Probabilistic Circuit Networks

Florian Peter Busch[1,2,†] **Moritz Willig**[1]
Kristian Kersting[1,2,3,4] **Devendra Singh Dhami**[5]

[1]Computer Science Department, Technical University of Darmstadt
[2]Hessian Center for AI (hessian.AI)
[3]German Research Center for AI (DFKI)
[4]Centre for Cognitive Science, Technical University of Darmstadt
[5]Department of Mathematics and Computer Science, Eindhoven University of Technology
[†]Correspondence to florian_peter.busch@tu-darmstadt.de

Abstract

As inference in Bayesian networks (BNs) can quickly become intractable, probabilistic circuits (PCs) have been established as the go-to choice for performing tractable inference. However, modeling with BNs remains popular as their graphical structure closely aligns with the semantics of the problem, making them modular and interpretable. As this generally does not apply to PCs trained from data, we propose probabilistic circuit networks (PCNs). PCNs model variables using separate PCs each covering the variable and its parents, so that the marginal distributions can be updated in response to changes upstream. To this end, an iterative proportional fitting (IPF) procedure can be employed, which remains tractable owing to the tractability properties of PCs. While the inference procedure of PCNs is only approximate and thus performs worse than BNs w.r.t. accuracy, we experimentally show that PCNs are more sample-efficient than full PCs and remain computationally tractable despite implementing the graphical structure directly.

1. Introduction

Bayesian networks (BNs) and probabilistic circuits (PCs) each have clear strengths that the other lacks. BNs explicitly model a directed acyclic graph (DAG) over the represented variables, which makes them interpretable, modular, and enforces (conditional) independencies directly through the graph structure. This modularity facilitates adaptable models where individual components can be inspected and changed, and enables interventions when the BN follows a causal graph. For PCs, such interventions require specialized approaches (Zečević et al., 2021; Busch et al., 2025; Wang and Kwiatkowska, 2023). On the other hand, marginal inference in BNs is generally intractable, scaling exponentially with the graph's treewidth. PCs address this by supporting tractable marginal inference (Choi et al., 2020), where computations across layers can be parallelized efficiently (Peharz et al., 2020). However, a PC does not follow a DAG over its variables, and with this, the modularity and structural guarantees of BNs are lost. While it is possible to transform BNs into PCs (Zhao et al., 2015), this results in a single, monolithic circuit that loses the modular structure. Ideally, one would want a model that retains the graph-based modularity and interpretability of BNs while leveraging the tractability of PCs for inference.

This paper introduces **Probabilistic Circuit Networks (PCNs)**, a model class that connects individual PCs along a DAG (Figure 1). In a PCN, every node[1] in the graph is represented by

1. At least conceptually. In practice, nodes without parents can be simplified and no PC is built for those.

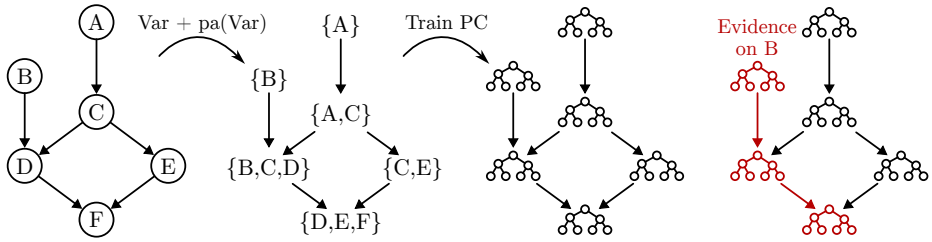

Figure 1. PC Networks. Given a DAG, the scope of each node consists of the current variable and its parents. Over this scope, a PC is trained for each node. The resulting set of PCs over the DAG is the PC network. For computing probabilistic queries on this network, a "message-passing" style of inference is applied, where evidence is passed down the edges, updating the respective PC nodes along the way.

a small PC whose scope covers the respective variable and its parents. In this work, we focus on the discrete case, where all variables have a finite number of states. Inference proceeds in a downstream manner: when evidence is observed or a parent's marginal distribution changes, the updated marginals are passed to their children nodes, and each receiving PC is adjusted accordingly. The key mechanism for this adjustment is the use of *scaling factors* applied to the leaf nodes of each PC. For tree-structured graphs, these scaling factors can be computed exactly in a single step. For arbitrary graphs, where parent variables can be correlated, we frame the problem as a matrix scaling problem and employ iterative proportional fitting (IPF; Deming and Stephan (1940)) to compute the correct factors. Crucially, each step of IPF requires marginalizing over the local PC, which, owing to the tractability of PCs, can be done in polynomial time. This turns what would normally be an intractable procedure into an efficient one, showcasing an interesting synergy between a traditional statistical method and PCs.

In their current form, PCNs have one major downside. The inference procedure used in PCNs is approximate, as downstream correlations not captured by the local factorization are lost (related to mean-field approximation (Jordan et al., 1999)). As our experiments show, this leads to higher prediction errors compared to exact BN inference on real-world benchmark graphs. Still, PCNs exhibit substantially higher sample efficiency than monolithic PCs and, most notably, scale gracefully to graph sizes where exact BN inference becomes infeasible. We see this work primarily as a methodological contribution, exploring the idea of having PCs interact and update each other's marginals along a graph structure.

Our contributions are as follows.

1. We introduce PCNs, a model class that places individual PCs at every node in a DAG and connects them through marginal updates, retaining the modularity and interpretability of BNs.

2. We show how scaling factors can update the local PCs to reflect changed marginals, and how IPF can be used to compute these factors by leveraging the tractability properties of PCs.

3. We conduct experiments showcasing the trade-offs of PCNs. While approximate inference introduces error, PCNs are more sample-efficient than full PCs and scale to problems where BN inference is no longer feasible.

2. Preliminaries

Probabilistic Circuits. A Probabilistic Circuit (PC, Choi et al. (2020)) is a computational graph that represents a joint probability distribution over a set of random variables $\mathbf{X} = \{X_1, \ldots, X_N\}$. A PC $\mathcal{C}$ consists of three types of nodes: leaf nodes, sum nodes, and product nodes. Each node n has a

scope $\phi(n) \subseteq \mathbf{X}$, denoting the set of variables modeled by that node.[2] Leaf nodes model distributions over (commonly) single variables using simple base distributions such as Gaussians or categorical distributions, i.e., $n_{\text{leaf}}(\mathbf{x}) = f(\mathbf{x}; \Theta)$, where f is a probability mass or density function and Θ its parameter(s). Sum nodes compute weighted mixtures of their children, $n_{\text{sum}}(\mathbf{x}) = \sum_{c \in \text{ch}(n)} w_c \cdot c(\mathbf{x})$ with weights $w_c \geq 0$ satisfying $\sum_{c \in \text{ch}(n)} w_c = 1$. Product nodes combine independent groups of variables via $n_{\text{prod}}(\mathbf{x}) = \prod_{c \in \text{ch}(n)} c(\mathbf{x})$. To ensure that the PC represents a valid and tractable probability distribution, we assume *smoothness*, i.e., all children of a sum node have the same scope, and *decomposability*, i.e., the scopes of the children of any product node are pairwise disjoint.

Related Work. It is possible to transform BNs into PCs (and the other way around) (Zhao et al., 2015), resulting in single, potentially exponentially large PCs, losing the modularity of BNs. Cutset Bayesian networks (Rahman et al., 2019) are similarly motivated by the generalization benefits of leveraging the true graphical structure, splitting the BN into two parts to speed up inference, however, some components can remain intractable. Lastly, neural components can be used to change the PC parameters in such a way that different marginal distributions are modeled depending on the input (Shao et al., 2022; Zečević et al., 2021; Busch et al., 2025). However, these approaches can fail to generalize to unseen inputs, as the neural component is not guaranteed to generalize to data points beyond the training distribution.

3. Probabilistic Circuit Networks

PCNs are graphical structures that implement PCs at every node. The scope of every PC within a PCN is the union of the current variable and its parents.

Definition 1 (Probabilistic Circuit Network). *A probabilistic circuit network (PCN) for some directed acyclic graph* $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ *with nodes* $\mathbf{V}$ *and edges* $\mathbf{E}$ *is a tuple* $\mathcal{C}^+ = (\mathbf{V}, \boldsymbol{\mathcal{C}})$*, where* $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ *is a set of PCs, such that for every node* $V_i \in \mathbf{V}$ *there exists a corresponding* $\mathcal{C}_i \in \boldsymbol{\mathcal{C}}$ *with scope* $\phi(\mathcal{C}_i) = (V_i \cup pa(V_i))$*.*

Generally, a PCN $\mathcal{C}^+$ factorizes in the same manner as a BN:

$$P(\mathbf{x}) = \prod_i \frac{\mathcal{C}_i\left(x_i, \text{pa}(x_i)\right)}{\mathcal{C}_i\left(\text{pa}(x_i)\right)}, \tag{1}$$

where $\mathcal{C}_i\left(\text{pa}(x_i)\right)$ is the marginal probability for $\text{pa}(x_i)$ which can be computed efficiently thanks to the tractability of $\mathcal{C}_i$. Naively, PCs could be employed instead of conditional probability tables and the PCN could otherwise be treated in the same manner as a BN. However, this would run into the general complexity issues of exact BN inference, where marginalizing variables requires summing over a potentially exponentially increasing number of variable assignments. Therefore, the idea behind PCNs is to pass information between the individual PC components in such a manner that their tractability can be leveraged to avoid the explicit "marginalization by summation" procedure.

To this end, there must be some way for the PC components in a PCN to "communicate", such that evidence or changes in some variable can propagate to other parts of the network. In the following, we propose an approach for updating the PC leaf parameters to take a specific parent distribution into account, allowing for changes in a PCN to flow downstream (computation of full queries is discussed later). To this end, we show how our inference procedure in PCNs ensures that, given arbitrary probability functions q_j corresponding to variable X_j, the PC components correctly compute $\sum_{\mathbf{z} \in \text{val}(\mathbf{Z})} P(x_i | \mathbf{e}, \mathbf{z}) \prod_{z_j \in \mathbf{z}} q_j(z_j)$, while leveraging the tractable marginal inference property of PCs to marginalize $\mathbf{Z} \subseteq \mathbf{pa}(X)$ and keeping the set of evidence $\mathbf{e}$ fixed. Specifically, we show that with PCs $\mathcal{C}_i$, we can employ an approach based on *scaling factors* α that builds upon these PCs such that for the resulting $\mathcal{C}_i^\alpha$:

2. Note that we use n to indicate a node in a PC and $\text{ch}(n)$ then denotes the children in the computational DAG of a PC. Later, we will talk about the DAG over a set of variables, in the same manner as is the case for BNs, where $\text{pa}(V)$ then indicates the set of parent variables for variable V.

$$\mathcal{C}_i^{\alpha}(x_i) = \sum_{\mathbf{z} \in \mathrm{val}(\mathbf{Z})} P(x_i|\mathbf{e}, \mathbf{z}) \prod_{z_j \in \mathbf{z}} q_j(z_j), \tag{2}$$

where $\mathrm{val}(\mathbf{Z})$ is the set of possible valuations $\mathbf{z}$ for $\mathbf{Z}$.

We employ scaling factors in such a manner that all leaf distributions are multiplied by their respective scaling factor that corresponds to a specific variable and value for all parent nodes. If for node i, there are $n = |\mathrm{pa}(i)|$ parent variables with maximum cardinality d, then there are at most dn scaling factors. We define $\alpha_j(\mathbf{z}, z_j) \to \mathbb{R}^+$ to return the scaling factor for the variable associated with index j obtaining the value z_j. Generally, this depends on all parent variables, which is why we need $\mathbf{z}$ as part of the function input. Using these scaling functions, we define our local PCs in the PCN as

$$\mathcal{C}_i^{\alpha}(x_i) := \sum_{\mathbf{z} \in \mathrm{val}(\mathbf{Z})} \left(\prod_{z_j \in \mathbf{z}} \alpha_j(\mathbf{z}, z_j) \right) \mathcal{C}_i(x_i, \mathbf{z}). \tag{3}$$

Note that $\mathcal{C}_i^{\alpha}(x_i) = \mathcal{C}_i(x_i)$ if we set $\alpha_j(\mathbf{z}, z_j) = 1$ for all j. Hence, our modified PCs are a generalization of the underlying PC $\mathcal{C}_i$.

Core Idea of PCNs

Scaling factors can be applied to the leaves corresponding to the parent variables in such a way that the resulting PC is a valid PC that correctly represents the updated probability distribution under a change of marginal distribution of the parent variables. These scaling factors can be computed in a tractable manner.

First, we consider the simple case where the underlying DAG is a tree.

Theorem 2. *For any PCN $\mathcal{C}^+$ where the underlying graph $\mathcal{G}$ is a tree, the scaling factors α can be computed exactly in a single step.*

Proof Idea. In a tree, each node has at most one parent. The scaling factor for each parent value is simply the ratio between the updated marginal $q(z)$ and the original marginal $\mathcal{C}_i(z)$. This ratio can be computed directly and applied to the corresponding leaf nodes. Full proof in Appendix A.1. □

In the case of arbitrary graphs, the simple ratio-and-update procedure no longer works, as parent variables can be correlated and changing the marginals of one variable also affects the others. However, we can frame this problem as a matrix scaling problem and apply iterative proportional fitting (IPF; Deming and Stephan (1940)) to compute the desired scaling factors.

Theorem 3. *For any PCN $\mathcal{C}^+$ over an arbitrary DAG and assuming positivity of the local joint distributions, the scaling factors α can be computed in polynomial time.*

Proof Idea. While applying the ratio update from the tree case to each parent individually does not yield the correct scaling factors, each such step corresponds to one step in IPF (Fig. 2). Under the assumption of positivity, IPF is known to converge to the true solution. While IPF generally scales exponentially with the number of dimensions (i.e., parent nodes), the tractable marginalization of PCs means the joint distribution does not have to be computed explicitly, as is usually required. This turns IPF into a polynomial-time algorithm. Full proof in Appendix A.2. □

Approximate Inference Procedure. The previous only proves the correctness of the updating procedure. For query computation, we use an inference procedure related to mean field approximation (Jordan et al., 1999). In short, we iterate over all variables in a hierarchical manner, passing down the marginal distributions by utilizing scaling factors. For any query, we multiply the local probabilities, similar to a factorization of a joint probability distribution, to compute the final query probability. As correlations that cannot be modeled downstream are not taken into account, it is generally not exact for all types of queries. For example, $P(B, C)$ on the graph $B \leftarrow A \rightarrow C$ would use the standard marginals passed down from A, which would result in computing the query as $P(B) \cdot P(C)$. Other queries that do not require such cross-branch correlations can still be computed exactly. The algorithm is included in Appendix B.

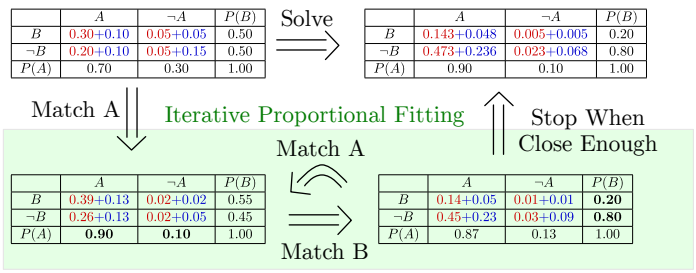

Figure 2. IPF for Fitting Marginals. Iteratively updating each parent's marginals following the IPF procedure eventually yields the desired scaling factors up to a specified tolerance ϵ.

4. Experiments

Error of the Approximate Inference Procedure. To investigate PCN accuracy in realistic settings, we use datasets from the Bayesian Network Repository (Scutari, 2010) and run conditional queries (assigning evidence to 1/3 of all variables and randomly choosing one as target variable). Note that for conditional queries especially, approximations might be particularly harmful w.r.t. the prediction error.[3] We compare with the true probabilities obtained from the ground truth BN computed using variable elimination (Zhang and Poole, 1994). In Table 1, we see that PCNs perform substantially worse than exact BN inference, with errors typically in the low single digits. While this highlights the cost of the approximation, the predictions remain in the vicinity of the true values. We illustrate positive properties of PCNs in the following experiments.

Dataset	BN	PCN
asia	0.30 ± 0.06	0.81 ± 0.17
child	0.60 ± 0.05	6.11 ± 0.90
alarm	0.36 ± 0.08	4.40 ± 0.76
insurance	0.55 ± 0.07	5.66 ± 0.97
win95pts	0.22 ± 0.04	2.59 ± 0.70
hepar2	0.41 ± 0.03	1.30 ± 0.26
hailfinder	1.06 ± 0.08	6.26 ± 0.90
water	0.31 ± 0.02	0.74 ± 0.19
barley	2.00 ± 0.21	8.50 ± 1.55
mildew	1.59 ± 0.28	6.30 ± 1.67

Table 1. Comparison BN and PCN on Real Graphs. Shown are the average L1 errors, scaled such that an error of 1 indicates that the predicted and true probabilities are, on average, off by 1%.

Higher Sample-Efficiency. We run the same setup as for the previous experiment but compare with a standard PC for asia, child, insurance, and alarm. The number of samples is increased in steps between $\{50, 100, 250, 500, 1000, 10000\}$ and the average L1 error is plotted. As can be seen from Fig. 3, PCN, despite the approximate inference procedure, performs substantially better than the single PC in the low sample regime. This is plausibly explained by the PCN enforcing all (conditional) independence by strictly enforcing the graph structure. The error bars between PCs and PCNs start overlapping for the large sample regime, but PCNs still perform better on the three larger datasets. While PC performance could likely be improved with better hyperparameters

3. In some preliminary experiments on synthetic data, we found that synthetic problems where conditional probabilities are set uniformly at random tend to result in very small approximation errors. While this makes PCN look great, it results in a misleadingly high prediction performance.

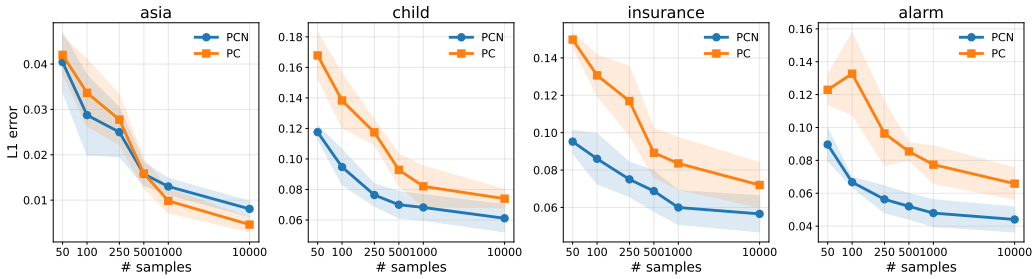

Figure 3. PCs and PCNs Sample Efficiency. Average L1 error for different sample sizes.

(we use identical settings across all experiments, only scaling the architecture with the number of variables), PCNs may be less sensitive to such choices, as they rely on many smaller PCs.

Scaling Due to Tractable Inference. We now employ the same setup but on grid graphs with width W, i.e., W^2 nodes where each node has at most 2 parents: its left and top neighbor. We track L1 error and inference time for a query where the first (top left) and last (bottom right) variables are given as evidence. Table 2 shows that the approximate inference procedure works well for this type of query with distant evidence, and that PCNs scale much better than BNs due to tractable (polynomial) inference time. BN inference time explodes as the treewidth scales with grid width, whereas PCNs still provide results in reasonable time, as the scope of any PC component $\mathcal{C}_i$ scales with the number of parents as opposed to the treewidth.

W	BN Error	PCN Error	BN Time	PCN Time
4	0.21 ± 0.06	0.22 ± 0.06	0.01 ± 0.00	0.38 ± 0.03
5	0.25 ± 0.04	0.23 ± 0.10	0.02 ± 0.01	0.63 ± 0.03
6	0.26 ± 0.09	0.27 ± 0.09	0.03 ± 0.00	0.92 ± 0.03
7	0.32 ± 0.13	0.30 ± 0.14	3.52 ± 4.96	1.22 ± 0.06
8	0.24 ± 0.05	0.25 ± 0.04	314.70 ± 509.33	1.67 ± 0.06

Table 2. Comparison BN and PCN on Grid Graphs. Shown are average L1 errors (scaled so that 1 corresponds to 1% average deviation) and inference times in seconds.

5. Conclusion

PCNs show how PCs can be connected along a DAG to create a probabilistic model that benefits from both the tractability of PCs and the modularity of BNs, with IPF leveraging PC tractability to efficiently update downstream components. While the approximate inference procedure limits practical application, we hope this methodology inspires future research on modular PC approaches.

Acknowledgements

The authors acknowledge the support of the German Science Foundation (DFG) research grant "Tractable Neuro-Causal Models" (KE 1686/8-1). This work has also benefited from the Hessian Ministry of Higher Education, Research, Science and the Arts (HMWK) via hessian.AI and also from the Cluster of Excellence "Reasonable AI" funded by the German Research Foundation (DFG) under Germany's Excellence Strategy—EXC-3057. The Eindhoven University of Technology authors received support from their Department of Mathematics and Computer Science and the Eindhoven Artificial Intelligence Systems Institute.

References

Johan Barthélemy and Thomas Suesse. mipfp: An r package for multidimensional array fitting and simulating multivariate bernoulli distributions. *Journal of Statistical Software*, 2018.

Yvonne MM Bishop, Stephen E Fienberg, and Paul W Holland. *Discrete multivariate analysis theory and practice*. Springer, 1975.

Florian Peter Busch, Moritz Willig, Matej Zečević, Kristian Kersting, and Devendra Singh Dhami. Structural causal circuits: Probabilistic circuits climbing all rungs of pearl's ladder of causation. *Transactions on Machine Learning Research (TMLR)*, 2025.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. *UCLA. URL: http://starai.cs.ucla.edu/papers/ProbCirc20.pdf*, 2020.

W Edwards Deming and Frederick F Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *The Annals of Mathematical Statistics*, 1940.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2), 1999.

Frederick Mosteller. Association and estimation in contingency tables. *Journal of the American Statistical Association*, 1968.

Kevin Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Uncertainty in Artificial Intelligence (UAI)*, 1999.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

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 *International Conference on Machine Learning (ICML)*, 2020.

Tahrima Rahman, Shasha Jin, and Vibhav Gogate. Cutset bayesian networks: A new representation for learning rao-blackwellised graphical models. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Ludger Rüschendorf. Convergence of the iterative proportional fitting procedure. *The Annals of Statistics*, 1995.

Marco Scutari. Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3), 2010. doi: 10.18637/jss.v035.i03.

Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. Conditional sum-product networks: Modular probabilistic circuits via gate functions. *International Journal of Approximate Reasoning (IJAR)*, 140, 2022.

Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 1964.

Benjie Wang and Marta Kwiatkowska. Compositional probabilistic and causal inference using tractable circuit models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023.

John Winn and Christopher M Bishop. Variational message passing. *Journal of Machine Learning Research*, 6(4), 2005.

Matej Zečević, Devendra Dhami, Athresh Karanam, Sriraam Natarajan, and Kristian Kersting. Interventional sum-product networks: Causal inference with tractable probabilistic models. In *Advances in neural information processing systems (NeurIPS)*, 2021.

Nevin Lianwen Zhang and David Poole. A simple approach to bayesian network computations. In *Proceedings of the biennial conference-Canadian society for computational studies of intelligence*. CANADIAN INFORMATION PROCESSING SOCIETY, 1994.

Han Zhao, Mazen Melibari, and Pascal Poupart. On the relationship between sum-product networks and bayesian networks. In *International Conference on Machine Learning (ICML)*, 2015.

Appendix

In this appendix, we first prove the theorems in Appendix A, show an algorithm for the approximate inference procedure in Appendix B, discuss the causal capabilities of PCNs in Appendix C and include a short ablation study on the runtime complexity of IPF in Appendix D. Code for all experiments can be found on https://github.com/olfub/pcn.

A. Proofs

A.1. Tree Scenario

Theorem 2. *For any PCN $\mathcal{C}^+$ where the underlying graph $\mathcal{G}$ is a tree, the scaling factors α can be computed exactly in a single step.*

Proof. We consider the scenario of tree-structured graphs, i.e., any node in the graph has no more than one parent node. We want to show that the actual probability of the marginal distribution of a node given its parent is the same as the marginal probability of that node in the respective PC after applying the scaling factors. The computation for any nodes without parents is trivial and results in $P(x_i) = \mathcal{C}_i(x_i)$. Since there is only one parent, the parent can either have evidence or be marginalized. If there is evidence, Equation 2 simplifies to $\mathcal{C}_i^\alpha(x_i) = P(x_i|e)$, which can be computed using a PC without any modifications. If the parent is marginalized, it remains to be shown that

$$\mathcal{C}_i^\alpha(x_i) = \sum_{z \in \text{val}(Z)} P(x_i|z)q(z) = \sum_{z \in \text{val}(Z)} \frac{P(x_i, z)}{P(z)} q(z) \tag{4}$$

holds for any probability function $q(X)$.

The idea for our scaling factors α entails that all probability values associated with $z = 0$ are multiplied by one scaling factor and all values associated with $z = 1$ are associated with another scaling factor (and extending this to whatever the cardinality of the parent variable is beyond the binary setting). In the PC, this can be achieved in a simple and efficient manner by multiplying all leaves corresponding to the respective value with the appropriate scaling factor (in a slight abuse of notation, we use $\alpha(z_j)$ instead of $\alpha_j(z_j, z_j)$, as there is only a single parent variable). Thus, we define

$$\mathcal{C}_i^\alpha(x_i) := \sum_{z \in \text{val}(Z)} \alpha(z)\mathcal{C}_i(x_i, z). \tag{5}$$

Let us set

$$\alpha(z) = q(z)/\mathcal{C}_i(z). \tag{6}$$

Since $\mathcal{C}_i(z) = P(z)$ and $\mathcal{C}_i(x_i, z) = P(X_i, z)$, it follows that

$$\mathcal{C}_i^\alpha(x_i) = \sum_{z \in \text{val}(Z)} \alpha(z)\mathcal{C}_i(x_i, z) = \sum_{z \in \text{val}(Z)} \frac{q(z)}{\mathcal{C}_i(z)} \mathcal{C}_i(x_i, z) = \sum_{z \in \text{val}(Z)} \frac{q(z)}{P(z)} P(x_i, z) \tag{7}$$

$$= \sum_{z \in \text{val}(Z)} \frac{P(x_i, z)}{P(z)} q(z), \tag{8}$$

proving Equation 4 and with it the correctness of our approach. □

A.2. General Graphs

Theorem 3. *For any PCN $\mathcal{C}^+$ over an arbitrary DAG and assuming positivity of the local joint distributions, the scaling factors α can be computed in polynomial time.*

Proof. For the general case, we again need to consider the general problem

$$\mathcal{C}_i^{\alpha}(x_i) = \sum_{\mathbf{z} \in \mathrm{val}(\mathbf{Z})} P(x_i | \mathbf{e}_{\mathbf{E}^i}, \mathbf{z}) \prod_{z_j \in \mathbf{z}} q_j(z_j). \tag{9}$$

For simplicity, we omit the evidence $\mathbf{e}_{\mathbf{E}^i}$. Note that each evidence could also be written as being part of $\mathbf{z}$, with the corresponding probability q_j being 1 if the evidence is matched and 0 otherwise. Therefore, we will focus on this problem:

$$\mathcal{C}_i^{\alpha}(x_i) = \sum_{\mathbf{z} \in \mathrm{val}(\mathbf{Z})} P(x_i | \mathbf{z}) \prod_{z_j \in \mathbf{z}} q_j(z_j). \tag{10}$$

We first investigate whether the same approach as for the tree scenario is still correct. By setting the scaling factors as $\alpha_j(\mathbf{z}, z_j) = q_j(z_j)/\mathcal{C}_i(z_j)$, we get

$$\mathcal{C}_i^{\alpha}(x_i) = \sum_{\mathbf{z} \in \mathrm{val}(\mathbf{Z})} \left(\prod_{z_j \in \mathbf{z}} \frac{q_j(z_j)}{\mathcal{C}_i(z_j)} \right) \mathcal{C}_i(x_i, \mathbf{z}) = \sum_{\mathbf{z} \in \mathrm{val}(\mathbf{Z})} \left(\prod_{z_j \in \mathbf{z}} \frac{q_j(z_j)}{P(z_j)} \right) P(x_i, \mathbf{z}). \tag{11}$$

We observe that independence between all $\mathbf{Z}$ where $P(\mathbf{z}) = \prod_{z_j \in \mathbf{z}} P(z_j)$ is sufficient for proving that our approach works as intended. We plug this in and obtain

$$\sum_{\mathbf{z} \in \mathrm{val}(\mathbf{Z})} \left(\frac{\prod_{z_j \in \mathbf{z}} q_j(z_j)}{P(\mathbf{z})} \right) P(x_i, \mathbf{z}) = \sum_{\mathbf{z} \in \mathrm{val}(\mathbf{Z})} P(x_i | \mathbf{z}) \prod_{z_j \in \mathbf{z}} q_j(z_j), \tag{12}$$

showing how independence of parent variables enables efficient computation of the correct scaling factors.

Without independence, the single-step ratio update is insufficient. However, the problem of adjusting a joint distribution to match given marginals while staying as close as possible to the original is a well-studied one, often referred to as matrix or tensor scaling (Deming and Stephan, 1940; Bishop et al., 1975; Sinkhorn, 1964). Iterative proportional fitting (IPF), or for more than two variables, multidimensional IPF (Barthélemy and Suesse, 2018), solves exactly this problem by yielding the unique distribution that matches the target marginals while minimizing the KL divergence to the original (Rüschendorf, 1995; Mosteller, 1968). Each step of IPF corresponds to applying the ratio update from the tree case to a single parent variable, which is exactly the operation shown above. Under the assumption that the original joint distribution is strictly positive, IPF is guaranteed to converge to the correct solution.

It remains to show that IPF can be executed in polynomial time. Consider a node with p parent variables, each with cardinality at most d. One iteration of IPF consists of updating the scaling factors for each of the p parents in turn. Each such update requires computing the marginal of $\mathcal{C}_i$ for each value of the respective parent variable, i.e., d marginals per parent, after which the scaling factors are applied as multiplicative corrections on the leaf nodes at negligible cost. This amounts to dp PC inference calls per iteration.[4] Let $S(p)$ denote the cost of a single inference call for a PC over $p+1$ variables. Since PC inference is linear in the circuit size, $S(p)$ is polynomial in p for standard PC architectures. For the number of iterations $k(p)$, IPF is known to exhibit linear convergence under positivity (Rüschendorf, 1995), meaning each iteration reduces the error by a constant factor. In practice, k grows modestly with p, as confirmed by our ablation in Appendix D. The per-node complexity is therefore $O(d \cdot p \cdot S(p) \cdot k(p))$, which is polynomial in p and d. This stands in contrast to the $O(d^p)$ scaling of variable elimination in BNs, which is what makes inference in PCNs feasible for graphs where BN inference is not. □

4. Without a PC, i.e., using an explicit joint probability table, computing these marginals would itself scale exponentially in p. This is precisely where the tractability of PCs becomes essential: each additional parent only increases the cost by a constant multiplicative factor.

Algorithm 1 Factorized Inference Procedure

Input: DAG G over n variables, Evidence $\mathbf{e} \in \text{val}(\mathbf{E})$ with $\mathbf{E} \subset \mathbf{X}$
$p_{\text{query}} \leftarrow 1$

{Iterate over all variables}
for $i = 1$ **to** n **do**
  $\mathbf{Z} \leftarrow \text{Pa}(X_i) \setminus \mathbf{E}$ {All marginalized parents}
  $\mathcal{E} \leftarrow \text{Pa}(X_i) \cap \mathbf{E}$ {All parents with evidence}

  if $X_i \in \mathbf{E}$ **then**
    $q(X_i = e_i) \leftarrow 1$
    $q(X_i = (1 - e_i)) \leftarrow 0$
    $p_{\text{local}} = \sum_{\mathbf{z} \in \text{val}(\mathbf{Z})} P(X_i = e_i | \mathbf{e}_{\mathcal{E}}, \mathbf{z}) \prod_{z_j \in \mathbf{z}} q_j(z_j)$
    $p_{\text{query}} \leftarrow p_{\text{query}} \cdot p_{\text{local}}$
  else
    $q(X_i = 0) \leftarrow \sum_{\mathbf{z} \in \text{val}(\mathbf{Z})} P(X_i = 0 | \mathbf{e}_{\mathcal{E}}, \mathbf{z}) \prod_{z_j \in \mathbf{z}} q_j(z_j)$
    $q(X_i = 1) \leftarrow \sum_{\mathbf{z} \in \text{val}(\mathbf{Z})} P(X_i = 1 | \mathbf{e}_{\mathcal{E}}, \mathbf{z}) \prod_{z_j \in \mathbf{z}} q_j(z_j)$
  end if
end for
Return: p_{query}

B. Approximate Inference

We include the algorithm on the approximate inference procedure (akin to mean field approximation (Jordan et al., 1999)) in Algorithm 1 for binary variables. The extension to different cardinalities follows analogously.

Let us briefly compare to other message-passing approaches. Belief propagation (BP) (Pearl, 1988) was originally introduced for BNs. In spirit, it is quite similar to the inference procedure of PCNs, as in both cases, information is passed between neighboring variables. However, BP guarantees exact inference on polytrees using a two-pass inference procedure and loopy belief propagation (LBP) (Murphy et al., 1999) constitutes an extension of BP that supports arbitrary DAGs with undirected cycles, on which inference is approximate. Currently, the PCN approach is optimized for efficiency, and the single-pass inference procedure avoids having to visit nodes multiple times. Of course, this comes at the cost of approximate results, which by its nature is more similar to variational message passing (VMP) (Winn and Bishop, 2005), although VMP does not restrict itself to a single pass through all variables. In both cases, information on the marginal distribution of the current variable is propagated and the partial probabilities resulting from this (q in the algorithm) are then multiplied assuming independence between them (following the mean field assumption). The idea is that the messages transferred before already propagated the required information. However, as discussed previously, this results in approximate behavior for specific graph structures and queries in which this leads to some correlations being ignored. PCNs also align with VMP more closely in that they always return a solution (the inference procedure only visits every node once and IPF is guaranteed to converge under positivity), while LBP can oscillate.

C. On Interventions

If the underlying DAG $\mathcal{G}$ is causal, it is tempting to view a PCN as a causal model. Indeed, a PCN encodes the same factorization as a causal Bayesian network. However, it does not correctly support interventions on arbitrary variables. We consider three cases.

Size	Iterations	Error	Runtime
2	3.00 ± 0.00	1.97 ± 0.27	0.09 ± 0.00
4	4.50 ± 0.50	2.99 ± 0.52	0.23 ± 0.02
8	5.00 ± 0.00	3.57 ± 0.10	0.58 ± 0.00
16	6.00 ± 0.00	6.96 ± 3.34	1.58 ± 0.00
32	8.50 ± 0.50	12.97 ± 3.29	5.02 ± 0.22
64	12.50 ± 0.50	25.40 ± 3.19	16.28 ± 0.65
128	27.00 ± 0.00	87.37 ± 5.36	77.23 ± 1.16

Table 3. IPF Statistics. Errors are scaled by 10^{-8}, with the convergence threshold set to $100 \cdot 10^{-8}$. "Size" refers to the number of parent variables. All experiments use cardinality 4. Runtime in seconds.

Root Variables. Intervening on a root variable is straightforward. Since a root has no parents, the intervention simply replaces the marginal q with the interventional distribution, which is then passed down the outgoing edges. All downstream computations remain correct.

Tree-Structured Graphs. In a tree, the intervened variable ignores its parent's distribution and passes down its intervened q to its children. Each child can then compute its updated distribution as in the root case, since it has only one parent.

General Multi-Parent Case. For arbitrary graphs, interventions are generally not handled correctly. Consider three nodes X, Y, Z with edges $Z \to X$, $Z \to Y$, and $X \to Y$. Intervening on X with $do(X = \hat{x})$ removes the edge $Z \to X$, changing the factorization from $P(X, Y, Z) = P(Z)P(X|Z)P(Y|X, Z)$ to $P^{do}(X, Y, Z) = P(Z)P(Y|X = \hat{x}, Z)\mathbf{1}[X = \hat{x}]$. However, the local PC $\mathcal{C}_Y$ still encodes the *observational* joint $P(Y, X, Z)$, which includes the correlation between X and Z induced by the edge $Z \to X$. When we apply scaling factors to update $\mathcal{C}_Y$ and marginalize over Z, the tractable marginalization procedure of the PC implicitly uses this observational correlation. The intervention should have removed it, but since $\mathcal{C}_Y$ was trained on observational data, it cannot distinguish between the pre- and post-intervention distributions over its parents. Correctly computing the interventional query would require enumerating all parent configurations explicitly, defeating the purpose of using PCs for tractable marginalization.

D. IPF Ablation

In this experiment, we isolate the IPF procedure to investigate its runtime and convergence behavior independently of the rest of the PCN pipeline. To this end, we generate random joint probability tables over $p = 2, 4, 8, \ldots, 128$ variables, each with cardinality 4, and apply IPF to fit new target marginals. We record the number of iterations until convergence, the final error, and the wall-clock runtime. Results are shown in Table 3.

All runs terminate correctly, with the final error falling below the specified convergence threshold. The number of iterations grows modestly with the problem size, confirming the linear convergence property discussed in Appendix A.2. Runtime scales superlinearly but remains far below exponential: even for 128 variables (where an explicit joint table would have 4^{128} entries and could never be processed by brute-force approaches), IPF with tractable PC marginalization terminates in under 80 seconds. This empirically supports the polynomial complexity analysis from our proof.