the example, we consider uncompressed PSDDs as more expressive dependence models.

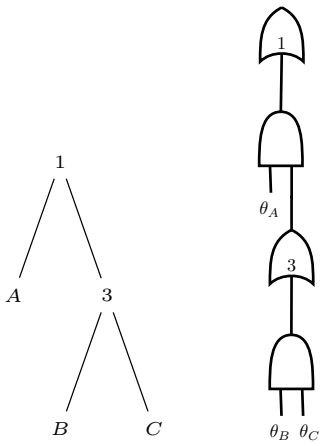

Figure 2. A vtree (left) and a compressed PSDD (right).

Building uncompressed PSDDs. The following procedure allows to explore the space of uncompressed PSDDs for a given vtree v over variables $\boldsymbol{X}$ representing a formula ϕ over $\boldsymbol{X}$.

1. Let $\boldsymbol{X}'$ be the variables of $v^{\leftarrow}$. Define, for each $\boldsymbol{x}'$ in $\{\bot, \top\}^{|\boldsymbol{X}'|}$, the formula:

$$\epsilon_{\boldsymbol{x}'} = \bigwedge_{X' \in \boldsymbol{X}'} \delta_{x'} \,, \tag{3}$$

where:

$$\delta_{x'} = \begin{cases} X' & \text{if} \quad x' = \top \\ \neg X' & \text{if} \quad x' = \bot \end{cases} \,, \tag{4}$$

and x' is the element of $\boldsymbol{x}'$ associated to X'. Formulae in Equation (3) are our primes (by construction they are consistent, mutually exclusive and exhaustive). The sub s of prime $\epsilon_{\boldsymbol{x}'}$ is $\phi(\boldsymbol{x}', \cdot)$, i.e., the restriction to $\boldsymbol{X}''$ of ϕ for instantiation $\boldsymbol{x}' \in \{\bot, \top\}^{|\boldsymbol{X}'|}$. This gives an $\boldsymbol{X}'$-partition which we will call *expanded*. Note that the elements of this partition that are inconsistent with the formula in the root of the vtree can be removed.

2. The size of the above decomposition is exponential in the number of variables in $\boldsymbol{X}'$. A compression can be achieved by merging together elements with the same sub, i.e.,

$$p = \bigvee_{\boldsymbol{x}' : \phi(\boldsymbol{x}', \cdot) = s} \epsilon_{\boldsymbol{x}'} \,, \tag{5}$$

for complete compression, while the disjunction is only over some $\boldsymbol{x}'$ such that $\phi(\boldsymbol{x}', \cdot) = s$ for partial compression.

The above two-step procedure can be iterated (every time with arbitrary choices of compression) for each prime (sub) with vtree $v^{\leftarrow}$ ($v^{\rightarrow}$) until $v^{\leftarrow}$ ($v^{\rightarrow}$) is empty.

Figure 3 shows an example of expanded PSDD obtained by the above procedure using a left linear vtree, with $\phi = \top$ in the root. This PSDD represents the factorisation $\Pr(ABCD) = \Pr(ABC) \cdot \Pr(D|ABC)$, and therefore does not encodes independences. It has size $2^4 - 1 = 15$, denoting the number of "learnable" parameters; the remaining parameters have their value fixed by the semantics. For example, the parameters in the subtree rooted at node 3 are all 0/1-valued (these subtrees can be removed for efficiency, if one is willing to cope with unnormalised circuits).

When the formula ϕ at the root is a tautology, every expanded PSDD generated in such way will have the maximum number of free parameters, hence will encode the same representation (they differ only in the way that the joint mass function is factorised, but do not imply any independence). For example, the expanded PSDD in Figure 4 represents $\Pr(ABCD) = \Pr(A) \Pr(B|A) \Pr(C|AB) \Pr(D|ABC)$. As with the PSDD in Figure 3, it has 15 free parameters (and 23 nodes). The PSDD in Figure 5 is obtained by a partial compression of the one in Figure 3. Such PSDD encodes the fact that, given A and C, B and D are independent.

Our procedure for constructing PSDDs depends on the formula ϕ. The following result is easy to prove and ensures that the above procedure allows to navigate through the whole set $\Pi(\phi, n, v)$

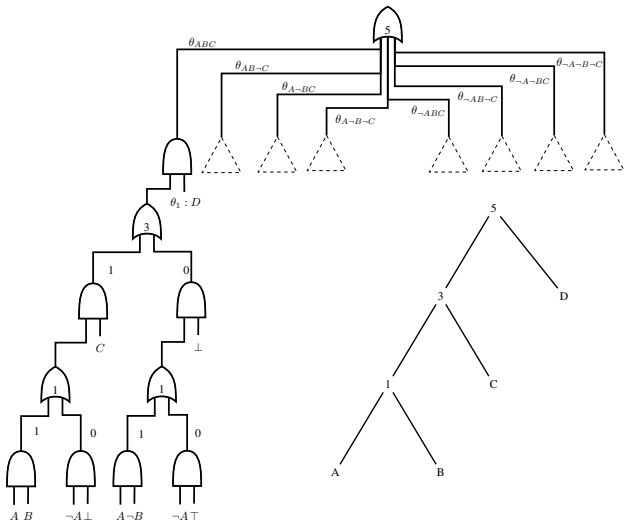

Figure 3. Fragment of the expanded PSDD generated using a tautology and the left linear vtree on the bottom right. The missing parts are copies of the left subtree.

Proposition 1. *Any $\boldsymbol{X}'$-partition of a formula ϕ can be obtained by starting from the expanded one and then proceeding with a number of compressions.*

4. Sampling the PSDDs Space

Vtree sampling. Following the discussion in the previous section, we might sample vtrees by separately sampling pre-vtrees and orderings (i.e., permutations) of the n model variables to be linked to the leaves of the pre-vtree. While a uniform sampler for permutations can be obtained by classical methods (e.g., Knuth's shuffle), doing the same in the space of pre-vtrees is more tricky. Here we achieve that by exploiting the one-to-one correspondence between pre-vtrees and *Dyck words*, i.e., a balanced string of square brackets. Sampling a Dyck word of length w can be achieved by w calls of a biased square bracket random generator, such that:

$$P(]) = \frac{r(k+r+2)}{2k(r+1)}\,, \tag{6}$$

and $P(]) = 1 - P([)$, where r is the number of unmatched left brackets and k is the difference between w and the current length of the word (Arnold & Sleep, 1980). The equivalence between Dyck words and pre-vtrees follows from the fact that their spaces have the same cardinality.

The following recursion converts a Dyck word of length w in a pre-vtree with $n = w/2 + 1$ leaves. Call *cut* of the word the leftmost symbol for which the number of left and right brackets is matched. Obtain two sub-words by splitting the word in the cut and initialize a tree with two children. The left child is associated to the first sub-word after the removal of the first and last symbol, while the right one is associated to the second sub-word. A recursive application of this procedure until only empty words appear produces the pre-vtree, whose leaves are eventually labelled according to the selected permutation by a depth-left-first traversal. An example is in Figure 6.

PSDDs sampling. Given a randomly generated vtree v and a formula ϕ, the procedure to build uncompressed PSDDs can be used as a sampling algorithm for these models based on different (random) choices of the level of compression in Equation (5). Since each compression makes the possible dependency of the sub on the two or more merged primes drop, by applying compression operations we eventually loose some dependence relations. As an alternative to random compression, for each iteration, we might perform independence tests in the dataset in order to decide whether to compress or not elements with identical subs. All the PSDD structures can be generated in this way because of Proposition 1.

A likelihood score for PSDDs. The above described sampling procedure might be used to search the best PSDD explaining the dataset $\mathcal{D}$. Analogously to what is commonly done for Bayesian networks, a likelihood based score can be used to select the best model. This typically gives higher scores to larger models, whose high number of parameters makes easier to fit the data. To prevent *over-fitting*, the (log) likelihood should be penalized by a term taking into account the number of parameters (i.e., the PSDD size) . Accordingly, we adopt the *Bayesian information criterion* score. Thus if $\mathcal{D} := \{\boldsymbol{x}^{(i)}\}_{i=1}^{d}$, the score of the PSDD α is

$$\mathrm{BIC}(\alpha) := \sum_{i=1}^{d} \ln \Pr_{\alpha}(\boldsymbol{x}^{(i)}) - \frac{\ln d}{2}|\alpha|\,, \tag{7}$$

where $|\alpha|$ denotes the size of the PSDD. Note that the probability of the complete observation $\boldsymbol{x}^{(i)}$ can be computed in time $O(|\alpha|)$. Such score is analogous to what has been proposed in (Bekker et al., 2015). The structural learning should be therefore intended as a combinatorial optimization task for which the score in Equation (7) should be optimized over the PSDD space $\Pi(\phi, n)$. The above described sampling procedure can be intended as a very preliminary approach to such a complex discrete optimization task.

5. A Demonstrative Example

As a very first and preliminary application of the previously discussed ideas we consider a small setup with $n = 4$ and $\phi = \top$. We generate a dataset of 3300 instances by sampling from a PSDD with structure and vtree as in Figure 7. We randomly select 300 instances for training the model parameters, and leave the rest for evaluation.

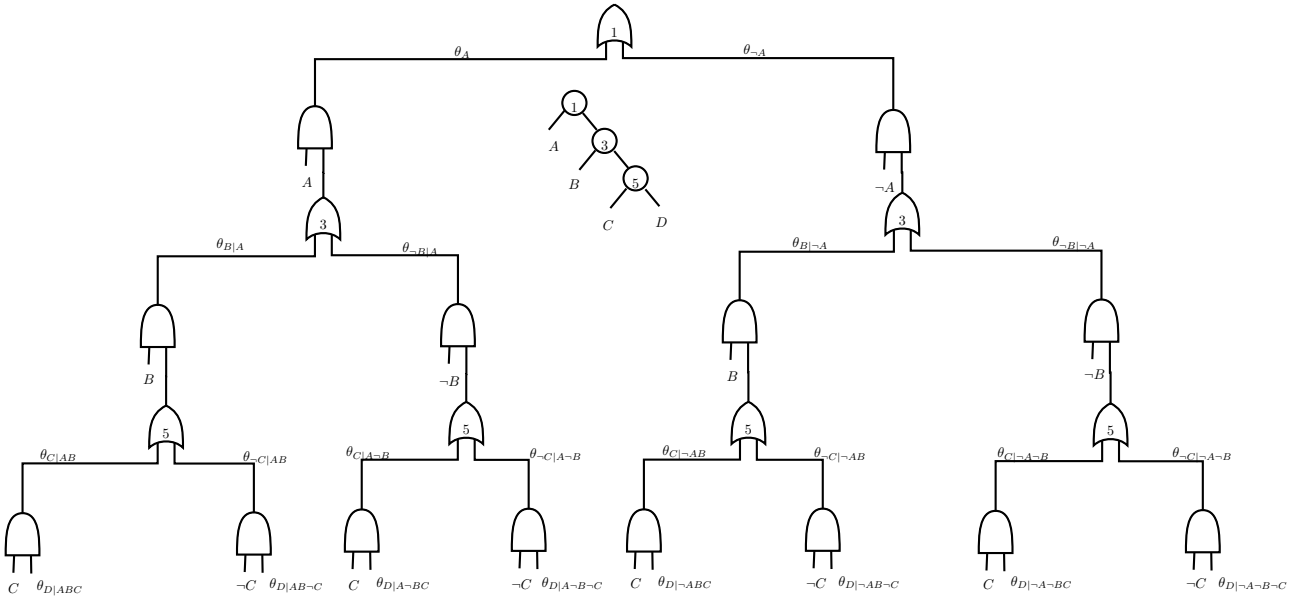

Figure 4. Expanded PSDD generated using a tautology and the right linear vtree in the middle.

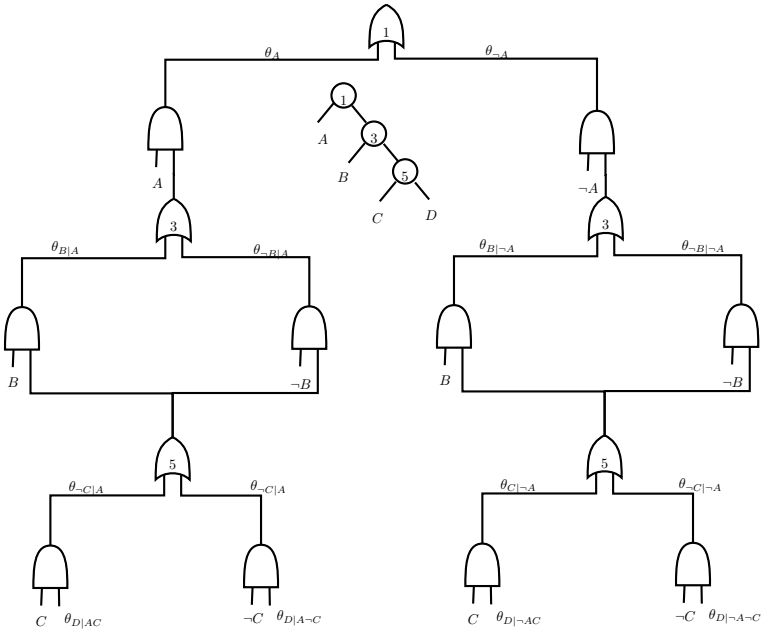

Figure 5. Partially compressed PSDDs obtained from the PSDD in Figure 4.

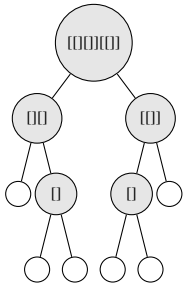

Figure 6. The pre-vtree associated to the word [[][]][[]].

Table 1. Results for PSDDs generated with our approach.

Vtree	V-Order	# compr.	LL(train)	LL(test)	Size	BIC
135	1234	0	-641.73	-6528.66	15	-6588.71
135	2341	0	-641.74	-6529.01	15	-6589.06
135	4231	0	-641.74	-6529.01	15	-6589.06
135	3142	0	-641.74	-6530.44	15	-6590.49
531	1234	0	**-641.73**	-6528.50	15	-6588.55
531	2341	0	-641.74	-6529.00	15	-6589.04
531	4231	0	-641.74	-6529.01	15	-6589.06
531	3142	0	-641.73	-6530.21	15	-6590.26
531	1234	1	-642.56	**-6527.93**	12	-6575.97
531	2341	1	-673.58	-6760.25	12	-6808.29
531	4231	1	-652.38	-6586.72	12	-6634.76
531	3142	1	-648.64	-6539.35	12	-6587.39
531	1234	2	-654.47	-6599.15	7	-6627.17
531	2341	2	-701.63	-6940.68	7	-6968.71
531	4231	2	-654.71	-6598.90	7	-6626.92
531	3142	2	-657.77	-6531.67	7	**-6559.69**

We sample PSDD structures using left and right linear pre-vtrees and permuting the variable ordering at the leaves (so as to obtain vtrees). We learn the parameters from data using a 0.1 smoothing to circumvent issues with zero counts.

Table 1 shows train and test log-likelihoods (LL), size and BIC scores for the PSDDs sampled. Column *# compr.* denotes the number of compression operations performed (0 corresponds therefore to expanded PSDDs). The highlighted row indicates the performance of the PSDD with same structure as the one we used to generate the data. One can see that it maximises test log-likelihood, and ranks second in BIC score.

6. Conclusions and Outlooks

This paper presents a preliminary analysis of the space of PSDDs. This suggests possible directions for the learning of PSDD structures by a search based on a likelihood score. As a future work we intend to derive efficient navigation strategies to achieve such a search and define a notion of equivalence for PSDDs analogous to what has been already done for Bayesian networks.

References

Amer, M. R. and Todorovic, S. Sum-product networks for modeling activities with stochastic structure. In *Proceedings of Computer Vision and Pattern Recognition*, pp. 1314–1321, 2012.

Arnold, D. B. and Sleep, M. R. Uniform random generation of balanced parenthesis strings. *ACM Transactions on Programming Languages and Systems*, 2(1):122–128, 1980.

Bekker, J., Davis, J., Choi, A., Darwiche, A., and Van den Broeck, G. Tractable learning for complex probability queries. In *Advances in Neural Information Processing Systems 28*, pp. 2242–2250. Curran Associates, Inc., 2015.

Chechetka, A. and Guestrin, C. Efficient principled learning of thin junction trees. In *Advances in Neural Information Processing Systems 20*, pp. 273–280. Curran Associates, Inc., 2007.

Choi, A. and Darwiche, A. Dynamic minimization of sentential decision diagrams. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

Choi, A., Van den Broeck, G., and Darwiche, A. Tractable learning for structured probability spaces: A case study in learning preference distributions. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Cooper, G. F. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial intelligence*, 42(2-3):393–405, 1990.

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

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

Elidan, G. and Gould, S. Learning bounded treewidth Bayesian networks. *Journal of Machine Learning Research*, 9:2699–2731, 2008.

Gens, R. and Domingos, P. Learning the structure of sum-product networks. In *International Conference on Machine Learning*, pp. 873–880, 2013.

Kisa, D., Van den Broeck, G., Choi, A., and Darwiche, A. Probabilistic sentential decision diagrams. In *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2014.

Koller, D. and Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

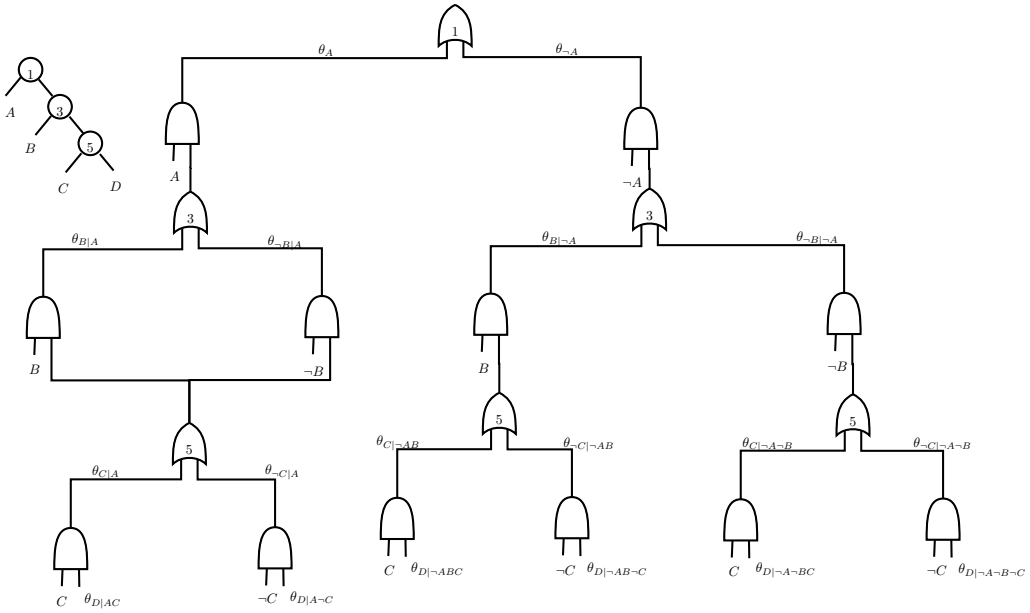

Figure 7. Partially compressed PSDDs obtained from the PSDD in Figure 4, used to generate data for the experiment.

Liang, Y. and Van den Broeck, G. Learning logistic circuits. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

Liang, Y., Bekker, J., and Van den Broeck, G. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, 2017.

Llerena, J. V. and Mauá, D. D. On using sum-product networks for multi-label classification. In *Proceedings of the Sixth Brazilian Conference on Intelligent Systems*, pp. 25–30, 2017.

Lowd, D. and Domingos, P. Learning arithmetic circuits. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI, pp. 383–392, 2008.

Meila, M. and Jordan, M. I. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

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

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Kersting, K., and Ghahramani, Z. Probabilistic deep learning using random sum-product networks. *arXiv preprint arXiv:1806.01910*, 2018.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops*, pp. 689–690. IEEE, 2011.

Rahman, T., Kothalkar, P., and Gogate, V. Cutset networks: a simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 630–645, 2014.

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

Roth, D. and Samdani, R. Learning multi-linear representations of distributions for efficient inference. *Machine Learning*, 76(2–3):195–209, 2009.

Scanagatta, M., Corani, G., de Campos, C., and Zaffalon, M. Approximate structure learning for large Bayesian networks. *Machine Learning*, 107(8-10):1209–1227, 2018.

Sguerra, B. M. and Cozman, F. G. Image classification using sum-product networks for autonomous flight of micro aerial vehicles. In *Proceeding of the Fifth Brazilian Conference on Intelligent Systems*, pp. 139–144, 2016.

Shen, Y., Choi, A., and Darwiche, A. A tractable probabilistic model for subset selection. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, 2017.

Shen, Y., Choi, A., and Darwiche, A. Conditional PSDDs: Modeling and learning with modular knowledge. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Shen, Y., Goyanka, A., Darwiche, A., and Choi, A. Structured bayesian networks: From inference to learning with routes. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

Spirtes, P. and Meek, C. Learning Bayesian networks with discrete variables from data. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp. 294–299, 1995.

Teyssier, M. and Koller, D. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pp. 584–590, 2005.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78, 2006.

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