Learning Bayesian Networks Under Sparsity Constraints: A Parameterized Complexity Analysis

Niels Grüttemeier[*] and **Christian Komusiewicz**
Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Marburg, Germany
{niegru, komusiewicz}@informatik.uni-marburg.de

Abstract

We study the problem of learning the structure of an optimal Bayesian network when additional structural constraints are posed on the network or on its moralized graph. More precisely, we consider the constraint that the moralized graph can be transformed to a graph from a sparse graph class Π by at most k vertex deletions. We show that for Π being the graphs with maximum degree 1, an optimal network can be computed in polynomial time when k is constant, extending previous work that gave an algorithm with such a running time for Π being the class of edgeless graphs [Korhonen & Parviainen, NIPS 2015]. We then show that further extensions or improvements are presumably impossible. For example, we show that when Π is the set of graphs in which each component has size at most three, then learning an optimal network is NP-hard even if $k = 0$. Finally, we show that learning an optimal network with at most k edges in the moralized graph presumably is not fixed-parameter tractable with respect to k and that, in contrast, computing an optimal network with at most k arcs can be computed is fixed-parameter tractable in k.

1 Introduction

Bayesian networks are graphical models for probability distributions in which the presence of conditional dependencies between a set of random variables are represented via a directed acyclic graph (DAG) $D = (N, A)$ over a set N of n random variables [Darwiche, 2009]. An arc (u, v) in a Bayesian network means that the distribution of v depends on the value of u. Once we have obtained a Bayesian network, one may infer the distribution of some random variables given the values of other random variables.

First, however, one needs to learn the network from observed data. An important step herein is to learn the *structure* of the network, that is, the arc set of the network. In this step, one is given for each network vertex v and each set of possible parents of v a parent score and the goal is to learn an acyclic network with a maximal sum of parent scores. To represent the observed data as closely as possible it may seem appropriate to learn a tournament, that is, a DAG in which every pair of vertices u and v is connected either by the arc (u, v) or by the arc (v, u). There are, however, several reasons why learning a tournament-like DAG should be avoided (see [Darwiche, 2009] for a detailed discussion): First, such a network gives no information about which variables are conditionally independent. Second, including too many dependencies in the model makes the model vulnerable to overfitting. Finally, the problem of inferring distributions on a given Bayesian network is intractable when the DAG is tournament-like. More precisely, the inference problem on Bayesian networks is NP-hard [Cooper, 1990]. The key to obtaining efficient inference algorithms is to exploit that the network is tree-like: If the moralized graph has small treewidth, the inference task can be solved more efficiently [Darwiche, 2009]; the moralized graph of a network D is the undirected graph on the same vertex set that is obtained by adding an edge between each pair of vertices that is adjacent or has a common child in D.

Motivated by these reasons for avoiding tournament-like networks and instead aiming for tree-like networks, it has been proposed to learn optimal networks under structural constraints that guarantee that the network or its moralized graph is tree-like [Elidan and Gould, 2008; Korhonen and Parviainen, 2013; 2015; Chow and Liu, 1968; Gaspers *et al.*, 2015]. We continue this line of research, focusing on exact algorithms with worst-case running time guarantees. In other words, we want to find out for which structural constraints there are fast algorithms for learning optimal Bayesian networks under these constraints and for which constraints this is presumably impossible.

1.1 Known Results

The problem of learning a Bayesian network without structural constraints, which we call VANILLA-BNSL, is NP-hard [Chickering, 1995] and can be solved in $2^n n^{\mathcal{O}(1)}$ time by dynamic programming over all subsets of N [Ott and Miyano, 2003; Silander and Myllymäki, 2006].

When the network is restricted to be a branching, that is, a directed tree in which every vertex has indegree at most one, then an optimal network can be computed in polynomial time [Chow and Liu, 1968; Gaspers *et al.*, 2015]. Note that learning a more restricted Bayesian network is not necessarily

[*]Contact Author

easier: While learning a branching is solvable in polynomial time, the problem becomes NP-hard if we aim to learn a directed path [Meek, 2001].

On the negative side, the BOUNDED-TW-BNSL problem, where the moralized graph of the network is restricted to have treewidth at most ω is NP-hard for every fixed $\omega \geq 2$ and can be solved in $3^n n^{\omega+\mathcal{O}(1)}$ time [Korhonen and Parviainen, 2013]. Finally, BOUNDED-VC-BNSL where the moralized graph is restricted to have a vertex cover of size at most k can be solved in $4^k \cdot n^{2k+\mathcal{O}(1)}$ time [Korhonen and Parviainen, 2015]; a vertex cover in a graph G is a vertex set S such that every edge of G has at least one endpoint in S. Since having a bounded vertex cover implies that the graph has bounded treewidth, the networks that are learned by BOUNDED-VC-BNSL allow for fast inference algorithms. An algorithm with running time $f(k) \cdot |I|^{\mathcal{O}(1)}$ is unlikely for BOUNDED-VC-BNSL, since BOUNDED-VC-BNSL is W[1]-hard with respect to the parameter k [Korhonen and Parviainen, 2015]. Here, $|I|$ denotes the total input size. In other words, it seems necessary that the degree of the running time polynomial depends on k.

1.2 Our Results

The results for BOUNDED-VC-BNSL [Korhonen and Parviainen, 2015] form the starting point for our work. An alternative view of vertex covers is as follows: A graph has a vertex cover of size k if and only if it can be transformed into an edgeless graph by k vertex deletions. Thus, in BOUNDED-VC-BNSL we learn a network whose moralized graph is close, in terms of the number of vertex deletions, to a sparse graph class. We investigate whether there are further positive examples for such constrained network learning problems.

First, we consider the constraint that the moralized graph can be transformed into a graph with maximum degree 1 by at most k vertex deletions. We show that under this constraint, one can learn an optimal network in $n^{\mathcal{O}(k^2)} \cdot |I|^{\mathcal{O}(1)}$ time and thus in polynomial time for every constant value of k. This extends the result for BOUNDED-VC-BNSL in the following sense: the value of k can be arbitrarily smaller than the vertex cover number and thus for fixed k our algorithm can learn an optimal network for a larger class of graphs than the algorithm for BOUNDED-VC-BNSL. Observe that the moralized graphs still have bounded treewidth and thus inference on the learned networks will still be solvable efficiently.

We then show that it is unlikely that this positive result can be improved much further. First, we show that an algorithm with running time $f(k) \cdot |I|^{\mathcal{O}(1)}$ is unlikely. Moreover, we show that learning an optimal network that has maximum degree 2 is NP-hard and that learning an optimal network in which every component has at most three vertices is NP-hard (in a graph with maximum degree one every connected component has at most two vertices).

We further extend these negative results by showing that even in the very restricted scenario where we aim to compute an optimal network whose moralized graph has at most k edges, an $f(k) \cdot |I|^{\mathcal{O}(1)}$-time algorithm is unlikely. In contrast, if we restrict instead the number of arcs in the network, we obtain an algorithm with a running time of $2^{\mathcal{O}(k)} \cdot |I|^{\mathcal{O}(1)}$. Thus, putting structural constraints on the moralized graph may make the problem much harder than putting similar structural constraints on the network itself. We obtain a further hardness result for VANILLA-BNSL: Under standard assumptions in complexity theory, it is impossible that we can transform a given instance of VANILLA-BNSL in polynomial time to an equivalent one of size $n^{\mathcal{O}(1)}$. Thus, it is sometimes necessary to keep an exponential number of parent scores to compute an optimal network.

2 Preliminaries

2.1 Notation

We consider directed and undirected graphs that do not contain multiple edges.

A directed graph $D = (N, A)$ consists of a *vertex set* N and an *arc set* $A \subseteq N \times N$. Let $D = (N, A)$ be a directed graph. If D does not contain directed cycles, then D is called *directed acyclic graph* (DAG). An arc $(u, v) \in A$ is called *incoming arc into* v and an arc $(v, u) \in A$ is called *outgoing arc from* v. A vertex without incoming arcs is a *source*. A vertex without outgoing arcs is a *sink*. The set $P_v^A := \{u \in N \mid (u, v) \in A\}$ is called *parent set of* v. The vertices in P_v^A are called *parents of* v and for every $u \in P_v^A$, the vertex v is called *child of* u. We call v_1 an *ancestor of* v_ℓ and v_ℓ a *descendant of* v_1 if there is a path $(v_1, v_2, \ldots, v_\ell)$ in D.

An undirected graph $G = (V, E)$ consists of a vertex set V and an *edge set* $E \subseteq \{\{u, v\} \mid u, v \in V\}$. For a vertex $v \in V$, we write $N_G(v) := \{u \mid \{u, v\} \in E\}$ to denote the neighborhood of v in G. The *degree of a vertex* v is defined as $\deg_G(v) := |N_G(v)|$. Given an edge-set $E' \subseteq E$, we let $G \setminus E'$ denote the graph we obtain after deleting the edges in E' from G. Given a vertex-set $V' \subseteq V$, we let $G - V'$ denote the graph we obtain after deleting the vertices in V' and their incident edges from G. A set $S \subseteq V$ is called *dissociation set*, if $G - S$ has maximum degree one. The size of the smallest possible dissociation set for G is called *dissociation number of* G.

A *graph class* Π is a set of undirected graphs. For a graph class Π and $k \in \mathbb{N}$, let $\Pi + kv$ denote the class of graphs that can be transformed into a graph in Π by performing at most k vertex deletions. Analogously, we define $\Pi + ke$ as the class of graphs that can be transformed into a graph in Π by performing at most k edge deletions. We call Π *monotone* if Π is closed under edge- and vertex deletions. Note that Π being monotone implies that for every $k \in \mathbb{N}_0$, the graph classes $\Pi + kv$ and $\Pi + ke$ are monotone.

2.2 Bayesian Network Structure Learning

Given a vertex set N, we call a family $\mathcal{F} = \{f_v : 2^{N \setminus \{v\}} \to \mathbb{N}_0 \mid v \in N\}$ a family of *local scores* for N. Intuitively, for a vertex $v \in N$ and some $P \in 2^{N \setminus \{v\}}$, the value $f_v(P) \in \mathbb{N}_0$ represents the score we obtain if we choose exactly the vertices of P as parents for v. Given a vertex set N, local scores $\mathcal{F}$, and some integer $t \in \mathbb{N}_0$, an arc set $A \subseteq N \times N$ is called $(N, \mathcal{F}, t)$*-valid* if (N, A) is a DAG and $\sum_{v \in N} f_v(P_v^A) \geq t$.

Given a directed graph $D = (N, A)$, we call the undirected graph $\mathcal{M}(D) := (V, E_1 \cup E_2)$ with $V :=$

N, $E_1 = \{\{u,v\} \mid (u,v) \in A\}$, and $E_2 = \{\{u,v\} \mid u \text{ and } v \text{ have a common child in } D\}$ the *moralized graph* of D. The edges in E_2 are called *moral edges*. We may now define the problem.

$(\Pi + v)$-BAYESIAN NETWORK STRUCTURE LEARNING ($(\Pi + v)$-BNSL)
Input: A set of vertices N, local scores $\mathcal{F} = \{f_v \mid v \in N\}$, and two integers $t, k \in \mathbb{N}_0$.
Question: Is there an $(N, \mathcal{F}, t)$-valid arc set $A \subseteq N \times N$ such that $\mathcal{M}((N, A)) \in \Pi + kv$?

Intuitively, $(\Pi + v)$-BNSL can be seen as a version of VANILLA-BNSL where we add an additional constraint to the moralized graph of the resulting network. The problem $(\Pi + e)$-BNSL is defined on the same input and we ask if there exists an $(N, \mathcal{F}, t)$-valid arc set A such that $(N, A) \in \Pi + ke$. For both problems we call the requested arc set A a *solution* of the instance $(N, \mathcal{F}, t, k)$. Note that, if Π is monotone, contains infinitely many graphs, and $k = n^2$, the property $\mathcal{M}((N, A)) \in \Pi + kv$ or $\mathcal{M}((N, A)) \in \Pi + ke$ always hold, since every edgeless graph and an empty graph belong to Π. Hence, $(\Pi + v)$-BNSL and $(\Pi + e)$-BNSL are generalizations of VANILLA-BNSL and thus NP-hard for every monotone and infinite Π. For formal reasons, the problems are stated as decision problems. However, the algorithms presented in this work solve the corresponding optimization problem within the same running time.

Throughout this work, we let $n := |N|$ denote the number of vertices given in an instance $I = (N, \mathcal{F}, t, k)$ of $(\Pi + v)$-BNSL or $(\Pi + e)$-BNSL. Furthermore, we assume that the local scores $\mathcal{F}$ are given in *non-zero representation* [Ordyniak and Szeider, 2013], that is, for every $f_v \in \mathcal{F}$, each value $f_v(P)$ is only given if it is different from zero. We assume that for $N = \{v_1, \dots, v_n\}$, the local scores $\mathcal{F}$ are given as a two-dimensional array $\mathcal{F} := [Q_1, Q_2, \dots, Q_n]$, 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 then defined as the number of bits we need to store this two-dimensional array. As the *size of* I we define $|I| := n + |\mathcal{F}| + \log(t) + \log(k)$.

In this work, we consider $(\Pi + v)$-BNSL and $(\Pi + e)$-BNSL for some monotone graph classes Π. Observe that in these cases the following holds.

Proposition 1 *Let Π be a monotone graph property, and let $(N, \mathcal{F}, t, k)$ be a yes-instance of $(\Pi + v)$-BNSL (or $(\Pi + e)$-BNSL). Then, there exists a solution A for $(N, \mathcal{F}, t, k)$ such that for every $v \in N$ it holds that $f_v(P_v^A) = 0$ implies $P_v^A = \emptyset$.*

Given an instance $I := (N, \mathcal{F}, t, k)$ and some $v \in N$, we define the *set of potential parents of* v by $\mathcal{P}_{\mathcal{F}}(v) := \{P \subseteq N \setminus \{v\} : f_v(P) > 0\} \cup \{\emptyset\}$, which are exactly the parent sets stored in $\mathcal{F}$ together with the empty set. If Π is monotone, we can assume by Proposition 1 that in a solution A for I, every v has a parent set $P_v^A \in \mathcal{P}_{\mathcal{F}}(v)$. An important measurement for the running times of our algorithms is the maximum number of potential parent sets $\delta_{\mathcal{F}} := \max_{v \in N} |\mathcal{P}_{\mathcal{F}}(v)|$ [Ordyniak and Szeider, 2013]. Given a vertex $v \in N$, we can iterate over all vertices in potential parent sets of v in $\mathcal{O}(\delta_{\mathcal{F}} \cdot n)$ time.

Another tool for designing algorithms for BNSL problems is the superstructure [Ordyniak and Szeider, 2013]. The *superstructure* of N and $\mathcal{F}$ is the directed graph $S_{\vec{\mathcal{F}}} = (N, A_{\mathcal{F}})$ with $A_{\mathcal{F}} = \{(u, v) \mid \exists P \in \mathcal{P}_{\mathcal{F}}(v) : u \in P\}$.

2.3 Parameterized Complexity

A problem is called *slicewise polynomial (XP)* for a parameter k if it can be solved in time $\mathcal{O}(|I|^{f(k)})$ for a computable function f. That is, the problem is solvable in polynomial time when k is constant. A problem is called *fixed-parameter tractable (FPT)* for a parameter k if it can be solved in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for a computable function f. If a problem is W[1]-hard then it is assumed to be fixed-parameter *intractable*. A problem has a polynomial kernel for a parameter k if there is a polynomial-time algorithm that, given an instance I with parameter k, computes an equivalent instance I' with parameter $k' \leq k$ of size $g(k)$ where g is a polynomial. For a detailed introduction into parameterized complexity we refer to [Cygan *et al.*, 2015].

3 Vertex Deletion Distances

Let $\Pi_1 := \{G \mid G \text{ has maximum degree } 1\}$, that is, every $G \in \Pi_1$ consists only of isolated edges and isolated vertices. Observe that Π_1 is monotone. In this section we consider $(\Pi_1 + v)$-BNSL. Note that, given some $k \in \mathbb{N}$, a DAG $D = (N, A)$ with $\mathcal{M}(D) \in \Pi_1 + kv$ is a DAG whose moralized graph has a dissociation set of size at most k. Since the treewidth of a graph is never bigger than the dissociation number plus one, the moralized graph of the resulting Bayesian network has treewidth at most $k + 1$. Before we describe the main idea of the algorithm we provide the following simple observation .

Proposition 2 *Let $D = (N, A)$ be a DAG and $S \subseteq N$ be a dissociation set of $\mathcal{M}(D)$. Then, at most $2|S|$ vertices in $N \setminus S$ have descendants in S.*

The idea is the following: If we know the dissociation set S, all their ancestors Q under A, and the arcs between them, the remaining arcs of A can be found in polynomial time. We start with the following .

Definition 3 *let N be a vertex set and let $S \subseteq N$. A set $Q \subseteq N \setminus S$ together with an arc-set $A_Q \subseteq (S \cup Q) \times (S \cup Q)$ is called* ancestor tuple for *S, if*

a) $D_Q := (S \cup Q, A_Q)$ is a DAG, and

b) for every $v \in Q$ exists some $w \in S$ such that w is a descendant of v in D_Q, and

c) in the moralized graph $\mathcal{M}(D_Q)$, every $v \in Q$ has at most one neighbor outside S.

Intuitively, the ancestor tuple $\langle Q, A_Q \rangle$ is the part of the solution that our algorithm finds via bruteforce. We next formally define an arc set containing the remaining arcs of a solution. To this end, we introduce some notation. Given an ancestor tuple $\langle Q, A_Q \rangle$ of some $S \subseteq N$, we let $R := N \setminus (S \cup Q)$ denote the remaining vertices of N. Moreover, we define $Q_0 := \{v \in Q \mid \deg_{\mathcal{M}(D_Q)-S}(v) = 0\}$ and $Q_1 := Q \setminus Q_0$. By Definition 3 c), the vertices of Q_1 are the vertices that have degree one in $\mathcal{M}(D_Q) - S$.

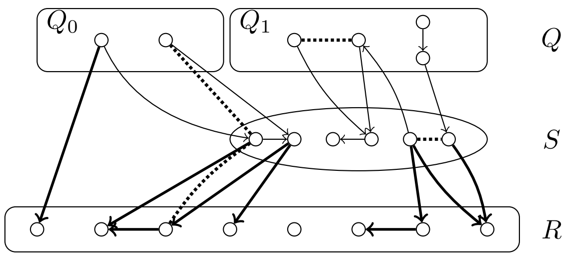

Figure 1: A DAG D whose moralized graph has a dissociation set S. The arc set of D is decomposed into the arc set of an ancestor tuple $\langle Q, A_Q\rangle$ and a suitable arc set A_R. The thin arrows correspond to the arcs of A_Q and the thick arrows correspond to the arcs of A_R. The dotted edges are the moral edges.

Definition 4 *Let N be a vertex set, let $S \subseteq N$, and let $\langle Q, A_Q\rangle$ be an ancestor tuple of S. An arc-set $A_R \subseteq (S \cup Q_0 \cup R) \times R$ is called* suitable for $\langle Q, A_Q\rangle$ *if A_R contains no self-loops, and every $w \in Q_0 \cup R$ has at most one incident arc in $A_R \cap ((R \cup Q_0) \times R)$.*

Next, we state the connection between Definitions 3, 4, and DAGs whose moralized graph has dissociation set S. First, an ancestor tuple for some S and a suitable arc set can be combined to a DAG where S is a dissociation set of the moralized graph.

Proposition 5 *Let N be a vertex set, let $S \subseteq N$, let $\langle Q, A_Q\rangle$ be an ancestor tuple of S, and let $A_R \subseteq (S \cup Q_0 \cup R) \times R$ be a suitable arc set for $\langle Q, A_Q\rangle$. Then, 1) $D := (N, A_Q \cup A_R)$ is a DAG, and 2) S is a dissociation set of $\mathcal{M}(D)$.*

Conversely, the arc set of every DAG whose moralized graph has a dissociation set S can be partitioned into the arc set of an ancestor tuple for S and a suitable arc set; Figure 1 shows an example.

Proposition 6 *Let $D = (N, A)$ be a DAG, let $S \subseteq N$ be a dissociation set of $\mathcal{M}(D)$. Then, $\langle Q, A_Q\rangle$ defined by*

$$Q := \{v \in N \setminus S \mid v \text{ has descendants in } S\},$$
$$A_Q := (S \cup Q) \times (S \cup Q) \cap A,$$

is an ancestor tuple of S with $|Q| \le 2|S|$ and $A \setminus A_Q$ is suitable for $\langle Q, A_Q\rangle$.

Next, we use ancestor tuples and suitable arc sets to decompose the local scores in $(\Pi_1 + v)$-BNSL. Let $(N, \mathcal{F}, t, k)$ be an instance of $(\Pi_1 + v)$-BNSL and let $S \subseteq N$. Furthermore, let $\langle Q, A_Q\rangle$ be an ancestor tuple for S and let A_R be suitable for $\langle Q, A_Q\rangle$. Then, we set $A := A_Q \cup A_R$ and by Proposition 5, $D := (N, A)$ is a DAG and S is a dissociation set of $\mathcal{M}(D)$. Observe that all arcs in A_R have endpoints in R and all arcs in A_Q have endpoints in $Q \cup S$. Hence, for every $v \in N$ it holds that either all incoming arcs are in A_Q or in A_R. Hence, the score of A under $\mathcal{F}$ is

$$\sum_{v \in N} f_v(P_v^A) = \sum_{v \in S \cup Q} f_v(P_v^{A_Q}) + \sum_{v \in R} f_v(P_v^{A_R}).$$

We next show that, if S and $\langle Q, A_Q\rangle$ are given, we can find A_R in polynomial time. More precisely, we solve the following problem.

$(\Pi_1 + v)$-BNSL-COMPLETION
Input: A set of vertices N, a subset $S \subseteq N$, an ancestor tuple $\langle Q, A_Q\rangle$ for S, local scores $\mathcal{F} = \{f_v \mid v \in N\}$, and an integer t.
Question: Is there an arc-set A_R that is suitable for $\langle Q, A_Q\rangle$ such that $\sum_{v \in R} f_v(P_v^{A_R}) \ge t$?

Proposition 7 $(\Pi_1 + v)$-BNSL-COMPLETION *can be solved in $\mathcal{O}(n^2 \cdot (n\delta_{\mathcal{F}} + \sqrt{n}))$ time.*

PROOF We give a polynomial-time reduction to MAXIMUM WEIGHT MATCHING, where one is given a graph $G = (V, E)$, edge-weights $\omega : E \to \mathbb{N}$, and $\ell \in \mathbb{N}$ and the question is if there exists a set $M \subseteq E$ of pairwise non-incident edges such that $\sum_{e \in M} \omega(e) \ge \ell$.

Construction: Let $I := (N, S, \langle Q, A_Q\rangle, \mathcal{F}, t)$ be an instance of $(\Pi_1 + v)$-BNSL-COMPLETION. We construct an equivalent instance (G, ω, ℓ) of MAXIMUM WEIGHT MATCHING. We first define $G := (V, E)$ with $V := Q_0 \cup R \cup R'$, where $R' := \{v' \mid v \in R\}$, and $E := X \cup Y \cup Z$, where

$$X := \{\{v, w\} \mid v, w \in R, v \neq w\},$$
$$Y := \{\{v, w\} \mid v \in R, w \in Q_0\}, \text{ and}$$
$$Z := \{\{v, v'\} \mid v \in R\}.$$

Next, we define edge-weights $\omega : E \to \mathbb{N}$: For $e = \{v, v'\} \in Z$, we set $\omega(e) := \max_{S' \subseteq S} f_v(S')$. Furthermore, for $e = \{v, w\} \in Y$ with $v \in R$ and $w \in Q_0$, we set $\omega(e) := \max_{S' \subseteq S} f_v(S' \cup \{w\})$. Finally, for $e = \{v, w\} \in X$, we set $\omega(e) := \max(\varphi(v, w), \varphi(w, v))$, where

$$\varphi(u_1, u_2) := \max_{S' \subseteq S} f_{u_1}(S' \cup \{u_2\}) + \max_{S' \subseteq S} f_{u_2}(S').$$

To complete the construction , we set $\ell := t$.

Due to lack of space, the correctness proof is deferred. We provide some intuition: A maximum weight matching M in G corresponds to the parent sets of vertices in R and therefore to arcs in A_R. An edge $\{v, v'\} \in Z$ with $v \in R$ corresponds to a parent set of v that contains only vertices from S. Moreover, an edge $\{v, w\} \in Y$ with $v \in R$ corresponds to a parent set of v that contains $w \in Q_0$ and vertices from S. Finally, an edge $\{v, w\} \in X$ means that either $v \in P_w^{A_R}$ or $w \in P_v^{A_R}$. Since M is a matching, the corresponding arc set is suitable for $\langle Q, A_Q\rangle$. □

Theorem 8 $(\Pi_1 + v)$-BNSL *can be solved in $\mathcal{O}((n\delta_{\mathcal{F}})^{3k} \cdot \text{poly}(|I|))$ time.*

PROOF (SKETCH) Iterate over all possible choices for S and $\langle Q, A_Q\rangle$ where $|S| \le k$ and $|Q| \le 2k$ in $\mathcal{O}((n\delta_{\mathcal{F}})^{3k})$ time and find the remaining arcs in polynomial time using Proposition 7. □

Note that in a Bayesian network whose moralized graph has a dissociation set of size k the maximum parent set size is $k+1$. Otherwise, the moralized graph has a clique of size at least $k+3$, contradicting the fact that it has a dissociation set of size k. Hence, we can delete every triple $(f_v(P), |P|, P)$ with $|P| > k+1$ from $\mathcal{F}$. Afterwards, $\delta_{\mathcal{F}} \le \binom{n}{k+1}$.

Corollary 9 $(\Pi_1 + v)$-BNSL *can be solved in* $n^{\mathcal{O}(k^2)} + \mathrm{poly}(|I|)$ *time.*

We complement the algorithm for $(\Pi_1 + v)$-BNSL by several hardness results. First, there is little hope to obtain an FPT algorithm for $(\Pi_1 + v)$-BNSL parameterized by $k + t$ since the problem is W[1]-hard. Observe that this is not implied by the W[1]-hardness of BOUNDED-VC-BNSL since $(\Pi_1 + v)$-BNSL is a different problem where we aim to find a Bayesian network with different restrictions. However, the proof is closely related to the W[1]-hardness-proof for BOUNDED-VC-BNSL [Korhonen and Parviainen, 2015].

Proposition 10 (Π_1+v)-BNSL *is W[1]-hard for $k+t$, even when the superstructure $S_{\vec{\mathcal{F}}}$ is a DAG and the maximum parent set size is three.*

We next consider monotone graph classes that are related to the class Π_1 . Let Π_2 be the class of graphs that have maximum degree two, and let Π_3^{COC} be the class of graphs where each connected component has size at most three. These graph classes are superclasses of Π_1, that is $\Pi_1 \subseteq \Pi_2$ and $\Pi_1 \subseteq \Pi_3^{\mathrm{COC}}$. Consequently, if a graph G belongs to the graph class $\Pi_1 + kv$ for some $k \in \mathbb{N}_0$, then there exist $k' \leq k$ and $k'' \leq k$ such that $G \in \Pi_2 + k'v$ and $G \in \Pi_3^{\mathrm{COC}} + k''v$. Moreover, observe that the treewidth of G is not bigger than $\min(k', k'') + \mathcal{O}(1)$.

With the next proposition we show that there is little hope that we can solve $(\Pi_2 + v)$-BNSL or $(\Pi_3^{\mathrm{coc}} + v)$-BNSL in XP time when parameterized by k. The hardness of (Π_2+v)-BNSL relies on the fact that learning paths is NP-hard [Meek, 2001].

Proposition 11 (Π_2+v)-BNSL *and* $(\Pi_3^{\mathrm{coc}}+v)$-BNSL *are NP-hard even if $k = 0$.*

4 Edge Deletion Distances

4.1 Bounded-Edges-BNSL

We now consider a version of BNSL, where we aim to learn a network whose moralized graph has a bounded number of edges. More precisely, we consider $(\Pi_0 + e)$-BNSL where Π_0 is the class of edgeless graphs. Clearly, Π_0 is monotone and $(\Pi_0 + e)$-BNSL can be solved in $\mathcal{O}(n^{2k} \cdot \mathrm{poly}(|I|))$ time by a brute-force algorithm. To put this into context, we show that there is little hope to solve $(\Pi_0 + e)$-BNSL in FPT time for $t + k$.

Theorem 12 $(\Pi_0 + e)$-BNSL *is W[1]-hard when parameterized by $t + k$, even when $S_{\vec{\mathcal{F}}}$ is a DAG and the maximum parent set size is three.*

Learning a Bayesian network whose moralized graph has a bounded feedback edge set is also W[1]-hard when parameterized by the size of the *feedback edge set*. More formally, let Π_F be the class of forests, which are undirected acyclic graphs.

Theorem 13 $(\Pi_F + e)$-BNSL *is W[1]-hard when parameterized by k, even when $S_{\vec{\mathcal{F}}}$ is a DAG and the maximum parent set size is four.*

For efficient inference it is desirable to have a small treewidth in the moralized graph. The size of a feedback edge set is a large upper bound for the treewidth. Since even learning Bayesian networks under this constraint is W[1]-hard, it appears to be unlikely to obtain fixed-parameter tractability for natural parameters that bound the treewidth of the moralized graph.

4.2 Bounded-Arcs-BNSL

Next, we consider a version of BAYESIAN NETWORK STRUCTURE LEARNING where we want to learn a Bayesian network with a bounded number of arcs. In contrast to $(\Pi_0 + e)$-BNSL, the additional sparsity constraint does not affect the moralized graph but only the arcs of the DAG itself. The problem is formally defined as follows.

BA-BNSL
Input: A set of vertices N, local scores $\mathcal{F} = \{f_v \mid v \in N\}$, and two integers $t, k \in \mathbb{N}$.
Question: Is there an $(N, \mathcal{F}, t)$-valid arc set $A \subseteq N \times N$ such that $|A| \leq k$?

BA-BNSL is a generalization of VANILLA-BNSL and therefore NP-hard. We prove that BA-BNSL becomes polynomial-time solvable if the superstructure is a DAG. The algorithm uses dynamic programming over a topological ordering of $S_{\vec{\mathcal{F}}}$.

Proposition 14 BA-BNSL *can be solved in $\mathcal{O}(\delta_{\mathcal{F}} \cdot k \cdot n)$ time if the superstructure $S_{\vec{\mathcal{F}}}$ is a DAG.*

PROOF We give a simple dynamic programming algorithm. Let $N := \{1, \ldots, n\}$, and let $(N, \mathcal{F}, t, k)$ be an instance of BA-BNSL such that $S_{\vec{\mathcal{F}}}$ is a DAG. Then, there exists such a topological ordering of $S_{\vec{\mathcal{F}}}$. Without loss of generality, let $(n, n-1, \ldots, 2, 1)$ be such topological ordering. Hence, for every arc (a, b) of $S_{\vec{\mathcal{F}}}$ it holds that $a > b$.

The dynamic programming table T has entries of the type $T[i, j]$ for all $i \in \{0, 1, \ldots, n\}$ and $j \in \{0, 1, \ldots, k\}$. Each entry stores the maximum sum of local scores of the vertices $(i, \ldots, 1)$ of the topological ordering that can be obtained by an arc set A of size at most j. For $i = 0$, we set $T[0, j] = 0$ for all $j \in \{0, \ldots, k\}$. The recurrence to compute an entry for $i > 0$ is

$$T[i, j] = \max_{P \in P_{\mathcal{F}}(i), |P| \leq j} (f_i(P) + T[i-1, j-|P|]),$$

and the result can then be computed by checking if $T[n, k] \geq t$. The corresponding network can be found by traceback. The correctness proof is straightforward and thus omitted. The size of T is $\mathcal{O}(n \cdot k)$ and each entry $T[i, j]$ can be computed in $\mathcal{O}(\delta_{\mathcal{F}})$ time by iterating over the at most $\delta_{\mathcal{F}}$ triples $(f_i(P), |P|, P)$ in $\mathcal{F}$ for the vertex i. Therefore, BA-BNSL can be solved in $\mathcal{O}(\delta_{\mathcal{F}} \cdot k \cdot n)$ time if $S_{\vec{\mathcal{F}}}$ is a DAG. □

4.3 A Randomized Algorithm for BA-BNSL

The dynamic programming algorithm behind Proposition 14 can be adapted to obtain an FPT algorithm for BA-BNSL when parameterized by the number of arcs k. The algorithm is based on color coding [Alon *et al.*, 1995]: In a Bayesian network with at most k arcs, there are at most $2k$ vertices which are endpoints of such arcs. The idea of color coding is to randomly color the vertices of N with $2k$ colors and find a solution A where all vertices that are incident with arcs of A are colored with pairwise distinct colors. To describe the color coding algorithm, we introduce some notation. Let N be a set of vertices. A function $\chi : N \to \{1, \ldots, 2k\}$ is called a *coloring (of N with $2k$ colors)*. Given a color $c \in \{1, \ldots, 2k\}$, we call $\chi^{-1}(c) := \{v \in N \mid \chi(v) = c\}$ the *color class of* c. For a subset $N' \subseteq N$, we let $\chi(N') := \{\chi(v) \mid v \in N'\}$, and for a subset $C \subseteq \{1, \ldots, 2k\}$ we let $\chi^{-1}(C) := \bigcup_{c \in C} \chi^{-1}(c)$. The following definition is important for our algorithm.

Definition 15 *Let N be a set of vertices and let $\chi : N \to \{1, \ldots, 2k\}$ be a coloring of N. An arc set $A \subseteq N \times N$ is called* color-loyal for χ *if for every color class $\chi^{-1}(c)$ it holds that*

a) there is no $(v, w) \in A$ with $v, w \in \chi^{-1}(c)$, and

b) there is at most one vertex $v \in \chi^{-1}(c)$ such that $P_v^A \neq \emptyset$.

Consider the following auxiliary problem.

COLORED BA-BNSL
Input: A set of vertices N, local scores $\mathcal{F} = \{f_v \mid v \in N\}$, two integers $t, k \in \mathbb{N}$, and a coloring $\chi : N \to \{1, \ldots, 2k\}$.
Question: Is there an $(N, \mathcal{F}, t)$-valid arc set $A \subseteq N \times N$ that is color-loyal for χ and $|A| \leq k$?

Intuitively, COLORED BA-BNSL is the problem that we solve after we randomly choose a coloring of N. The correspondence between BA-BNSL and COLORED BA-BNSL is as follows.

Proposition 16 *Let $I = (N, \mathcal{F}, t, k)$ be an instance of* BA-BNSL. *If I is a yes-instance of* BA-BNSL, *then there exist at least $(2k)!(2k)^{(n-2k)}$ colorings $\chi : N \to \{1, 2, \ldots, 2k\}$ such that $(N, \mathcal{F}, t, k, \chi)$ is a yes-instance of* COLORED BA-BNSL.

Proposition 17 COLORED BA-BNSL *can be solved in $\mathcal{O}(4^k k^2 n^2 \delta_{\mathcal{F}})$ time.*

PROOF We fill a dynamic programming table T with entries of type $T[C', k']$ where $C' \subseteq C$ and $k' \in \{0, 1, \ldots, k\}$. Every entry stores the maximum value of $\sum_{v \in \chi^{-1}(C')} f_v(P_v^A)$ over all possible DAGs $D = (N, A)$, where $A \subseteq \chi^{-1}(C') \times \chi^{-1}(C')$ is color-loyal for χ and contains at most k' arcs. We set $T[\{c\}, k'] := \sum_{w \in \chi^{-1}(c)} f_w(\emptyset)$ for every $c \in C$ and $k' \in \{0, 1, \ldots, 2k\}$. The recurrence to compute the entry for $C' \subseteq C$ with $|C'| > 1$ is

$$T[C', k'] = \max_{c \in C'} \max_{v \in \chi^{-1}(c)} \max_{\substack{P \in \mathcal{P}_{\mathcal{F}}(v) \\ |P| \leq k' \\ \chi(P) \subseteq C' \setminus \{c\}}} H_{C'}^{k'}(c, v, P),$$

where

$$H_{C'}^{k'}(c, v, P) = T[C' \setminus \{c\}, k' - |P|] + f_v(P) + \sum_{w \in \chi^{-1}(c) \setminus \{v\}} f_w(\emptyset).$$

The result can be computed by checking if $T[C, k] \geq t$. Note that the corresponding network can be found via traceback. The correctness proof is straightforward and thus omitted. The size of T is $\mathcal{O}(2^{2k} \cdot k)$. We omit the proof of the polynomial running time part. □

Propositions 16 and 17 give the following.

Theorem 18 *There exists a randomized algorithm for* BA-BNSL *that, in time $\mathcal{O}((2e)^{2k} \cdot k^2 n^2 \delta_{\mathcal{F}})$ returns* no, *if given a no-instance and returns* yes *with a constant probability of at least $1 - \frac{1}{e}$, if given a yes-instance.*

The algorithm can be derandomized with standard techniques [Naor *et al.*, 1995; Cygan *et al.*, 2015].

Corollary 19 BA-BNSL *can be solved in* $(2e)^{2k} \cdot k^{\mathcal{O}(\log(k))} \cdot \text{poly}(|I|)$ *time.*

Bounding the number of arcs appears to be not so relevant for practical use. However, the algorithm might be useful as a heuristic upper-bound: If we want to add a restricted number of dependencies to a given Bayesian network, the result of BA-BNSL gives an upper bound for the profit we can expect from that modification. The above algorithm is complemented by the following negative result.

Theorem 20 BA-BNSL *parameterized by $t + k$ does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly *even when $k = n^2$.*

Observe that for instances of BA-BNSL with $k = n^2$, the budget of arcs can never be exceeded since a DAG has at most $\binom{n}{2} < n^2$ arcs. Hence, on instances with $k = n^2$ we ask for an $(N, \mathcal{F}, t)$-valid arc set without additional sparsity constraints. Thus, BA-BNSL and VANILLA-BNSL are the same when $k = n^2$. Then, Theorem 20 implies the following.

Corollary 21 VANILLA-BNSL *parameterized by n does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly.

5 Conclusion

We have outlined the tractability borderline of BAYESIAN NETWORK STRUCTURE LEARNING with respect to several structural constraints on the learned network or on its moralized graph. In particular, we have shown that putting structural sparsity constraints on the moralized graph may make the problem harder than putting similar constraints on the network. This is somewhat counterintuitive since the moralized graph is a supergraph of the underlying undirected graph of the network. It seems interesting to investigate this issue further, that is, to find structural constraints such that putting these constraints on the network leads to an easier problem than putting them on the moralized graph.

While none of our algorithms have direct practical applications, they may be useful as bounds on the score that can be achieved for example by adding k arcs to a network that is currently considered in the search. Thus, it would be interesting to explore variants of BAYESIAN NETWORK STRUCTURE LEARNING where the input contains a partial network and the aim is to extend it. Do the positive results for BAYESIAN NETWORK STRUCTURE LEARNING also hold for this more general problem?

References

[Alon *et al.*, 1995] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[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. Information Theory*, 14(3):462–467, 1968.

[Cooper, 1990] Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.*, 42(2-3):393–405, 1990.

[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.

[Darwiche, 2009] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[Elidan and Gould, 2008] Gal Elidan and Stephen Gould. Learning bounded treewidth Bayesian networks. In *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, (NIPS'08)*, pages 417–424. Curran Associates, Inc., 2008.

[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.

[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.

[Meek, 2001] Christopher Meek. Finding a path is harder than finding a tree. *J. Artif. Intell. Res.*, 15:383–389, 2001.

[Naor *et al.*, 1995] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the Thirty-Sixth Annual Symposium on Foundations of Computer Science, (FOCS'95)*, pages 182–191. IEEE Computer Society, 1995.

[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.

[Ott and Miyano, 2003] Sascha Ott and Satoru Miyano. Finding optimal gene networks using biological constraints. *Genome Informatics*, 14:124–133, 2003.

[Silander and Myllymäki, 2006] Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the Twenty-Second Conference in Uncertainty in Artificial Intelligence (UAI'06)*. AUAI Press, 2006.