# On the Parameterized Complexity of Polytree Learning

**Niels Grüttemeier***, **Christian Komusiewicz** and **Nils Morawietz**†
Philipps-Universität Marburg, Marburg, Germany
{niegru, komusiewicz, morawietz}@informatik.uni-marburg.de

## Abstract

A Bayesian network is a directed acyclic graph that represents statistical dependencies between variables of a joint probability distribution. A fundamental task in data science is to learn a Bayesian network from observed data. POLYTREE LEARNING is the problem of learning an optimal Bayesian network that fulfills the additional property that its underlying undirected graph is a forest. In this work, we revisit the complexity of POLYTREE LEARNING. We show that POLYTREE LEARNING can be solved in single-exponential FPT time for the number of variables. Moreover, we consider the influence of $d$, the number of variables that might receive a nonempty parent set in the final DAG on the complexity of POLYTREE LEARNING. We show that POLYTREE LEARNING is presumably not fixed-parameter tractable for $d$, unlike Bayesian network learning which is fixed-parameter tractable for $d$. Finally, we show that if $d$ and the maximum parent set size are bounded, then we can obtain efficient algorithms.

## 1 Introduction

Bayesian networks are the most important tool for modelling statistical dependencies in joint probability distributions. A Bayesian network consists of a DAG $(N, A)$ over the variable set $N$ and a set of condiditional probability tables. Given a Bayesian network and the observed values on some of its variables, one may infer the probability distributions of the remaining variables under the observations. One of the drawbacks of using Bayesian networks is that this inference task is NP-hard. Moreover, the task of constructing a Bayesian network with an optimal network structure is NP-hard as well, even on very restricted instances [Chickering, 1995]. In this problem, we are given a local parent score function $f_v : 2^{N\setminus\{v\}} \to \mathbb{N}_0$ for each variable $v$ and the task is to learn a DAG $(N, A)$ such that the sum of the parent scores over all variables is maximal.

In light of the hardness of handling general Bayesian networks, the learning and inference problems for Bayesian networks fulfilling some specific structural constraints have been studied extensively [Pearl, 1989; Korhonen and Parviainen, 2013; Korhonen and Parviainen, 2015; Grüttemeier and Komusiewicz, 2020; Ramaswamy and Szeider, 2021].

One of the earliest special cases that has received attention are *tree* networks, also called *branchings*. A tree is a Bayesian network where every vertex has at most one parent. In other words, every connected component is a directed in-tree. Learning and inference can be performed in polynomial time on trees [Chow and Liu, 1968; Pearl, 1989]. Trees are, however, very limited in their modeling power since every variable may depend only on at most one other variable. To overcome this problem, a generalization of branchings called polytrees has been proposed. A polytree is a DAG whose underlying undirected graph is a forest. An advantage of polytrees is that the inference task can be performed in polynomial time on them [Pearl, 1989; Guo and Hsu, 2002]. POLYTREE LEARNING, the problem of learning an optimal polytree structure from parent scores, however, remains NP-hard [Dasgupta, 1999]. We study exact algorithms for POLYTREE LEARNING.

**Related Work.** POLYTREE LEARNING is NP-hard even if every parent set with strictly positive score has size at most 2 [Dasgupta, 1999]. Motivated by the contrast between the NP-hardness of POLYTREE LEARNING and the fact that learning a tree has a polynomial-time algorithm, the problem of optimally learning polytrees that are close to trees has been considered. More precisely, it has been shown that the best polytree among those that can be transformed into a tree by at most $k$ edge deletions can be found in $n^{\mathcal{O}(k)}|I|^{\mathcal{O}(1)}$ time [Gaspers *et al.*, 2015; Safaei *et al.*, 2013] where $n$ is the number of variables and $|I|$ is the overall input size. Thus, the running time of these algorithms is polynomial for every fixed $k$. As noted by Gaspers et al. [2015], a brute-force algorithm for POLYTREE LEARNING would need to consider $n^{n-2} \cdot 2^{n-1}$ directed trees. Polytrees have been used, for example, in image-segmentation for microscopy data [Fehri *et al.*, 2019].

**Our Results.** We obtain an algorithm that solves POLYTREE LEARNING in $3^n \cdot |I|^{\mathcal{O}(1)}$ time. This running time is a substantial improvement over the brute-force algorithm

*Contact Author

†Supported by the Deutsche Forschungsgemeinschaft (DFG), project OPERAH, KO 3669/5-1.

mentioned above, thus positively answering a question of Gaspers et al. [2015] on the existence of such algorithms. We then study whether POLYTREE LEARNING is amenable to polynomial-time data reduction that shrinks the instance $I$ if $|I|$ is much bigger than $n$. We show that such a data reduction algorithm is unlikely. More precisely, we show that (under standard complexity-theoretic assumptions) there is no polynomial-time algorithm that replaces any $n$-variable instance $I$ by an equivalent one of size $n^{\mathcal{O}(1)}$. In other words, it seems necessary to keep an exponential number of parent sets to compute the optimal polytree.

We then consider a parameter that is potentially much smaller than $n$ and determine whether POLYTREE LEARNING can be solved efficiently when this parameter is small. The parameter $d$, which we call the number of *dependent* variables, is the number of variables $v$ for which at least one entry of $f_v$ is strictly positive. The parameter essentially counts how many variables might receive a nonempty parent set in an optimal solution. We show that POLYTREE LEARNING can be solved in polynomial time when $d$ is constant but that an algorithm with running time $g(d) \cdot |I|^{\mathcal{O}(1)}$ is unlikely for any computable function $g$. Consequently, in order to obtain positive results for the parameter $d$, one needs to consider further restrictions on the structure of the input instance. We make a first step in this direction and consider the case where all parent sets with a strictly positive score have size at most $p$. Using this parameterization, we show that every input instance can be solved in $2^{\omega dp} \cdot |I|^{\mathcal{O}(1)}$ time where $\omega$ is the matrix multiplication constant. With the current-best known value for $\omega$ this gives a running time of $5.18^{dp} \cdot |I|^{\mathcal{O}(1)}$. We then consider again data reduction approaches. This time we obtain a positive result: Any instance of POLYTREE LEARNING where $p$ is constant can be reduced in polynomial time to an equivalent one of size $d^{\mathcal{O}(1)}$. Informally, this means that if the instance has only few dependent variables, the parent sets with strictly positive score are small, and there are many nondependent variables, then we can identify some nondependent variables that are irrelevant for an optimal polytree representing the input data. We note that this result is tight in the following sense: Under standard complexity-theoretic assumptions it is impossible to replace each input instance in polynomial time by an equivalent one with $(d + p)^{\mathcal{O}(1)}$ variables. Thus, the assumption that $p$ is a constant is necessary.

## 2 Preliminaries

**Notation.** An *undirected graph* is a tuple $(V, E)$, where $V$ is a set of vertices and $E \subseteq \{\{u, v\} \mid u, v \in V\}$ is a set of edges. Given a vertex $v \in V$, we define $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$ as the *neighborhood of* $v$. A *directed graph* is a tuple $(N, A)$, where $N$ is a set of vertices and $A \subseteq N \times N$ is a set of arcs. If $(u, v) \in A$ we call $u$ a *parent of* $v$ and $v$ a *child of* $u$. Given a vertex $v$, we let $P_v^A := \{u \mid (u, v) \in A\}$ denote the *parent set* of $v$. The *skeleton* of a directed graph $(N, A)$ is the undirected graph $(N, E)$ where $\{u, v\} \in E$ if and only if $(u, v) \in A$ or $(v, u) \in A$. A directed acyclic graph is a *polytree* if its skeleton is a forest, that is, the skeleton is acyclic [Dasgupta, 1999]. As a shorthand, we write $P \times v := P \times \{v\}$ for a vertex $v$ and a set $P \subseteq N$.

**Problem Definition.** Given a vertex set $N$, a family $\mathcal{F} := \{f_v : 2^{N \setminus \{v\}} \to \mathbb{N}_0 \mid v \in N\}$ is a *family of local scores for* $N$. Intuitively, $f_v(P)$ is the score that a vertex $v$ obtains if $P$ is its parent set. Given a directed graph $D := (N, A)$ we define $\text{score}(A) := \sum_{v \in N} f_v(P_v^A)$. We study the following computational problem.

POLYTREE LEARNING
**Input**: A set of vertices $N$, local scores $\mathcal{F} = \{f_v \mid v \in N\}$, and an integer $t \in \mathbb{N}_0$.
**Question**: Is there an arc-set $A \subseteq N \times N$ such that $(N, A)$ is a polytree and $\text{score}(A) \geq t$?

Given an instance $I := (N, \mathcal{F}, t)$ of POLYTREE LEARNING, an arc-set $A$ is a *solution of* $I$ if $(N, A)$ is a polytree and $\text{score}(A) \geq t$. Without loss of generality we may assume that $f_v(\emptyset) = 0$ for every $v \in N$ [Grüttemeier and Komusiewicz, 2020].

**Solution Structure and Input Representation.** We assume that the local scores $\mathcal{F}$ are given in *non-zero representation*. That is, $f_v(P)$ is part of the input if it is different from 0. For $N = \{v_1, \dots, v_n\}$, the local scores $\mathcal{F}$ are represented by a two-dimensional array $[Q_1, Q_2, \dots, Q_n]$, where each $Q_i$ is an array containing all triples $(f_{v_i}(P), |P|, P)$ where $f_{v_i}(P) > 0$. The size $|\mathcal{F}|$ is defined as the number of bits needed to store this two-dimensional array. Given an instance $I := (N, \mathcal{F}, t)$, we define $|I| := n + |\mathcal{F}| + \log(t)$.

For a vertex $v$, we call $\mathcal{P}_{\mathcal{F}}(v) := \{P \subseteq N \setminus \{v\} \mid f_v(P) > 0\} \cup \{\emptyset\}$ the *set of potential parents of* $v$. Given a yes-instance $I := (N, \mathcal{F}, t)$ of POLYTREE LEARNING, there exists a solution $A$ such that $P_v^A \in \mathcal{P}_{\mathcal{F}}(v)$ for every $v \in N$: If there is a vertex with $P_v^A \notin \mathcal{P}_{\mathcal{F}}(v)$ we can simply replace its parent set by $\emptyset$. The running times presentend in this paper will also be measured in the maximum number of potential parent sets $\delta_{\mathcal{F}} := \max_{v \in N} |\mathcal{P}_{\mathcal{F}}(v)|$ [Ordyniak and Szeider, 2013].

A tool for designing algorithms for POLYTREE LEARNING is the *directed superstructure* [Ordyniak and Szeider, 2013] which is the directed graph $S_{\mathcal{F}} := (N, A_{\mathcal{F}})$ with $A_{\mathcal{F}} := \{(u, v) \mid \exists P \in \mathcal{P}_{\mathcal{F}}(v) : u \in P\}$. In other words, there is an arc $(u, v) \in A_{\mathcal{F}}$ if and only if $u$ is a potential parent of $v$.

**Parameterized Complexity.** A problem is in the class XP for a parameter $k$ if it can be solved in $|I|^{g(k)}$ time for some computable function $g$. In other words, a problem is in XP if it can be solved within polynomial time for every fixed $k$. A problem is fixed-parameter tractable (FPT) for a parameter $k$ if it can be solved in $g(k) \cdot |I|^{\mathcal{O}(1)}$ time for some computable $g$. If a problem is W[1]-hard for a parameter $k$, then it is assumed to not be fixed-parameter tractable for $k$. A *problem kernel* is an algorithm that, given an instance $I$ with parameter $k$ computes in polynomial time an equivalent instance $I'$ with parameter $k'$ such that $|I'| + k' \leq h(k)$ for some computable function $h$. If $h$ is a polynomial, then the problem admits a *polynomial kernel*. Strong conditional running time bounds can also be obtained by assuming the *Exponential Time Hypothesis (ETH)*, a standard conjecture in complexity theory [Impagliazzo *et al.*, 2001]. For a detailed introduction into parameterized complexity we refer to the standard textbook [Cygan *et al.*, 2015].

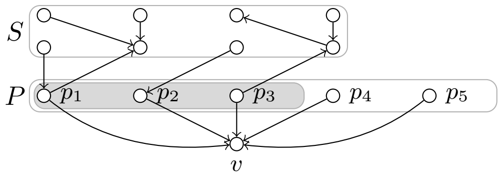

Figure 1: Illustration of an entry $T[v, P, S, i]$ where $i = 3$.

## 3 Parameterization by the Number of Vertices

In this section we study the complexity of POLYTREE LEARNING when parameterized by $n$, the number of vertices. Note that there are up to $n \cdot 2^{n-1}$ entries in $\mathcal{F}$ and thus, the total input size of an instance of POLYTREE LEARNING might be exponential in $n$.

**Theorem 1.** POLYTREE LEARNING *can be solved in* $3^n \cdot |I|^{\mathcal{O}(1)}$ *time.*

*Proof.* Let $I := (N, \mathcal{F}, t)$ be an instance of POLYTREE LEARNING. We describe a dynamic programming algorithm to solve $I$. We suppose an arbitrary fixed total ordering on the vertices of $N$. For every $P \subseteq N$, and every $i \in [1, |P|]$, we denote with $p_i$ the $i$th smallest element of $P$ according to the total ordering.

The dynamic programming table $T$ has entries of type $T[v, P, S, i]$ with $v \in N$, $P \in \mathcal{P}_{\mathcal{F}}(v)$, $S \subseteq N \setminus (P \cup \{v\})$, and $i \in [0, |P|]$.

Each entry stores the maximal score of an arc set $A$ of a polytree on $\{v\} \cup P \cup S$ where $v$ has no children, $v$ learns exactly the parent set $P$ under $A$, and for each $j \in [i+1, |P|]$, only the arc $(p_j, v)$ is incident with $p_j$ under $A$. See Figure 1 for an illustration.

We initialize the table $T$ by setting $T[v, P, \emptyset, i] := f_v(P)$ for all $v \in N$, $P \in \mathcal{P}_{\mathcal{F}}(v)$, and $i \in [0, |P|]$. The recurrence to compute an entry for $v \in N$, $P \in \mathcal{P}_{\mathcal{F}}(v)$, $S \subseteq N \setminus (P \cup \{v\})$, and $i \in [1, |P|]$ is

$$\begin{aligned} T[v, P, S, i] &:= \max_{S' \subseteq S} T[v, P, S \setminus S', i-1] + \\ &\max_{v' \in S' \cup \{p_i\}} \max_{\substack{P' \in \mathcal{P}_{\mathcal{F}}(v') \\ P' \subseteq S' \cup \{p_i\}}} T[v', P', (S' \cup \{p_i\}) \setminus (P' \cup \{v'\}), |P'|]. \end{aligned}$$

Note that the two vertex sets $P \cup (S \setminus S') \cup \{v\}$ and $P' \cup (S' \cup \{p_i\}) \setminus (P' \cup \{v'\}) \cup \{v'\} = S' \cup \{p_i\}$ share only the vertex $p_i$. Hence, combining the polytree on these two vertex sets results in a polytree.

If $i$ is equal to zero, then the recurrence is

$$\begin{aligned} &T[v, P, S, 0] := \\ &f_v(P) + \max_{v' \in S} \max_{\substack{P' \in \mathcal{P}_{\mathcal{F}}(v') \\ P' \subseteq S}} T[v', P', S' \setminus (P' \cup \{v'\}), |P'|]. \end{aligned}$$

Thus, to determine if $I$ is a yes-instance of POLYTREE LEARNING, it remains to check if $T[v, P, N \setminus (P \cup \{v\}), |P|] \geq t$ for some $v \in N$ and some $P \in \mathcal{P}_{\mathcal{F}}(v)$. The corresponding polytree can be found via traceback. The correctness proof is straightforward and thus omitted.

The table $T$ has $2^n \cdot n^2 \cdot \delta_{\mathcal{F}}$ entries. Each of these entries can be computed in $2^{|S|} \cdot n^2 \cdot \delta_{\mathcal{F}}$. Consequently, all entries can be computed in $\sum_{i=0}^{n} \binom{n}{i} 2^i \cdot |I|^{\mathcal{O}(1)} = 3^n \cdot |I|^{\mathcal{O}(1)}$ time in total. To evaluate if there is some $v \in N$ and some $P \in \mathcal{P}_{\mathcal{F}}(v)$ such that $T[v, P, N \setminus (P \cup \{v\}), |P|] \geq t$ can afterwards be done in $\mathcal{O}(n \cdot \delta_{\mathcal{F}})$ time. Hence, the total running time is $3^n \cdot |I|^{\mathcal{O}(1)}$. $\square$

The proof is closely related to a similar kernel lower bound for BAYESIAN NETWORK STRUCTURE LEARNING parameterized by $n$ [Grüttemeier and Komusiewicz, 2020].

**Theorem 2.** POLYTREE LEARNING *does not admit a polynomial kernel when parameterized by $n$, unless* NP $\subseteq$ coNP/poly.

## 4 Dependent Vertices

We now introduce a new parameter $d$ called *number of dependent vertices*. Given an instance $(N, \mathcal{F}, t)$ of POLYTREE LEARNING, a vertex $v \in N$ is called *dependent* if there is a nonempty potential parent-set $P \in \mathcal{P}_{\mathcal{F}}(v)$. Thus, a vertex is dependent if it might learn a nonempty parent set in a solution. A vertex that is not dependent is called *nondependent vertex*. Observe that $d$ is potentially smaller than $n$. We start with a simple XP-result.

**Theorem 3.** POLYTREE LEARNING *can be solved in* $(\delta_{\mathcal{F}})^d \cdot n^{\mathcal{O}(1)}$ *time.*

*Proof.* Choose for each dependent vertex $v_i$ one of its potential parent sets $P_i \in \mathcal{P}_{\mathcal{F}}(v_i)$ and check afterwards if $(N, \cup_{i=1}^{d} P_i \times v_i)$ is a polytree of score at least $t$. This is the case for some combination of potential parent sets if and only if the instance is a yes-instance. Since each check can be done in polynomial time and there are $(\delta_{\mathcal{F}})^d$ many combinations of potential parent sets, we obtain the stated running time. $\square$

We next show that there is little hope for a significant running time improvement on this simple brute-force algorithm. More precisely, we show that there is no $g(d) \cdot |I|^{\mathcal{O}(1)}$-time algorithm for some computable function $g$ (unless FPT $=$ W[1]) and a stronger ETH-based running time bound.

**Theorem 4.** POLYTREE LEARNING *is W[1]-hard when parameterized by the number of dependent vertices d; if the ETH holds, then it has no* $(\delta_{\mathcal{F}})^{o(d)} \cdot |I|^{\mathcal{O}(1)}$*-time algorithm. Both results even hold for instances where the directed superstructure* $S_{\mathcal{F}}$ *is a DAG.*

*Proof.* We reduce from INDEPENDENT SET where one is given an undirected graph $G = (V, E)$ and an integer $k$ and the question is whether there is a subset $S \subseteq V$ of size at least $k$ such that no two vertices in $S$ are connected by an edge. INDEPENDENT SET is W[1]-hard when parameterized by $k$ [Cygan *et al.*, 2015].

Given an instance $I = (G = (V, E), k)$ of INDEPENDENT SET, we describe how to construct an equivalent instance $I' = (N, \mathcal{F}, t)$ of POLYTREE LEARNING in polynomial time such that at most $k$ vertices have a nonempty potential parent set. Note that we can assume that every vertex of $G$ has degree at least one. We start with an empty set $N$ and add $k + 1$ vertices $v_1, \ldots, v_k$, and $v^*$. Moreover, we add a vertex $w_e$ for each edge $e \in E$. For every vertex $v \in V$, we set $P_v := \{w_{\{v,u\}} \mid u \in N_G(v)\} \cup \{v^*\}$ and we set $f_{v_i}(P_v) := 1$ for each $i \in [1, k]$. All other local scores are set to 0. Finally, we set $t := k$. This completes the construction of $I'$. We omit the correctness proof.

In the constructed instance, $d = k$ and $\delta_{\mathcal{F}} = n + 1$. Unless the ETH fails, INDEPENDENT SET cannot be solved in $n^{o(k)}$ time [Chen *et al.*, 2006] and, hence, POLYTREE LEARNING cannot be solved in $(\delta_{\mathcal{F}})^{o(d)} \cdot |I|^{\mathcal{O}(1)}$ time. $\square$

Theorem 4 points out a difference between POLYTREE LEARNING and BAYESIAN NETWORK STRUCTURE LEARNING (BNSL), where we aim to learn a DAG. In BNSL, a nondependent vertex $v$ can be easily removed from the input instance $(N, \mathcal{F}, t)$ by setting $N' := N \setminus \{v\}$ and modifying the local scores to $f'_u(P) := \max(f_u(P), f_u(P \cup \{v\}))$.

## 5 Dependent Vertices and Small Parent Sets

Due to Theorem 4, fixed-parameter tractability for POLYTREE LEARNING parameterized by $d$ is presumably not possible. However, in instances constructed in the proof of Theorem 4 the maximum parent set size $p$ is not bounded by some computable function in $d$. In practice there are many instances where $p$ is relatively small or upper-bounded by some small constant [van Beek and Hoffmann, 2015]. First, we provide an FPT algorithm for the parameter $d + p$. Second, we provide a polynomial kernel for the parameter $d$ if the maximum parent set size $p$ is constant. Both results are based on computing max $q$-representative sets in a matroid [Fomin *et al.*, 2014; Lokshtanov *et al.*, 2018].

To apply the technique of representative sets we assume that there is a solution with exactly $d \cdot p$ arcs and every nonempty potential parent set contains exactly $p$ vertices. This can be obtained with the following simple modification of an input instance $(N, \mathcal{F}, t)$: For every dependent vertex $v$ we add vertices $v_1, v_2, \ldots, v_p$ to $N$ and set $f_{v_i}(P) := 0$ for all their local scores. Then, for every potential parent set $P \in P_{\mathcal{F}}(v)$ with $|P| < p$ we set $f_v(P \cup \{v_1, \ldots, v_{p-|P|}\}) := f_v(P)$ and, afterwards, we set $f_v(P) := 0$. Then, the given instance is a yes-instance if and only if the modified instance has a solution with exactly $d \cdot p$ arcs. Furthermore, note that $f_v(\emptyset) = 0$ for every dependent vertex and every nonempty potential parent set has size exactly $p$ after applying the modification. Before we present the results of this section we state the definition of a matroid.

**Definition 1.** *A pair $M = (E, \mathcal{I})$, where $E$ is a set and $\mathcal{I}$ is a family of subsets of $E$ is a* matroid *if*

1. *$\emptyset \in \mathcal{I}$,*
2. *if $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$, and*
3. *if $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists some $b \in B \setminus A$ such that $A \cup \{b\} \in \mathcal{I}$.*

Given a matroid $M = (E, \mathcal{I})$, the sets in $\mathcal{I}$ are called *independent sets*. A *representation of $M$ over a field* $\mathbb{F}$ is a mapping $\varphi : E \to V$ where $V$ is some vector space over $\mathbb{F}$ such that $A \in \mathcal{I}$ if and only if the vectors $\varphi(a)$ with $a \in A$ are linearly independent in $V$. A matroid with a representation is called *linear matroid*. Given a set $B \subseteq E$, a set $A \subseteq E$ *fits* $B$ if $A \cap B = \emptyset$ and $A \cup B \in \mathcal{I}$.

**Definition 2.** *Let $M = (E, \mathcal{I})$ be a matroid, let $\mathcal{A}$ be a family of subsets of $E$, and let $w : \mathcal{A} \to \mathbb{N}_0$ be a weight function. A subfamily $\widehat{\mathcal{A}} \subseteq \mathcal{A}$* max $q$-represents *$\mathcal{A}$ (with respect to $w$) if for every set $B \subseteq E$ with $|B| = q$ the following holds: If there is a set $A \in \mathcal{A}$ that fits $B$, there exists some $\widehat{A} \in \widehat{\mathcal{A}}$ that fits $B$, and $w(\widehat{A}) \geq w(A)$. If $\widehat{\mathcal{A}}$ max $q$-represents $\mathcal{A}$ we write $\widehat{\mathcal{A}} \subseteq^q \mathcal{A}$.*

We refer to a set family $\mathcal{A}$ where every $A \in \mathcal{A}$ has size exactly $x \in \mathbb{N}_0$ as an *$x$-family*. Our results rely on the fact that max $q$-representative sets of an $x$-family can be computed efficiently as stated in a theorem by Lokshtanov et al. [2018] that is based on an algorithm by Fomin et al. [2014]. In the following, $\omega < 2.373$ is the matrix multiplication constant [Williams, 2012].

**Theorem 5** ([Lokshtanov *et al.*, 2018])**.** *Let $M = (E, \mathcal{I})$ be a linear matroid which representation can be encoded with a $k \times |E|$ matrix over the field $\mathbb{F}_2$ for some $k \in \mathbb{N}$. Let $\mathcal{A}$ be an $x$-family containing $\ell$ sets, and let $w : \mathcal{A} \to \mathbb{N}_0$ be a weight function. Then,*

*a) there exists some $\widehat{\mathcal{A}} \subseteq^q \mathcal{A}$ of size $\binom{x+q}{x}$ that can be computed with $\mathcal{O}\left(\binom{x+q}{x}^2 \cdot \ell x^3 k^2 + \ell \binom{x+q}{q}^{\omega} kx\right) + (k + |E|)^{\mathcal{O}(1)}$ operations in $\mathbb{F}_2$, and*

*b) there exists some $\widehat{\mathcal{A}} \subseteq^q \mathcal{A}$ of size $\binom{x+q}{x} \cdot k \cdot x$ that can be computed with $\mathcal{O}\left(\binom{x+q}{x} \cdot \ell x^3 k^2 + \ell \binom{x+q}{q}^{\omega-1} (kx)^{\omega-1}\right) + (k + |E|)^{\mathcal{O}(1)}$ operations in $\mathbb{F}_2$.*

We next define the matroid we use in this work. Recall that, given an instance $(N, \mathcal{F}, t)$ of POLYTREE LEARNING, the directed superstructure $S_{\mathcal{F}}$ is defined as $S_{\mathcal{F}} := (N, A_{\mathcal{F}})$ where $A_{\mathcal{F}}$ is the set of arcs that are potentially present in a solution, and we set $m := |A_{\mathcal{F}}|$. In this work we consider the *super matroid* $M_{\mathcal{F}}$ which we define as the graphic matroid [Tutte, 1965] of the super structure. Formally, $M_{\mathcal{F}} := (A_{\mathcal{F}}, \mathcal{I})$ where $A \subseteq A_{\mathcal{F}}$ is independent if and only if $(N, A)$ is a polytree. The super matroid is closely related to the *acyclicity matroid* that has been used for a constrained version of POLYTREE LEARNING [Gaspers *et al.*, 2015]. The proof of the following proposition is along the lines of the proof that the graphic matroid is a linear matroid.

**Proposition 1.** *Let $(N, \mathcal{F}, t)$ be an instance of* POLYTREE LEARNING*. Then, the super matroid $M_{\mathcal{F}}$ is a linear matroid and its representation can be encoded by an $n \times m$ matrix over the field $\mathbb{F}_2$.*

**An FPT Algorithm.** We now use the super matroid $M_{\mathcal{F}}$ to show that POLYTREE LEARNING can be solved in $2^{\omega dp} \cdot |I|^{\mathcal{O}(1)}$ time where $\omega$ is the matrix multiplication constant. The idea of the algorithm is simple: Let $H := \{v_1, \ldots, v_d\}$ be the set of dependent vertices, and for $i \in \{0, 1, \ldots, d\}$ let $H_i := \{v_1, \ldots, v_i\}$ be the set containing the first $i$ dependent vertices. The idea is that, for every $H_i$, we compute a family $\mathcal{A}_i$ of possible polytrees where only the vertices from $\{v_1, \ldots, v_i\}$ learn a nonempty potential parent set. We

**Algorithm 1** FPT-algorithm for parameter $d + p$

```
1: Input: $(N, \mathcal{F}, t)$ and dependent vertices $v_1, \ldots, v_d$
2: $\widehat{\mathcal{A}}_0 := \{\emptyset\}$
3: for $i = 1 \ldots d$ do
4:     $\widetilde{\mathcal{A}}_i = \bigcup_{P \in \mathcal{P}_\mathcal{F}(v_i) \setminus \{\emptyset\}} \widehat{\mathcal{A}}_{i-1} \oplus_{v_i} P$
5:     $\widehat{\mathcal{A}}_i :=$ ComputeRepresentation$(\widetilde{\mathcal{A}}_i, (d-i) \cdot p)$
6: return $\widehat{A} \in \widehat{\mathcal{A}}_d$ such that $(N, \widehat{A})$ is a polytree
   and score$(\widehat{A})$ is maximal
```

use the algorithm behind Theorem 5 as a subroutine to delete arc-sets from $\mathcal{A}_i$ that are not necessary to find a solution. We next define the operation $\oplus$. Intuitively, $\mathcal{A} \oplus_v P$ means that we extend each possible solution in the family $\mathcal{A}$ by the arc-set that defines $P$ as the parent set of a vertex $v$.

**Definition 3.** *Let $v \in N$, and let $\mathcal{A}$ be an $x$-family of subsets of $A_\mathcal{F}$ such that $P_v^A = \emptyset$ for every $A \in \mathcal{A}$. For a vertex set $P \subseteq N$ we define*

$$\mathcal{A} \oplus_v P := \{A \cup (P \times v) \mid A \in \mathcal{A} \text{ and } A \cup (P \times v) \in \mathcal{I}\}.$$

Observe that for every $A \in \mathcal{A}$, the set $P \times v$ is disjoint from $A$ since $P_v^A = \emptyset$. Consequently, $\mathcal{A} \oplus_v P$ is an $(x + |P|)$-family. The next lemma ensures that some operations (including $\oplus$) are compatible with representative sets.

**Lemma 1.** *Let $w : 2^{A_\mathcal{F}} \to \mathbb{N}_0$ be a weight function with $w(A) :=$ score$(A)$. Let $\mathcal{A}$ be an $x$-family of subsets of $A_\mathcal{F}$.*

*a) If $\widehat{\mathcal{A}} \subseteq^q \mathcal{A}$ and $\widehat{\mathcal{A}} \subseteq^q \widetilde{\mathcal{A}}$, then $\widehat{\mathcal{A}} \subseteq^q \mathcal{A}$.*

*b) If $\widehat{\mathcal{A}} \subseteq^q \mathcal{A}$ and $\mathcal{B}$ is an $x$-family of subsets of $A_\mathcal{F}$ with $\widehat{\mathcal{B}} \subseteq^q \mathcal{B}$, then $\widehat{\mathcal{A}} \cup \widehat{\mathcal{B}} \subseteq^q \mathcal{A} \cup \mathcal{B}$.*

*c) Let $v \in N$ and let $P \subseteq N$ such that $P_v^A = \emptyset$ for every $A \in \mathcal{A}$. Then, if $\widehat{\mathcal{A}} \subseteq^{q+|P|} \mathcal{A}$ it follows that $\widehat{\mathcal{A}} \oplus_v P \subseteq^q \mathcal{A} \oplus_v P$.*

We now describe the FPT algorithm. Let $I := (N, \mathcal{F}, t)$ be an instance of POLYTREE LEARNING. Let $H := \{v_1, v_2, \ldots, v_d\}$ denote the set of dependent vertices of $I$, and for $i \in \{0, 1, \ldots, d\}$ let $H_i := \{v_1, \ldots, v_i\}$ contain the first $i$ dependent vertices. Observe that $H_0 = \emptyset$ and $H_d = H$. We define $\mathcal{A}_i$ as the family of possible directed graphs (even the graphs that are no polytrees) where only the vertices in $H_i$ learn a nonempty potential parent set. Formally, this is

$$\mathcal{A}_i := \left\{ A \subseteq A_\mathcal{F} \middle| \begin{array}{l} P_v^A \in \mathcal{P}_\mathcal{F}(v) \setminus \{\emptyset\} \text{ for all } v \in H_i \\ P_v^A = \emptyset \text{ for all } v \in N \setminus H_i \end{array} \right\}.$$

The algorithm is based on the following recurrence.

**Lemma 2.** *If $i = 0$, then $\mathcal{A}_i = \{\emptyset\}$. If $i > 0$, $\mathcal{A}_i$ can be computed by $\mathcal{A}_i = \bigcup_{P \in \mathcal{P}_\mathcal{F}(v_i) \setminus \{\emptyset\}} \mathcal{A}_{i-1} \oplus_{v_i} P$.*

Intuitively, Lemma 2 states that $\mathcal{A}_i$ can be computed by considering $\mathcal{A}_{i-1}$ and combining every $A \in \mathcal{A}_{i-1}$ with every arc-set that defines a nonempty potential parent set of $v_i$. The correctness proof is straightforward and thus omitted. We next present the FPT algorithm.

**Theorem 6.** POLYTREE LEARNING *can be solved in $2^{\omega dp} \cdot |I|^{\mathcal{O}(1)}$ time.*

*Proof.* Let $I := (N, \mathcal{F}, t)$ be an instance of POLYTREE LEARNING with dependent vertices $H = \{v_1, \ldots, v_d\}$, let the families $\mathcal{A}_i$ for $i \in \{0, 1, \ldots, d\}$ be defined as above, and let $w : 2^{A_\mathcal{F}} \to \mathbb{N}_0$ be defined by $w(A) :=$ score$(A)$. All representing families considered in this proof are max representing families with respect to $w$. We prove that Algorithm 1 computes an arc-set $A$ such that $(N, A)$ is a polytree with maximal score.

The subroutine ComputeRepresentation$(\widetilde{\mathcal{A}}_i, (d - i) \cdot p)$ in Algorithm 1 is an application of the algorithm behind Theorem 5 *b*). It computes a max $((d - i) \cdot p)$-representing family for $\widetilde{\mathcal{A}}_i$. As a technical remark we mention that the algorithm as described by Lokshtanov et al. [2018] evaluates the weight $w(A)$ for $|\widetilde{\mathcal{A}}_i|$ many arc-sets $A$. We assume that each such evaluation $w(A)$ is replaced by the computation of score$(A) = \sum_{v \in H} f_v(P_v^A)$ which can be done in $|I|^{\mathcal{O}(1)}$ time.

*Correctness.* We first prove the following invariant.

**Claim 1.** *The family $\widehat{\mathcal{A}}_i$ max $((d - i) \cdot p)$-represents $\mathcal{A}_i$ and $|\widehat{\mathcal{A}}_i| \leq \max(1, \binom{dp}{ip}) \cdot n \cdot i \cdot p)$.*

*Proof*. The loop-invariant holds before entering the loop since $\widehat{\mathcal{A}}_0 := \{\emptyset\}$ in Line 2 and $\mathcal{A}_0 = \{\emptyset\}$. Suppose that the loop-invariant hold for the $(i-1)$th execution of the loop. We show that the loop-invariant holds after the $i$th execution.

First, consider Line 4. Since we assume that every nonempty potential parent set contains exactly $p$ vertices, Lemma 1 and Lemma 2 imply that $\widetilde{\mathcal{A}}_i$ max $((d - i) \cdot p)$-represents $\mathcal{A}_i$. Note that at this point $\widetilde{\mathcal{A}}_i$ contains up to $\max(1, \binom{dp}{(i-1)p}) \cdot n \cdot (i - 1) \cdot p) \cdot \delta_\mathcal{F}$ sets.

Next, consider Line 5. Since we assume that every nonempty potential parent set contains exactly $p$ vertices, the family $\widetilde{\mathcal{A}}_i$ is an $(i \cdot p)$-family. Then, the algorithm behind Theorem 5 computes a $((d - i) \cdot p)$-representing family $\widehat{\mathcal{A}}_i$. Then, by Theorem 5 *b*) and Lemma 1 *a*), $\widehat{\mathcal{A}}_i$ max $((d-i) \cdot p)$-represents $\mathcal{A}_i$ and $|\widehat{\mathcal{A}}_i| \leq \binom{dp}{ip} \cdot n \cdot i \cdot p$ after the execution of Line 5. ♢

We next show that $\widehat{\mathcal{A}}_d$ contains an arc-set that defines a polytree with maximum score and thus, a solution is returned in Line 6. Since we assume that there is an optimal solution $A$ that consists of exactly $d \cdot p$ arcs, this solution is an element of the family $\mathcal{A}_d$. Then, since $\widehat{\mathcal{A}}_d \subseteq^0 \mathcal{A}_d$, there exists some $\widehat{A} \in \widehat{\mathcal{A}}_d$ with $\widehat{A} \cup \emptyset \in \mathcal{I}$ and $w(\widehat{A}) \geq w(A)$. Since $\widehat{A} \cup \emptyset \in \mathcal{I}$, the graph $(N, \widehat{A})$ is a polytree, and since $w(\widehat{A}) \geq w(A)$ the score of $\widehat{A}$ is maximal.

*Running time.* We next analyze the running time of the algorithm. For this analysis, we use the inequality $\binom{a}{b} \leq 2^a$ for every $b \leq a$. Let $i$ be fixed.

We first analyze the running time of one execution of Line 4. Since $\widehat{\mathcal{A}}_{i-1}$ has size at most $\binom{dp}{(i-1)p} \cdot n \cdot i \cdot p$ due to Claim 1, Line 4 can be executed in $2^{dp} \cdot |I|^{\mathcal{O}(1)}$ time.

We next analyze the running time of one execution of Line 5. Recall that $\widetilde{\mathcal{A}}_i$ is an $(i \cdot p)$-family of size at most $\binom{dp}{(i-1)p} \cdot n \cdot i \cdot p \cdot \delta_{\mathcal{F}}$. Furthermore, recall that there are $|\widetilde{\mathcal{A}}_i|$ many evaluations of the weight function. Combining the running time from Theorem 5 $b$) with the time for evaluating $w$, the subroutine takes time

$$\mathcal{O}\left(\binom{dp}{ip}\binom{dp}{(i-1)p}\delta_{\mathcal{F}}(i\cdot p)^4 n^3 + \binom{dp}{(i-1)p}\delta_{\mathcal{F}}\binom{dp}{ip}^{\omega-1}(n\cdot i\cdot p)^{\omega}\right) + (n+m)^{\mathcal{O}(1)} + \underbrace{\binom{dp}{(i-1)p}\cdot n\cdot i\cdot p\cdot\delta_{\mathcal{F}}\cdot|I|^{\mathcal{O}(1)}}_{\text{evaluating } w}.$$

Therefore, one execution of Line 5 can be done in $2^{\omega dp}|I|^{\mathcal{O}(1)}$ time. Since there are $d$ repetitions of Lines 4–5, and Line 6 can be executed in $|I|^{\mathcal{O}(1)}$ time, the algorithm runs within the claimed running time. $\square$

**Problem Kernelization.** We now study problem kernelization for Polytree Learning parameterized by $d$ when the maximum parent set size $p$ is constant. We provide a problem kernel consisting of at most $(dp)^{p+1} + d$ vertices where each vertex has at most $(dp)^p$ potential parent sets which can be computed in $(dp)^{\omega p} \cdot |I|^{\mathcal{O}(1)}$ time. Observe that both, the running time and the kernel size, are polynomial for every constant $p$. Note also that, since $d + p \in \mathcal{O}(n)$, Theorem 2 implies that there is presumably no kernel of size $(d+p)^{\mathcal{O}(1)}$ that can be computed in $(d+p)^{\mathcal{O}(1)}$ time.

The basic idea of the kernelization is that we use max $q$-representations to identify nondependent vertices that are not necessary to find a solution.

**Theorem 7.** *There is an algorithm that, given an instance $(N, \mathcal{F}, t)$ of* Polytree Learning *computes in time $(dp)^{\omega p} \cdot |I|^{\mathcal{O}(1)}$ an equivalent instance $(N', \mathcal{F}', t)$ such that $|N'| \leq (dp)^{p+1} + d$ and $\delta_{\mathcal{F}'} \leq (dp)^p$.*

*Proof.* Let $H$ be the set of dependent vertices.

*Computation of the reduced instance.* We describe how we compute $(N', \mathcal{F}', t)$. We define the family $\mathcal{A}_v := \{P \times v \mid P \in \mathcal{P}_{\mathcal{F}}(v)\}$ for every $v \in H$ and the weight function $w : \mathcal{A}_v \to \mathbb{N}_0$ by $w(P \times v) := f_v(P)$. We then apply the algorithm behind Theorem 5 $a$) and compute a max $((d-1)\cdot p)$-representing family $\widehat{\mathcal{A}}_v$ for every $\mathcal{A}_v$.

Given all $\widehat{\mathcal{A}}_v$, a vertex $w$ is called *necessary* if $w \in H$ or if there exists some $v \in H$ such that $(w, v) \in A$ for some $A \in \widehat{\mathcal{A}}_v$. We then define $N'$ as the set of necessary vertices. Next, $\mathcal{F}'$ consists of local score functions $f'_v : 2^{N' \setminus \{v\}} \to \mathbb{N}_0$ with $f'_v(P) := f_v(P)$ for every $P \in 2^{N' \setminus \{v\}}$. In other words, $f'_v$ is the limitation of $f_v$ on parent sets that contain only necessary vertices.

Next, consider the running-time of the computation of $(N', \mathcal{F}', t)$. Since each $\mathcal{A}_v$ contains at most $\delta_{\mathcal{F}}$ arc sets and we assume that every potential parent set has size exactly $p$, each $\widehat{\mathcal{A}}_v$ can be computed in time

$$\mathcal{O}\left(\binom{dp}{p}^2 \cdot \delta_{\mathcal{F}} \cdot p^3 \cdot n^2 + \delta_{\mathcal{F}} \cdot \binom{dp}{p}^{\omega} \cdot n \cdot p\right) + (n+m)^{\mathcal{O}(1)}.$$

Observe that we use the symmetry of the binomial coefficient to analyze this running time. After computing all $\widehat{\mathcal{A}}_v$, we compute $N'$ and $\mathcal{F}'$ in polynomial time in $|I|$. The overall running time is $(dp)^{\omega p} \cdot |I|^{\mathcal{O}(1)}$.

*Correctness.* We next show that $(N, \mathcal{F}, t)$ is a yes-instance if and only if $(N', \mathcal{F}', t)$ is a yes-instance.

($\Leftarrow$) Let $(N', \mathcal{F}', t)$ be a yes-instance. Then, there exists an arc-set $A'$ such that $(N', A')$ is a polytree with score at least $t$. Since $N' \subseteq N$, $f'_v(P) = f_v(P)$ for every $v \in N'$, and $P \subseteq N' \setminus \{v\}$ we conclude that $(N, A')$ is a polytree with score at least $t$.

($\Rightarrow$) Let $(N, \mathcal{F}, t)$ be a yes-instance. We choose a solution $A$ for $(N, \mathcal{F}, t)$ such that $P_v^A \subseteq N'$ for as many dependent vertices $v$ as possible. We prove that this implies that $P_v^A \subseteq N'$ for all dependent vertices. Assume towards a contradiction that there is some $v \in H$ with $P_v^A \not\subseteq N'$. Observe that $(P_v^A \times v) \in \mathcal{A}_v \setminus \widehat{\mathcal{A}}_v$. We then define the arc set $B := \bigcup_{w \in H \setminus \{v\}} P_w^A \times w$. Since we assume that all nonempty potential parent sets have size exactly $p$, we conclude $|B| = (d-1)p$. Then, since $\widehat{\mathcal{A}}_v$ max $((d-1)\cdot p)$-represents $\mathcal{A}_v$ and $(P_v^A \times v) \in \mathcal{A}_v$ fits $B$ we conclude that there is some $(P \times v) \in \widehat{\mathcal{A}}_v$ such that $B \cap (P \times v) = \emptyset$, $(N, B \cap (P \times v))$ is a polytree, and $f_v(P) \geq f_v(P_v^A)$. Thus, $C := B \cup (P \times v)$ is a solution of $(N, \mathcal{F}, t)$ and the number of vertices $v$ that satisfy $P_v^C \subseteq N'$ is bigger than the number of vertices $v$ that satisfy $P_v^A \subseteq N'$. This contradicts the choice of $A$.

*Bound on the size of $|N'|$ and $\delta_{\mathcal{F}'}$.* By Theorem 5, each $\widehat{\mathcal{A}}_i$ has size at most $\binom{(d-1)p+p}{p} = \binom{dp}{p}$. Consequently, $\delta_{\mathcal{F}'} \leq (dp)^p$ and $N' \leq d \cdot \binom{dp}{p} \cdot p + d \leq (dp)^{p+1} + d$. $\square$

## 6 Conclusion

We believe that there is potential for practically relevant exact Polytree Learning algorithms and that this work could constitute a first step. Next, one should aim for improved parameterizations. For example, Theorem 3 gives an FPT algorithm for the parameter $\delta_{\mathcal{F}} + d$. Can we replace $\delta_{\mathcal{F}}$ or $d$ by smaller parameters? Instead of $\delta_{\mathcal{F}}$, one could consider parameters that are small when only few dependent vertices have many potential parent sets. Instead of $d$, one could consider parameters of the directed superstructure, for example the size of a smallest vertex cover; this parameter never exceeds $d$. We think that the algorithm with running time $3^n \cdot |I|^{\mathcal{O}(1)}$ might be practical for $n$ up to 20 based on experience with dynamic programs with a similar running time [Komusiewicz *et al.*, 2011]. A next step should be to combine this algorithm with heuristic data reduction and pruning rules to further increase the range of tractable values of $n$.

## References

[Chen *et al.*, 2006] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006.

[Chickering, 1995] David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Statistics, (AISTATS'95)*, pages 121–130. Springer, 1995.

[Chow and Liu, 1968] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inf. Theory*, 14(3):462–467, 1968.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Dasgupta, 1999] Sanjoy Dasgupta. Learning polytrees. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pages 134–141. Morgan Kaufmann, 1999.

[Fehri *et al.*, 2019] Hamid Fehri, Ali Gooya, Yuanjun Lu, Erik Meijering, Simon A. Johnston, and Alejandro F. Frangi. Bayesian polytrees with learned deep features for multi-class cell segmentation. *IEEE Trans. Image Process.*, 28(7):3246–3260, 2019.

[Fomin *et al.*, 2014] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*, pages 142–151. SIAM, 2014.

[Gaspers *et al.*, 2015] Serge Gaspers, Mikko Koivisto, Mathieu Liedloff, Sebastian Ordyniak, and Stefan Szeider. On finding optimal polytrees. *Theor. Comput. Sci.*, 592:49–58, 2015.

[Grüttemeier and Komusiewicz, 2020] Niels Grüttemeier and Christian Komusiewicz. Learning Bayesian networks under sparsity constraints: A parameterized complexity analysis. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*, pages 4245–4251. ijcai.org, 2020.

[Guo and Hsu, 2002] Haipeng Guo and William Hsu. A survey of algorithms for real-time Bayesian network inference. In *Working Notes of the Joint AAAI/UAI/KDD Workshop on Real-Time Decision Support and Diagnosis Systems*, 2002.

[Impagliazzo *et al.*, 2001] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[Komusiewicz *et al.*, 2011] Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann. Deconstructing intractability - A multivariate complexity analysis of interval constrained coloring. *J. Discrete Algorithms*, 9(1):137–151, 2011.

[Korhonen and Parviainen, 2013] Janne H. Korhonen and Pekka Parviainen. Exact learning of bounded tree-width Bayesian networks. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, (AISTATS'13)*, pages 370–378. JMLR.org, 2013.

[Korhonen and Parviainen, 2015] Janne H. Korhonen and Pekka Parviainen. Tractable Bayesian network structure learning with bounded vertex cover number. In *Proceedings of the Twenty-Eighth Annual Conference on Neural Information Processing Systems, (NIPS'15)*, pages 622–630. MIT Press, 2015.

[Lokshtanov *et al.*, 2018] Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. *ACM Trans. Algorithms*, 14(2):14:1–14:20, 2018.

[Ordyniak and Szeider, 2013] Sebastian Ordyniak and Stefan Szeider. Parameterized complexity results for exact Bayesian network structure learning. *J. Artif. Intell. Res.*, 46:263–302, 2013.

[Pearl, 1989] Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989.

[Ramaswamy and Szeider, 2021] Vaidyanathan Peruvemba Ramaswamy and Stefan Szeider. Turbocharging treewidth-bounded Bayesian network structure learning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*, 2021.

[Safaei *et al.*, 2013] Javad Safaei, Ján Manuch, and Ladislav Stacho. Learning polytrees with constant number of roots from data. In *Proceedings of the 26th Australasian Joint Conference on Advances in Artificial Intelligence (AI '13)*, volume 8272 of *Lecture Notes in Computer Science*, pages 447–452. Springer, 2013.

[Tutte, 1965] William Thomas Tutte. Lectures on matroids. *J. Research of the National Bureau of Standards (B)*, 69:1–47, 1965.

[van Beek and Hoffmann, 2015] Peter van Beek and Hella-Franziska Hoffmann. Machine learning of Bayesian networks using constraint programming. In *Proceedings of the twenty-first Conference of Principles and Practice of Constraint Programming (CP'15)*, volume 9255 of *Lecture Notes in Computer Science*, pages 429–445. Springer, 2015.

[Williams, 2012] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC '12)*, pages 887–898. ACM, 2012.