Tractable Uncertainty for Structure Learning

Benjie Wang [1] **Matthew Wicker** [1] **Marta Kwiatkowska** [1]

Abstract

Bayesian structure learning allows one to capture uncertainty over the causal directed acyclic graph (DAG) responsible for generating given data. In this work, we present Tractable Uncertainty for STructure learning (TRUST), a framework for approximate posterior inference that relies on probabilistic circuits as the representation of our posterior belief. In contrast to sample-based posterior approximations, our representation can capture a much richer space of DAGs, while also being able to tractably reason about the uncertainty through a range of useful inference queries. We empirically show how probabilistic circuits can be used as an augmented representation for structure learning methods, leading to improvement in both the quality of inferred structures and posterior uncertainty. Experimental results on conditional query answering further demonstrate the practical utility of the representational capacity of TRUST.

1. Introduction

Understanding the causal and probabilistic relationship between variables of underlying data-generating processes can be a vital step in many scientific inquiries. Such systems are often represented by causal Bayesian networks (BNs), probabilistic models with structure expressed using a directed acyclic graph (DAG). The basic task of structure learning is to identify the underlying BN from a set of observational data, which, if successful, can provide useful insights about the relationships between random variables and the effects of potential interventions. However, even under strong assumptions such as causal sufficiency and faithfulness, it is typically impossible to identify a single causal DAG from purely observational data. Further, while consistent methods exist for producing a point estimate DAG in the limit of infinite data (Chickering, 2002), in practice, when data is scarce many BNs can fit the data well. It thus becomes vitally important to quantify the uncertainty over causal structures, particularly in safety-critical scenarios.

Bayesian methods for structure learning tackle this problem by defining a prior and likelihood over DAGs, such that the posterior distribution can be used to reason about the uncertainty surrounding the learned causal edges, for instance by performing Bayesian model averaging. Unfortunately, the super-exponential space of DAGs makes both representing and learning such a posterior extremely challenging. A major breakthrough was the introduction of order-based representations (Friedman & Koller, 2003), in which the state space is reduced to the space of topological orders.

Unfortunately, the number of possible orders is still factorial in the dimension d, making it infeasible to represent the posterior as a tabular distribution over orders. Approximate Bayesian structure learning methods have thus mostly sought to approximate the distribution using samples of DAGs or orders (Lorch et al., 2021; Agrawal et al., 2018). However, such sample-based representations have very limited coverage of the posterior, restricting the information they can provide. Consider, for instance, the problem of finding the most probable graph extension, given an arbitrary set of required edges. Given the super-exponential space, even a large sample may not contain even a *single* order consistent with the given set of edges, making answering such a query impossible.

A natural question, therefore, is whether it is possible to more compactly represent distributions over orders (and thus DAGs) while retaining the ability to perform useful inference queries tractably (in the size of the representation). We answer in the affirmative, by proposing a novel representation, OrderSPNs, for distributions over orders and graphs. Under the assumption of order-modularity, we show that OrderSPNs form a natural and flexible approximation to the target distribution. The key component is the encoding of hierarchical conditional independencies into the form of a sum-product network (SPN) (Poon & Domingos, 2011), a well-known type of tractable probabilistic circuit. Based on this, we develop an approximate Bayesian structure learning framework, TRUST, for efficiently querying OrderSPNs and learning them from data. Empirical results corroborate the increased representational capacity and coverage

[1]Department of Computer Science, University of Oxford, Oxford, United Kingdom. Correspondence to: Benjie Wang <benjie.wang@cs.ox.ac.uk>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

of TRUST, while also demonstrating improved performance compared to competing methods on standard metrics. Our contributions are as follows:

- We introduce a novel representation, OrderSPNs, for Bayesian structure learning based on sum-product networks. In particular, we exploit exact hierarchical conditional independencies present in order-modular distributions. This allows OrderSPNs to express distributions over a potentially exponentially larger set of orders relative to their size.
- We show that OrderSPNs satisfy desirable properties that enable tractable and exact inference. In particular, we present methods for computation of a range of useful inference queries in the context of structure learning, including marginal and conditional edge probabilities, graph sampling, maximal probability graph completions, and pairwise causal effects. We further provide complexity results for these queries; notably, all take at most linear time in the size of the circuit.
- We demonstrate how our method, TRUST, can be used to approximately learn a posterior over DAG structures given observational data. In particular, we utilize a two-step procedure, in which we (i) propose a structure for the SPN using a seed sampler; and (ii) optimize the parameters of the SPN in a variational inference scheme. Crucially, the tractable properties of the circuit enable the ELBO and its gradients to be computed *exactly* without sampling.

2. Related Work

Bayesian approaches to structure learning infer a distribution over possible causal graphs. Such distributions can then be queried to extract useful information, such as estimating causal effects, which can aid investigators in understanding the domain, or to plan interventions (Castelletti & Consonni, 2021; Maathuis et al., 2010; Viinikka et al., 2020). Unfortunately, due to the super-exponential space, exact Bayesian inference methods for structure learning do not scale beyond $d = 20$ (Koivisto & Sood, 2004; Koivisto, 2006). As a result, there has been much interest in approximate methods, most notably performing MCMC sampling over the space of DAGs (Madigan et al., 1995; Giudici & Castelo, 2003). Notable works in this direction include those by Friedman & Koller (2003), who operate over the much smaller space and smoother posterior landscape of topological orders, Tsamardinos et al. (2006), who reduce the state-space by considering conditional independence, and Kuipers et al. (2018), who reduce the per-step computational cost associated with scoring.

Alternatively, some recent works have applied variational inference to the Bayesian structure learning problem, where an approximate distribution over graphs is obtained by optimizing over some variational family describing distributions over graphs. Unfortunately, existing representations are typically not very tractable; Annadani et al. (2021); Cundy et al. (2021) utilize neural autoregressive and energy-based models respectively, while Lorch et al. (2021) employ sample-based approximations and particle variational inference (Liu & Wang, 2016). This presents significant challenges for gradient-based optimization, since the reparameterization trick is not applicable for the discrete space of graphs. Further, downstream inference queries can only be estimated approximately through sampling. In contrast, our proposed variational family based on tractable models makes optimization and inference exact and efficient.

Probabilistic circuits (Choi et al., 2020) are a general class of tractable probabilistic models which represent distributions using computational graphs. The key advantage of circuits, compared to other probabilistic models such as Bayesian networks, VAEs (Kingma & Welling, 2013), or GANs (Goodfellow et al., 2014), is their ability to perform tractable and exact inference, for instance, computing marginal probabilities. While the typical use case is to learn a distribution over a set of variables from data (Gens & Pedro, 2013; Rooshenas & Lowd, 2014), in this work we consider learning a circuit to approximate a given (intractable) posterior distribution over the space of DAGs, thus requiring different structure and parameter learning routines.

3. Background

3.1. Bayesian Structure Learning

Bayesian Networks A Bayesian network (BN) $\mathcal{N} = (G, \Theta)$ is a probabilistic model $p(\boldsymbol{X})$ over d variables $\boldsymbol{X} = \{X_1, ...X_d\}$, specified using the directed acyclic graph (DAG) G, which encodes conditional independencies in the distribution p, and Θ, which parameterizes the mechanisms (conditional probability distributions) constituting the Bayesian network. The conditional probabilities take the form $p(X_i|\text{pa}_G(X_i), \Theta_i)$, giving rise to the joint data distribution:

$$p(\boldsymbol{X}|G, \Theta) = \prod_i p(X_i|\text{pa}_G(X_i), \Theta_i)$$

where $\text{pa}_G(X)$ denotes the parents of X in G. One of the most popular types of BN model is the linear Gaussian model, under which the distribution is given by the structural equation $\boldsymbol{X} = \boldsymbol{X}B + \boldsymbol{\epsilon}$, where $B \in \mathbb{R}^{d\times d}$ is a matrix of real weights parameterizing the mechanisms, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{b}, \Sigma)$ where $\boldsymbol{b} \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d\times d}_{\geq 0}$ is a diagonal matrix of noise variances. In particular, for a given DAG G, we have $B_{ij} = 0$ for all i, j such that i is not a parent of j in G.

Whereas Bayesian networks typically only express probabilistic (conditional independence) information, causal

Bayesian networks (Spirtes et al., 2000; Pearl, 2009) are additionally imbued with a causal interpretation, where, intuitively, the directed edges in G represent direct causation. More formally, causal BNs can predict the effect (change in joint distribution) of interventions in the system, where some mechanism is changed, for instance by setting a variable X to some value x independent of its parents.

Bayesian Structure Learning *Structure learning* (Koller & Friedman, 2009; Glymour et al., 2019) is the problem of learning the DAG G of the (causal) Bayesian network responsible for generating some given data $\mathcal{D}$. Typically, strong assumptions are required for structure learning; in this work, we make the common assumption of causal sufficiency, meaning that there are no latent (unobserved) confounders. Even given this assumption, it is often not possible to reliably infer the causal DAG, whether due to limited data, or non-identifiability within a Markov equivalence class. Instead of learning a single DAG, Bayesian approaches to structure learning express uncertainty over structures in a unified fashion, through defining a prior $p_{pr}(G)$ and (marginal) likelihood $p_{lh}(\mathcal{D}|G)$ over directed graphs G.

A common assumption is that the prior and likelihood scores are *modular*, that is, they decompose into a product of terms for each mechanism G_i of the graph, where G_i specifies the set of parents of variable i in G. In such cases, the overall posterior decomposes as:

$$\begin{aligned} p_G(G|\mathcal{D}) &\propto \mathbb{1}_{\text{DAG}(G)} p_{pr}(G) p_{lh}(\mathcal{D}|G) \\ &= \mathbb{1}_{\text{DAG}(G)} \prod_i p_{pr}(G_i) p_{lh}(\mathcal{D}_i|G_i) \end{aligned}$$

The acyclicity constraint $\mathbb{1}_{\text{DAG}(G)}$ induces correlations between different mechanisms and presents the key computational challenge for posterior inference. The prior and likelihood can be chosen based on knowledge about the domain; for example, for linear Gaussian models, we can employ the BGe score (Kuipers et al., 2014), a closed form expression for the marginal likelihood of a variable given its parent set (marginalizing over weights of the linear model). The prior is typically chosen to penalize larger parent sets.

3.2. Sum-Product Networks

Sum-product networks (SPN) are probabilistic circuits over a set of variables $\boldsymbol{V}$, represented using a rooted DAG consisting of three types of nodes: leaf, sum and product nodes. These nodes can each be viewed as representing a distribution over some subset of variables $\boldsymbol{W} \subseteq \boldsymbol{V}$, where the root node specifies an overall distribution $q_\phi(\boldsymbol{V})$. Each leaf node L specifies an input distribution over some subset of variables $\boldsymbol{W} \subseteq \boldsymbol{V}$, which is assumed to be tractable. Each product node P multiplies the distributions given by its children, i.e., $P = \prod_{C_i \in ch(P)} C_i$, while each sum node is defined by a weighted sum of its children, i.e., $T = \sum_{C_i \in ch(S)} \phi_i C_i$. The weights ϕ_i for each sum node satisfy $\phi_i > 0, \sum \phi_i = 1$, and are referred to as the *parameters* of the SPN. The *scope* of a node N denotes the set of variables N specifies a distribution over, and can be defined recursively as follows. Each leaf node N has scope $sc(N) = \{V\}$, where V is the variable it specifies its distribution over, and each product or sum node N has scope $sc(N) = \cup_{C \in ch(N)} sc(C)$.

SPNs provide a computationally convenient representation of probability distributions, enabling efficient and exact inference for many types of queries, given certain structural properties (Poon & Domingos, 2011; Peharz et al., 2014):

- A SPN is *complete* if, for every sum node T, and any two children C_1, C_2 of T, it holds that $sc(C_1) = sc(C_2)$. In other words, all the children of T, and thus T itself, have the same scope.
- A SPN is *decomposable* if, for each product node P, and any two children C_1, C_2 of P, it holds that $sc(C_1) \cap sc(C_2) = \emptyset$. In other words, the scope of P is partitioned by its children.
- A SPN is *deterministic* if, for each sum node T, and any instantiation $\boldsymbol{w}$ of its scope $sc(T) = \boldsymbol{W}$, at most one of its children $C_i(\boldsymbol{w})$ evaluates to a non-zero probability.

Given completeness and decomposability, marginal inference becomes tractable, that is, we can compute $q_\phi(\boldsymbol{W})$ for any $\boldsymbol{W} \subseteq \boldsymbol{V}$ in linear time in the number of edges of the SPN. Conditional probabilities can be computed as the ratio of two marginal probabilities. If the SPN additionally satisfies determinism, MPE inference, i.e., $\max_{\boldsymbol{v}:\boldsymbol{W}=\boldsymbol{w}} q_\phi(\boldsymbol{v})$, also becomes tractable (Peharz et al., 2017).

4. Tractable Representations for Bayesian Structure Learning

In this work, we consider Bayesian structure learning over the joint space of topological orders and DAGs, where each order σ is a permutation of $\{1, ..., d\}$. Let $\sigma^{<i}$ be the set of variables preceding variable i in σ. We say that a parent set G_i is consistent with an order σ if $G_i \subseteq \sigma^{<i}$, and that graph G is consistent if all of its parent sets are consistent (written $G \models \sigma$). It follows that any DAG is consistent with at least one order, and further any directed graph consistent with an order must be acyclic. Thus we can specify a joint distribution over orders and DAGs as follows:

$$\begin{aligned} p(\sigma, G|\mathcal{D}) &\propto p_G(G|\mathcal{D}) \mathbb{1}_{G \models \sigma} \\ &= p_{pr}(G) p_{lh}(\mathcal{D}|G) \prod_i \mathbb{1}_{G_i \subseteq \sigma^{<i}} \end{aligned}$$

Notice that the marginal $p(G|\mathcal{D})$ is not the same as $p_G(G|\mathcal{D})$, as p will favour graphs which are consistent with

more orders. This imparts a bias for learning with respect to p_G. On the other hand, the space of orders is much smaller than the space of DAGs, enabling more efficient exploration of the distribution (Friedman & Koller, 2003).

In the case where the prior and likelihood are modular, the resulting distribution p is said to be *order-modular*. In this case, $p_G(G)$ factorizes as $p_G(G) = \prod_i p_{G_i}(G_i)$, giving:

$$p(\sigma, G) \propto p_G(G)\mathbb{1}_{G\models\sigma} = \prod_i p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq\sigma^{<i}} \quad (1)$$

where we have omitted the dependence on the dataset and write $p(\sigma, G)$ for the Bayesian posterior.

4.1. Hierarchical CIs

Unfortunately, the representation of the order-modular distribution in Equation 1 is not *tractable*: we cannot easily sample from it, nor can we efficiently deduce, for instance, the marginal probability of a given edge. Our goal is thus to obtain a representation approximating this distribution which does possess tractable properties. The key idea is that by exploiting exact conditional independences (CIs) in the distribution, we can *hierarchically* break the approximation of the original distribution into smaller subproblems.

To illustrate this, we first define some notation. Given any variable subset $S \subseteq \{1, ..., d\}$, let σ_S denote an ordering (permutation) over variables in S, and $G_S \triangleq \{G_i : i \in S\}$ denote the set of parent sets for each variable i in S.

Now, suppose we partition the set of BN variables $\{1, ..., d\}$ into two subsets (S_1, S_2), and consider conditioning on the event that all variables in S_1 come before S_2 in the ordering, that is, the order partitions as $\sigma = (\sigma_{S_1}, \sigma_{S_2})$. In this case, the conditional distribution can be written as:

$$p(\sigma, G|\sigma = (\sigma_{S_1}, \sigma_{S_2})) \propto \prod_i p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq(\sigma_{S_1},\sigma_{S_2})^{<i}}$$
$$= \prod_{i\in S_1} p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq\sigma_{S_1}^{<i}} \prod_{i\in S_2} p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq S_1\cup\sigma_{S_2}^{<i}}$$

Notice that the distribution has factorized into two terms, which respectively include only (σ_{S_1}, G_{S_1}) and (σ_{S_2}, G_{S_2}). In fact, if we define the following (unnormalized) distribution over (σ_{S_2}, G_{S_2}):

$$\tilde{p}_{S_1,S_2}(\sigma_{S_2}, G_{S_2}) \triangleq \prod_{i\in S_2} p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq S_1\cup\sigma_{S_2}^{<i}}$$

the previous factorization can be written as:

$$p(\sigma, G|\sigma = (\sigma_{S_1}, \sigma_{S_2}))$$
$$\propto \tilde{p}_{\emptyset,S_1}(\sigma_{S_1}, G_{S_1})\tilde{p}_{S_1,S_2}(\sigma_{S_2}, G_{S_2})$$

Thus, conditional on $\sigma = (\sigma_{S_1}, \sigma_{S_2})$, we have split the distribution over (σ, G) into distributions over only (σ_{S_1}, G_{S_1}) and (σ_{S_2}, G_{S_2}) respectively.

Now, let us consider arbitrary disjoint subsets $S_1, S_2 \subseteq \{1, ..., d\}$. Then $\tilde{p}_{S_1,S_2}$ is a distribution over σ_{S_2}, G_{S_2}. We can apply a similar method to conditionally decompose $\tilde{p}_{S_1,S_2}(\sigma_{S_2}, G_{S_2})$ into distributions over S_{21}, S_{22}, where S_{21}, S_{22} partition S_2, given by the following Proposition:

Proposition 1. *Let $p(\sigma, G) \propto p_G(G)\mathbb{1}_{G\models\sigma}$ be an order-modular distribution. Suppose that S_1, S_2 are any disjoint subsets of the variables $\{1, ..., d\}$, and let (S_{21}, S_{22}) be a partition of S_2. Then the following CI holds:*

$$\tilde{p}_{S_1,S_2}(\sigma_{S_2}, G_{S_2}|\sigma_{S_2} = (\sigma_{S_{21}}, \sigma_{S_{22}}))$$
$$\propto \tilde{p}_{S_1,S_{21}}(\sigma_{S_{21}}, G_{S_{21}})\tilde{p}_{S_1\cup S_{21},S_{22}}(\sigma_{S_{22}}, G_{S_{22}})$$

These conditional independencies suggest an approximation strategy: select K partitions (S_1, S_2) of $\{1, ..., d\}$ to form the approximation and, conditional on a partition, then independently approximate the resulting distributions $\tilde{p}_{\emptyset,S_1}(\sigma_{S_1}, G_{S_1}), \tilde{p}_{S_1,S_2}(\sigma_{S_2}, G_{S_2})$, which are simpler problems of dimensions $|S_1|, |S_2|$, respectively. Using Proposition 1, this can be done recursively, until we obtain distributions where S_2 is a singleton $\{i\}$, where:

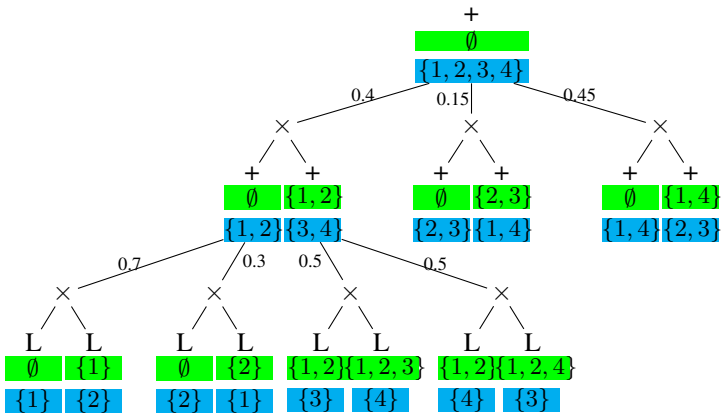
$$\tilde{p}_{S_1,\{i\}}(\sigma_{\{i\}}, G_i) = p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq S_1\cup\sigma_{\{i\}}^{<i}}$$
$$= p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq S_1}$$

4.2. OrderSPNs

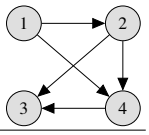
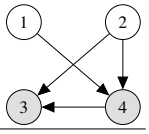
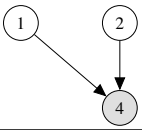
The decomposition process can be viewed as a rooted tree, where we alternate between nodes that select partitions, and those which decompose the conditional distribution. This naturally induces a sum-product network structure, which we formalize in the following definition:

Definition 1. *An OrderSPN q_ϕ is a sum-product network over (σ, G) with the following structure:*

- *Each leaf node L is associated with $(S_1, \{i\})$, for some subset S_1 of $\{1, ..., d\}$ and $i \notin S_1$, and has scope $sc(L) = (\sigma_{\{i\}}, G_i)$. In addition, the leaf node distribution must have support only over graphs $G_i \subseteq S_1$.*
- *Each sum node T is associated with two disjoint subsets (S_1, S_2) of $\{1, ..., d\}$, where $|S_2| > 1$, and has scope $sc(T) = (\sigma_{S_2}, G_{S_2})$. It has K_T children and weights $\phi_{T,i}$ for $i = 1, ..., K_T$, where the i^{th} child is a product node P associated with $(S_1, S_{21,i}, S_{22,i})$ for some partition $(S_{21,i}, S_{22,i})$ of S_2.*
- *Each product node P is associated with three disjoint subsets (S_1, S_{21}, S_{22}) of $\{1, ...d\}$, and has scope $sc(P) = (\sigma_{S_{21}\cup S_{22}}, G_{S_{21}\cup S_{22}})$, where $\sigma_{S_{21}\cup S_{22}}$ takes the form $(\sigma_{S_{21}}, \sigma_{S_{22}})$. It has two children, where the first child is associated with (S_1, S_{21}), and the second with $(S_1 \cup S_{21}, S_{22})$. These children are either sum-nodes or leaves.*



(a) Regular OrderSPN, with expansion factors $\boldsymbol{K} = (3, 2)$. Each sum/leaf node is labelled with its associated (S_1, S_2). Only one expansion beyond the first level is shown for clarity.

(S_1, S_2)	Example Orders	Example Graph
($\emptyset$, $\{1,2,3,4\}$)	**(1, 2, 4, 3)** (2, 3, 1, 4) (4, 1, 3, 2)	
($\{1,2\}$, $\{3,4\}$)	(3, 4) **(4, 3)**	
($\{1,2\}$, $\{4\}$)	**(4)**	

(b) Example orders and graphs for 3 sum/leaf nodes. Graphs only include parent sets of S_2 (filled) variables.

Figure 1. Example of regular OrderSPN for $d = 4$. Best viewed in color.

We can interpret each sum (or leaf) node T associated with (S_1, S_2) as representing a distribution over DAGs over variables S_2, where these variables can additionally have parents from among S_1. In other words, every sum node represents a (smaller) Bayesian structure learning problem over a set of variables S_2 and a set of potential confounders S_1.

In practice, we organize the SPN into alternating layers of sum and product nodes, starting with the root sum node. In the j^{th} sum layer, we create a fixed number K_j of children for each sum node T in the layer. Further, for each child i of each sum node T, we choose $(S_{21,i}, S_{22,i})$ such that $|S_{21,i}| = \lfloor \frac{|S_2|}{2} \rfloor$, $|S_{22,i}| = \lceil \frac{|S_2|}{2} \rceil$, and further require that the partitions are distinct for different children i of T. Under these conditions, the OrderSPN will have $\lceil \log_2(d) \rceil$ sum (and product) layers. This ensures compactness of the representation, and enables efficient tensorized computation over layers. We call such OrderSPNs *regular*, and the associated list $\boldsymbol{K}$ of numbers of children for each layer are called the *expansion factors*. An example of a regular OrderSPN is shown in Figure 1. At the top sum layer, we create a child for $K_1 = 3$ different partitions of $\{1, 2, 3, 4\}$ into equally sized subsets, each of which has an associated weight. Sum and product layers alternate until we reach the leaf nodes.

The leaf nodes L represent distributions over some column of the graph: if L is associated with (S_1, i), then it expresses a distribution over the parents G_i of variable i. The interpretation of S_1 is that this distribution should only have support over sets $G_i \subseteq S_1$. This restriction ensures that OrderSPNs are consistent, in the sense that they represent distributions over valid (σ, G) pairs (in particular, all graphs are acyclic):

Proposition 2. *Let q_ϕ be an OrderSPN. Then, for all pairs (σ, G) in the support of an OrderSPN, it holds that $G \models \sigma$.*

By design, (regular) OrderSPNs satisfy the standard SPN properties that make then an efficient representation for inference, which we show in the following Proposition. In the following sections, we use these properties for query computation, as well as for learning the SPN parameters.

Proposition 3. *Any OrderSPN is complete and decomposable, and regular OrderSPNs are additionally deterministic.*

4.3. Leaf Distributions

Given a leaf node associated with (S_1, i), corresponding to a distribution on G_i, the only restriction imposed by the definition is that the distribution has support only on graphs $G_i \subseteq S_1$. Given that we are approximating an order-modular distribution $p(\sigma, G) \propto \prod_i p_{G_i}(G_i) \mathbb{1}_{G_i \subseteq \sigma^{<i}}$, the natural choice of (unnormalized) leaf distributions is $p_{S_1,\{i\}}(\sigma_{\{i\}}, G_i) \propto p_{G_i}(G_i) \mathbb{1}_{G_i \subseteq S_1}$; we provide formal justification for this choice in Proposition 7 in the Appendix.

For tractable inference on the overall distribution over (σ, G), we require that the leaf distributions can be computed tractably. In particular, we will be interested in three types of tasks: marginal/conditional inference, MPE inference, and (conditional) sampling. To formalize this, let $a_{i,j}$ be a Boolean variable indicating whether $j \in G_i$, i.e., j is a parent of i. Further, let c_i be any logical conjunction of the corresponding positive or negative literals, i.e. $a_{i,j}$ or $\neg a_{i,j}$. For instance, $c_i = a_{i,0} \wedge a_{i,1} \wedge \neg a_{i,2}$ represents the event that $0, 1$ are parents of i, but not 2. Then, the task of marginal inference is to evaluate the probability $p_{S_1,\{i\}}(c_i = 1)$. Conditional inference is the task of $p_{S_1,\{i\}}(c_i = 1 | c'_i = 1)$ for two conjunctions c_i, c'_i. MPE inference is $\max_{G_i} p_{S_1,\{i\}}(G_i | c_i = 1)$, while conditional sampling is the task of sampling from $p_{S_1,\{i\}}(G_i | c_i = 1)$.

Unfortunately, these inference queries are intractable to compute without further assumptions. Following previous work (Kuipers et al., 2018; Viinikka et al., 2020), we limit the parents of each variable i to a fixed set of candidates parents $C_i \subset \{1, ..., d\} \setminus \{i\}$, where the size of $|C_i|$ is chosen to be manageable (around 16). These candidate sets are chosen to maximize the coverage of the distribution mass. Given this, we approximate $\tilde{p}_{S_1,\{i\}}(G_i) \approx p_{G_i}(G_i)\mathbb{1}_{G_i \subseteq S_1 \cap C_i}$.

Given this approximation, we can then perform a precomputation taking $O(3^{|C_i|})$ time and space complexity, after which all of these queries require just a $O(1)$ lookup, except conditional sampling, which takes time $O(|C_i|)$. We provide further details of the method in Appendix B.

4.4. Tractable Queries

In general, being able to compute inference queries tractably individually for single-node distributions is not sufficient to perform inference on the overall distribution over DAGs. While previous works have tackled this problem by sampling single DAGs or orders, our key insight is that we can leverage the tractable properties of SPNs to hierarchically aggregate over order components. We now characterize the classes of queries that can be computed tractably for OrderSPNs, and their interpretation in the context of structure learning. Below we will write $q_\phi(G)$ to denote the marginal of G in $q_\phi(\sigma, G)$, and denote the size of the SPN by M.

Marginal and conditional inference Let c_i, c_i' be conjunctions over the graph column G_i. Then the *marginal inference* problem is to compute $q_\phi(\bigwedge_{i=1}^d c_i)$. This can be interpreted as the probability of any arbitrary combination of edges (direct causal relations) simultaneously being present. It is well known that marginal/conditional inference queries can be computed exactly for a complete and decomposable SPN in linear time in the size of the circuit (Poon & Domingos, 2011). Since marginal inference for the individual leaves requires just a constant-time lookup, the overall complexity is $O(M)$.

MPE inference MPE inference is the problem of finding the most likely instantiation of the variables, given some evidence. More precisely, we wish to compute $\max_G q_\phi(G|\bigwedge_{i=1}^d c_i)$, which allows us to, for instance, find the most likely extension of a partially specified DAG. This is tractable (in linear-time) provided that the SPN is *deterministic* (Choi & Darwiche, 2017). As MPE inference on the individual leaves requires just a constant-time lookup, the overall complexity is once again $O(M)$.

Sampling Unconditional sampling from the OrderSPN is straightforward and efficient; we traverse the SPN top-down, choosing one child of each sum-node, and all children of each product-node, until we reach the leaf nodes, taking linear time in d. Coupled with the cost of sampling the leaf-node distributions, the overall complexity is $O(d \max_i |C_i|)$ per sample. Conditional sampling is more involved, and requires an $O(M)$ bottom-up computation which updates the SPN weights/probabilities according to the evidence, before sampling via top-down traversal (Vergari et al., 2019).

Query	Time	Space
Marginal/Conditional	$O(M)$	$O(M)$
MPE	$O(M)$	$O(M)$
Sampling	$O(d^2)$	$O(d^2)$
Conditional Sampling	$O(d^2 + M)$	$O(d^2 + M)$
Pairwise Causal Effects	$O(d^3 M)$	$O(d^2 M)$

Table 1. Per-query complexity for OrderSPNs, for d variables and OrderSPN of size M

Causal effects We now turn to the computation of other types of queries specific to the Bayesian network setting. In the well-studied case of linear Gaussian Bayesian networks, one of the most important quantities for causal inference is pairwise causal effects, first studied by Wright (1934) as the "method of path coefficients". In particular, for a given graph G and weights B, the causal effect of X_i on X_j, written $E_{ij}(B)$, is given by summing the weight of all directed paths from i to j, where the weight of a path is given by the product of the weights of the edges along that path. Notice that, in cases where i is not an ancestor of j, $E_{ij}(B) = 0$. Now, a priori, when we do not know the graph or weights, the causal effect is a random variable given by:

$$E_{ij} = \sum_{\pi \in F(\{1,\dots,d\} \setminus \{i,j\})} B_{i,\pi_1} B_{\pi_{|\pi|},j} \prod_{i=1}^{|\pi|-1} B_{\pi_i,\pi_{i+1}}$$

where $F(S)$ is the family of all ordered subsets of the variables S. From the Bayesian perspective, we would like to employ Bayesian model averaging to estimate the causal effect. This is given by:

$$\text{BCE}(i,j) \triangleq \mathbb{E}_{G \sim q_\phi(G)}[\mathbb{E}_{B \sim q(B|G)}[E_{ij}(B)]]$$

While the other queries we have analyzed describe properties of the distribution $q_\phi(G)$ over causal graphs, here we are concerned with causal inference *on the domain variables* $\boldsymbol{X}$ themselves (induced by $q_\phi(G)$). This is a significant distinction for two reasons. Firstly, it is often the case that such quantities are of great practical interest, for instance, to estimate the effect of various types of treatments/interventions on patient outcomes. Secondly, even with full knowledge of the causal graph, inference in Bayesian networks is NP-hard in general, making it unclear how to efficiently transfer knowledge about the distribution over causal graphs to distributions over the domain variables. Fortunately, we find

that in the case of linear Gaussian BNs, OrderSPNs possess the appropriate structure to compute Bayesian averaged causal effects efficiently:

Proposition 4. *Given an OrderSPN representation q_ϕ of the distribution over DAGs, the matrix of all pairwise Bayesian averaged causal effects BCE(i, j) with respect to q_ϕ can be computed in $O(d^3M)$ time and $O(d^2M)$ space, where M is the size of the SPN.*

The factor of $O(d^3)$ is unsurprising and arises from the inference cost of causal effects in linear Gaussian BNs (Koller & Friedman, 2009); the significance is in the linear complexity in the size of the OrderSPN, given that BCE(i, j) averages over potentially exponentially more DAGs. Intuitively, this is achieved by "summing-out" over different causal (directed) paths between variables i, j at each node. We provide more details and a formal proof in Appendix C.

5. Structure Learning with OrderSPNs

In this section we propose a framework for learning OrderSPNs from data. This consists of two components. Firstly, we learn a structure for the OrderSPN, which characterizes the support of the distribution. Then, we optimize the parameters of the SPN using a variational inference scheme.

5.1. SPN structure learning

We focus on regular OrderSPNs, where the topology of the SPN is fixed, but for each sum node T in layer j we must choose the K_j partitions $(S_{21,i}, S_{22,i})$ of S_2. We define the problem as choosing an oracle $\mathcal{O}$ which takes as input some data $\mathcal{D}$, disjoint sets S_1, S_2, and a number of samples K, and returns K partitions $(S_{21,i}, S_{22,i})$ of S_2. The goal of the oracle is to maximize coverage of the posterior distribution, i.e. the posterior mass of orders consistent with at least one of the sampled partitions. In practice, we can instantiate the oracle with any Bayesian structure learning method that can be modified to produce a DAG over S_2, which can additionally have parents from S_1. In particular, we adapt two recent Bayesian structure learners, DIBS (Lorch et al., 2021) and GADGET (Viinikka et al., 2020). Given such a method, we can define the oracle by (i) taking K samples of such DAGs; (ii) for each sample, choosing a random ordering consistent with the DAG; and (iii) splitting the ordering into a partition. Each partition $(S_{21,i}, S_{22,i})$ influences the support of the corresponding child of T, by restricting that S_{21} comes before S_{22} in the ordering.

The proposed strategy involves calling the oracle $\mathcal{O}$ for each sum-node in the OrderSPN. This improves exploration of the space over the base structure learning method, by recursively exploring subspaces of DAGs over smaller subsets of variables $S_2 \subseteq \{1, ..., d\}$. However, it also appears to introduce a computational challenge since the number of sum-nodes in the SPN could be very large. Thankfully, though each successive sum-layer has $2K_j$ more sum-nodes, the dimension of the DAG space is halved, meaning that the oracle requires much less time. In practice, we ensure efficient implementation by the following methods: (i) we set a time budget appropriately for the oracle in each layer; (ii) for small dimensions $|S_2| \leq d'$ (chosen to be 4), we avoid the constant-time overhead of each oracle run by instead explicitly enumerating over all partitions.

5.2. Parameter Learning via Variational Inference

Given a SPN structure, we now consider the task of learning the parameters of the SPN. We formulate this as a discrete variational inference (VI) problem. Given an unnormalized order-modular distribution $\tilde{p}(\sigma, G) = p_G(G)\mathbb{1}_{G \models \sigma}$, the evidence lower bound (ELBO) is given by:

$$\mathbb{E}_{q_\phi(\sigma,G)}[\log \tilde{p}(\sigma, G)] + H(q_\phi(\sigma, G))$$

where $H(q_\phi(\sigma, G)) = -\mathbb{E}_{q_\phi(\sigma,G)}[\log q_\phi(\sigma, G)]$ is the entropy of the OrderSPN q. The goal of VI is then to maximize the ELBO with respect to ϕ. Typically, such discrete VI problems are difficult, since the ELBO requires computing (gradients of) the expectation using high-variance estimators such as REINFORCE (due to the discrete space, the reparameterization trick is not applicable). Fortunately, due to the tractable properties of the SPN, this is not an issue:

Proposition 5. *The ELBO and its gradients for any regular OrderSPN q_ϕ and order-modular distribution p can be computed in linear time in the size of the SPN.*

We provide the proof in Appendix E, which is based on the corresponding result (Thm 1) from Shih & Ermon (2020) for deterministic SPNs. This allows us to learn the SPN parameters using gradient-based optimization. Further, due to the layered structure of regular OrderSPNs, we can leverage tensor learning frameworks and hardware acceleration.

6. Experiments

In this section, we perform an empirical validation of the TRUST framework.[1] We implement two state-of-the-art Bayesian structure learning methods, (marginal) DIBS (Lorch et al., 2021) and GADGET (Viinikka et al., 2020), and compare them with their TRUST-enhanced counterparts, TRUST-D and TRUST-G, which use the respective method as the oracle for OrderSPN structure learning. Each inference method is applied to synthetic structure learning problems, where the ground truth causal structures are Erdős-Rényi random graphs with dimension $d \in \{16, 32\}$ and $2d$ expected edges, and the Bayesian network distribution is linear

[1] Our implementation is available at https://github.com/wangben88/trust.

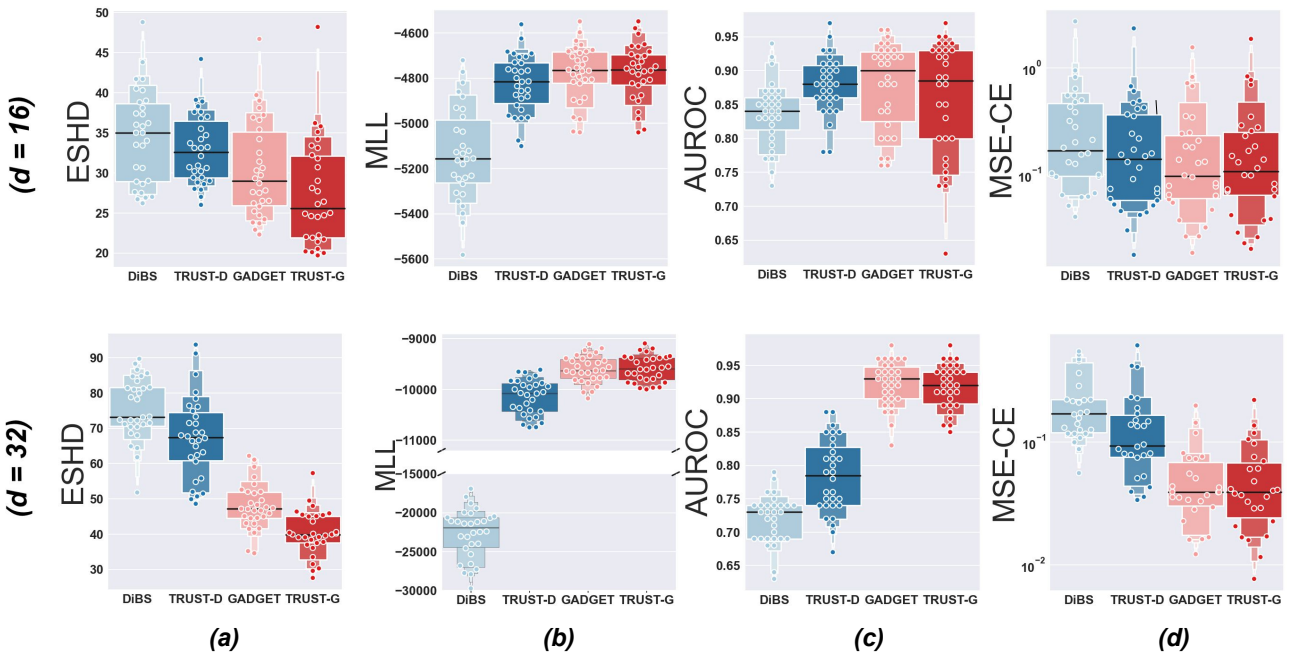


Figure 2. Performance evaluation of the TRUST framework. We find that across all metrics and for both dimensionalities that the TRUST framework outperforms the seed method, in some instances considerably. **Top Row:** Learning structures with $d = 16$. **Bottom Row:** Learning structures with $d = 32$. **(a)** Expected Structural Hamming Distance, lower is better. **(b)** Marginal Log Likelihood (higher is better). **(c)** Area Under the Receiver Operator Characteristic curve (higher is better). **(d)** MSE of Causal Effects (lower is better).

Gaussian. All methods tested employ the BGe marginal likelihood. For each experiment, a dataset $\mathcal{D}^{\text{train}}$ of $N = 100$ datapoints is generated for each graph for inference.

6.1. Learning Performance

We begin by evaluating the quality of the inferred posterior $q(G|\mathcal{D})$ for each inference method, over a variety of standard metrics. In what follows, we use G, B to denote the true graph/edge weights respectively, and $\mathcal{D}^{\text{test}}$ to denote a held-out dataset of 1000 datapoints.

The *expected structural Hamming distance* E-SHD(q, G) measures the expected number of edge changes (SHD) between the essential graphs of G and G', where G' is sampled from the posterior q:

$$\text{E-SHD}(q, G) = \mathbb{E}_{G' \sim q}[\text{SHD}(\text{essential}(G'), \text{essential}(G))]$$

The *area under the receiver operating characteristic curve* AUROC(q, G) for Bayesian structure learning (Friedman & Koller, 2003) is computed using marginal edge probabilities $q(G'_{ij} = 1)$ for each potential edge G'_{ij}, while varying the confidence threshold to construct the ROC curve.

The *marginal log-likelihood* MLL$(q, G, \mathcal{D}^{\text{test}})$ measures how well the posterior fits the held-out test data, using the BGe marginal likelihood p:

$$\text{MLL}(q, G, \mathcal{D}^{\text{test}}) = \mathbb{E}_{G' \sim q}[\log p(\mathcal{D}^{\text{test}}|G')]$$

Finally, the *mean-squared error of causal effects* MSE-CE(q, B) measures the squared difference between the expected posterior causal effect $BCE_q(i, j)$, and the true causal effect $E_{ij}(B)$ (for variable pair i, j). This is then averaged over all (distinct) pairs i, j:

$$\text{MSE-CE}(q, B) = \frac{1}{d(d-1)} \sum_{i \neq j} |BCE_q(i, j) - E_{ij}(B)|^2$$

We show the results in Figure 2 for all methods. TRUST-D and TRUST-G match or outperform their counterparts across all metrics, with especially strong performance on E-SHD, where TRUST-G is best by a clear margin for both $d = 16, 32$. Interestingly, we find that both the oracle methods used for SPN structure learning and the subsequent VI parameter learning are important for achieving the best posterior approximation; see Appendix G for further details.

6.2. Coverage and Query Answering

We now compare the query answering capabilities of TRUST to DIBS and GADGET, for $d = 16$ networks. We set up the task by selecting n edges randomly from the true graph, which we use to form the condition $\bigwedge_{i=1}^{d} c'_i$ in Section 4.4 (requiring that all n edges are present). A good representation of the posterior should consistently have posterior mass over this condition. For DIBS and GADGET, we obtain

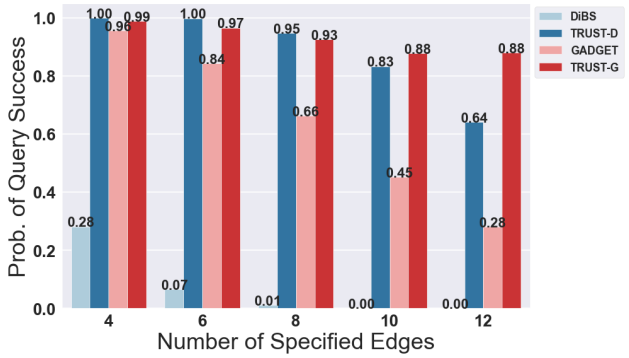


Figure 3. As we specify more edges in our query, the probability that sample-based posteriors (DIBS and GADGET) have support over the queried edges drops. TRUST-D and TRUST-G, in contrast, maintain much greater coverage.

sample-based approximations q of the posterior, for which we take 30 and 10000 samples respectively, as indicated by the respective papers and reference implementations. For TRUST-D and TRUST-G we directly perform the inference queries on the learned OrderSPN.

We begin by considering the marginal probability $q(\bigwedge_{i=1}^{d} c_i')$. In Figure 3, we compute this over 30 different runs and 50 random edge selections for each run, for different values of n, and plot the proportion of times that the probability is non-zero. We see that, as n increases, both methods based on TRUST consistently outperform their counterparts. This demonstrates how TRUST can be used to augment an oracle method to significantly improve the reliability of posterior coverage. This is particularly noteworthy for DIBS, whose coverage is otherwise limited by its quadratic time complexity in the number of samples.

From a practical perspective, this is especially important for conditional inference. In Table 2, we simulate a scenario where we obtain information on the true causal graph after learning. In particular, given $n = 4, 8, 16$ randomly specified edges from the true graph as a condition, we compute conditional probabilities for all unspecified (potential) edges. This can be viewed as "injecting" causal information, which, for instance, could permit distinguishing between DAGs in the same Markov equivalence class where observational data would not suffice. To evaluate, we compute the AUROC given the computed probabilities for each edge. In the case where the representation q has no probability over the condition, we simply take the overall AUROC for the unconditional distribution. Table 2 shows mean and standard deviation for AUROC over 30 runs for each method. As the number of specified edges increases, we see that the performance of GADGET degrades despite the extra information, since the sample-based representation suffers from prohibitively high variance when estimating conditional probability. On the other hand, the greater coverage of TRUST-G ensures that we can take advantage of the extra information, improving the quality of inferences.

No. Edges	Method	AUROC
4	GADGET	0.905 ± 0.073
	TRUST-G	0.903 ± 0.057
8	GADGET	0.888 ± 0.089
	TRUST-G	0.933 ± 0.048
16	GADGET	0.876 ± 0.081
	TRUST-G	0.957 ± 0.077

Table 2. Quality of inference for conditional queries. Results show TRUST-G is significantly better at inferring conditional distributions, especially as the condition becomes more restrictive.

7. Conclusion

We study the problem of tractable representations in Bayesian structure learning. Such representations are crucial for being able to effectively learn and reason about causal structures with uncertainty. In particular, we introduce OrderSPNs, a new approximate representation of distributions over orders and structures. We show that OrderSPNs enable tractable and exact inference over the representation for a variety of important classes of queries, including, remarkably, inference of causal effects for linear Gaussian networks. Our experimental results demonstrate that OrderSPNs can indeed improve upon the representations of state-of-the-art Bayesian structure learning methods, with greater posterior coverage and query answering capabilities.

Our findings illustrate the potential of using tractable probabilistic representations to represent distributions over causal hypotheses. We anticipate that such representations could be applicable to a variety of tasks in causal inference, such as designing optimal interventions (Agrawal et al., 2019), though we leave the investigation of these to future work. Also, while we have chosen to focus on order-modular distributions and SPNs, it is an interesting question whether other types of distributions and tractable representations could be implemented. For instance, probabilistic sentential decision diagrams (Kisa et al., 2014) are a type of probabilistic circuit that admit a wider range of queries than SPNs. Such representations could offer alternative tractability properties that make them suitable for differing applications.

Acknowledgements We thank the anonymous reviewers for their valuable feedback and suggestions. This project was funded by the ERC under the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No.834115).

References

Agrawal, R., Uhler, C., and Broderick, T. Minimal i-map mcmc for scalable structure discovery in causal dag models. In *International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 89–98, 2018.

Agrawal, R., Squires, C., Yang, K. D., Shanmugam, K., and Uhler, C. Abcd-strategy: Budgeted experimental design for targeted causal structure discovery. In *International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 3400–3409, 2019.

Annadani, Y., Rothfuss, J., Lacoste, A., Scherrer, N., Goyal, A., Bengio, Y., and Bauer, S. Variational causal networks: Approximate bayesian inference over causal structures. *arXiv preprint arXiv:2106.07635*, 2021.

Castelletti, F. and Consonni, G. Bayesian inference of causal effects from observational data in gaussian graphical models. *Biometrics*, 77(1):136–149, 2021.

Chan, H. and Darwiche, A. On the robustness of most probable explanations. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, pp. 63–71, Arlington, Virginia, USA, 2006. AUAI Press. ISBN 0974903922.

Chickering, D. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 01 2002. doi: 10.1162/153244303321897717.

Choi, A. and Darwiche, A. On relaxing determinism in arithmetic circuits. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 825–833, 2017.

Choi, Y., Vergari, A., and Van den Broeck, G. Probabilistic circuits: A unifying framework for tractable probabilistic models. Technical report, oct 2020. URL http://starai.cs.ucla.edu/papers/ProbCirc20.pdf.

Cundy, C., Grover, A., and Ermon, S. Bcd nets: Scalable variational approaches for bayesian causal discovery. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Dennis, A. and Ventura, D. Greedy structure search for sum-product networks. In *International Joint Conference on Artificial Intelligence*, pp. 932–938, 2015.

Eggeling, R., Viinikka, J., Vuoksenmaa, A., and Koivisto, M. On structure priors for learning bayesian networks. In *International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1687–1695, 16–18 Apr 2019.

Friedman, N. and Koller, D. Being bayesian about network structure. A bayesian approach to structure discovery in bayesian networks. *Machine Learning*, 50(1-2):95–125, 2003. doi: 10.1023/A:1020249912095.

Gens, R. and Pedro, D. Learning the structure of sum-product networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 873–880, 17–19 Jun 2013.

Giudici, P. and Castelo, R. Improving markov chain monte carlo model search for data mining. *Machine Learning*, 50(1):127–158, 2003.

Glymour, C., Zhang, K., and Spirtes, P. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10:524, 2019. ISSN 1664-8021. doi: 10.3389/fgene.2019.00524.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Koivisto, M. Advances in exact bayesian structure discovery in bayesian networks. In *Conference on Uncertainty in Artificial Intelligence*, 2006.

Koivisto, M. and Sood, K. Exact bayesian structure discovery in bayesian networks. *The Journal of Machine Learning Research*, 5:549–573, 2004.

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

Kuipers, J., Moffa, G., and Heckerman, D. Addendum on the scoring of gaussian directed acyclic graphical models. *The Annals of Statistics*, 42(4), Aug 2014. ISSN 0090-5364. doi: 10.1214/14-aos1217.

Kuipers, J., Suter, P., and Moffa, G. Efficient sampling and structure learning of bayesian networks. *arXiv preprint arXiv:1803.07859*, 2018.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Lorch, L., Rothfuss, J., Schölkopf, B., and Krause, A. Dibs: Differentiable bayesian structure learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Maathuis, M. H., Colombo, D., Kalisch, M., and Bühlmann, P. Predicting causal effects in large-scale systems from observational data. *Nature Methods*, 7(4):247–248, 2010.

Madigan, D., York, J., and Allard, D. Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, 63(2):215–232, 1995.

Pearl, J. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X.

Peharz, R., Gens, R., and Domingos, P. Learning selective sum-product networks. In *ICML Workshop on Learning Tractable Probabilistic Models*, 06 2014.

Peharz, R., Gens, R., Pernkopf, F., and Domingos, P. M. On the latent variable interpretation in sum-product networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(10):2030–2044, 2017.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *Conference on Uncertainty in Artificial Intelligence*, 2011.

Rooshenas, A. and Lowd, D. Learning sum-product networks with direct and indirect variable interactions. In *International Conference on International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, 2014.

Shih, A. and Ermon, S. Probabilistic circuits for variational inference in discrete graphical models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Spirtes, P., Glymour, C. N., Scheines, R., and Heckerman, D. *Causation, prediction, and search*. MIT press, 2000.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

Vergari, A., Di Mauro, N., and Esposito, F. Visualizing and understanding sum-product networks. *Machine Learning*, 108(4):551–573, apr 2019. ISSN 0885-6125. doi: 10.1007/s10994-018-5760-y.

Viinikka, J., Hyttinen, A., Pensar, J., and Koivisto, M. Towards scalable bayesian learning of causal dags. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Wright, S. The method of path coefficients. *The Annals of Mathematical Statistics*, 5(3):161–215, 1934. ISSN 00034851.

Appendix

A. Proofs of OrderSPN results

In this section, we provide further details on the properties of OrderSPNs. Firstly, we prove Proposition 1, regarding decompositions for order-modular distributions. Then, we provide proofs for Propositions 2 and 3 from the main paper, which show that (regular) OrderSPNs are consistent over orders and graphs, and satisfy the required properties for efficient inference. Finally, we formally characterize the compactness of the OrderSPN representation in a new result.

A.1. Hierarchical Decomposition

Recall that in Section 4, we showed that the distribution on orders and graphs (σ, G) could be decomposed into a product of two distributions on (σ_{S_1}, G_{S_1}) and (σ_{S_2}, G_{S_2}) respectively, conditional on $\sigma = (\sigma_{S_1}, \sigma_{S_2})$, i.e. the event that all variables in S_1 come before S_2 in the ordering. We now prove the generalization of that result, which allows us to *hierarchically* decompose the distribution, giving rise to the proposed OrderSPN structure.

Proposition 1. *Let $p(\sigma, G) \propto p_G(G)\mathbb{1}_{G\models\sigma}$ be an order-modular distribution. Suppose that S_1, S_2 are any disjoint subsets of the variables $\{1, ..., d\}$, and let (S_{21}, S_{22}) be a partition of S_2. Then the following CI holds:*

$$\tilde{p}_{S_1,S_2}(\sigma_{S_2}, G_{S_2}|\sigma_{S_2} = (\sigma_{S_{21}}, \sigma_{S_{22}})) \propto \tilde{p}_{S_1,S_{21}}(\sigma_{S_{21}}, G_{S_{21}})\tilde{p}_{S_1\cup S_{21},S_{22}}(\sigma_{S_{22}}, G_{S_{22}})$$

Proof. By definition, we have that $\tilde{p}_{S_1,S_2}(\sigma_{S_2}, G_{S_2}) = \prod_{i\in S_2} p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq S_1\cup\sigma^{<i}_{S_2}}$. Conditioning on the event $\sigma_{S_2} = (\sigma_{S_{21}}, \sigma_{S_{22}})$, we have that:

$$\begin{aligned}
&\tilde{p}_{S_1,S_2}(\sigma_{S_2}, G_{S_2}|\sigma_{S_2} = (\sigma_{S_{21}}, \sigma_{S_{22}})) \\
&\quad\propto \prod_{i\in S_2} p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq S_1\cup\sigma^{<i}_{(S_{21},S_{22})}} \\
&\quad= \prod_{i\in S_{21}} p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq S_1\cup\sigma^{<i}_{S_{21}}} \\
&\qquad\prod_{i\in S_{22}} p_{G_i}(G_i)\mathbb{1}_{G_i\subseteq S_1\cup S_{21}\cup\sigma^{<i}_{S_{22}}} \\
&\quad= \tilde{p}_{S_1,S_{21}}(\sigma_{S_{21}}, G_{S_{21}})\tilde{p}_{S_1\cup S_{21},S_{22}}(\sigma_{S_{22}}, G_{S_{22}})
\end{aligned}$$

as required. □

A.2. OrderSPN properties

Proposition 2. *Let q_ϕ be an OrderSPN. Then, for all pairs (σ, G) in the support of an OrderSPN, it holds that $G \models \sigma$.*

Proof. A complete subcircuit C (Chan & Darwiche, 2006; Dennis & Ventura, 2015) is obtained by traversing the circuit top-down and i) selecting one child of every sum-node; ii) selecting all children of every product-node; iii) selecting all leaf nodes reached. By removing (or equivalently, setting to 1) all sum-node weights, C is itself an OrderSPN expressing a distribution over (σ, G). The key point is that the order is determined in any complete subcircuit. At the leaf nodes, the orders $\sigma_{\{i\}}$ over singletons are trivially deterministic. At the product nodes in the subcircuit, the order is determined by the order specified by the first (left) and second (right) child. That is, for a product node P in the subcircuit associated with (S_1, S_{21}, S_{22}), if the left child specifies an order $\sigma_{S_{21}}$ and the right child an order $\sigma_{S_{22}}$, then the order for P is determined as $(\sigma_{S_{21}}, \sigma_{S_{22}})$. Finally, the sum nodes in the subcircuit only have one child, so the order is determined from its child. Let the uniquely determined order for subcircuit C be denoted σ^C.

Now, consider any path from the root node to a leaf node in the subcircuit. Label the sum nodes (and leaf node) reached T_i for $i = 1, ..., m$ (for some m), associated with $(S_{1,i}, S_{2,i})$ respectively. We will now show, by induction, that for each sum node T_i, it is the case that all variables in $S_{1,i}$ come before $S_{2,i}$ in the ordering σ^C.

- The root is associated with $(S_{1,1}, S_{2,1}) = (\emptyset, \{1, ...d\})$, so the condition is trivially satisfied.
- Now, given node T_i with $i < m$, by definition T_i has a product node child P_i such that T_{i+1} is either the first or second child of P_i. Let P_i be associated with $(S_{1,i}, S_{21,i}, S_{22,i})$. Then, (i) if T_{i+1} is the first child of P_i, then $(S_{1,i+1}, S_{2,i+1}) = (S_{1,i}, S_{21,i})$, while (ii) if T_{i+1} is the second child of P_i, then $(S_{1,i+1}, S_{2,i+1}) = (S_{1,i} \cup S_{21,i}, S_{22,i})$. Now, σ_C has the property that all nodes in $S_{21,i}$ come before those in $S_{22,i}$. Given the inductive hypothesis that $S_{1,i}$ comes before $S_{2,i}$ in the ordering, in both cases (i) and (ii) we have that all nodes in $S_{1,i+1}$ come before nodes in $S_{2,i+1}$ in the ordering.

This means that, at any leaf node associated with some $(S_1, \{i\})$, it will be the case that S_1 comes before i in σ^C. Since the leaf distribution only has support over graphs with $G_i \subseteq S_1$, it follows that all graphs G in the support satisfy $G \models \sigma^C$.

The overall distribution of the OrderSPN is given by a (weighted) sum of all complete subcircuits, so the result follows.

□

Proposition 3. *Any OrderSPN is complete and decomposable, and regular OrderSPNs are additionally deterministic.*

Proof. Given any sum node T in the OrderSPN, completeness follows since the i^{th} product node has scope

$(\sigma_{S_{21,i} \cup S_{22,i}}, G_{S_{21,i} \cup S_{22,i}}) = (\sigma_{S_2}, G_{S_2})$, as $S_{21,i}, S_{22,i}$ partitions S_2 by definition. Decomposability follows immediately from the scopes of the product nodes P and their children, where the variables $(\sigma_{S_{21} \cup S_{22}}, G_{S_{21} \cup S_{22}}$ are split into sum (or leaf) nodes with scope $(\sigma_{S_{21}}, G_{S_{21}})$ and $(\sigma_{S_{22}}, G_{S_{22}})$, where S_{21}, S_{22} are disjoint. Determinism holds for regular OrderSPNs since every sum node has children which split the order into different partitions, so that the children have distinct support over orders (in fact, the choice of child at each sum node can be viewed as determining the order). □

A.3. Compactness of OrderSPNs

When organized as a regular OrderSPN, we can further characterize the compactness of the representation. The following result shows that OrderSPNs can be *exponentially* more compact than sample representations of orders:

Proposition 6. *Given a regular OrderSPN q_ϕ over $d = 2^l$ variables, with l sum (and product) layers and expansion factors $(K_0, ...K_{l-1})$ as above, then we have that:*

- *The size (number of edges) of q_ϕ is given by: $\sum_{i=1}^{l}(2^i + 2^{i-1}) \prod_{j<i} K_j$*
- *The size (number of orders) of the support of q_ϕ is given by: $\prod_{i=0}^{l-1} K_i^{2^i}$*

Proof. Let T_i, P_i be the i^{th} layer of the OrderSPN, with $|T_i|, |P_i|$ nodes respectively, for $i = 0, ..., l-1$. We will also write T_l to denote the leaf layer following all of the other layers. Then, by definition, the nodes in the l^{th} sum layer each have K_j children. Thus, $|P_i| = K_i|T_i|$. Each product node has two children, so we have the relation $|T_{l+1}| = 2|P_l|$. Since the first sum layer P_1 consists of just a single root node, $|P_0| = 1$, and it can be easily checked that $|T_i| = 2^i \prod_{j<i} K_j$ and $|P_i| = 2^i \prod_{j<i+1} K_j$. Thus the total number of nodes is given by:

$$\sum_{i=0}^{l}|T_i| + \sum_{i=0}^{l-1}|L_i| = \sum_{i=0}^{l} 2^i \prod_{j<i} K_j + \sum_{i=0}^{l-1} 2^i \prod_{j<i+1} K_j$$
$$= 1 + \sum_{i=1}^{l}(2^i + 2^{i-1}) \prod_{j<i} K_j$$

Note that the structure of an OrderSPN takes the form of a tree, i.e., each node has a unique parent (except the root). Thus, the number of edges in the OrderSPN is equal to the number of nodes, excluding the root. Now, each node represents a distribution over orders and graphs restricted to some subset of variables. Let $N(T_i)$ denote the number of distinct orders in the support of the first node in T_i (similar for $N(P_i)$). Notice that, since $d = 2^l$, all nodes in the layer T_i have support over the same number of orders. Thus, we need only consider how the number of orders covered changes as we move through the layers. Firstly note that, for the leaf layer T_l, all nodes express distributions over $(\sigma_{\{i\}}, G_i)$ for some variable i. There is only one possible permutation over a singleton set, so $N(T_l) = \sigma_{\{i\}}$. Then, for any sum-node in T_i, by determinism, each child of T_i has disjoint support, so it follows that $N(T_i) = K_i \times N(P_i)$. For any product-node in P_i, we have that the two children of P_i express distributions over orders/permutations $\sigma_{S_{21}}, \sigma_{S_{22}}$, where S_{21}, S_{22} are disjoint sets. Since each of the children have support over $N(T_{i+1})$ orders, the product node expressing a distribution over $\sigma_{S_{21} \cup S_{22}}$ has $N(P_i) = N(T_{i+1})^2$. It is worth comparing this to the corresponding relation $|T_{l+1}| = 2|P_l|$ above; the conditional independence asserted by the OrderSPN results in the compactness of the representation. We can now see (by induction) that $N(P_i) = \prod_{j=i+1}^{l-1} K_j^{2^{j-i+1}}$ and $N(T_i) = \prod_{j=i}^{l-1} K_j^{2^{j-i}}$, and so the root node has support size:

$$N(T_0) = \prod_{j=0}^{l-1} K_j^{2^j}$$

□

B. Computation of leaf distributions

We now explain in detail how to perform marginal, conditional, MPE and sampling inference for leaf distributions.

Recall that leaf distributions for variable i in an OrderSPN are given by the following density:

$$p_{S_1,\{i\}}(G_i) = \frac{p_{G_i}(G_i)\mathbb{1}_{G_i \subseteq S_1}}{\sum_{G_i \subseteq S_1} p_{G_i}(G_i)}$$

where $S_1 \subset C_i$ is the set of potential parents of variable i, and where we have explicitly included the normalizing constant.

As the dimension d increases, this is challenging to compute due to the (exponential) sum over subsets in the normalizing constant. Further, different leaves of the OrderSPNs will in general have different sets S_1, due to the conditions on variable ordering imposed by the SPN structure.

Thus, following previous work (Friedman & Koller, 2003; Kuipers et al., 2018), we globally limit the parents of variable i to a *candidate set* C_i. That is, for each leaf node for variable i with distribution $p_{S_1,\{i\}}(G_i)$, we replace S_1 with $S_1 \cap C_i$. While this inevitably restricts the coverage of the distribution over DAGs, we can choose the candidate parents C_i in such a way as to preserve as much posterior mass as possible[2]. As we will shortly see, this enables us

[2](Viinikka et al., 2020) studied a number of different strategies for selecting these candidate parents; we use the *Greedy* heuristic, which was found empirically to be most effective.

to design precomputation schemes that then allow for inference queries on $p_{S_1,\{i\}}(G_i)$ to be answered efficiently for *any* $S_1 \subseteq C_i$.

In contrast to previous work, we are interested not just in the densities/normalizing constants, but also more complex forms of inference. For this, we define $a_{i,j}$ to be the event that $j \in G_i$, i.e. j is a parent of i. Then, the key component is to precompute the following function, previously proposed in the Appendix of Viinikka et al. (2020) (for a different purpose):

$$f_i(A_i, A_i') = \sum_{G_i \models \left(\bigwedge_{j \in A_i} a_{i,j} \wedge \bigwedge_{j \in A_i'} \neg a_{i,j}\right)} p_{G_i}(G_i)$$

where A_i, A_i' are disjoint subsets of C_i. Intuitively, this is the (unnormalized) probability that all variables in A_i are parents of i, and all those in A_i' are not parents of i.

This function can be precomputed in time and space $O(3^{|C_i|})$ as follows. In the base case where A_i, A_i' partition C_i, then we simply have:

$$f_i(A_i, A_i') = p_{G_i}(A_i)$$

since A_i, A_i' fully specify the parents of i.

In any other case, we have the recurrence:

$$f_i(A_i, A_i') = f_i(A_i \cup \{b\}, A_i') + f_i(A_i, A_i' \cup \{b\})$$

for any $b \in C_i \setminus (A_i \cup A_i')$. This can be seen from the definition of f_i; the RHS corresponds to conditioning on the cases where b either is or is not a parent of i. Notice that, for each A_i, A_i', we need just a constant-time addition; thus the overall complexity is given by the number of partitions of C_i into three subsets, i.e. $O(3^{|C_i|})$.

Now, let c_i be any conjunction of (positive or negative) literals of the atoms $\{a_{i,j} : j \in C_i\}$, i.e., a partial specification of which edges can and can't be transformed. We now propose methods for performing marginal, conditional, MPE and sampling inferences:

- **Marginal/Conditional**: For distribution $p_{S_1,\{i\}}$, the marginal for formula c_i is given by:

$$\begin{aligned} p_{S_1,\{i\}}(c_i = 1) &= \frac{\sum_{G_i \models c_i} p_{G_i}(G_i) \mathbb{1}_{G_i \subseteq S_1}}{\sum_{G_i \subseteq S_1} p_{G_i}(G_i)} \\ &= \frac{\sum_{G_i \models \left(c_i \wedge \bigwedge_{j \in C_i \setminus S_1} \neg a_{i,j}\right)} p_{G_i}(G_i)}{\sum_{G_i \models \bigwedge_{j \in C_i \setminus S_1} \neg a_{i,j}} p_{G_i}(G_i)} \end{aligned}$$

  where we have expressed the condition that $G_i \subseteq S_1$ as the logical formula $\bigwedge_{j \in C_i \setminus S_1} \neg a_{i,j}$. Notice that both the numerator and denominator are of the form of the precomputed f_i, so we can compute the marginal probability simply by two lookups, i.e. $O(1)$ per query.

  Any conditional probability $p_{S_1,\{i\}}(c_i = 1 | c_i' = 1)$ can be computed from marginals as $p_{S_1,\{i\}}(c_i = 1 | c_i' = 1) = \frac{p_{S_1,\{i\}}(c_i \wedge c_i' = 1)}{p_{S_1,\{i\}}(c_i' = 1)}$.

- **MPE**: For distribution $p_{S_1,\{i\}}$, the MPE for formula c_i is given by:

$$\begin{aligned} p_{S_1,\{i\}}(c_i = 1) &= \max_{G_i} p_{S_1,\{i\}}(G_i | c_i = 1) \\ &= \max_{G_i \models c_i} p_{S_1,\{i\}}(G_i | c_i = 1) \\ &= \frac{\max_{G_i \models c_i \wedge \bigwedge_{j \in C_i \setminus S_1} \neg a_{i,j}} p_{G_i}(G_i)}{\sum_{G_i \models c_i \wedge \bigwedge_{j \in C_i \setminus S_1} \neg a_{i,j}} p_{G_i}(G_i)} \end{aligned}$$

  The maximum is over G_i satisfying a logical conjunction, similarly to how f_i expresses sums over G_i satisfying logical conjunctions. Thus, we propose to precompute another function $f_i^{\max}$, which is entirely similar to f_i except that the recurrence is given by:

$$\begin{aligned} &f_i^{\max}(A_i, A_i') \\ &= \max(f_i^{\max}(A_i \cup \{b\}, A_i'), f_i^{\max}(A_i, A_i' \cup \{b\})) \end{aligned}$$

  Analogously to f_i, $f_i^{\max}$ computes the maximal probability $p_{G_i}(G_i)$ for all G_i satisfying the logical formula. Thus, once this function is precomputed, we can compute any MPE query through a lookup of $f_i^{\max}$ and a lookup of f_i, i.e. $O(1)$ per query.

- **Sampling**: Given the condition c_i, we would like to sample G_i from $p_{S_1,\{i\}}(G_i | c_i = 1)$. Let $B \subseteq C_i$ contain the variables which c_i does not specify (as either definitely being a parent, or definitely not being a parent).

  Then, given any ordering $b_1, ...b_K$ of the elements of B, we can sample whether b_k is present sequentially. When sampling b_k, let $d_i^{(k)}$ be a conjunction formula representing the sampling of $b_1, ...b_{k-1}$, e.g., $d_i = a_{i,b_1} \wedge \neg a_{i,b_2} \wedge ... \wedge \neg a_{i,b_{k-1}}$. Then we have:

$$\begin{aligned} &p_{S_1,\{i\}}(a_{b_k} = 1 | d_i^{(k)} = 1, c_i = 1) \\ &\qquad = p_{S_1,\{i\}}(a_{b_k} = 1 | d_i^{(k)} \wedge c_i = 1) \end{aligned}$$

  This takes the form of a conditional probability, which we can compute in constant time. We must apply this operation $K = O(|C_i|)$ times, which leads to an overall complexity of $O(|C_i|)$ per sampling query.

C. Causal Effect Computation

In this section, we show how to tractably compute Bayesian averaged causal effects with respect to OrderSPN representations. The computation of BCE differs from the other

queries, as E_{ij} involves terms which are not localized to a leaf-node distribution; thus standard SPN inference routines are not applicable. Nonetheless, we find that it is possible to compute $\text{BCE}(i,j)$ for all i,j *exactly* with respect to the probabilistic circuit representation over orders and graphs.

Proposition 4. *Given an OrderSPN representation q_ϕ of the distribution over DAGs, the matrix of all pairwise Bayesian averaged causal effects $\text{BCE}(i,j)$ with respect to q_ϕ can be computed in $O(d^3M)$ time and $O(d^2M)$ space, where M is the size of the SPN.*

Proof. Recall that all nodes t in the SPN can be associated with the variable subsets (S_1, S_2), and represent a distribution over the set of edges G_{S_2} (in the case of product nodes, we define $S_2 = S_{21} \cup S_{22}$). Thus, they also define a distribution over causal effects, given by:

$$E_{ij}^{(t)} = \sum_{\pi \in F(S_2 \setminus \{j\})} B_{i,\pi_1} B_{\pi_{|\pi|},j} \prod_{i=1}^{|\pi|-1} B_{\pi_i,\pi_{i+1}}$$

which is defined for any distinct $i \in S_1 \cup S_2$, $j \in S_2$. Notice that this only counts paths which immediately enter (and stay in) S_2; thus all edges are in G_{S_2}.

By taking the expectation, we can similarly define Bayesian averaged causal effects for node t:

$$\text{BCE}(i,j)^{(t)} \triangleq \mathbb{E}_{G_{S_2} \sim q_\phi^{(t)}(G_{S_2})}[\mathbb{E}_{B \sim q(B_{S_2}|G_{S_2})}[E_{ij}^{(t)}(B_{S_2})]]$$

Given this, we now show how it is possible to decompose the computation of $\text{BCE}(i,j)$ according to the structure of the SPN.

If t is a sum node, with children nodes $t_1, .., t_k$ and corresponding weights $\phi_1^{(t)}, ...\phi_C^{(t)}$ we simply have that:

$$\begin{aligned}
&\text{BCE}(i,j)^{(t)} \triangleq \mathbb{E}_{G_{S_2} \sim q_\phi^{(t)}(G_{S_2})}[\mathbb{E}_{B \sim q(B_{S_2}|G_{S_2})}[E_{ij}(B_{S_2})]] \\
&= \sum_{c=1,...,C} \phi_c^{(t)} \mathbb{E}_{G_{S_2} \sim q_\phi^{(t_c)}(G_{S_2})}[\mathbb{E}_{B \sim q(B_{S_2}|G_{S_2})}[E_{ij}(B_{S_2})]] \\
&= \sum_{c=1,...,C} \phi_c^{(t)} \text{BCE}(i,j)^{(t_c)}
\end{aligned}$$

where we have used linearity of expectations to bring the sum outside.

If t is instead a product node, then it has two children t_1, t_2, which are associated with variable subsets (S_1, S_{21}), $(S_1 \cup S_{21}, S_{22})$, respectively. We now consider three separate cases, depending on where $i \in S_1 \cup S_2, j \in S_2$ are located within the subsets.

- If $i \in S_{22}$, $j \in S_{21}$, then $\text{BCE}(i,j)^{(t)} = 0$ since by construction edges (and by extension paths) from S_{22} to S_{21} are disallowed.
- If $i \in S_1 \cup S_{21}$ and $j \in S_{21}$, or alternatively $i \in S_{22}$ and $j \in S_{22}$, then notice that all paths between i, j must stay within S_{21} or S_{22} respectively, since there are no edges from S_{22} to S_{21}. Thus, we have that $E_{ij}^{(t)} = E_{ij}^{(t_1)}$ or $E_{ij}^{(t_2)}$ (respectively) and

$$\text{BCE}(i,j)^{(t)} = \text{BCE}(i,j)^{(t_1)} \text{ or } \text{BCE}(i,j)^{(t_2)}$$

- In the final case, $i \in S_1 \cup S_{21}$ while $j \in S_{22}$. Here we must consider all possible paths between i and j. To do so, we will condition on the last variable in $S_1 \cup S_{21}$ ("exit-point") k along a path. Then we have:

$$\begin{aligned}
E_{ij}^{(t)} &= \sum_{\pi \in F(S_2 \setminus \{j\})} B_{i,\pi_1} B_{\pi_{|\pi|},j} \prod_{i=1}^{|\pi|-1} B_{\pi_i,\pi_{i+1}} \\
&= \sum_{k \in F(S_{21})} \\
&\left(\sum_{\pi \in F(S_{21} \setminus \{k\})} B_{i,\pi_1} B_{\pi_{|\pi|},k} \prod_{i=1}^{|\pi|-1} B_{\pi_i,\pi_{i+1}} \right) \\
&\left(\sum_{\pi \in F(S_{22} \setminus \{j\})} B_{k,\pi_1} B_{\pi_{|\pi|},j} \prod_{i=1}^{|\pi|-1} B_{\pi_i,\pi_{i+1}} \right) \\
&= \sum_{k \in F(S_{21})} E_{ik}^{(t_1)} E_{kj}^{(t_2)}
\end{aligned}$$

  The last equality follows as the two summations are precisely the causal effects $i \to k$ and $k \to j$ for t_1, t_2, respectively, which correspond to variable subsets (S_1, S_{21}) and $(S_1 \cup S_{21}, S_{22})$. Now, by linearity of expectations, and the independence of $E_{ik}^{(t_1)}, E_{kj}^{(t_2)}$, this gives the matrix multiplication:

$$\text{BCE}(i,j)^{(t)} = \sum_{k \in F(S_{21} \setminus \{i\})} \text{BCE}(i,k)^{(t_1)} \text{BCE}(k,j)^{(t_2)}$$

Finally, we consider the leaf nodes t of the SPN, where $|S_2| = 1$ (say, $S_2 = \{j\}$). In such cases, the causal effect reduces to $a_{ij}^{(t)} = B_{i,j}$, and the expectation is given by:

$$\text{BCE}(i,j)^{(t)} = \mathbb{E}_{G_j \sim q_\phi^{(t)}(G_j)}[\mathbb{E}_{B_j \sim q(B_j|G_j)}[B_j]]$$

Given the graph column G_j, the distribution of B_j is given by a multivariate t-distribution (Viinikka et al., 2020), and so the inner expectation can be computed exactly for a given G_j. The outer expectation can be approximated using sampling from the leaf distribution. Though this involves sampling, the crucial aspect of our method is that the expectation through the OrderSPN (and thus through different orders) is exact, unlike *Beeps* (Viinikka et al., 2020), which computes causal effects using sampled DAGs.

At each node t corresponding to variable subsets (S_1, S_2), we must maintain an array $BCE(i,j)^{(t)}$ for $i \in S_1 \cup S_2, j \in S_2$, i.e., of size $(|S_1|+|S_2|) \times |S_2| < d^2$. Computations at any node t take linear time in the number of children (outgoing edges) of the node, except for the matrix multiplication at product nodes, which takes $(|S_1|+|S_{21}|) \times |S_{21}| \times |S_{22}| < d^3$ time. Thus, the overall space and time complexity is $O(d^2 M)$ and $O(d^3 M)$ respectively.

□

D. OrderSPN Structure Learning Oracles

In this section we elaborate further the oracles $\mathcal{O}$ used for generating the structure of the OrderSPN in Section 5.1. As previously defined, the $\mathcal{O}$ takes as input some data $\mathcal{D}$, disjoint sets S_1, S_2, and a number of samples K, and returns K partitions $(S_{21,i}, S_{22,i})$ of S_2. The goal of the oracle is to maximize coverage of the posterior distribution, i.e. the posterior mass of orders consistent with at least one of the sampled partitions. Solving such a problem exactly is clearly intractable; thus we would like heuristic methods which can obtain good coverage.

A possible oracle would simply be to take K random partitions of S_2. However, this does not make efficient usage of the capacity of the OrderSPN. Thus, we consider adapting other Bayesian structure learning methods to take the role of the oracle. This can be done by sampling DAGs from the method; each such DAG naturally induces an order over the variables S_2, and thus a partition. Intuitively, we utilize their ability to find promising areas of the space of orders and DAGs to choose a better structure for our SPN.

The key practical challenge is that, is in contrast to the typical use case, we are not just interested in learning a DAG over a set S_2, but also want to allow the variables in S_2 to have parents from some disjoint set S_1. This will require adaptations specific to the particular method chosen. In the rest of this section, we provide brief descriptions of how this can be done for DIBS and GADGET.

DIBS is a Bayesian structure learning approach based on particle variational inference (Liu & Wang, 2016). In particular, in the marginal form, it assumes the following latent-variable generative model:

$$p(Z, G, \mathcal{D}) = p(Z)p(G|Z)p(\mathcal{D}|G)$$

where $Z = [U, V]$ with $U, V \in \mathbb{R}^{k \times d}$ (for some $k < d$) is a latent variable, generating the graph $G \in \{0,1\}^{d \times d}$ and $\mathcal{D}$ is the dataset. In particular, the distribution for the graph takes the form:

$$p(G|Z) = \prod_{i,j} p(G_{ij}|Z) = \prod_{i,j} \sigma(\boldsymbol{u}_i^T \boldsymbol{v}_j)$$

Now suppose that we want to learn a DAG over S_2, where variables can additionally have parents in S_1. In this case, the natural generative model is to simply restrict to the components of the graph which are being modelled:

$$p_{S_1,S_2}(G|Z) = \prod_{i \in S_1 \cup S_2} \prod_{j \in S_2} \sigma(\boldsymbol{u}_i^T \boldsymbol{v}_j)$$

The marginal likelihood $p(\mathcal{D}|G)$ is modular, so we can additionally restrict the likelihood to only concern the likelihood of S_2:

$$p_{S_2}(\mathcal{D}|G) = \prod_{j \in S_2} p(\mathcal{D}_j|G_j)$$

With these modifications, we have a valid generative model for any (S_1, S_2), to which the DIBS particle variational inference scheme can be applied with no further changes, giving us an oracle.

GADGET is a MCMC method which samples over the space of *ordered partitions* of the set of variables (note this is distinct from the 2-partitions we use in OrderSPNs). Informally speaking, the ordered partition represents a partial ordering of the variables, where a variable must have a parent from the partition directly preceding its partition. For example, for $d = 6$, a partition might be $3, 4, 5, 1, 2, 6$, where variable 1 must have one of $3, 4, 5$ as parent (but not any of $2, 6$). A $k-$partition R is scored using a modular score:

$$\pi(R) = \prod_{t=1}^{k} \prod_{j \in R_t} \tau_j(\cup_{i=1}^{t-1} R_i, R_{t-1})$$

where $\tau_j(U, T)$ is the summed score (posterior probability) that variable j has all parents contained in the set U, and at least one parent in the set T.

As mentioned in Appendix B, GADGET (Viinikka et al., 2020) uses a similar type of precomputation to that used in TRUST to precompute the functions $\tau_j(U, T)$, where a candidate parent set C_j of each variable is chosen in advance (using a heuristic) so that we actually compute $\tau_j(U \cap C_j, T \cap C_j)$.

Now, suppose we are given sets (S_1, S_2), and as usual seek to learn DAGs over S_2 which additionally have parents from S_1. This can be achieved by simply restricting the MCMC to only learn ordered partitions over S_2, while also allowing the parent sets of variables S_2 to be contained in $S_1 \cup S_2$. In particular, if we have precomputed $\tau_j(U \cap C_j, T \cap C_j)$ for all j and $U, T \subseteq \{1, ..., d\}$, this includes all of the necessary scores $\tau_j(U \cap C_j, T \cap C_j)$ for all $j \in S_2$ and $U, T \subseteq S_1 \cup S_2$, for any restriction (S_1, S_2).

The MCMC proceeds as if it were over a $|S_2|$ dimensional problem, over the set of variables S_2, but with modified scores involving S_1 as above, thus providing an oracle for TRUST.

E. Parameter Learning and Tractable ELBO Computation

In this section, we provide further details on the variational inference scheme used to learn parameters of the OrderSPN. First, we provide a proof of Proposition 5, which is based on Theorem 1 from (Shih & Ermon, 2020).

Proposition 5. *The ELBO and its gradients for any regular OrderSPN q_ϕ and order-modular distribution p can be computed in linear time in the size of the SPN.*

Proof. We assume an order-modular distribution over the form $\tilde{p}(\sigma, G) = \prod_i p_{G_i}(G_i)\mathbb{1}_{G \models \sigma}$. Define $\tilde{p}_{S_2}(\sigma_{S_2}, G_{S_2}) \triangleq \prod_{i \in S_2} p_{G_i}(G_i)\mathbb{1}_{G_{S_2} \models \sigma_{S_2}}$ for any $S_2 \subseteq \{1, ..., d\}$. For any node N in the OrderSPN with scope (σ_{S_2}, G_{S_2}), we will define the following quantity, which is the evidence lower-bound when using the distribution $N(\sigma_{S_2}, G_{S_2})$ to approximate $\tilde{p}_{S_2}(\sigma_{S_2}, G_{S_2})$:

$$ELBO(N) = \mathbb{E}_N[\tilde{p}_{S_2}(\sigma_{S_2}, G_{S_2})] + H(N(\sigma_{S_2}, G_{S_2}))$$

We now show that the ELBO for q_ϕ can be computed efficiently (i.e. in linear time in the size of the SPN) as a function of the ELBO of the leaf node distributions.

Let T be a sum node associated with (S_1, S_2), with children $C_1, ..., C_K$ and corresponding weights $\phi_1, ..., \phi_K$. We can write the expectation of $\tilde{p}_{S_2}$ and entropy in terms of corresponding quantities of the child distributions:

$$\mathbb{E}_T[\log \tilde{p}_{S_2}(\sigma_{S_2}, G_{S_2})] = \sum_{i=1}^{K} \phi_i \mathbb{E}_{C_i}[\log \tilde{p}_{S_2}(\sigma_{S_2}, G_{S_2})]$$

$$\begin{aligned}
H(T(\sigma_{S_2}, G_{S_2})) &= -\mathbb{E}_T[\log T(\sigma_{S_2}, G_{S_2})] \\
&= -\sum_{i=1}^{K} \phi_i \mathbb{E}_{C_i}[\sum_{j=1}^{K} \log \phi_j C_j(\sigma_{S_2}, G_{S_2})] \\
&= -\sum_{i=1}^{K} \phi_i \mathbb{E}_{C_i}[\log \phi_i C_i(\sigma_{S_2}, G_{S_2})] \\
&= -\sum_{i=1}^{K} \phi_i \log \phi_i + \sum_{i=1}^{K} \phi_i \mathbb{E}_{C_i}[-C_i(\sigma_{S_2}, G_{S_2})] \\
&= -\sum_{i=1}^{K} \phi_i \log \phi_i + \sum_{i=1}^{K} \phi_i H(C_i(\sigma_{S_2}, G_{S_2}))
\end{aligned}$$

$$\begin{aligned}
&ELBO(T) = \mathbb{E}_T[\log \tilde{p}_{S_2}(\sigma_{S_2}, G_{S_2})] + H(T(\sigma_{S_2}, G_{S_2})) \\
&= -\sum_{i=1}^{K} \phi_i \log \phi_i \\
&\quad + \sum_{i=1}^{K} \phi_i \left[\mathbb{E}_{C_i}[\log \tilde{p}_{S_2}(\sigma_{S_2}, G_{S_2})] + H(C_i(\sigma_{S_2}, G_{S_2}))\right] \\
&= -\sum_{i=1}^{K} \phi_i \log \phi_i + \sum_{i=1}^{K} \phi_i ELBO(C_i)
\end{aligned}$$

In other words, the expectation decomposes as a weighted sum over expectations with respect to the child distributions, and the entropy decomposes as a sum of the entropy of the sum-node weights, and a weighted sum over entropies with respect to the child distributions. Note that the third equality in the derivation of the entropy decomposition holds only due to the fact that OrderSPNs are deterministic; this means that the children C_i have disjoint supports, and thus $\mathbb{E}_{C_i}[C_j(\sigma, G)] = 0$ for all $i \neq j$. Together, we have that the ELBO of a sum-node can be expressed in terms of the ELBO of its children.

Let P be a product node, associated with $((S_1, S_{21}), (S_{21}, S_{22}))$, with children C_1, C_2. P expresses a distribution over (σ_{S_2}, G_{S_2}), where $S_2 = S_{21} \cup S_{22}$. Then we have that:

$$\begin{aligned}
&\mathbb{E}_P[\log \tilde{p}_{S_2}(\sigma_{S_2}, G_{S_2})] = \mathbb{E}_P[\log \prod_{i \in S_2} p_{G_i}(G_i)\mathbb{1}_{G_{S_2} \models \sigma_{S_2}}] \\
&= \mathbb{E}_P[\log \prod_{i \in S_{21}} p_{G_i}(G_i)\mathbb{1}_{G_{S_{21}} \models \sigma_{S_{21}}} \\
&\qquad \log \prod_{i \in S_{22}} p_{G_i}(G_i)\mathbb{1}_{G_{S_{22}} \models \sigma_{S_{22}}}] \\
&= \mathbb{E}_P[\log \tilde{p}_{S_{21}}(\sigma_{S_{21}}, G_{S_{21}})] + \mathbb{E}_P[\log \tilde{p}_{S_{22}}(\sigma_{S_{22}}, G_{S_{22}})]
\end{aligned}$$

$$\begin{aligned}
&H(P(\sigma_{S_2}, G_{S_2})) = -\mathbb{E}_P[\log P(\sigma_{S_2}, G_{S_2})] \\
&= -\mathbb{E}_P[\log C_1(\sigma_{S_{21}}, G_{S_{21}}) + \log C_2(\sigma_{S_{22}}, G_{S_{22}})] \\
&= -\mathbb{E}_{C_1}[\log C_1(\sigma_{S_{21}}, G_{S_{21}})] - \mathbb{E}_{C_2}[\log C_2(\sigma_{S_{22}}, G_{S_{22}})] \\
&= H(C_1(\sigma_{S_{21}}, G_{S_{21}})) + H(C_2(\sigma_{S_{22}}, G_{S_{22}}))
\end{aligned}$$

$$\begin{aligned}
&ELBO(P) = \mathbb{E}_P[\log \tilde{p}_{S_2}(\sigma_{S_2}, G_{S_2})] + H(P(\sigma_{S_2}, G_{S_2})) \\
&= \mathbb{E}_P[\log \tilde{p}_{S_{21}}(\sigma_{S_{21}}, G_{S_{21}})] + H(C_1(\sigma_{S_{21}}, G_{S_{21}})) \\
&\quad + \mathbb{E}_P[\log \tilde{p}_{S_{22}}(\sigma_{S_{22}}, G_{S_{22}})] + H(C_2(\sigma_{S_{22}}, G_{S_{22}})) \\
&= ELBO(C_1) + ELBO(C_2)
\end{aligned}$$

This follows from decomposability, which ensures that the child distributions are over disjoint sets of variables (and are thus independent).

By recursively applying the above equalities, we can express the ELBO for the overall OrderSPN q_ϕ in terms of the SPN

weights ϕ and ELBO for the leaf node distributions. Since each equality involves a sum/product over the children of the node (i.e., the outgoing edges), the overall computation takes linear time in the size (number of edges) of the SPN. □

E.1. ELBO for Leaf Node Distributions

In the above Proposition, we have not mentioned how to compute the ELBO for the leaf node distributions. For a leaf node L associated with (S_1, i), which is a distribution $L(G_i)$ over the parents of variable i, we have that:

$$\begin{aligned} ELBO(L) &= \mathbb{E}_L[\log \tilde{p}(\sigma_{\{i\}}, G_i)] + H(L(\sigma_{\{i\}}, G_i)) \\ &= \mathbb{E}_L[\log p_{G_i}(G_i)] + H(L(G_i)) \end{aligned} \tag{2}$$

Recall that, for OrderSPNs, it is required that $L(G_i)$ has support only over $G_i \subseteq S_1$. In the main paper, we chose to set $L(G_i) \propto p_{G_i}(G_i)\mathbb{1}_{G_i \subseteq S_1}$. We now provide justification for this choice:

Proposition 7. *$L(G_i) \propto p_{G_i}(G_i)\mathbb{1}_{G_i \subseteq S_1}$ maximizes (2) subject to the support condition.*

Proof. The ELBO for a leaf distribution (2) can be written as:

$$\begin{aligned} ELBO(L) &= \mathbb{E}_L[\log p_{G_i}(G_i)] + H(L(G_i))) \\ &= \mathbb{E}_L[\log p_{G_i}(G_i)] - \mathbb{E}_L[\log L(G_i)] \\ &= -KL(L||p_{G_i}) \end{aligned}$$

where KL is the KL-divergence. Thus, to maximize the ELBO, we need to minimize this KL-divergence. Let $C = \sum_{G_i \subseteq S_1} p_{G_i}(G_i)$. Assuming L satisfies the support condition, this can be written as:

$$\begin{aligned} KL(L||p_{G_i}) &= \mathbb{E}_L\left[\log \frac{L(G_i)}{p_{G_i}(G_i)}\right] \\ &= \mathbb{E}_L\left[\log \frac{L(G_i)}{p_{G_i}(G_i)}\right] \\ &= \mathbb{E}_L\left[\log \frac{L(G_i)}{p_{G_i}(G_i)/C}\right] - \log C \end{aligned}$$

This KL-divergence is minimized by $L(G_i) \propto p_{G_i}(G_i)\mathbb{1}_{G_i \subseteq S_1}$, as required. In this case, the ELBO is given by:

$$\begin{aligned} ELBO(L) &= \log C - KL(L||\frac{p_{G_i}\mathbb{1}_{G_i \subseteq S_1}}{C}) \\ &= \log C \end{aligned}$$

□

We see that, with this choice of L, the ELBO is a constant $\log C$ that we can precompute using the methods for computation of leaf distribution described in Appendix B. Thus, the computation of ELBO for leaf distributions can be done in an $O(1)$ lookup, and the overall ELBO computation is linear in the size of the OrderSPN (in particular, independent of the dimension).

F. Experimental Details

Bayesian network hyperparameters In our experiments, we consider linear Gaussian Bayesian networks, and generate Erdos-Renyi random structures, with expected numbers of edges given by $2d$. We generate data using fixed observation noise $\sigma^2 = 0.1$, and edge weights drawn independently from $\mathcal{N}(0, 1)$.

Posterior setup We use the fair prior over graph structures (Eggeling et al., 2019), where the prior probability of a mechanism having k edges is proportional to the inverse of the number of different parents sets of size k. In addition, we use the BGe marginal likelihood (Kuipers et al., 2014) with hyperparameters $\alpha_\mu = 1$, $\alpha_w = d + 2$, and $T = \frac{1}{2}I$ where I is the $d \times d$ identity matrix.

Implementation details Our implementations of DIBS and GADGET are based on the reference implementations with the default settings of hyperparameters. In particular, we ran DIBS with $N = 30$ particles and 3000 epochs using the marginal inference method, while GADGET was run using 16 coupled chains and for 320000 MCMC iterations, extracting $N = 10000$ samples.

Our implementation of TRUST uses the PyTorch framework to tensorize passes through the SPN, following the regular OrderSPN structure described in the main paper. In the $d = 16, 32$ cases, we used expansion factors of $\boldsymbol{K} = [64, 16, 6, 2], [32, 8, 2, 6, 2]$ respectively; these were chosen empirically to approximately match oracle computation across layers. Parameter learning in the SPN was performed by optimizing the ELBO objective using the Adam optimizer with learning rate 0.1 and for 700 iterations. Operations in the circuit are performed in log-space for numerical stability.

Inference Queries We perform inference for DiBS and Gadget by applying the appropriate calculation over the sample (for instance, the marginal probability of an edge G_ij is simply the proportion of sampled DAGs in which it appears), while for TRUST, we perform inference directly on the OrderSPN using the queries described in the paper when this is possible, and by sampling otherwise (e.g. E-SHD).

G. Ablation Study on OrderSPN Learning

In Section 5, we proposed to use a two-step procedure for learning OrderSPNs, in which we (i) propose a structure for

the OrderSPN using an oracle method; and (ii) further learn the parameters of the OrderSPN via variational inference. We now perform an ablation study to examine the each of these steps and their impact on performance.

We evaluate five different methods:

- **Random** In this case, instead of using an oracle method $\mathcal{O}$ to split S_2 into a partition $(S_{21,i}, S_{22,i}$, we instead perform this split *randomly* throughout the OrderSPN. We also do not perform any parameter learning, instead setting the parameters at each sum-node in the OrderSPN to be equal (e.g. if a sum-node has 4 children, we set each parameter to 0.25).
- **Parameter Only** We randomly propose the structure as above, but do perform parameter learning using VI.
- **Structure Only** We do perform structure learning using GADGET as an oracle, but do not learn parameters.
- **Gadget** As in the main paper.
- **TRUST-G** As in the main paper.

The first step of structure learning determines the support of the OrderSPN, i.e. the orders and DAGs to which it assigns positive probability, while the second step of parameter learning aims to optimize the fit to the posterior given the support constraints imposed by the first step. By randomizing one (or both) of these steps, we can see how this affects the approximation.

The results are shown in Figure 4. As expected, the fully random method performs by far the worst, on all metrics. Both performing parameter learning only and structure learning only provide significant improvements, but interestingly on different metrics. Structure learning only performs quite well on AUROC, while parameter learning only performs comparatively better on E-SHD and MLL (even outperforming GADGET on E-SHD). The performance of using parameter learning only is quite remarkable, given that the graphs covered by the OrderSPN were chosen *at random*. We hypothesize that this can be attributed to the compactness and capacity of OrderSPNs as a representation; as a result, even the randomly chosen structure will contain some orders/DAGs which are close to the ground truth DAG. Nonetheless, adding structure learning as well, as in TRUST-G, does provide the best overall performance, and shows that both steps are important to obtain the best possible representation.

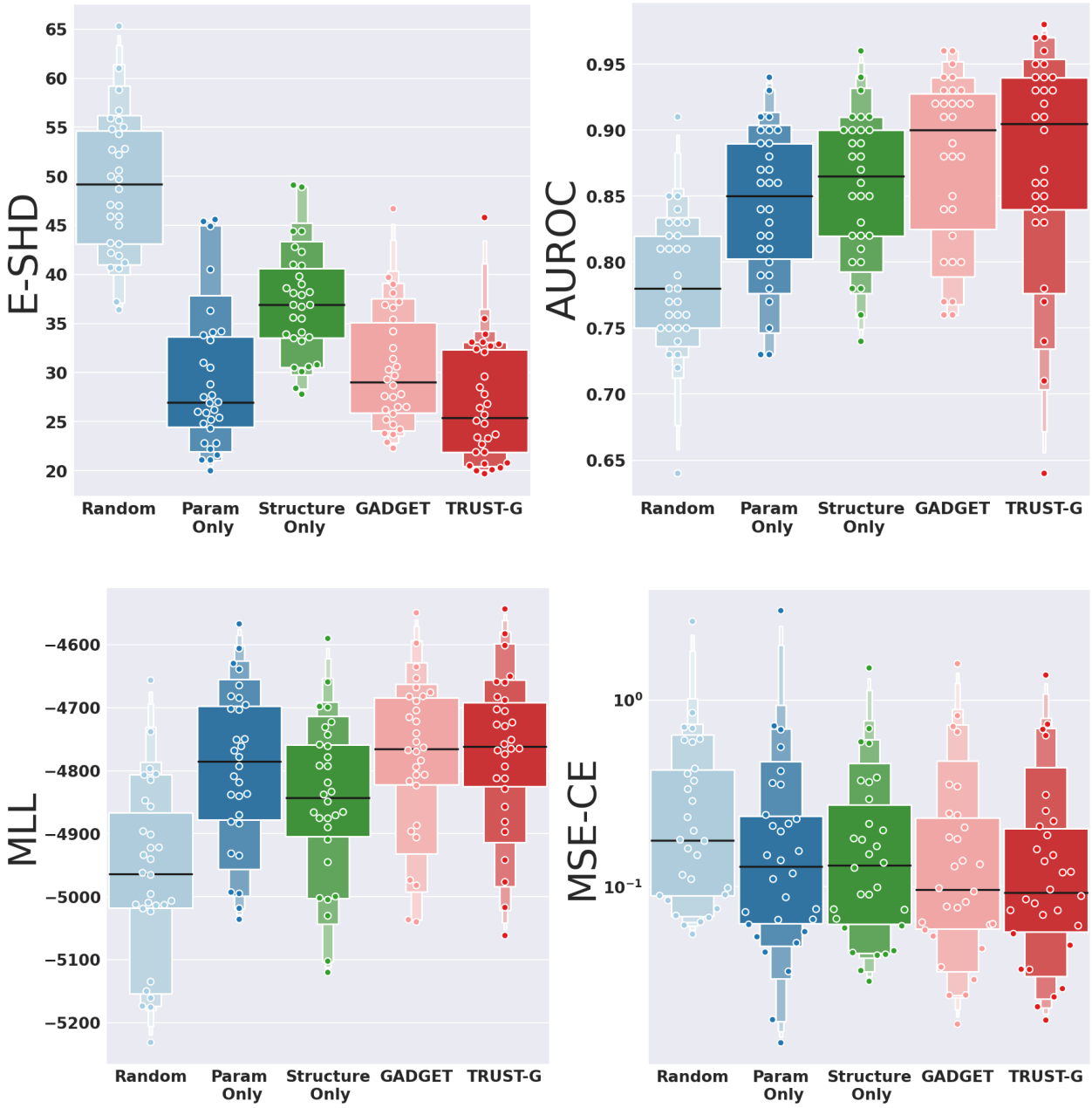


Figure 4. Ablation study evaluating performance of different variants of TRUST-G (and GADGET), for $d = 16$.