TU WIEN Informatics

Parametrisierte Algorithmen für das Bayesianisches Netzwerklernen

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Logic and Computation

eingereicht von

Viktoriia Korchemna
Matrikelnummer 11931227

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Robert Ganian

Wien, 3. Juli 2021

Viktoriia Korchemna	Robert Ganian

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Parameterized Algorithms for Bayesian Network Learning

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Logic and Computation

by

Viktoriia Korchemna
Registration Number 11931227

to the Faculty of Informatics

at TU Wien

Advisor: Robert Ganian

Vienna, 3rd July, 2021

Viktoriia Korchemna	Robert Ganian

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Viktoriia Korchemna

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 3. Juli 2021

Viktoriia Korchemna

Acknowledgements

I owe a deep sense of gratitude to my teacher and advisor Robert Ganian. Last year I got carried away by his highly interactive lectures on parameterized complexity and since this spring we have been working in a research project. Robert picks up really challenging and relevant problems. He gives me a full freedom and encourages creative approach. We often start just by sharing ideas and finally come up with an elegant solutions.

It is a pleasure for me to express the highest appreciation to my lyceum teacher Svitlana Kushnir. I believe that she is the one who had awakened a curiosity for mathematics and logic within me. Besides turning our everyday lessons into bright and exciting events, she held the facultative program. There we were walked through beautiful and non-standard tools and had a chance to try our hands at solving olympiad problems.

Concerning informatics and algorithms, I had two great teachers in my life. The first one is my father Serhiy Ageyev, who dedicated his life to computer programming and architecture. Designing algorithms and gluing them into huge projects is rather an art then a job for him. Thanks to my father I learned to see the beauty in building up "alive" systems from simple and clear components. My second teacher, Ernest Volkov, provided me with a rich algorithmic toolbox. In particular he showed us various dynamic programming techniques and algorithms on graphs and always filled us with a competitive aspiration to invent ours.

I would like to say a special thank you to my teacher and bachelor thesis advisor Andriy Oliynyk, who introduced me into computability and complexity theory. His positive mood and friendly attitude towards the students made us feel deeply involved into the lectures. Instead of presenting complete proofs and solutions, he often improvised, asking for our guesses and ideas.

Last but not least, I am grateful to my friend Denys Chergykalo. He has opened the world of artificial intelligence for me. Although he has just finished his bachelor, he already performs an extensive research in the area. I was especially inspired by his intuitive explanations regarding neural networks.

Abstract

We investigate the parameterized complexity of Bayesian Network Structure Learning (BNSL), a classical problem that has received significant attention in empirical but also purely theoretical studies. We follow up on previous works that have analyzed the complexity of BNSL w.r.t. the so-called *superstructure* of the input. While known results imply that BNSL is unlikely to be fixed-parameter tractable even when parameterized by the size of a vertex cover in the superstructure, here we show that a different kind of parameterization—notably by the size of a feedback edge set—yields fixed-parameter tractability. We proceed by showing that this result can be strengthened to a localized version of the feedback edge set. We adapt corresponding algorithms to the closely related problem of Polytree Learning. Concerning the lower bounds, we establish W[1]-hardness of BNSL parameterized by tree-cut width.

We then analyze how the complexity of BNSL depends on the representation of the input. In particular, while the bulk of past theoretical work on the topic assumed the use of the so-called *non-zero representation*, here we prove that if an *additive representation* can be used instead then BNSL becomes fixed-parameter tractable even under significantly milder restrictions to the superstructure, notably when parameterized by the treewidth alone.

Contents

CHAPTER 1

Introduction

The key feature of Bayesian networks is that they represent conditional dependencies between random variables via a directed acyclic graph; the vertices of this graph are the variables, and an arc ab means that the distribution of variable b depends on the value of a. One beneficial property of Bayesian networks is that they can be used to infer the distribution of dependent random variables in the network based on the values of the remaining variables. An example of a Bayesian network is depicted at Figure 1.1.

The problem of constructing a Bayesian network with an optimal network structure is NP-hard, and remains NP-hard even on highly restricted instances [Chi96]. This initial negative result has prompted an extensive investigation of the problem's complexity, with the aim of identifying new tractable fragments as well as the boundaries of its intractability [KP13, OS13, KP15, GK20, EG08, Das99, GKL$^+$15]. The problem—which we simply call Bayesian Network Structure Learning (BNSL)—can be stated as follows: given a set of V of variables (represented as vertices), a family $\mathcal{F}$ of *score functions* which assign each variable $v \in V$ a score based on its *parents*, and a target value ℓ, determine if there exists a directed acyclic graph over V that achieves a total score of at least ℓ.

To obtain a more refined understanding of the complexity of BNSL, past works have analyzed the problem not only in terms of classical complexity but also from the perspective of *parameterized complexity* [DF13, CFK$^+$15]. In parameterized complexity analysis, the tractability of problems is measured with respect to the input size n and additionally with respect to a specified numerical *parameter* k. In particular, a problem that is NP-hard in the classical sense may—depending on the parameterization used—be *fixed-parameter tractable* (FPT), which is the parameterized analogue of polynomial-time tractability and means that a solution can be found in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f, or W[1]-*hard*, which rules out fixed-parameter tractability under standard complexity assumptions. The use of parameterized complexity as a refinement of classical complexity is becoming increasingly common and has been employed not only for

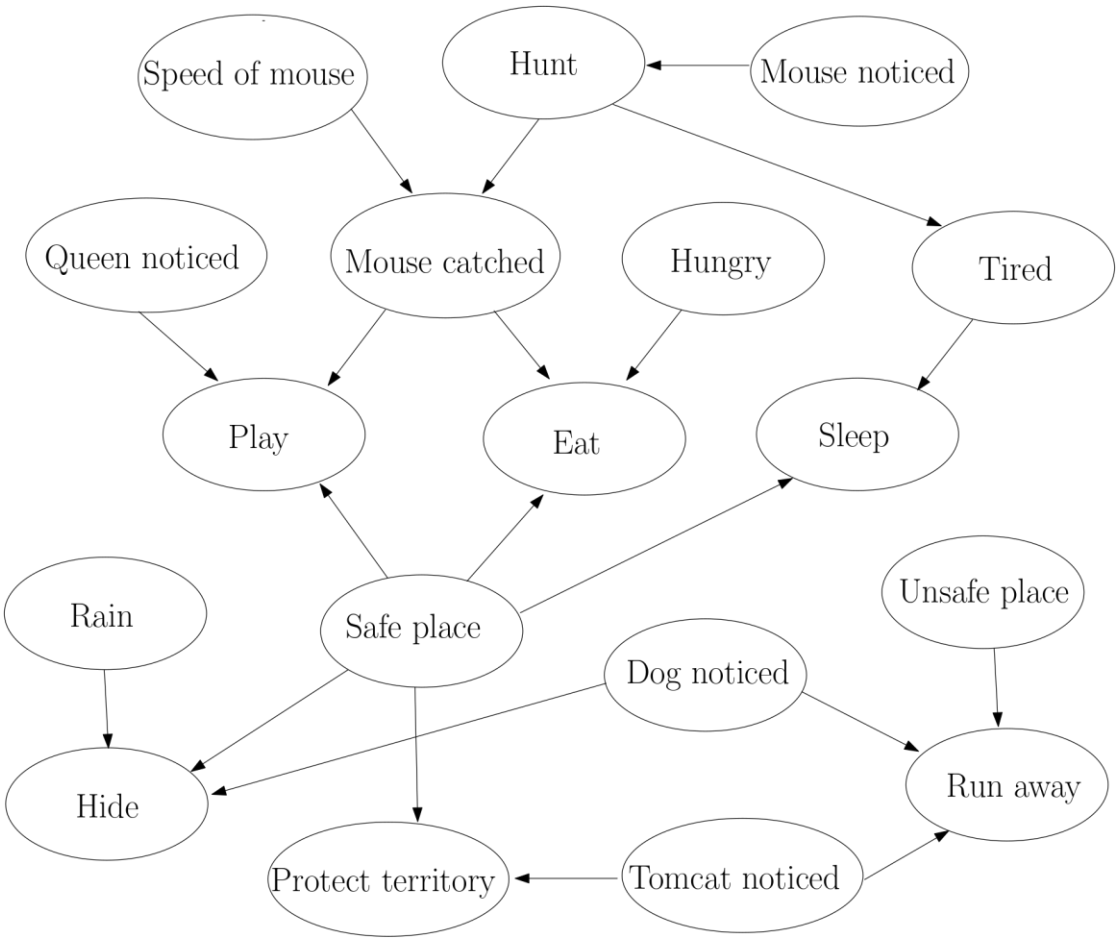

Figure 1.1: One of possible Bayesian networks representing the dependencies between environment, events and tomcat's behaviour. Sometimes an action can not be fully determined by a single factor. If another tomcat is noticed in a safe place, it leads to protection of the territory. However, an encounter in an unsafe location would likely result in the tomcat fleeing instead.

BNSL [KP13, OS13, KP15], but also for numerous other problems arising in the context of neural networks and artificial intelligence [GKOS18, SFGP19, EGKS19, GO18].

In Chapter 3 we provide an overview of known results concerning parameterized complexity of BNSL. We supplement some of them by proof sketches and mention general structures of the algorithms.

While analysing the existing literature, we noticed that today two main approaches can be distinguished. The first one is to impose restrictions on the constructed graph. However, most attempts to use such restrictions do not lead to FPT algorithms. For example, BNSL remains NP-hard if we require to construct a directed path instead of arbitrary acyclic digraph, as Meek proves in [Mee01]. The complexity of BNSL parameterized by vertex cover number (vcn) of the resulting network was studied as well. Although vcn is a highly restrictive parameter and means in fact that the skeleton of obtained digraph can be made edgeless by deleting bounded number of vertices, the problem is W[1]-hard, as was shown by Korhonen and Parviainen in [KP15]. In contrast, BNSL becomes FPT if parameterized by maximal number of arcs in the resulting network [GK20].

Another common restriction is to learn not an arbitrary acyclic digraph, but a polytree, i.e. a digraph with underlying graph being a forest. Such a variant of Bayesian network learning has been extensively studied recently an is now often referred to as Polytree Learning (PL). Among the novel results concerning the problem, we briefly mention [GKM21]. In the article PL is shown to be FPT when parameterized by the number of dependent vertices (vertices that may have in-neighbours in the resulting digraph) and maximal size of parent sets yielding a non-zero score for a vertex. However, parameterization by the number of dependent vertices alone leads to W[1]-hardness.

Sometimes restrictions are imposed not on the resulting acyclic digraph, but on its moralised graph, i.e. graph consisting of its skeleton and in addition edges between any two vertices with common out-neighbours. Grüttemeier and Komusiewicz provide in [GK20] a detailed complexity analysis for the cases when the moralised graph belongs to a particular sparsity class or can be done a member of such a class by deleting bounded number of vertices or edges. Again, in most of cases the problem remains hard. For example, BNSL is W[1]-hard if parameterized by minimal number of edges to be deleted to make the moralized graph acyclic. Moreover, it remains NP-hard if the moralised graph of the resulting DAG is required to have maximal degree 2, as well as if it must have connected components of size at most 3.

A second approach is to impose restrictions on the superstructure graph instead of the potentially learned network. For instance, in [OS13] Ordyniak and Szeider studied the complexity of BNSL parameterized by treewidth and/or maximal degree of the superstructure. It turns out that problem is W[1]-hard when parameterized by treewidth only, and in fact the superstructure graph provided in their reduction has bounded vertex cover number. But the parameterization by treewidth and maximal degree leads to an FPT algorithm that proceeds via leaf-to-root dynamic programming along the tree-decomposition.

Motivated by the last results, we were looking for a single simple and natural restriction on the superstructure graph leading to tractability. Hardness for the vertex cover number established in [OS13] together with FPT algorithm for bounded number of arcs provided in [GK20] encouraged us to concentrate on edge-cut based parameters.

In Section 4.1 of this article, we show that BNSL is FPT when parameterized by the feedback edge set number of the superstructure, i.e. minimal number of edges to make the graph acyclic. In fact, we present a poly-time procedure that allows to shrink the original instance and obtain an equivalent instance with number of vertices linear in feedback edge set number. In the next Section 4.2 we adapt the reduction procedure for PL. In Chapter 5 we lift the FPT results to the localized version of feedback edge number by providing efficient dynamic programming algorithms. We complete the analysis of edge-cut based parameters by establishing the lower bounds in Chapter 6. Our reduction strengthens the result from [OS13] and, in particular, implies that BNSL is W[1]-hard when parameterized by the tree-cut width.

Finally, in Chapter 7 we study an alternative input representation of BNSL based on assumption that the scoring functions decompose on scores of single arcs. In contrast to hardness results provided in [OS13], this variant of the problem turns out to be FPT when parameterized by the treewidth of the superstructure alone. We establish the tractability by presenting the dynamic programming algorithm on the tree-decomposition.

CHAPTER 2

Preliminaries

For an integer i, we let $[i] = \{1, 2, \ldots, i\}$ and $[i]_0 = [i] \cup \{0\}$. We denote by $\mathbb{N}$ the set of natural numbers, by $\mathbb{N}_0$ the set $\mathbb{N} \cup \{0\}$. For a set S, we denote by 2^S the set of all subsets of S.

2.1 Binary Relations

Let X be a set, then every $R \subseteq X \times X$ is called a *binary relation on* X. The binary relation R is termed:

- *symmetric* if $(x, y) \in R$ whenever $(y, x) \in R$ for every $x, y \in X$;
- *reflexive* if $(x, x) \in R$ for every $x \in X$;
- *transitive* if $(x, z) \in R$ whenever $(x, y) \in R$ and $(y, z) \in R$ for every $x, y, z \in X$;

A transitive relation R' is called *a transitive closure* of R (denote $R' = \texttt{trcl}(R)$) if it is a minimal transitive relation containing R, i.e. if there is no transitive relation R'' such that $R \subseteq R'' \subsetneq R'$. If R is reflexive, symmetric and transitive, it is called an *equivalence relation*. Given an equivalence relation R on X, we denote by $[x]_R = [x] = \{y | (x, y) \in R\}$ the *equivalence class* of x in X.

2.2 Graphs and Digraphs

We refer to the handbook by Diestel [Die12] for standard graph terminology. *Undirected graph* G is a pair (V, E) where V is a *vertex set* of G and E is some set of two-element subsets of V (*edge set* of G). *Directed graph*, or *digraph*, is a pair $D = (V, A)$ where V is a *vertex set* of D and $A \subseteq V \times V$ is an *arc set* of D. For a given undirected graph G we refer to its vertex and edge sets as $V(G)$ and $E(G)$ correspondingly. Similarly we denote

by $V(D)$ and $A(D)$ the sets of vertices and arcs of a digraph D. We will consider directed as well as undirected graphs. If another is not specified, by graphs we mean undirected graphs. If $G = (V, E)$ is a graph and $\{v, w\} \in E$, we will often use vw as shorthand for $\{v, w\}$. Moreover, we let $N_G(v)$ denote the set of *neighbors* of v, i.e., $\{\, u \in V \mid vu \in E \,\}$. We extend this notation to sets as follows: $N_G(X) = \{\, u \in V \setminus X \mid \exists x \in X : ux \in E(G)$.

If $D = (V, A)$ is a directed graph and $(v, w) \in A$, we will similarly use vw as shorthand for (v, w) and say that an arc vw *starts* in v and *finishes* in w. We also let $P_D(v)$ denote the set of *parents* of v, i.e., $\{\, u \in V \mid uv \in A \,\}$ (they are sometimes called *in-neighbors* in the literature). In both cases, we may drop G or D from the subscript if the (di)graph is clear from the context. The *degree* of v is $|N(v)|$, and for digraphs we use the notions of *in-degree* (which is equal to $|P(v)|$) and *out-degree* (the number of arcs originating from the given vertex).

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two (di)graphs, then we define their *union* as $G = (V_1 \cup V_2, E_1 \cup E_2)$. If $V_1 \cap V_2 = E_1 \cap E_2 = \emptyset$, G is a *disjoint* union of G_1 and G_2. The (disjoint) union of more then two graphs is defined analogously.

For a given (di)graph $G = (V, E)$ we say that $G' = (V', E')$ is its subgraph if $V' \subseteq V$ and $E' \subseteq E$. Let G be a graph G and V' be a subset of V, we define the *induced subgraph of* G *on* V' to be $G[V'] := (V', E')$ where $E' = \{uv \in E | u \in V', v \in V'\}$, i.e. $G[V']$ contains precisely those edges of G which have both endpoints in V'. Similarly for a digraph $D = (V, A)$ and $V' \subseteq V$ we denote by $D[V'] := (V', A \cap (V' \times V'))$ its *induced subgraph* on V'. For convenience we may omit the parenthesis and write $G[x_0, \ldots, x_k]$ instead of $G[\{x_0, \ldots, x_k\}]$. The *skeleton* (sometimes called the *underlying undirected graph*) of a digraph $G = (V, A)$ is the undirected graph $G' = (V, E)$ such that $vw \in E$ if and only if $vw \in A$ or $wv \in A$.

A *path* (*directed path* correspondingly) in a graph (digraph) G is a non-empty subgraph $P = (V, E)$ of G of the form $V = \{x_0, \ldots, x_k\}$, $E = \{x_0x_1, \ldots, x_{k-1}x_k\}$, where the x_i are distinct. We call P a path *from* x_0 *to* x_k and denote it by $P = x_0 \ldots x_k$. In this case x_0 and x_k are *connected by* P in G. For convenience we say that x_0 and x_k are connected by an undirected path P in a digraph G if they are connected by P in the skeleton of G. If in addition x_kx_0 belongs to an edge (arc) set of G, $x_0 \ldots x_kx_0$ is called a *cycle* (*directed cycle*) in G. An undirected graph is called *a forest* if it is *acyclic*, i.e. doesn't contain cycles.

If every two vertices of a graph (digraph) are connected by an undirected path, G is a *connected* graph (digraph). Note that every (di)graph G is a disjoint union of connected graphs, which are called the *connected components* of G. Connected forest is called *a tree*, in general, any forest is a disjoint union of trees. A subgraph T of G is called a *spanning tree* if T is a tree and $V(T) = V(G)$. A digraph is a *polytree* if its skeleton is a forest.

We use a shorthand DAG for acyclic digraph. For a set X of vertices, let A_X denote the set of all possible arcs over X.

2.3 Problem Statement

The general task is, given the set V of random variables and the set of observations, to construct a DAG on V (depicting dependencies by arcs) that maximizes posterior probability to get the original set of observations. In some cases it is equivalent to maximizing the sum of local scores of the variables, determined by sets of their in-neighbours in the constructed DAG. In practice, local scores can be computed from the set of observations. In our case, they are given as a part of an input. Formally, the problem is stated as follows:

Bayesian Network Structure Learning (BNSL)	
Input:	A set V of vertices, a family $\mathcal{F} = \{f_v : 2^{V\setminus\{v\}} \to \mathbb{N}_0\}$ of local score functions, and an integer ℓ.
Question:	Does there exist a DAG $D = (V, A)$ such that $f(D) := \sum_{v\in V} f_v(P_v) \geq \ell$, where P_v is the set of in-neighbours of v in D?

Another closely related problem which has been recently studied is Polytree Learning (PL). It is stated similarly to BNSL with an additional requirement that the learned network must form a polytree:

Polytree Learning (PL)	
Input:	A set V of vertices, a family $\mathcal{F} = \{f_v : 2^{V\setminus\{v\}} \to \mathbb{N}_0\}$ of local score functions, and an integer ℓ.
Question:	Does there exist a polytree $D = (V, A)$ such that $f(D) := \sum_{v\in V} f_v(P_v) \geq \ell$, where P_v is the set of in-neighbours of v in D?

There are different ways to encode f_v in the input. If we explicitly specify $f_v(P)$ for every $P \subseteq V \setminus \{v\}$, total size of the input automatically becomes exponential in the number of vertices. However, the complexity of BNSL which is commonly studied is based on number of vertices and sets of their in-neighbours yielding non-zero local scores. For this reason we assume that $\mathcal{F}$ is given as a set of all tuples $(v, P, f_v(P))$ where $f_v(P) > 0$. We refer to this variant of a problem as BNSL in a *non-zero representation*, or BNSL$^{\neq 0}$. This model has been used in a large number of works studying the complexity of BNSL and PL [KP13, OS13, KP15, GK20, GKL$^+$15, GKM21] and is known to be strictly more general than, e.g., the bounded-arity representation where one only considers parent sets of arity bounded by a constant [OS13, Section 3].

Let $\Gamma_f(v)$ be the set of candidate parents of v which yield a non-zero score; formally, $\Gamma_f(v) = \{\, Z \mid f_v(Z) \neq 0 \,\}$, and the input size $|\mathcal{I}|$ of an instance $\mathcal{I} = (V, \mathcal{F}, \ell)$ is simply defined as $|V| + \ell + \sum_{v\in V, P\in\Gamma_f(v)} |P|$. Let $P_{\to}(v)$ be the set of all parents which appear in $\Gamma_f(v)$, i.e., $a \in P_{\to}(v)$ if and only if $\exists Z \in \Gamma_f(v) : a \in Z$. A natural way to think about and exploit the structure of inter-variable dependencies laid bare by the non-zero

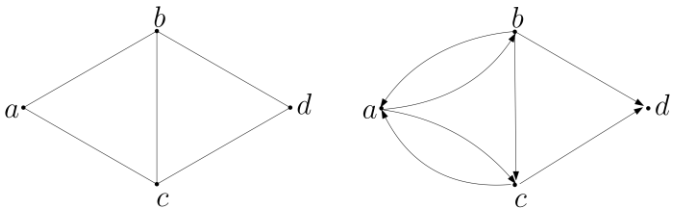

Figure 2.1: Example of a superstructure graph (on the left) and directed superstructure graph (on the right) when:
$\Gamma_f(a)$ consists of $\{b\}, \{c\}$, and $\{b, c\}$
$\Gamma_f(b)$ contains only $\{a\}$
$\Gamma_f(c)$ consists of $\{b\}$ and $\{a, b\}$
$\Gamma_f(d)$ contains only $\{b, c\}$

representation is to consider the *superstructure graph* $G_{\mathcal{I}} = (V, E)$ of a BNSL (or PL) instance $\mathcal{I} = (V, \mathcal{F}, \ell)$, where $ab \in E$ if and only if either $a \in P_{\to}(b)$, or $b \in P_{\to}(a)$, or both. A potentially more precise way to determine possible structure of constructed digraphs is to consider a *directed superstructure graph* $\bar{G}_{\mathcal{I}} = (V, E)$ where $ab \in E$ if and only if $a \in P_{\to}(b)$. See Figure 2.1 for an illustration.

Naturally, families of local score functions may be exponentially larger than $|V|$ even when stored using the non-zero representation. In this paper, we also consider a second representation of $\mathcal{F}$ which is guaranteed to be polynomial in $|V|$: in the *additive representation*, we require that for every vertex $v \in V$ and set $Q = \{q_1, \ldots, q_m\} \subseteq V \setminus \{v\}$, $f_v(Q) = f_v(\{q_1\}) + \cdots + f_v(\{q_m\})$. Hence, each score function f_v can be fully characterized by storing at most $|V|$-many entries of the form $f_v(x) := f_v(\{x\})$ for each $x \in V \setminus \{v\}$. To avoid overfitting, one may optionally impose an additional constraint: an upper bound q on the size of any parent set in the solution (or, equivalently, q is a maximum upper-bound on the in-degree of the sought-after acyclic digraph D).

While not every family of local score functions admits an additive representation, the additive model is similar in spirit to the models used by some practical algorithms for BNSL. For instance, the algorithms of Scanagatta, de Campos, Corani and Zaffalon [SdCCZ15, SCdCZ16], which can process BNSL instances with up to thousands of variables, approximate the real score functions by adding up the known score functions for two parts of the parent set and applying a small, logarithmic correction. Both of these algorithms also use the aforementioned bound q for the parent set size. In spite of this connection to practice and the representation's streamlined nature, we are not aware of any prior works that considered the additive representation in complexity-theoretic studies of BNSL and PL.

As before, in the additive representation we will also only store scores for parents of v which yield a non-zero score, and can thus define $P_{\to}(v) = \{\, z \mid f_v(z) \neq 0 \,\}$, as for the non-zero representation. This in turn allows us to define the superstructure graphs as before: $G_{\mathcal{I}} = (V, E)$ where $ab \in E$ if and only if $a \in P_{\to}(b)$, $b \in P_{\to}(a)$, or both.

To distinguish between these models, we use BNSL$^{\neq 0}$, BNSL^{+}, and BNSL$^{+}_{\leq}$ to denote Bayesian Network Structure Learning with the non-zero representation, the

additive representation, and the additive representation and the parent set size bound q, respectively. The same notation will also be used for Polytree Learning—for example, an instance of $\mathrm{PL}^{+}_{\leq}$ will consist of V, a family $\mathcal{F}$ of local score functions in the additive representation, and integers ℓ, q, and the question is whether there exists a polytree $D = (V, A)$ with in-degree at most q and $\texttt{score}(D) \geq \ell$.

2.4 Parameterized Complexity

In parameterized algorithmics [CFK+15, DF13, Nie06] the running-time of an algorithm is studied with respect to a parameter $k \in \mathbb{N}_0$ and input size n. The basic idea is to find a parameter that describes the structure of the instance such that the combinatorial explosion can be confined to this parameter. In this respect, the most favorable complexity class is FPT (*fixed-parameter tractable*) which contains all problems that can be decided by an algorithm running in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function. Algorithms with this running-time are called *fixed-parameter algorithms* or just FPT *algorithms*. A less favorable outcome is an XP *algorithm*, which is an algorithm running in time $\mathcal{O}(n^{f(k)})$; problems admitting such algorithms belong to the class XP.

Another important complexity class is W[1]. We will not provide a proper definition of the class here, as it is quiet involved and is not needed to establish and prove our results. For a general impression, let us mention that W[1] contains XP and showing W[1]-hardness of a problem rules out the existence of a fixed-parameter algorithm under the well-established assumption that W[1] $\neq$ FPT. This is usually done via a *parameterized reduction* [CFK+15, DF13] to some known W[1]-hard problem. A parameterized reduction from a parameterized problem $\mathcal{P}$ to a parameterized problem $\mathcal{Q}$ is a function:

- which maps Yes-instances to Yes-instances and No-instances to No-instances,
- which can be computed in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function, and
- where the parameter of the output instance can be upper-bounded by some function of the parameter of the input instance.

2.5 Numerical Parameters

When comparing two numerical parameters α, β of graphs, we say that α is more *restrictive* than β if there exists a function f such that $\beta(G) \leq f(\alpha(G))$ holds for every graph G. In other words, α is more restrictive than β if and only if the following holds: whenever all graphs in some graph class $\mathcal{H}$ have α upper-bounded by a constant, all graphs in $\mathcal{H}$ also have β upper-bounded by a constant. Observe that in this case a fixed-parameter algorithm parameterized by β immediately implies a fixed-parameter algorithm parameterized by α, while W[1]-hardness behaves in the opposite way.

Among the most common graph parameters we will concentrate on are those which, roughly speaking, measure how far the graph is from a forest. One highly restrictive

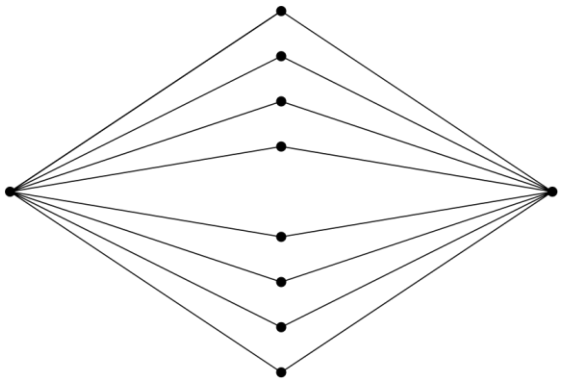

Figure 2.2: Example of a graph with small vertex cover number but large feedback edge number. Left and right vertices form the vertex cover of size 2.

parameter is *feedback edge number* (fen), specifying minimal number of edges to be deleted to make the graph acyclic. For example, forests have fen of zero while for a cycle it is equal to one. Another parameter is *vertex cover number* (vcn), it shows minimal size of subset of vertices which contains at least one endpoint of each edge. Note that these two parameters are incomparable in general. Indeed, for a long path fen is obviously equal to zero, but vcn is linear in the path length. From another side, graph depicted on the Figure 2.2 has vertex cover of size two but fen linear in number of edges. Another parameter measuring the distance to acyclic graph is *treewidth*. We will describe it in more details further, but now mention that it is less restrictive then both vcn and fen.

2.6 Treewidth

A *tree-decomposition* $\mathcal{T}$ of a graph $G = (V, E)$ is a pair (T, χ), where T is a tree (whose vertices we call *nodes*) rooted at a node r and χ is a function that assigns each node t a set $\chi(t) \subseteq V$ such that the following holds:

- For every $uv \in E$ there is a node t such that $u, v \in \chi(t)$.
- For every vertex $v \in V$, the set of nodes t satisfying $v \in \chi(t)$ forms a subtree of T.
- $|\chi(\ell)| = 1$ for every leaf ℓ of T and $|\chi(r)| = 0$.

The tree-decomposition $\mathcal{T}$ is called *nice* if there are only 3 kinds of non-leaf nodes in T:

- **Introduce node:** a node t with exactly one child t' such that $\chi(t) = \chi(t') \cup \{v\}$ for some vertex $v \notin \chi(t')$.
- **Forget node:** a node t with exactly one child t' such that $\chi(t) = \chi(t') \setminus \{v\}$ for some vertex $v \in \chi(t')$.
- **Join node:** a node t with two children t_1, t_2 such that $\chi(t) = \chi(t_1) = \chi(t_2)$.

The *width* of a nice tree-decomposition (T, χ) is the size of a largest set $\chi(t)$ minus 1, and the *treewidth* of the graph G, denoted $\mathrm{tw}(G)$, is the minimum width of a nice

tree-decomposition of G. Fixed-parameter algorithms are known for computing a nice tree-decomposition of optimal width:

Fact 1 ([BDD+16, Klo94])**.** *There exists an algorithm which, given an n-vertex graph G and an integer k, in time $2^{\mathcal{O}(k)} \cdot n$ either outputs a nice tree-decomposition of G of width at most $5k+4$ and $\mathcal{O}(n)$ nodes, or determines that* $\mathrm{tw}(G) > k$.

For $t \in V(T)$, let T_t be the subtree of T rooted at t. We denote by F_t the set of vertices *forgotten in* t, i.e. the vertices in T_t that were forgotten in some node of T_t:

$$F_t = \{v |\ \chi(t') = \chi(t'') \setminus \{v\},\ t' \in T_t,\ t'' \text{ is a child of } t'\}.$$

2.7 Local Feedback Edge Set

Up to now, these were the only two edge-cut based graph parameters that have been considered in the broader context of algorithm design: feedback edge number and *tree-cut width* (see [GKO21, Subsection 2.4] for the definition). Here, we propose a new parameter that can be seen as a localized relaxation of the feedback edge number: instead of measuring the total size of the feedback edge set, it only measures how many feedback edges can "locally interfere with" any particular part of the graph.

Formally, for a connected graph $G = (V, E)$ and a spanning tree T of G, let the *local feedback edge set* at $v \in V$ be

$$E^T_{\mathrm{loc}}(v) = \{uw \in E \setminus E(T) \mid \text{ the unique path between } u \text{ and } w \text{ in } T \text{ contains } v\}.$$

The *local feedback edge number of* the pair (G, T) (denoted $\mathrm{lfen}(G, T)$) is then equal to $\max_{v \in V} |E^T_{\mathrm{loc}}(v)|$, and the *local feedback edge number of* G is simply the smallest local feedback edge number among all possible spanning trees of G, i.e.,

$$\mathrm{lfen}(G) = \min\{\mathrm{lfen}(G, T) | T \text{ is a spanning tree of } G\}.$$

CHAPTER 3

Related Work

In this chapter we present an overview of latest complexity results for BNSL. The survey is not exhaustive, but it captures most of basic graph parameters, motivates our work and hopefully displays an overall picture of state of the art. As we have already mentioned in the introduction, the most common approach is to impose restrictions on the structural properties of either the superstructure graph or the learned network.

3.1 Exploiting the Superstructure: Treewidth, Maximal Degree and Acyclicity

In [OS13], Ordyniak and Szeider study the parameterized complexity of BNSL by imposing restrictions on the structural properties of the superstructure graph. This approach allows to reduce the size of a search space: one may only consider acyclic digraphs whose skeleton is a subgraph of a superstructure graph. Indeed, any acyclic directed graph can be transformed to satisfy the property without decreasing the score. Moreover, by the same arguments we may assume that every vertex either receives a parent set of non-zero score or doesn't have in-neighbours at all. Similarly, in case of directed superstructure graph it is sufficient to consider only its subgraphs as potential digraphs of optimal score.

One of the main results obtained in [OS13] is an FPT algorithm for parameterization by treewidth ω of the superstructure and maximal number of candidate parent sets per vertex. Formally, let $\mathcal{I}$ be the instance of BNSL, we denote $\delta_f = \max_{v \in V} |\Gamma_f(v)|$. Then the following holds [OS13, Theorem 1] :

Theorem 2. *Given an instance $\mathcal{I} = (V, \mathcal{F}, \ell)$ of BNSL such that $\mathrm{tw}(G_{\mathcal{I}}) \leq \omega$, a DAG D on V of maximal score $f(D)$ can be found in time $2^{\mathcal{O}(\omega^2)} \cdot \delta_f^{\mathcal{O}(\omega)} \cdot |V|$*

Let us sketch the main proof ideas. We start from computing a nice tree-decomposition (T, χ) of $G_{\mathcal{I}}$ of nearly optimal treewidth, which can be done efficiently according to Fact 1. Then, moving in T from leaves to the root, we construct an acyclic digraph of the maximal score. For every node $t \in T$, let $\chi_t^{\downarrow}$ be the set of all vertices occurring in bags of the rooted subtree T_t, i.e., $\chi_t^{\downarrow} = \{u \mid \exists t' \in T_t \text{ such that } u \in \chi(t')\}$. Let $G_t^{\downarrow}$ be the subgraph of $G_{\mathcal{I}}$ induced on $\chi_t^{\downarrow}$. Intuitively, we guess a restriction of a solution DAG to the vertex set $\chi_t^{\downarrow}$ along with parent sets of vertices in $\chi(t)$ in the solution. Every such a partial DAG D_t in node t can be described by a record storing the following information:

- parent sets of vertices in $\chi(t)$;
- connectivity relation on $\chi(t)$ specifying for every $x, y \in \chi(t)$ whether D_t contains a directed path from x to y.
- sum of scores of vertices in D_t that are forgotten in t.

Obviously, it is sufficient to keep only records with maximal sums of scores when the parent sets and connectivity relation are fixed. Note that the number of possible records in t is $2^{\mathcal{O}(\omega^2)} \cdot \delta_f^{\mathcal{O}(\omega)}$. Indeed, $\chi(t)$ contains $\mathcal{O}(\omega)$ vertices, resulting in $2^{\mathcal{O}(\omega^2)}$ potential connectivity relations. Further, every vertex in $\chi(t)$ has at most δ_f candidates for the parent set. To prove the theorem, a leaf-to-root dynamic programming algorithm is designed. It computes and stores a set of records at each node of T, whereas once we ascertain the records for r we will have the information required to output a correct answer. Intuitively:

- **Introduce node:** let t be a node with exactly one child t' such that $\chi(t) = \chi(t') \cup \{v\}$ for some vertex $v \notin \chi(t')$. Then we branch over all records R in t' and parent sets P of v. If the combination of R and arcs induced by P doesn't lead to a directed cycle, we add a new record for t. The record is constructed from R by adding v with parent set P and updating the connectivity relation with new arcs incident to v.
- **Forget node:** let t be a node with exactly one child t' such that $\chi(t) = \chi(t') \setminus \{v\}$ for some vertex $v \in \chi(t')$. Then we can construct records in t from the ones in t' by simply omitting the parent set P of v, restricting the connectivity relations to the remaining vertices and adding $f_v(P)$ to the score of forgotten vertices.
- **Join node:** let t be a node with two children t_1, t_2 such that $\chi(t) = \chi(t_1) = \chi(t_2)$. Then we branch over all possible pairs of records R_1 and R_2 in t_1 and t_2 correspondingly that agree in the parent set choice for every $v \in \chi(t)$. If combination of their connectivity relations doesn't yield a directed cycle, we add a new record R with the score of forgotten vertices equal to the sum of scores in R_1 and R_2. Note that according to the definition of a tree-decomposition $F_{t_1} \cap F_{t_2} = \emptyset$, so we count every forgotten vertex precisely once.

As we assume the root to have an empty bag, all the vertices are forgotten in the root and therefore sum of their scores in the unique record of the root is precisely the maximal score achievable by DAGs in $\mathcal{I}$. Simple back-tracking along the tree-decomposition now allows to reconstruct a DAG of this score.

Observe that the running time $2^{\mathcal{O}(\omega^2)} \cdot \delta_f^{\mathcal{O}(\omega)} \cdot |V|$ actually shows an XP membership of BNSL parameterized by the treewidth of the superstructure alone. The natural question which arises is whether the problem is FPT if parameterized only by ω. However, the following result [OS13, Theorem 3] rules out this possibility, under standard complexity-theoretical assumptions:

Theorem 3. *BNSL parameterized by the treewidth of the superstructure is* W[1]*-hard.*

Authors prove the statement by parameterized reduction from the following well-known W[1]-hard problem [DF13, CFK+15]:

Multicolored Clique (MCC)	
Input:	A k-partite graph $G = (V_1 \cup ... \cup V_k, E)$
Parameter:	The integer k
Question:	Are there nodes v^i that form a k-colored clique in G, i.e. $v^i \in V_i$ and $v^i v^j \in E$ for all $i, j \in [k], i \neq j$?

Given an instance $G = (V_1 \cup ... \cup V_k, E)$ of MCC, it is possible to construct in FPT time the instance $\mathcal{I}$ of BNSL satisfying the following properties:

- V consists of nodes n_v, encoding vertices $v \in V_1 \cup ... \cup V_k$ and nodes a_{ij} for every $i, j \in [k]$, $i \neq j$, encoding edges between V_i and V_j
- a_{ij} receives a non-zero score only if its parent set consists of two vertices—$v_i \in V_i$ and $v_j \in V_j$—that are connected by an edge in G
- node n_{v_i} for every $v_i \in V_i$ receives a non-zero score only if it takes into a parent set all a_{lk} with $l = i$ or $k = i$
- the score of ℓ can be achieved only if every a_{ij} achieves its maximal score and all but at most k vertices n_v achieve their maximal scores

Intuitively, in case of Yes-instance the vertices n_v that receive a score of zero will be taken into parent sets of a_{ij} and determine the k-colored clique.

Note that every edge in the superstructure graph is adjacent to some a_{ij}, so $G_{\mathcal{I}}$ has vcn of at most $\frac{k(k-1)}{2}$. In particular, it results in bounded treewidth.

The authors also point out in Theorem 2. that a reduction by Chickering [Chi96]

implies NP-hardness of BNSL even if every vertex in the superstructure graph has a degree of at most 4. Note that bounding the degree of the superstructure graph automatically bounds the maximum size of $\Gamma_f(v)$. Indeed, if $G_\mathcal{I}$ has a maximum degree of d then for every $v \in V$ it holds that $|\Gamma_f(v)| \leq 2^d$, as every $P \in \Gamma_f(v)$ is a subset of $N_{G_\mathcal{I}}(v)$. Therefore BNSL is not in FPT and even not in XP when parameterized only by δ_f : if this would be the case, setting δ_f equal to 2^4 would result in polynomial algorithm, which is impossible under the standard assumption P $\neq$ NP.

It would be natural to expect that restricting the structural properties of the directed superstructure graph makes the problem of learning the optimal network easier. Indeed, according to [OS13, Corollary 2], BNSL is solvable in quadratic time if $\bar{G}_\mathcal{I}$ does not contain directed cycles. To establish this, a greedy approach can be used: for every $v \in V$ we choose a parent set yielding the maximal score. Acyclicity of $\bar{G}_\mathcal{I}$ ensures that resulting digraph is acyclic. However, already a slight complication of the directed superstructure graph leads to NP-hardness, as [OS13, Theorem 8] shows:

Theorem 4. *Bayesian Network Structure Learning is* NP*-hard for instances* $\mathcal{I}$ *where* $\bar{G}_\mathcal{I}$ *can be made acyclic by deleting one node. Hardness even holds if we additionally bound the maximal in-degree and the maximal out-degree of* $\bar{G}_\mathcal{I}$ *by 3.*

3.2 Restrictions on the Constructed Network

3.2.1 Learning a Polytree.

Polytree Learning (PL) is a variant of BNSL where we require the resulting network to form a polytree. The complexity of PL has been studied in several works [GKM21, GKL+15, SMS13], we will briefly describe the results of this year's article [GKM21]. Let $\mathcal{I} = (V, \mathcal{F}, \ell)$ be an instance of PL. We call the vertex $v \in V$ *dependent* if there is at least one parent set for v yielding a non-zero score, i.e. $\Gamma_f(v) \neq \emptyset$. According to [GKM21, Theorem 4.2] Polytree Learning is W[1]-hard when parameterized by the number of dependent vertices. Original proof proceeds by reduction from the well known $W[1]$-hard problem:

Independent Set	
Input:	Graph $G = (V, E)$
Parameter:	The integer k
Question:	Is there subset $S \subseteq V$ of size at least k such that no two vertices of S are connected by edge in E?

Such a subset is called *independent set*. Given an instance $G = (V, E)$ of Independent Set, we construct an equivalent instance $(V', \mathcal{F}, \ell)$ of PL where:

- V' consists of vertices $v_1, \ldots, v_k, v^*$ and vertices w_e for each edge $e \in E$

- for every $i \in [k]$, $\Gamma_f(v_i) = \{E_v \cup \{v^*\} | v \in V\}$, where $E_v = \{w_{vu} | vu \in E\}$ and $f_{v_i}(E_v \cup \{v^*\}) = 1$, all the rest of scores are zero
- $\ell = k$

Note that the score of k is achieved if and only if every v_i receives some parent set with the score of 1. Intuitively, if G contains an independent set $S = \{s_i | i \in [k]\}$, we may choose the parent sets for v_i equal to the sets of edges ajacent to s_i. However, if S is not an independent set (some s_i and s_j are connected by edge), then the parent sets for v_i and v_j chosen as before would share at least two vertices: $w_{s_i s_j}$ and v^*, which is forbidden for a polytree.

However, the problem becomes tractable if we in addition parameterize by the maximal size of a parent set [GKM21, Theorem 5.8]:

Theorem 5. *Polytree Learning can be solved in time $2^{\mathcal{O}(kp)} \cdot |\mathcal{I}|^{\mathcal{O}(1)}$, where k is the number of dependent vertices and p is the maximum size of a parent set.*

3.2.2 Learning a Path.

Another, even more restrictive version of BNSL is the one where the resulting network is required to be a path (we will call it Path Learning). However, even this version of the problem is NP-hard, as was shown in [Mee01]. The hardness is proved via reduction from the well-known NP-complete problem:

Hamiltonian path	
Input:	Graph $G = (V, E)$
Question:	Is there a path in G visiting every $v \in V$ precisely one time?

Given an instance $G = (V, E)$ of Hamiltonian path, the author constructs an equivalent instance of Path Learning with the same vertex set. We keep the construction but slightly modify the scores from original proof for convenience. Let every vertex $v \in V$ receive the score of 1 for empty parent set and the score of 2 for any parent set $\{w\}$ where $w \in N_G(v)$. Further, the goal score ℓ is set equal to $1 + 2(|V| - 1)$. Obviously the score can be achieved by a directed path network D if and only if the path contains all the vertices $v \in V$ and all but one of them have precisely one in-neighbour in D corresponding to some neighbour in G. If we now start from the only vertex with empty parent set and move along the arcs of D, it will result in a hamiltonian path in G.

3.2.3 Sparse Moralized Graphs.

The *moralized* graph of a digraph D is an (undirected) graph consisting of the skeleton of D and, in addition, edges between any two vertices that share a child (an out-neighbour) in D. In [GK20] Grüttemeier and Komusiewicz provide a detailed complexity analysis for

the cases when the moralised graph of learned network belongs to a particular sparsity class or can be done a member of such a class by deleting bounded number of vertices or edges. More precisely, they study the following classes of problem:

$(\Pi + v)$-Bayesian Network Structure Learning	
Input:	A set of vertices V, local scores $\mathcal{F}$ and integer ℓ
Parameter:	k
Question:	Is there a DAG D on V with score $f(D) \geq \ell$ such that moralized graph of D can be transformed into a graph from a class Π by deleting at most k vertices?

The problem $(\Pi+e)$-Bayesian Network Structure Learning is defined analogously, in this case not vertices but edges can be deleted. Π here should be replaced by a particular sparsity class. For example, let us denote by Π_1 the set of graphs with maximum degree 1. Then $(\Pi_1 + v) - \text{BNSL}$ asks whether there exist a DAG such that deletion of at most k vertices from its moralized graph results in a graph of maximum degree 1. [GK20, Corollary 9] shows that the problem is in XP:

Theorem 6. *$(\Pi_1 + v) - BNSL$ can be solved in time $n^{\mathcal{O}(k^2)} + |\mathcal{I}|^{\mathcal{O}(1)}$.*

However, an FPT algorithm is unlikely to exists, even if we in additional parameterize by a goal score ℓ [GK20, Proposition 10]:

Theorem 7. *$(\Pi_1 + v) - BNSL$ is $W[1]$-hard for $k + \ell$, even if the directed superstructure is a DAG and the maximum parent set size is 3.*

3.2.4 Bounding the Number of Arcs.

Let us conclude this overview by an interesting FPT result concerning learning networks with small number of arcs. Formally, the problem is stated as follows:

Bounded-Arcs-BNSL	
Input:	A set of vertices V, local scores $\mathcal{F}$ and integer ℓ
Parameter:	k
Question:	Is there a DAG D on V with score $f(D) \geq \ell$ containing at most k arcs?

[GK20, Proposition 17] presents an FPT algorithm for the *colored* version of the problem. In Colored Bounded-Arcs-BNSL, every vertex $v \in V$ has a color in $C = \{1, \ldots, 2k\}$. For $C' \subseteq C$, let us denote by $V_{C'}$ the vertices from V which have colors in C'. The task is to compute a network consisting of at most k arcs such that (1) the endpoints of every arc have different colors (2) there is at most one vertex with non-empty parent set per color. The algorithm proceeds by dynamic programming; for each $C' \subseteq C$ and $k' \in [k]_0$ it computes the entry $T[C', k']$ equal to the maximal score achievable by DAGs on $V_{C'}$

with at most k' arcs, satisfying (1) and (2). Assume that the entries are computed for all color sets of cardinality at most m (for all $k' \in [k]_0$) and $C' \subseteq C$ has the cardinality of $m+1$. Intuitively, to compute $T[C', k']$, we guess three things:

- what color c does the sink have
- which vertex v of color c is a sink
- which is a parent set P of v

When c, v and P are guessed right, rest of vertices of color c have empty parent sets and therefore:
$T[C', k'] = T[C' \setminus \{c\}, k' - |P|] + f_v(P) + \sum_{w \neq v: \text{w has color c}} f_w(\emptyset)$. We refer to the book "Parameterized Algorithms" [CFK+15] for the description of *color coding* technique. This powerful tool in parameterized complexity in some cases allows to lift FPT results for a colored problem to the original one.

CHAPTER 4

Parameterization by Feedback Edge Number

4.1 Data Reduction for BNSL$^{\neq 0}$

We say that two instances $\mathcal{I}$, $\mathcal{I}'$ of BNSL are *equivalent* if (1) they are either both Yes-instances or both No-instances, and furthermore (2) a solution to one instance can be transformed into a solution to the other instance in polynomial time. Our aim here is to prove the following theorem:

Theorem 8. *There is an algorithm which takes as input an instance $\mathcal{I}$ of BNSL$^{\neq 0}$ whose superstructure has* fen k*, runs in time $\mathcal{O}(|\mathcal{I}|^2)$, and outputs an equivalent instance $\mathcal{I}' = (V', \mathcal{F}', \ell')$ of BNSL$^{\neq 0}$ such that $|V'| \leq 16k$.*

In parameterized complexity theory, such data reduction algorithms with performance guarantees are called *kernelization algorithms* [DF13, CFK$^+$15]. These may be applied as a polynomial-time preprocessing step before, e.g., more computationally expensive methods are used. The fixed-parameter tractability of BNSL$^{\neq 0}$when parameterized by the fen of the superstructure follows as an immediate corollary of Theorem 8 (one may solve $\mathcal{I}$ by, e.g., exhaustively looping over all possible DAGs on V' via a brute-force procedure). We also note that even though the number of variables of the output instance is polynomial in the parameter k, the instance $\mathcal{I}'$ need not have size polynomial in k.

We begin our path towards a proof of Theorem 8 by computing a feedback edge set E_F of G of size k in time $\mathcal{O}(|\mathcal{I}|^2)$ by, e.g., Prim's algorithm. Let T be the spanning tree of G, $E_F = E(G) \setminus E(T)$. The algorithm will proceed by the recursive application of certain reduction rules, which are polynomial-time operations that alter ("simplify") the input instance in a certain way. A reduction rule is *safe* if it outputs an instance which

is equivalent to the input instance. We start by describing a rule that will be used to prune T until all leaves are incident to at least one edge in E_F.

Reduction Rule 1. *Let $v \in V$ be a vertex and let Q be the set of neighbors of v with degree 1 in G. We construct a new instance $\mathcal{I}' = (V', \mathcal{F}', \ell)$ by setting:* **1.** *$V' := V \setminus Q$;* **2.** *$\Gamma_{f'}(v) := \{\emptyset\} \cup \{\, (P \setminus Q) \mid P \in \Gamma_f(v) \,\}$;* **3.** *for all $w \in V' \setminus \{v\}$, $f'_w = f_w$;* **4.** *for every $P' \in \Gamma_{f'}(v)$:*

$$f'_v(P') := \max_{P : P \setminus Q = P'} \Big(f_v(P) + \sum_{v_{\text{in}} \in P \cap Q} f_{v_{\text{in}}}(\emptyset) + \sum_{v_{\text{out}} \in Q \setminus P} \max(f_{v_{\text{out}}}(\emptyset), f_{v_{\text{out}}}(v)) \Big).$$

Lemma 9. *Reduction Rule 1 is safe.*

Proof. For the forward direction, assume that $\mathcal{I}'$ admits a solution D', and let λ be the score D' achieves on v. By the construction of $\mathcal{I}'$, there must be a parent set $Z \in \Gamma_f(v)$ such that $Z \cap V' = P_{D'}(v)$ (i.e., Z agrees with v's parents in D') and λ is the sum of the following scores: (1) $f_v(Z)$, (2) the maximum achievable score for each vertex in $Q \setminus Z$, and (3) the score of $\{\emptyset\}$ for each vertex in $Z \cap Q$. Let D be obtained from D' by adding the following arcs: zv for each $z \in Z$, and vq for each $q \in Q \setminus Z$ such that q achieves its maximum score with v as its parent. By construction, $\lambda = \sum_{w' \in \{v\} \cup Q} f_w(P_D(w))$. Since the scores of D and D' coincide on all vertices outside of $\{v\} \cup Q$ and D is acyclic, we conclude that $\texttt{score}(D) = \texttt{score}(D')$, and hence $\mathcal{I}$ is a Yes-instance.

For the converse direction, assume that $\mathcal{I}$ admits a solution D. Let $D' = D - Q$. By the construction of f'_v, it follows that $f'_v(P_{D'}(v))$ is greater or equal to the score D achieves on $\{v\} \cup Q$. Thus, D' is a solution to $\mathcal{I}'$, and we conclude that Reduction Rule 1 is safe. □

Observe that the superstructure graph G' obtained after applying one step of Reduction Rule 1 is simply $G - Q$; after its exhaustive application we obtain an instance $\mathcal{I}$ such that all the leaves of the tree T are endpoints of E_F. Our next step is to get rid of long paths in G whose internal vertices have degree 2. We note that this step is more complicated than in typical kernelization results using feedback edge set as the parameter, since a directed path Q in G can serve multiple "roles" in a hypothetical solution D and our reduction gadget needs to account for all of these. Intuitively, Q may or may not appear as a directed path in D (which impacts what other arcs can be used in D due to acyclicity), and in addition the total score achieved by D on the internal vertices of Q needs to be preserved while taking into account whether the endpoints of Q have a neighbor in the path or not. Because of this we will not be replacing Q merely by a shorter path, but by a more involved gadget.

Reduction Rule 2. *Let $a, b_1, \ldots, b_m, c$ be a path in G such that for each $i \in [m]$, b_i has degree precisely 2. For each $B \subseteq \{a, c\}$, let $\ell_{\max}(B)$ be the maximum sum of scores that can be achieved by $b_1, \ldots, b_m$ under the condition that b_1 (and analogously b_m) takes a (c) into its parent set if and only if $a \in B$ ($c \in B$). In other words,*

$\ell_{\max}(B) = \max_{D_B} \sum_{b_i | i \in [m]} f_{b_i}(P_{D_B}(b_i))$ where D_B is a DAG on $\{b_1, \ldots, b_m\} \cup B$ such that B does not contain any vertices of out-degree 0 in D_B. Moreover, let $\ell_{\text{noPath}}(a)$ (and analogously $\ell_{\text{noPath}}(c)$) be the maximum score that can be achieved on the vertices $b_1, \ldots, b_m$ by a DAG on $a, b_1, \ldots, b_m, c$ with the following properties: a (c) has out-degree 1, c (a) has out-degree 0, and there is no directed path from a to b_m (from c to b_1).

We construct a new instance $\mathcal{I}' = (V', \mathcal{F}', \ell)$ as follows:

- $V' := V \cup \{b\} \setminus \{b_2 ... b_{m-1}\}$;
- $\Gamma_{f'}(b) = \{B \cup \{b_1, b_m\} | B \subseteq \{a, c\}\}$ *with scores* $f'_b(B \cup \{b_1, b_m\}) := \ell_{\max}(B)$;
- *The scores for a and c are obtained from $\mathcal{F}$ by simply adding b to any parent set containing either b_1 or b_m; formally:*
  - $\Gamma_{f'}(a)$ *is a union of* $\{P \in \Gamma_f(a) | b_1 \notin P\}$, *where* $f'_a(P) := f_a(P)$ *and* $\{P \cup \{b\} | b_1 \in P, P \in \Gamma_f(a)\}$, *where* $f'_a(P \cup \{b\}) := f_a(P)$;
  - $\Gamma_{f'}(c)$ *is a union of* $\{P \in \Gamma_f(c) | b_m \notin P\}$, *where* $f'_c(P) := f_c(P)$, *and* $\{P \cup \{b\} | b_m \in P, P \in \Gamma_f(c)\}$, *where* $f'_c(P \cup \{b\}) := f_c(P)$.
- $\Gamma_{f'}(b_1)$ *contains only* $\{a, b, b_m\}$ *with score* $\ell_{\text{noPath}}(a)$;
- $\Gamma_{f'}(b_m)$ *contains only* $\{c, b, b_1\}$ *with score* $\ell_{\text{noPath}}(c)$;
- *for all* $w \in V' \setminus \{a, b_1, b, b_m, c\}$, $f'_w = f_w$.

An Illustration of Reduction Rule 2 is provided in Figure 4.1. The rule can be applied in linear time, since the 6 values of ℓ_{noPath} and ℓ_{max} can be computed in linear time by a simple dynamic programming subroutine that proceeds along the path $a, b_1, \ldots, b_m, c$ (alternatively, one may instead invoke the fact that paths have treewidth 1 [OS13]).

Lemma 10. *Reduction Rule 2 is safe.*

Proof. Note that the superstructure graph of reduced instance is obtained from $G_{\mathcal{I}}$ by contracting $b_2 ... b_{m-1}$, adding b and connecting it by edges to a, c, b_1, b_m. We will show that a score of at least ℓ can be achieved in the original instance $\mathcal{I}$ if and only if a score of at least ℓ can be achieved in the reduced instance $\mathcal{I}'$.

Assume that D is a DAG that achieves a score of ℓ in $\mathcal{I}$. We will construct a DAG D', called the *reduct* of D, with $f'(D') \geq \ell$. To this end, we first modify D by removing the vertices $b_2 ... b_{m-1}$ and adding b (let us denote the DAG obtained at this point D^*). Further modifications of D^* depend only on $D[a, b_1 ... b_m, c]$, and we distinguish the 6 cases listed below (see also Figure 4.1):

- case 1: D contains both arcs ab_1 and cb_m. We add to D^* arcs from a, c, b_1, b_m to b, denote resulting graph by D'. As D' is obtained from DAG by making b a sink, it is a DAG as well. Parent set of b in D' is $\{a, c, b_1, b_m\}$, so its score is

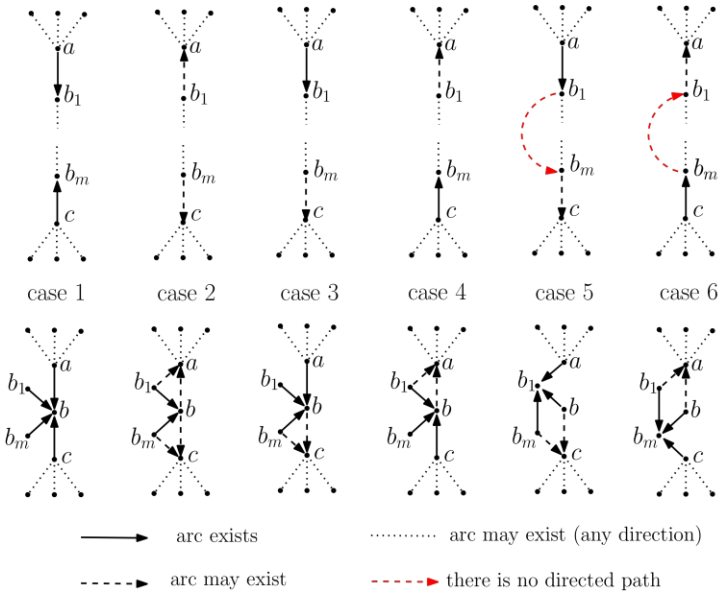

Figure 4.1:
Top: The six possible scenarios that give rise to the values of ℓ_{max} (Cases 1-4) and ℓ_{noPath} (Cases 5-6). Bottom: The corresponding arcs in the gadget after the application of Reduction Rule 2.

$\ell_{max}(a,c) \geq \sum_{i=1}^{m} f_{b_i}(P_D(b_i))$, which means that it achieves the highest scores all of b_i's can achieve in D. The remaining vertices in $V(D') \setminus \{b_1, b_m, b\}$ have the same scores as in D, so $f'(D') \geq f(D) = \ell$.

- case 2: D contains none of the arcs ab_1 and cb_m. To keep the scores of a and c the same as in D, we add to D^* the arc ba iff D contains b_1a, add arc bc iff D contains b_mc. Furthermore, we add arcs b_1b and b_mb and denote resulting graph D'. As D' is obtained from D by making b a source and then adding sources b_1 and b_m, it is a DAG as well. The parent set of b in D' is $\{b_1, b_m\}$, so its score is $\ell_{max}(\emptyset) \geq \sum_{i=1}^{m} f_{b_i}(P_D(b_i))$. Rest of vertices in $V(D') \setminus \{b_1, b_m, b\}$ have the same scores as in D, so $f'(D') \geq f(D) = \ell$.

- case 3: D doesn't contain the arc cb_m, but contains ab_1 and all the arcs b_ib_{i+1}, $i \in [m-1]$. We add to D^* arcs ab, b_1b and b_mb. We also add bc iff D contains b_mc, to preserve the score of c. Denote resulting graph by D'. D' can be considered as D where long directed path $a \to b_1 \to ... \to b_m$ was replaced by $a \to b$ and then sources b_1 and b_m were added, so it is a DAG. Arguments for scores are similar to cases 1 and 2.

- case 4: D doesn't contain the arc ab_1, but contains cb_m and all the arcs $b_{i+1}b_i$, $i \in [m-1]$. This case is symmetric to case 3.

- case 5: D contains the arc ab_1 but does not contain the arc cb_m and at least one of the arcs b_ib_{i+1}, $i \in [m-1]$ is also missing (i.e., there is no directed path from a to b_m). We add to D' arcs bb_1 and b_mb_1. If $b_mc \in A(D)$, add also bc. Denote the resulting graph D'. As D' is obtained from D^* by making b_1 a sink and b, b_m sources, it is a DAG. The parent set of b_1 in D' is $\{a, b, b_m\}$ so its score is $\ell_{noPath}(a) \geq \sum_{i=1}^{m} f_{b_i}(P_D(b_i))$. Rest of vertices in $V(D') \setminus \{b_1, b_m, b\}$ have the same

scores as in D, so $f'(D') \geq f(D) = \ell$.

- case 6: D contains the arc cb_m but does not contain the arc ab_1 and at least one of the arcs $b_{i+1}b_i$, $i \in [m-1]$ is also missing. This case is symmetric to case 5.

The considered cases exhaustively partition all possible configurations of $D[a, b_1 ... b_m, c]$, so we always can construct D' with a score at least ℓ. For the converse direction, note that the DAGs constructed in cases 1-6 cover all optimal configurations on $\{a, b_1, b, b_m, c\}$: if there is a DAG D'' in $\mathcal{I}'$ with a score of ℓ', we can always modify the construction to obtain a DAG D' with score at least ℓ' such that $D'[a, b_1, b, b_m, c]$ has one of the forms depicted at the bottom line of the figure. The claim for the converse direction follows from the fact that every such D' is a reduct of some DAG D of the original instance with the same score. □

We are now ready to prove the desired result.

Proof of Theorem 8. We begin by exhaustively applying Reduction Rule 1 on an instance whose superstructure graph has a feedback edge set of size k, which results in an instance with the same feedback edge set but whose spanning tree T has at most $2k$ leaves. It follows that there are at most $2k$ vertices with a degree greater than 2 in T.

Let us now "mark" all the vertices that either are endpoints of the edges in E_F or have a degree greater then 2 in T; the total number of marked vertices is upper-bounded by $4k$. We now proceed to the exhaustive application of Reduction Rule 2, which will only be triggered for sufficiently long paths in T that connect two marked vertices but contain no marked vertices on its internal vertices; there are at most $4k$ such paths due to the tree structure of T. Reduction Rule 2 will replace each such path with a set of 3 vertices, and therefore after its exhaustive application we obtain an equivalent instance with at most $4k + 4k \cdot 3 = 16k$ vertices, as desired. Correctness follows from the safeness of Reduction Rules 1, 2, and the runtime bound follows by observing that the total number of applications of each rule as well as the runtime of each rule are upper-bounded by a linear function of the input size. □

4.2 Data Reduction for PL$^{\neq 0}$

Recall that the proof of Theorem 8 used two data reduction rules. While Reduction Rule 1 carries over to PL$^{\neq 0}$, Reduction Rule 2 has to be completely redesigned to preserve the (non-)existence of undirected paths between a and c. By doing so, we obtain:

Theorem 11. *There is an algorithm which takes as input an instance $\mathcal{I}$ of PL$^{\neq 0}$ whose superstructure has feedback edge number k, runs in time $\mathcal{O}(|\mathcal{I}|^2)$, and outputs an equivalent instance $\mathcal{I}' = (V', \mathcal{F}', \ell')$ of PL$^{\neq 0}$ such that $|V'| \leq 24k$.*

Proof. Note that Reduction Rule 1 acts on the superstructure graph by deleting leaves and therefore preserves not only optimal scores but also (non-)existance of polytrees

achiving the scores. Hence we can safely apply the rule to reduce the instance of $\mathrm{PL}^{\neq 0}$. After the exhaustive application, all the leaves of the superstructure graph G are the endpoints of edges in feedback edge set, so there can be at most $2k$ of them. To get rid of long induced paths in G, we introduce the following rule:

Reduction Rule 3. *Let $a, b_1, \ldots, b_m, c$ be a path in G such that for each $i \in [m]$, b_i has degree precisely 2. For every $B \subseteq \{a, c\}$ and $p \in \{0, 1\}$, let $\ell_p(B)$ be the maximum sum of scores that can be achieved by $b_1, \ldots, b_m$ under the conditions that (1) there exists an undirected path between b_1 and b_m if and only if $p = 1$; (2) b_1 (and analogously b_m) takes a (c) into its parent set if and only if $a \in B$ ($c \in B$).*

We construct a new instance $\mathcal{I}' = (V', \mathcal{F}', \ell)$ as follows:

- $V' := (V \cup \{b, b'_1, b''_1, b'_m, b''_m\}) \setminus \{b_1 \ldots b_m\}$;
- $\Gamma_{f'}(b'_1) = \Gamma_{f'}(b''_1) = \Gamma_{f'}(b'_m) = \Gamma_{f'}(b''_m) = \emptyset$;
- *The scores for a (analagously c) are obtained from $\mathcal{F}$ by simply replacing every occurence of b_1 by b'_1 and b''_1 (b_m by b'_m and b''_m), formally:*
  - $\Gamma_{f'}(a)$ *is a union of* $\{P \in \Gamma_f(a) | b_1 \notin P\}$, *where* $f'_a(P) := f_a(P)$ *and* $\{(P \setminus b_1) \cup \{b'_1, b''_1\} | b_1 \in P, P \in \Gamma_f(a)\}$, *where* $f'_a((P \setminus b_1) \cup \{b'_1, b''_1\}) := f_a(P)$;
  - $\Gamma_{f'}(c)$ *is a union of* $\{P \in \Gamma_f(c) | b_m \notin P\}$, *where* $f'_c(P) := f_c(P)$, *and* $\{(P \setminus b_m) \cup \{b'_m, b''_m\} | b_m \in P, P \in \Gamma_f(c)\}$, *where* $f'_c((P \setminus b_m) \cup \{b'_m, b''_m\}) := f_c(P)$.
- $\Gamma_{f'}(b)$ *consists of eight sets, yielding corresponding scores f'_b:*
  $\{a, c, b'_1, b''_1, b'_m, b''_m\} \to l_1(\{a, c\})$, $\{b'_1, b''_1, b'_m, b''_m\} \to l_0(\{a, c\})$,
  $\{b'_1, b'_m\} \to l_1(\emptyset)$, $\emptyset \to l_0(\emptyset)$,
  $\{a, b'_1, b''_1, b'_m\} \to l_1(\{a\})$, $\{b'_1, b''_1\} \to l_0(\{a\})$,
  $\{b'_1, b'_m, b''_m, c\} \to l_1(\{c\})$. $\{b'_m, b''_m\} \to l_0(\{c\})$,

Parent sets of b are defined in a way to cover all the possible configurations on solutions to $\mathcal{I}$ restricted to $a, b_1, \ldots, b_m, c$; the corresponding scores of b are intuitively the sums of scores that b_i, $i \in [m]$, receive in the solutions. The eight cases that may arise are illustrated in Figure 4.2.

Claim 1. *Reduction Rule 3 is safe.*

Proof. We will show that a score of at least ℓ can be achieved in the original instance $\mathcal{I}$ if and only if a score of at least ℓ can be achieved in the reduced instance $\mathcal{I}'$.

Assume that D is a polytree that achieves the score of ℓ in $\mathcal{I}$. We will construct a polytree D', called the *reduct* of D, with $f'(D') \geq \ell$. To this end, we first modify D by removing the vertices $b_1, \ldots, b_m$ and adding $b, b'_1, b''_1, b'_m, b''_m$. We also add arcs $b'_1 a$ and $b''_1 a$ ($b'_m c$ and $b''_m c$ correspondingly) if and only if $b_1 a \in A(D)$ ($b_m c \in A(D)$). Let us denote the DAG obtained at this point D^*. Note that scores of a and c in D^* are the same as in D.

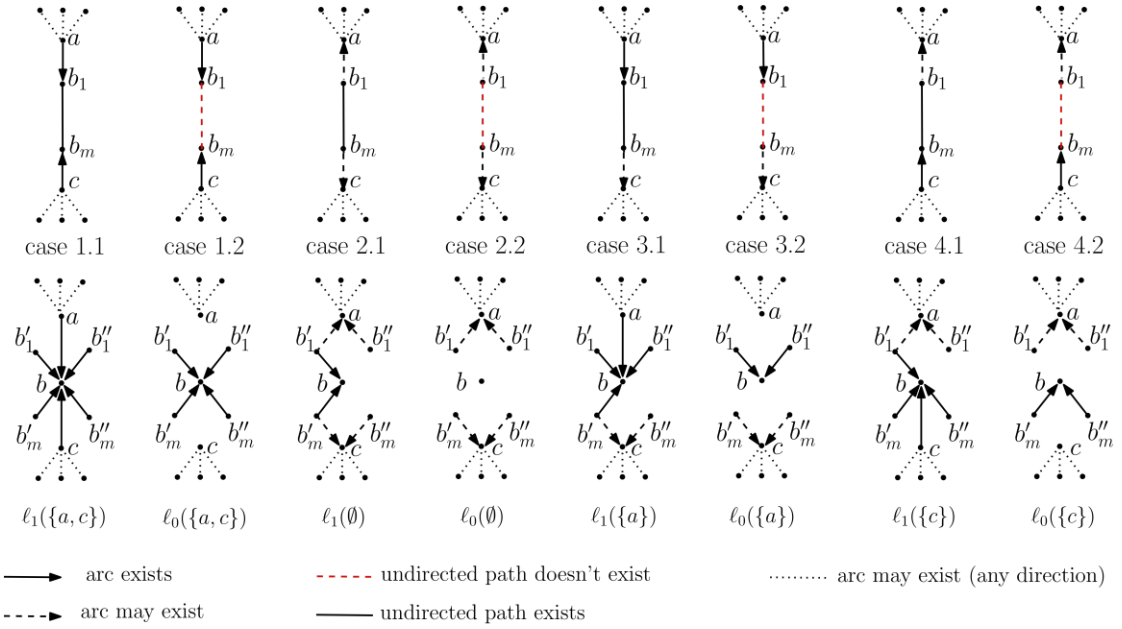

Figure 4.2: Top: The eight possible scenarios for solutions to $\mathcal{I}$. Bottom: The corresponding arcs in the gadget after the application of Reduction Rule 2' (the scores of b are specified below).

Further modifications of D^* depend only on $D[a, b_1 ... b_m, c]$ and change only the parent set of b. We distinguish the 8 cases listed below (see also Figure 4.2):

- case 1.1 (1.2): $ab_1, cb_m \in A(D)$, b_1 and b_m are (not) connected by an undirected path in D. We add incoming arcs to b from $a, c, b'_1, b''_1, b'_m, b''_m$ (b'_1, b''_1, b'_m, b''_m only) resulting in $f'_b(P_{D'}(b)) = l_1(\{a, c\})$ ($f'_b(P_{D'}(b)) = l_0(\{a, c\})$).
- case 2.1 (2.2): $ab_1, cb_m \notin A(D)$, b_1 and b_m are (not) connected by an undirected path in D. We add incoming arcs to b from b'_1 and b'_m (leave D^* unchanged) yielding $f'_b(P_{D'}(b)) = l_1(\emptyset)$ ($f'_b(P_{D'}(b)) = l_0(\emptyset)$).
- case 3.1 (3.2): $ab_1 \in A(D)$, $cb_m \notin A(D)$, b_1 and b_m are (not) connected by an undirected path in D. We add incoming arcs to b from a, b'_1, b''_1, b'_m (b'_1 and b''_1 only), then $f'_b(P_{D'}(b)) = l_1(\{a\})$ ($f'_b(P_{D'}(b)) = l_0(\{a\})$).
- case 4.1 (4.2): $ab_1 \notin A(D)$, $cb_m \in A(D)$, b_1 and b_m are (not) connected by an undirected path in D. The cases are symmetric to 3.1 (3.2)

Note that D' contains a path between a and c if and only if D does. By definition of l_0 and l_1, the score of b in D' is at least as large as the sum of scores of b_i, $i \in [m]$, in D. Moreover, each vertice in $V(D) \cap V(D')$ receives equal scores in D and D'. Hence D' is a polytree with $f'(D') \geq \ell$, as desired.

For the converse direction, note that the polytrees constructed in cases 1.1-4.2 cover all optimal configurations which may arise in $\mathcal{I}'$: if there is a polytree D'' in $\mathcal{I}'$ with a score of ℓ', we can always modify it to a polytree D' with a score of at least ℓ' such that $D'[a, b'_1, b''_1, b, b'_m, b''_m, c]$ has one of the forms depicted at the bottom line of the figure. But every such D' is a reduct of some polytree D of the original instance with the same score. □

We apply Reduction Rule 3 exhaustively, until there is no more path to shorten. Bounds on the running time of the procedure and size of the reduced instance can be obtained similarly to the case of BNSL$^{\neq 0}$. In particular, every long path is replaced with a set of 5 vertices, resulting in at most $4k + 4k \cdot 5 = 24k$ vertices. □

CHAPTER 5

Local Feedback Edge Number

5.1 Fixed-Parameter Algorithm for BNSL$^{\neq 0}$

Our aim here will be to lift the fixed-parameter tractability of BNSL$^{\neq 0}$ established by Theorem 8 by relaxing the parameterization to lfen. In particular, we will prove:

Theorem 12. *If a spanning tree T of G such that $\mathrm{lfen}(G,T) = k$ is provided as part of the input, BNSL$^{\neq 0}$ can be solved in time $2^{O(k^3)} \cdot n^3$, where $n = |\mathcal{I}|$.*

Since fen is a more restrictive parameter than lfen, this results in a strictly larger class of instances being identified as tractable. However, the means we will use to establish Theorem 12 will be fundamentally different: we will not use a polynomial-time data reduction algorithm as the one provided in Theorem 8, but instead apply a dynamic programming approach.

As our first step towards proving Theorem 12, we provide general conditions for when the union of two DAGs is a DAG as well. Let $D = (V, A)$ be a directed graph and $V' \subseteq V$. Denote by $\mathtt{Con}(V', D)$ the binary relation on $V' \times V'$ which specifies whether vertices from V' are connected by a path in D: $\mathtt{Con}(V', D) = \{(v_1, v_2) \subseteq V' \times V' | \exists$ directed path from v_1 to v_2 in $D\}$. Similarly to arcs, we will use $v_1v_2 \in$ as shorthand for (v_1, v_2); we will also use `trcl` to denote the transitive closure.

Lemma 13. *Let D_1, D_2 be digraphs with common vertices $V_{\mathrm{com}} = V(D_1) \cap V(D_2)$, $V_{\mathrm{com}} \subseteq V_1 \subseteq V(D_1)$, $V_{\mathrm{com}} \subseteq V_2 \subseteq V(D_2)$. Then:*

- *(i)* $\mathtt{Con}(V_1 \cup V_2, D_1 \cup D_2) = \mathtt{trcl}(\mathtt{Con}(V_1, D_1) \cup \mathtt{Con}(V_2, D_2))$*;*
- *(ii) If D_1, D_2 are DAGs and* $\mathtt{Con}(V_1 \cup V_2, D_1 \cup D_2)$ *is irreflexive, then $D_1 \cup D_2$ is a DAG.*

Proof. (i) Denote $R_i := \texttt{Con}(V_i, D_i)$, $i = 1, 2$. Obviously $\texttt{trcl}(R_1 \cup R_2)$ is a subset of $\texttt{Con}(V_1 \cup V_2, D_1 \cup D_2)$. Assume that for some $x, y \in V_1 \cup V_2$ there exists a directed path P from x to y in $D_1 \cup D_2$. We will show (by induction on the length l of shortest P) that $xy \in \texttt{trcl}(R_1 \cup R_2)$.

- $l = 1$: in this case there is an arc xy in some D_i, so $xy \in R_i \subseteq \texttt{trcl}(R_1 \cup R_2)$
- $l \to l + 1$. If P is completely contained in some D_i, then $xy \in R_i \subseteq \texttt{trcl}(R_1 \cup R_2)$. Otherwise P must contain arcs $e \notin A(D_1)$, $f \notin A(D_2)$. Then there is $w \in V_{com} \subseteq V_1 \cup V_2$ between them. By the induction hypothesis $xw \in \texttt{trcl}(R_1 \cup R_2)$ and $wy \in \texttt{trcl}(R_1 \cup R_2)$, so $xy \in \texttt{trcl}(R_1 \cup R_2)$

(ii) The precondition implies that the digraph $D_1 \cup D_2$ induced on $V_1 \cup V_2$ is a DAG. Assume that $D_1 \cup D_2$ is not a DAG and let C be a shortest directed cycle in $D_1 \cup D_2$. As D_1 and D_2 are DAGs, C must contain arcs $e \notin A(D_1)$, $f \notin A(D_2)$. So there are least 2 different vertices x, y from V_{com} in C. By (i) we have that $xy \in \texttt{trcl}(R_1 \cup R_2)$ and $yx \in \texttt{trcl}(R_1 \cup R_2)$, then also $xx \in \texttt{trcl}(R_1 \cup R_2)$, which contradicts irreflexivity. □

Towards proving Theorem 12, assume that we are given an instance $\mathcal{I} = (V, \mathcal{F}, \ell)$ of BNSL$^{\neq 0}$ with connected superstructure graph $G = (V, E)$. Let T be a fixed rooted spanning tree of G such that $\text{lfen}(G, T) = \text{lfen}(G) = k$, denote the root by r. For $v \in V(T)$, let T_v be the subtree of T rooted at v, let $V_v = V(T_v)$, and let $\bar{V}_v = N_G(V_v) \cup V_v$. We define the *boundary* $\delta(v)$ of v to be the set of endpoints of all edges in G with precisely one endpoint in V_v (observe that the boundary can never have a size of 1).

v is called *closed* if $|\delta(v)| \leq 2$ and *open* otherwise. We begin by establishing some basic properties of the local feedback edge set.

Observation 14. *Let v be a vertex of T. Then:*

1. *For every closed child w of v in T, it holds that $\delta(w) = \{v, w\}$ and vw is the only edge between V_w and $V \setminus V_w$ in G.*
2. *$|\delta(v)| \leq 2k + 2$.*
3. *Let $\{v_i | i \in [t]\}$ be the set of all open children of v in T. Then $t \leq 2k$ and $\delta(v) \subseteq \cup_{i=1}^{t} \delta(v_i) \cup \{v\} \cup N_G(v)$*

Proof. The first claim follows by the connectivity assumption on G and the definition of boundary.

For the second claim, clearly $\delta(r) = \emptyset$. Let $v \neq r$ have the parent u, and consider an arbitrary $w \in \delta(v) \setminus \{u, v\}$. Then there is an edge $ww' \in E(G)$ with precisely one endpoint in V_v and $ww' \neq uv$. Hence $ww' \notin E(T)$ and the path between w and w' in T contains v, and this implies $ww' \in E^T_{loc}(v)$ by definition. Consequently, $w \in V^T_{loc}(v)$. For the claimed bound we note that $|V^T_{loc}(v)| \leq 2|E^T_{loc}(v)| \leq 2k$.

For the third claim, let $w = v_i$ for some $i \in [t]$. As w is open, there exists an edge $e \neq vw$ between V_w and $V \setminus V_w$ in G. By definition of local feedback edge set, $e \in E^T_{loc}(v)$. Let x_w be the endpoint of e that belongs to V_w, then $x_w \in V^T_{loc}(v)$ and $x_w \notin V_{w'}$ for any open child $w' \neq w$ of v. But $|V^T_{loc}(v)| \leq 2k$, which yields the bound on number t of open children.
For the boundary inclusion, consider any edge c in G with precisely one endpoint x_v in V_v. Note that x_v can not belong to V_w for any closed child w of v. If $x_v \in V_{v_i}$ for some $i \in [t]$, then endpoints of c belong to $\delta(v_i)$. Otherwise $x_v = v$ and therefore the second endpoint of c is in $N_G(v)$. □

With Observation 14 in hand, we can proceed to a definition of the records used in our dynamic program. Intuitively, these records will be computed in a leaf-to-root fashion and will store at each vertex v information about the best score that can be achieved by a partial solution that intersects the subtree rooted at v.

Let R be a binary relation on $\delta(v)$ and s an integer. For $s \in \mathbb{Z}$, we say that $(R : s)$ is a *record* for a vertex v if and only if there exists a DAG D on $\bar{V}_v$ such that (1) $w \in V_v$ for each arc $uw \in A(D)$, (2) $R = \texttt{Con}(\delta(v), D)$ and (3) $\sum_{u \in V_v} f_u(P_D(u)) = s$. The records (R, s) where s is maximal for fixed R are called *valid*. Denote the set of all valid records for v by $\mathcal{R}(v)$, and note that $|\mathcal{R}(v)| \leq 2^{\mathcal{O}(k^2)}$.

Observe that if v_i is a closed child of v, then by Observation 14.1 $\mathcal{R}(v_i)$ consists of precisely two valid records: one for $R = \emptyset$ and one for $R = \{vv_i\}$. Moreover, the root r of T has only a single valid record $(\emptyset : s_{\mathcal{I}})$, where $s_{\mathcal{I}}$ is the maximum score that can be achieved by a solution in $\mathcal{I}$. The following lemma lies at the heart of our result and shows how we can compute our records in a leaf-to-root fashion along T.

Lemma 15. *Let $v \in V(G)$ have m children in T where $m > 0$, and assume we have computed $\mathcal{R}(v_i)$ for each child v_i of v. Then $\mathcal{R}(v)$ can be computed in time at most $m \cdot |\Gamma_f(v)| \cdot 2^{\mathcal{O}(k^3)}$.*

Proof. Without loss of generality, let the open children of $v \in V(G)$ be $v_1, \ldots, v_t$ and let the remaining (i.e., closed) children of v be $v_{t+1}, \ldots, v_m$; recall that by Point 3. of Observation 14, $t \leq 2k$. For each closed child v_j, $j \in [m] \setminus [t]$, let $s^{\emptyset}_j$ be the second component of the valid record for $\emptyset \in \mathcal{R}(v_j)$, and let $s^{\times}_j$ be the second component of the valid record for the single non-empty relation in $\mathcal{R}(v_j)$. Consider the following procedure $\mathbb{A}$.

First, $\mathbb{A}$ branches over all choices of $P \in \Gamma_f(v)$ and all choices of $(R_i, s_i) \in \mathcal{R}(v_i)$ for each individual open child v_i of v. Let $R_0 = \{\, pv \mid p \in P \,\}$ and let $R' = \bigcup_{j \in [t]_0} R_j$. If $\texttt{trcl}(R')$ is not irreflexive, we discard this branch; otherwise, we proceed as follows. Let R_{new} be the subset of R' containing all arcs uw such that $w \in V_v$. Moreover, let $s_{new} = f_v(P) + (\sum_{i \in [t]} s_i) + (\sum_{i \in [m] \setminus [t] \mid v_i \in P} s^{\emptyset}_i) + (\sum_{i \in [m] \setminus [t] \mid v_i \notin P} (\max(s^{\emptyset}_i, s^{\times}_i))$.

The algorithm $\mathbb{A}$ gradually constructs a set $\mathcal{R}^*(v)$ as follows. At the beginning, $\mathcal{R}^*(v) = \emptyset$. For each newly obtained tuple (R_{new}, s_{new}), $\mathbb{A}$ checks whether $\mathcal{R}^*(v)$ already contains a

tuple with R_{new} as its first element; if not, we add the new tuple to $\mathcal{R}^*(v)$. If there already exists such a tuple $(R_{new}, s_{old}) \in \mathcal{R}^*(v)$, we replace it with $(R_{new}, \max(s_{old}, s_{new}))$.

For the running time, recall that in order to construct $\mathcal{R}^*(v)$ the algorithm branched over $|\Gamma_f(v)|$-many possible parent sets of v and over the choice of at most $2k$-many binary relations R_i on the boundaries of open children. According to Observation 14.2, there are at most $3^{(2k+2)^2}$ options for every such relation, so we have at most $\mathcal{O}((3^{(2k+2)^2})^{2k} \cdot |\Gamma_f(v)|) \leq 2^{\mathcal{O}(k^3)} \cdot |\Gamma_f(v)|$ branches. In every branch we compute $\texttt{trcl}(R')$ in time $k^{\mathcal{O}(1)}$ and then compute the value of s_{new} using the equation provided above before updating $\mathcal{R}^*(v)$, which takes time at most $\mathcal{O}(m)$.

Finally, to establish correctness it suffices to prove following claim:

Claim 2. *$(R : s)$ is a record for v if and only if there exist $P \in \Gamma_f(v)$ and records $(R_i : s_i)$ for v_i, $i \in [m]$, such that:*

- $\texttt{trcl}(\cup_{i=0}^{t} R_i)$ *is irreflexive;*
- $R_i = \emptyset$ *for any closed child* $v_i \in P$*;*
- $\sum_{i=1}^{m} s_i + f_v(P) = s$*;*
- $R = (\texttt{trcl}(\cup_{i=0}^{t} R_i)) \cap (\delta(v) \times \delta(v))$.

Moreover, if $(R : s) \in \mathcal{R}(v)$ then in addition:

- $(R_i : s_i) \in \mathcal{R}(v_i)$, $i \in [t]$*;*
- *for every closed child* $v_i \notin P$, $s_i = \max(s_i^{\emptyset}, s_i^{\times})$.

Proof of the Claim. (a) ($\Leftarrow$) Denote $V_i = V_{v_i}$ and $\bar{V}_i = \bar{V}_{v_i}$, $i \in [m]$. For every $i \in [m]$ there exists DAG D_i on $\bar{V}_i$ such that all its arcs finish in V_i, $R_i = \texttt{Con}(\delta(v_i), D_i)$ and $\sum_{u \in V_i} f_u(P_{D_i}(u)) = s_i$. Denote by D_0 DAG on $V_0 = v \cup N_G(v)$ with arc set R_0. We will construct the witness D of (R, s) by gluing together all D_i, $i \in [m]_0$.

We start from D_0 and DAGs of open children. Note that $\texttt{Con}(V_0, D_0) = R_0$ and $\texttt{Con}(\delta(v_i), D_i) = R_i$ for $i \in [t]$. Inductive application of Lemma 13 to DAGs D_i, $i \in [t]$, yields $\texttt{Con}(\cup_{i=1}^{t} \delta(v_i) \cup V_0, D^*) = \texttt{trcl}(\cup_{i=0}^{t} R_i)$. In particular, as $\delta(v) \subseteq \cup_{i=1}^{t} \delta(v_i) \cup V_0$ by Observation 14.3, we have that $\texttt{Con}(\delta(v), D^*) = (\texttt{trcl}(\cup_{i=0}^{t} R_i))|_{\delta(v) \times \delta(v)} = R$. As $\texttt{trcl}(\cup_{i=0}^{t} R_i)$ is irreflexive, $D^* = \cup_{i=0}^{t} D_i$ is DAG by Lemma 13.

Now we add to D^* DAGs for closed children and finally obtain $D = \cup_{i=t+1}^{m} D_i \cup D^*$. For every closed child v_i, D_i is by Observation 14.1 the union of v and $D_i \setminus v$, plus at most one of arcs vv_i, v_iv between them (recall $R_i = \emptyset$ for any closed child $v_i \in P$). Note that $D_i \setminus v$ can share only v_i with D_0 and doesn't have common vertices with any other D_j. Therefore any directed path in D starting and finishing outside outside of V_i, $i > t$, doesn't intersect V_i. In particular, acyclicity of D^* and D_i, $i \in [m] \setminus [t]$, implies acyclicity

of D; $\texttt{Con}(\delta(v), D) = \texttt{Con}(\delta(v), D^*) = R$.

All the arcs in D_i finish in V_i, so parent set for every $x_i \in D_i$ in D is the same as in D_i, $i \in [m]$. Also parent set of v in D is the same as in D_0. So

$$\sum_{u \in V_v} f_u(P_D(u)) = \sum_{i=1}^{m} \sum_{u \in V_i} f_u(P_{D_i}(u)) + f_v(P_{D_0}(v)) = \sum_{i=1}^{m} s_i + f_v(P) = s$$

($\Rightarrow$) Let D be a witness for $(R : s)$, i.e. D is DAG on $\bar{V}_v$ with all arcs finishing in V_v such that $\sum_{u \in V_v} f_u(P_D(u)) = s$ and $\texttt{Con}(\delta(v), D) = R$. For $i = 1 \in [m]$ define $D'_i = D[\bar{V}_i]$ and let D_i be obtained from D'_i by deleting arcs that finish outside V_i. Note that $\cup_{i=1}^{m} D_i = D$. Let $R_i = \texttt{Con}(\delta(v_i), D_i)$, as in ($\Leftarrow$) we have that $R = \texttt{Con}(\delta(v), D) = \texttt{trcl}(\cup_{i=0}^{t} R_i))|_{\delta(v) \times \delta(v)}$. As D is DAG, $\texttt{trcl}(\cup_{i=0}^{t} R_i)$ is irreflexive and $R_i = \emptyset$ for any closed child $v_i \in P$. Local score for D_i is

$$s_i = \sum_{u \in V_i} f_u(P_{D_i}(u)) = \sum_{u \in V_i} f_u(P_{D'_i}(u)) = \sum_{u \in V_i} f_u(P_D(u))$$

So v_i has record $(R_i : s_i)$. Denote $P = P_D(v)$. Then:

$$s = \sum_{u \in V_v} f_u(P_D(u)) = \sum_{i=1}^{m} \sum_{u \in V_i} f_u(P_D(u)) + f_v(P_D(v)) = \sum_{i=1}^{m} s_i + f_v(P)$$

(b) Let $(R : s) \in \mathcal{R}(v)$ and all D, P, D_i, R_i, s_i are as in $(a)(\Rightarrow)$. Assume that for some i (R_i, s_i) is not valid record of v_i. In this case v_i must have a record $(R_i : s_i + \Delta)$ with $\Delta > 0$. But then $(a)(\Leftarrow)$ implies that v has record $(R : s + \Delta)$, which contradicts to validity of $(R : s)$

Assume that some closed $v_i \notin P$ has valid record $(R'_i, s_i + \Delta)$ with $\Delta > 0$. R' and R differ only by arc vv_i, so addition or deletion of the arc to D would increase the total score by $\Delta > 0$ without creating cycles. This would result in record $(R : s + \Delta)$ and yield a contradiction with validity of $(R : s)$. ■ □

We are now ready to prove the main result of this section.

Proof of Theorem 12. We provide an algorithm that solves BNSL$^{\neq 0}$ in time $2^{O(k^3)} \cdot n^3$, where $n = |\mathcal{I}|$, assuming that a spanning tree T of G such that $\text{lfen}(G, T) = k$ is given as part of the input.

The algorithm computes $\mathcal{R}(v)$ for every node v in T, moving from leaves to the root:

- For a leaf v, compute $\mathcal{R}^*(v) := \{(R_P : f_v(P)) | \, P \in \Gamma_f(v), R_P = \{uv | u \in P\}\}$. This can be done by simply looping over $\Gamma_f(v)$ in time $\mathcal{O}(n)$. Note that $\mathcal{R}^*(v)$ is the set of all records of v, so we can correctly set $\mathcal{R}(v) := \{(R : s) \in \mathcal{R}^*(v) |$ there is no $(R : s') \in \mathcal{R}^*(v)$ with $s' > s\}$.

- Let $v \in V(G)$ have at least one child in T, and assume we have computed $\mathcal{R}(v_i)$ for each child v_i of v. Then we invoke Lemma 15 to compute $\mathcal{R}(v)$ in time at most $m \cdot |\Gamma_f(v)| \cdot 2^{\mathcal{O}(k^2)} \leq 2^{\mathcal{O}(k^2)} \cdot n^2$. □

5.2 Fixed-Parameter Algorithm for PL$^{\neq 0}$

Similarly as for BNSL we can provide an FPT algorithm using the same ideas as in the proof of Theorem 12. The algorithm proceeds by dynamic programming on the spanning tree T of G with $\mathrm{lfen}(G,T) = \mathrm{lfen}(G) = k$. The records will, however, need to be modified: for each vertex v, instead of the path-connectivity relation on $\delta(v)$, we store connected components of the *inner boundary* $\delta(v) \cap V_v$ and incoming arcs to T_v. We provide a full description of the algorithm below.

Theorem 16. *If a spanning tree T of G such that $\mathrm{lfen}(G,T) = k$ is provided as part of the input, PL$^{\neq 0}$ can be solved in time $2^{O(k^3)} \cdot n^3$, where $n = |\mathcal{I}|$.*

Proof. We pick a root r in T and keep all the notations T_v, V_v, $\bar{V}_v$, $\delta(v)$ for $v \in V(T)$ from the section 5.1. In addition, we define the *inner boundary* of $v \in V(T)$ to be $\delta_{in}(v) := \delta(v) \cap V_v$ i.e. part of boundary that belongs to subtree of T rooted in v. The remaining part we call the *outer boundary* of v and denote by $\delta_{out}(v) := \delta(v) \setminus \delta_{in}(v)$. For any set A of arcs, we define $\widetilde{A} = \{uv | uv \in A \text{ or } vu \in A\}$. Obviously, the claims of Observation 14 still hold. Moreover, for every closed v, $\delta_{in}(v)$ contains only v itself and $\delta_{out}(v)$ is either the parent of v in T or $\emptyset$ (for $v = r$).

Let R_v be binary relation on $\delta_{in}(v)$, $A_v \subseteq \delta_{out}(v) \times \delta_{in}(v)$, s_v is integer. Then (R_v, A_v, s_v) is a *record* for v if and only if there exist a polytree D on $\bar{V}_v$ with all arcs oriented inside V_v such that:

- $A_v = \{xy \in A(D) |\, x \in \delta_{out}(v), y \in \delta_{in}(v)\}$
- $R_v = \{xy |\, x, y \in \delta_{in}(v)$ are in the same connected component of $D[V_v]\}$
- $s_v = \sum_{u \in V_v} f_u(P_D(u))$

Note that R_v is an equivalence relation on $\delta_{in}(v)$, number of its equivalence classes is equal to number of connected components of $D[V_v]$ that intersect $\delta(v)$.

Record (R_v, A_v, s_v) is called *valid* if and only if s_v is maximal for fixed R_v, A_v among all the records for v. Denote by $\mathcal{R}(v)$ the set of all valid records for v, then $|\mathcal{R}(v)| \leq 2^{(2k+2)^2}$. Indeed, R_v and A_v can be uniquely determined by the choice of some relation on $\delta(v) \times \delta(v)$. As $|\delta(v)| \leq 2k+2$, there are at most $2^{(2k+2)^2}$ possible relations.

The root r of T has a single valid record $(\emptyset, \emptyset, s_{\mathcal{I}})$, where $s_{\mathcal{I}}$ is the maximum score that can be achieved by a solution to $\mathcal{I}$. For any closed $v \neq r$, $\mathcal{R}(v)$ consists of precisely two valid records: one for $A_v = \emptyset, R_v = \{vv\}$ and another for $A_v = \{wv\}, R_v = \{vv\}$, where w is a parent of v in T.

We proceed by computing our records in a leaf-to-root fashion along T.

Let v be a leaf. Start by innitiating $\mathcal{R}^*(v) := \emptyset$, then for each $P \in \Gamma_f(v)$ add to $\mathcal{R}^*(v)$ the triple $(\{vv\}, \{uv | u \in P\}, f_v(P))$. Note that $\mathcal{R}^*(v)$ is by definition precisely the set of all records for v, so we can correctly set $\mathcal{R}(v) = \{(R_v, A_v, s_v) \in \mathcal{R}^*(v) | s_v \text{ is maximal for fixed } R_v, A_v\}$.

Assume that v has m children $\{v_i : i \in [m]\}$ in T, where v_i, $i \in [t]$, are open and v_i, $i \in [m] \setminus [t]$, are closed. The following claim shows how (and under which conditions) the records of children of v can be composed into a record of v.

Claim 3. *Let $P \in \Gamma_f(v)$, D_0 is a polytree on $V_0 = v \cup P$ with arc set $A_0 = \{uv | u \in P\}$, (R_i, A_i, s_i) are records for v_i witnessed by D_i, $i \in [m]$. Let A^{in}_{loc} be the set of arcs in $\bigcup_{i\in[t]_0} A_i$ which have both endpoints in V_v, $R = \mathtt{trcl}(\widetilde{A}^{in}_{loc} \cup \bigcup_{i\in[t]_0} R_i)$. Then $D = \cup_{i=0}^{m} D_i$ is a polytree if and only if the following two conditions hold:*

1. *$A_i = \emptyset$ for each closed child $v_i \in P$.*
2. *$\sum_{i=0}^{t} N_i - |A^{in}_{loc}| - \sum_{y\in Y}(n_y - 1) = N$, where*
   - *N is the number of equivalence classes in $\mathtt{trcl}(\bigcup_{i\in[t]_0}(\widetilde{A}_i \cup R_i))$*
   - *N_i is the number of equivalence classes in R_i, $i \in [t]$*
   - *Y is the set of endpoints of arcs in $\bigcup_{i\in[t]_0} A_i$ which don't belong to any V_i, $i \in [m]$.*
   - *For every $y \in Y$, n_y is the number of arcs in $A_0 \cup ... \cup A_t$ having endpoint y.*

In this case D witnesses the record (R_v, A_v, s_v), where:
$R_v = R|_{\delta_{in}(v)\times\delta_{in}(v)}$, $A_v = (\bigcup_{i\in[t]_0} A_i)|_{\delta_{out}(v)\times\delta_{in}(v)}$, $s_v = \sum_{i=0}^{m} s_i + f_v(P)$.

If $(R_v, A_v, s_v) \in \mathcal{R}(v)$, then $(R_i, A_i, s_i) \in \mathcal{R}(v_i)$, $i \in [m]$. Moreover, for any closed child $v_i \notin P$, there is no $(R'_i, A'_i, s'_i) \in \mathcal{R}(v_i)$ with $s'_i > s_i$.

We will prove the claim at the end, let us show how it can be exploited to compute valid records of v. We start from initial setting $\mathcal{R}^*(v) := \emptyset$, then branch over all parent sets $P \in \Gamma_f(v)$ and triples $(R_i, A_i, s_i) \in \mathcal{R}(v_i)$ for open children v_i. For each closed child $v_i \notin P$ take $(R_i, A_i, s_i) \in \mathcal{R}(v_i)$ with maximal s_i, for each closed child $v_i \in P$ take $(R_i, A_i, s_i) \in \mathcal{R}(v_i)$ with $A_i = \emptyset$. Now the first condition of Claim3 holds, if the second one holds as well, we add to $\mathcal{R}^*(v)$ the triple (R_v, A_v, s_v).

According to Claim 3, $\mathcal{R}^*(v)$ computed in such a way consists only of records for v and, in particular, contains all the valid records. Therefore we can correctly set $\mathcal{R}(v) = \{(R_v, A_v, s_v) \in \mathcal{R}^*(v) | \, s_v \text{ is maximal for fixed } R_v, A_v\}$.

To construct $\mathcal{R}^*(v)$ for node v with children v_i, $i \in [m]$, we branch over at most n possible parent sets of v and at most $2^{(2k+2)^2}$ valid records for every open child of v. Number of open children is bounded by $2k$, so we have at most $\mathcal{O}((2^{(2k+2)^2})^{2k} \cdot n) \leq 2^{\mathcal{O}(k^3)} \cdot n$

branches. In a fixed branch we compute scores for closed children in $\mathcal{O}(n)$, application of Claim 3 requires time polynomial in k. So $\mathcal{R}^*(v)$ is computed in time $2^{\mathcal{O}(k^3)} \cdot n^2$ that majorizes running time for leaves. As the number of vertices in T is at most n, total running time of the algorithm is $2^{\mathcal{O}(k^3)} \cdot n^3$ assuming that T is given as a part of the input.

proof of Claim 3 ($\Leftarrow$). We start from checking whether $D = \cup_{i=0}^{m} D_i$ is a polytree. As the first condition implies that a polytree of every closed child v_i is connected to the rest of D by at most one arc $v_i v$ or $v v_i$, it is sufficient to check whether $D^t = \cup_{i=0}^{t} D_i$ is polytree. Number of connected components of D^t is $N' + N$, where N' is the total number of connected components of D_i that don't intersect $\delta(v_i)$, $i \in [t]$. Note that D^t can be constructed as follows:

1. Take a disjoint union of polytrees $D'_i = D_i[V_i]$, $i \in [t]_0$, then the resulting polytree has $N' + \sum_{i=0}^{t} N_i$ connected components.

2. Add arcs between D'_i and D'_j that occur in D for every $i, j \in [t]_0$, i.e. the arcs specified by A^{in}_{loc}. Resulting digraph is a polytree if and only if every added arc decreases the number of connected components by 1, i.e. the number of connected components after this step is $N' + \sum_{i=0}^{t} N_i - |A^{in}_{loc}|$.

3. Add all remaining vertices y of D together with their adjacent arcs in D. Note that such y precisely form the set Y, so D^t is a polytree if and only if we obtained a polytree after the previous step and every $y \in Y$ decreased it's number of connected components by $(n_y - 1)$, i.e. the number $N' + N$ of connected components in D^t is equal to $N' + \sum_{i=0}^{t} N_i - |A^{in}_{loc}| - \sum_{y \in Y}(n_y - 1)$. But this is precisely the condition 2 of the claim.

Now, assuming that D is a polytree, we will show that it witnesses (R_v, A_v, s_v). Parent sets of vertices from each V_i in D are the same as in D_i, parent set of v in D is P. So $s_v = \sum_{i=0}^{m} s_i + f_v(P)$ is indeed the sum of scores over V_v in D.

There are two kinds of arcs in D starting outside of V_v: incoming arcs to v and incoming arcs to the subtrees of open children. Thus $A(D)|_{\delta_{out}(v) \times \delta_{in}(v)} = (\bigcup_{i \in [t]_0} A_i)|_{\delta_{out}(v) \times \delta_{in}(v)} = A_v$.

Take any $u, w \in \delta_{in}(v), u \neq w$, note that u and w can not belong to subtrees of closed children. So u and w are in the same connected component of $D[V_v]$ if and only if they are connected by some undirected path π in the skeleton of D using only vertices from $D^t \cap V_v$. In this case R_i captures the segmens of π which are completely contained in $D_i[V_i], i \in [t]$. Rest of edges in π either connect v to some V_i, $i \in [t]$, or have enpoints in different V_i and V_j for some $i, j \in [t]$. Edges of this kind precisely form the set $\widetilde{A}^{in}_{loc}$, so uw belongs to $R = \texttt{trcl}(\bigcup_{i \in [t]} R_i \cup \widetilde{A}^{in}_{loc})$. Therefore $R_v = R|_{\delta_{in}(v) \times \delta_{in}(v)}$ indeed represents connected components of $\delta_{in}(v)$ in $D[V_v]$.

($\Rightarrow$) Condition 1 obviously holds, otherwise D would contain a pair of arcs with the same endpoints and different directions. In ($\Leftarrow$) we actually showed the necessity of condition 2 when 1 holds.

For the last statement, assume that $(R_v, A_v, s_v) \in \mathcal{R}(v)$ but $(R_i, A_i, s_i) \notin \mathcal{R}(v_i)$ for some i. Then there is $(R_i, A_i, s_i + \Delta) \in \mathcal{R}(v_i)$ for some $\delta > 0$. Let D_i' be a witness of $(R_i, A_i, s_i + \Delta)$, then $D' = \bigcup_{j \in [m] \setminus \{i\}} D_j \cup D_i'$ is a polytree witnessing $(R_v, A_v, s_v + \Delta)$. But this contradicts to validity of (R_v, A_v, s_v). By the same arguments records for closed children $v_i \notin P$ are the ones with maximal s_i among two $(R_i, A_i, s_i) \in \mathcal{R}(v_i)$. ■ □

CHAPTER 6

Lower Bounds: Tree-Cut Width

It is not difficult to show that the local feedback edge number is "sandwiched" between the feedback edge number and tree-cut width.

Proposition 17. *For every graph G,* $\mathrm{tcw}(G) \leq \mathrm{lfen}(G) + 1$ *and* $\mathrm{lfen}(G) \leq \mathrm{fen}(G)$.

Proof. Let us begin with the second inequality. Consider an arbitrary spanning tree T of G. Then for every $v \in V(G)$, $E^T_{\mathrm{loc}}(v)$ is a subset of a feedback edge set corresponding to the spanning tree T, so $|E^T_{\mathrm{loc}}(v)| \leq \mathrm{fen}(G)$ and the claim follows.

To establish the first inequality, we will use the notation and definition of tree-cut width from [GKO21, Subsection 2.4]. Let T be the spanning tree of G with $\mathrm{lfen}(G,T) = \mathrm{lfen}(G)$. We construct a tree-cut decomposition $(T, \mathcal{X})$ where each bag contains precisely one vertex, notably by setting $X_t = \{t\}$ for each $t \in V(T)$. Fix any node t in T other than root, let u be the parent of t in T. All the edges in $G \setminus ut$ with one endpoint in the rooted subtree T_t and another outside of T_t belong to $E^T_{loc}(t)$, so $\mathrm{adh}_T(t) = |\operatorname{cut}(t)| \leq |E^T_{loc}(t)| \leq \mathrm{lfen}(G)$.

Let H_t be the torso of $(T, \mathcal{X})$ in t, then $V(H_t) = \{t, z_1 ... z_l\}$ where z_i correspond to connected components of $T \setminus t$, $i \in [l]$. In $\tilde{H}(t)$, only z_i with degree at least 3 are preserved. But all such z_i are the endpoints of at least 2 edges in $|E^T_{loc}(t)|$, so $\mathrm{tor}(t) = |V(\tilde{H}_t)| \leq 1 + |E^T_{loc}(t)| \leq 1 + \mathrm{lfen}(G)$. Thus $\mathrm{tcw}(G) \leq \mathrm{lfen}(G) + 1$. □

Since lfen lies between fen and tree-cut width in the parameter hierarchy (see Proposition 17) and BNSL$^{\neq 0}$ is FPT when parameterized by lfen, the next step would be to ask whether this tractability result can be lifted to tree-cut width. Below, we answer this question negatively. In fact, we strengthen the hardness result established in [OS13] by bounding the degree of vertices outside of the vertex cover:

Theorem 18. *$BNSL^{\neq 0}$ is* W[1]*-hard when parameterized by the vertex cover number of the superstructure even when all vertices outside of the vertex cover are required to have degree at most* 2*.*

Proof. We reduce from the following well-known W[1]-hard problem [DF13, CFK+15]:

Regular Multicolored Clique (RMC)	
Input:	A k-partite graph $G = (V_1 \cup ... \cup V_k, E)$ such that $\lvert N_G(v)\rvert = m$ for every $v \in V$
Parameter:	The integer k
Question:	Are there nodes v^i that form a k-colored clique in G, i.e. $v^i \in V_i$ and $v^i v^j \in E$ for all $i, j \in [k], i \neq j$?

We say that vertices in V_i have color i. Let $G = (V_1 \cup ... \cup V_k, E)$ be an instance of RMC. We will construct an instance $(V, \mathcal{F}, \ell)$ of $\text{BNSL}^{\neq 0}$ such that $\mathcal{I}$ is a Yes-instance if and only if G is a Yes-instance of RMC. V consists of one vertex v_i for each color $i \in [k]$ and one vertex v_e for every edge $e \in E$. For each edge $e \in E$ that connects a vertex of color i with a vertex of color j, the constructed vertex v_e will have precisely one element in its score function that achieves a non-zero score, in particular: $f_{v_e}(\{v_i, v_j\}) = 1$.

Next, for each $i \in [k]$, we define the scores for v_i as follows. For every $v \in V_i$, let E_v be the set of all edges incident to v in G, and let $P_i^v = \{v_e : e \in E_v\}$. We now set $f_{v_i}(P_i^v) = m + 1$ for each such v; all other parent sets will receive a score of 0. Note that $\{ v_i \mid i \in [k] \}$ forms a vertex cover of the superstructure graph and that all vertices outside of this vertex cover have degree at most 2, as desired. We will show that G has a k-colored clique if and only if there is a Bayesian network D with score at least $\ell = |E| + k + \binom{k}{2}$. (In fact, it will later become apparent that the score can never exceed ℓ.)

Assume first that G has a k-colored clique on $v^i, i \in [k]$, consisting of a set E_X of $\binom{k}{2}$ edges. Consider the digraph D on V obtained as follows. For each vertex v_i, $i \in [k]$, and each vertex v_e where $e \in E$, D contains the arc $v_e v_i$ if v_e is incident to v^i and otherwise D contains the arc $v_i v_e$. This completes the construction of D. Now notice that the construction guarantees that each v_i receives the parent set $P_i^{v^i}$ and hence contributes a score of $m + 1$. Moreover, for every edge e not incident to a vertex in the clique, the vertex v_e contributes a score of 1; note that the number of such edges is $|E| - km + \binom{k}{2}$; indeed, every v_i is incident to m edges but since $v^i, i \in [k]$, was a clique we are guaranteed to double-count precisely $\binom{k}{2}$ many edges. Hence the total score is $k(m + 1) + |E| - km + \binom{k}{2} = |E| + k + \binom{k}{2}$, as desired.

Assume that $\mathcal{I} = (V, \mathcal{F}, \ell)$ is a Yes-instance and let $s_{\text{opt}} \geq \ell = |E| + k + \binom{k}{2}$ be the maximum score that can be achieved by a solution to $\mathcal{I}$; let D be a dag witnessing such a score. Then all v_i, $i \in [k]$, must receive a score of $m + 1$ in D. Indeed, assume that some v_i receives a score of 0 and let P_v be any parent set of v_i with a score $m + 1$. Modify D

by orienting edges $v_i v_e$ for every $v_e \in P_v$ inside v_i. Now local score of v_i is $m+1$, total score of the rest of vertices decreased by at most m (maximal number of v_e that had local score 1 in D and lost it after the modification). So the modified DAG has a score of at least $s_{\texttt{opt}} + 1$, which contradicts the optimality of $s_{\texttt{opt}}$. Therefore all v_i, $i \in [k]$, get score $m+1$ in D.

Let P_i be parent set of v_i in D, then $|P_i| = m$, $P_i = P_i^{v^i}$ for some $v^i \in V_i$. For every $v_e \in P_i$, the local score of v_e in D is 0. Denote by E_{unsat} the set of all v_e that have a score of 0 in D. Every v_e belongs to at most 2 different P_i and $P_i \cap P_j \leq 1$ for every $i \neq j$, so $|E_{unsat}| \geq km - \binom{k}{2}$. If $|E_{unsat}| > km - \binom{k}{2}$, sum of local scores of e_v in D would be smaller then $|E| - km + \binom{k}{2}$, which results in $s_{\texttt{opt}} < |E| + k + \binom{k}{2}$. Therefore $|E_{unsat}| = km - \binom{k}{2}$. But this means that $P_i \cap P_j \neq \emptyset$ for any $i \neq j$, i.e. v^i, $i \in [k]$ form a k-colored clique in G. In particular $s_{\texttt{opt}} = \ell$. □

CHAPTER 7

Additive Scores and Treewidth

While the previous two sections focused on the complexity of BNSL when the non-zero representation was used (i.e., BNSL$^{\neq 0}$), here we turn our attention to the complexity of the problem with respect to the additive representation. Recall from Section 2.3 that there are two variants of interest for this representation: BNSL^{+} and BNSL$^{+}_{\leq}$. In contrast to BNSL$^{\neq 0}$, both of them are fixed-parameter tractable when parameterized by the treewidth of the superstructure alone.

Theorem 19. *BNSL^{+} is can be solved in time $2^{\mathcal{O}(k^2)} \cdot n$, where k is the treewidth of the superstructure and $n = |\mathcal{I}|$. Moreover, BNSL$^{+}_{\leq}$ can be solved in time $2^{\mathcal{O}(k^2)} \cdot q^{\mathcal{O}(k)} \cdot n$.*

Proof. We begin by proving the latter statement, and will then explain how that result can be straightforwardly adapted to obtain the former. As our initial step, we apply Bodlaender's algorithm [Bod96, Klo94] to compute a nice tree-decomposition $(\mathcal{T}, \chi)$ of $G_{\mathcal{I}}$ of width $k = \mathrm{tw}(G_{\mathcal{I}})$. In this proof we use T to denote the set of nodes of $\mathcal{T}$ and let $r \in T$ be the root of $\mathcal{T}$. Given a node $t \in T$, let $\chi_t^{\downarrow}$ be the set of all vertices occurring in bags of the rooted subtree T_t, i.e., $\chi_t^{\downarrow} = \{u \mid \exists t' \in T_t \text{ such that } u \in \chi(t')\}$.

To prove the theorem, we will design a leaf-to-root dynamic programming algorithm which will compute and store a set of records at each node of T, whereas once we ascertain the records for r we will have the information required to output a correct answer. Intuitively, the records will store all information about each possible set of arcs between vertices in each bag, along with relevant connectivity information provided by arcs between vertices in $\chi_t^{\downarrow}$ and information about the partial score. They will also keep track of parent set sizes in each bag.

Formally, the records will have the following structure. For a node t, let $S(t) = \{(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) \mid \mathrm{loc}, \mathrm{con} \subseteq A_{\chi(t)}, \mathrm{inn} : \chi(t) \to [q]_0\}$ be the set of *snapshots* of t. The record $\mathcal{R}_t$ of t is then a mapping from $S(t)$ to $\mathbb{N}_0 \cup \{\perp\}$. Observe that $|S(t)| \leq 4^{k^2}(q+1)^k$.

To introduce the semantics of our records, let $\mathcal{D}_t$ be the set of all directed acyclic graphs over the vertex set $\chi_t^{\downarrow}$ with maximal in-degree at most q, and let $D_t = (\chi_t^{\downarrow}, A)$ be a directed acyclic graph in $\mathcal{D}_t$. We say that the *snapshot of* D_t in t is the tuple (α, β, p) where $\alpha = A \cap A_{\chi(t)}$, $\beta = \texttt{Con}(\chi(t), D_t)$ and p specifies numbers of parents of vertices from $\chi(t)$ in D, i.e. $p(v) = |\{w \in \chi_t^{\downarrow} | wv \in A\}|$, $v \in \chi(t)$. We are now ready to define the record $\mathcal{R}_t$. For each snapshot $(\text{loc}, \text{con}, \text{inn}) \in S(t)$:

- $\mathcal{R}_t(\text{loc}, \text{con}, \text{inn}) = \perp$ if and only if there exists no directed acyclic graph in $\mathcal{D}_t$ whose snapshot is $(\text{loc}, \text{con}, \text{inn})$, and
- $\mathcal{R}_t(\text{loc}, \text{con}, \text{inn}) = \tau$ if $\exists D_t \in \mathcal{D}_t$ such that
  - the snapshot of D_t is $(\text{loc}, \text{con}, \text{inn})$,
  - $\text{score}(D_t) = \tau$, and
  - $\forall D'_t \in \mathcal{D}_t$ such that the snapshot of D'_t is $(\text{loc}, \text{con}, \text{inn})$: $\text{score}(D_t) \geq \text{score}(D'_t)$.

Recall that for the root $r \in T$, we assume $\chi(r) = \emptyset$. Hence $\mathcal{R}_r$ is a mapping from the one-element set $\{(\emptyset, \emptyset, \emptyset)\}$ to an integer τ such that τ is the maximum score that can be achieved by any DAG $D = (V, A)$ with all in-degrees of vertices upper bounded by q. In other words, $\mathcal{I}$ is a YES-instance if and only if $\mathcal{R}_r(\emptyset, \emptyset, \emptyset) \geq \ell$. To prove the theorem, it now suffices to show that the records can be computed in a leaf-to-root fashion by proceeding along the nodes of T. We distinguish four cases:

t **is a leaf node.** Let $\chi(t) = \{v\}$. By definition, $S(t) = \{(\emptyset, \emptyset, \emptyset)\}$ and $\mathcal{R}_t(\emptyset, \emptyset, \emptyset) = f_v(\emptyset)$.

t **is a forget node.** Let t' be the child of t in $\mathcal{T}$ and let $\chi(t) = \chi(t') \setminus \{v\}$. We initiate by setting $\mathcal{R}_t^0(\text{loc}, \text{con}, \text{inn}) = \perp$ for each $(\text{loc}, \text{con}, \text{inn}) \in S(t)$.

For each $(\text{loc}', \text{con}', \text{inn}') \in S(t')$, let loc_v, con_v be the restrictions of loc′, con′ to tuples containing v. We now define $\text{loc} = \text{loc}' \setminus \text{loc}_v$, $\text{con} = \text{con}' \setminus \text{con}_v$, $\text{inn} = \text{inn}'|_{\chi(t)}$ and say that $(\text{loc}, \text{con}, \text{inn})$ is *induced* by $(\text{loc}', \text{con}', \text{inn}')$. Set $\mathcal{R}_t^0(\text{loc}, \text{con}, \text{inn}) := \max(\mathcal{R}_t^0(\text{loc}, \text{con}, \text{inn}), \mathcal{R}_{t'}(\text{loc}', \text{con}', \text{inn}'))$, where $\perp$ is assumed to be a minimal element.

For correctness, it will be useful to observe that $\mathcal{D}_t = \mathcal{D}_{t'}$. Consider our final computed value of $\mathcal{R}_t^0(\text{loc}, \text{con}, \text{inn})$ for some $(\text{loc}, \text{con}, \text{inn}) \in S(t)$.

If $\mathcal{R}_t(\text{loc}, \text{con}, \text{inn}) = \tau$ for some $\tau \neq \perp$, then there exists a DAG D which witnesses this. But then D also admits a snapshot $(\text{loc}', \text{con}', \text{inn}')$ at t' and witnesses $\mathcal{R}_{t'}(\text{loc}', \text{con}', \text{inn}') \geq \tau$. Note that $(\text{loc}, \text{con}, \text{inn})$ is induced by $(\text{loc}', \text{con}', \text{inn}')$. So in our algorithm $\mathcal{R}_t^0(\text{loc}, \text{con}, \text{inn}) \geq \mathcal{R}_{t'}(\text{loc}', \text{con}', \text{inn}') \geq \tau$.

If on the other hand $\mathcal{R}_t^0(\text{loc}, \text{con}, \text{inn}) = \tau$ for some $\tau \neq \perp$, then there exists a snapshot $(\text{loc}', \text{con}', \text{inn}')$ such that $(\text{loc}, \text{con}, \text{inn})$ is induced by $(\text{loc}', \text{con}', \text{inn}')$ and $\mathcal{R}_{t'}(\text{loc}', \text{con}', \text{inn}') = \tau$. $\mathcal{R}_t(\text{loc}, \text{con}, \text{inn}) \geq \tau$ now follows from the existence of a DAG witnessing the value of $\mathcal{R}_{t'}(\text{loc}', \text{con}', \text{inn}')$.

Hence, we can correctly set $\mathcal{R}_t = \mathcal{R}_t^0$.

t **is an introduce node.** Let t' be the child of t in $\mathcal{T}$ and let $\chi(t) = \chi(t') \cup \{v\}$. We initiate by setting $\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) = \bot$ for each $(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) \in S(t)$.

For each $(\mathrm{loc}', \mathrm{con}', \mathrm{inn}') \in S(t')$ and each $Q \subseteq \{ab \in A_{\chi(t)} \mid \{a, b\} \cap \{v\} \neq \emptyset\}$, we define:

- $\mathrm{loc} := \mathrm{loc}' \cup Q$
- $\mathrm{con} := \mathtt{trcl}(\mathit{con}' \cup Q)$
- $\mathrm{inn}(x) := \mathrm{inn}'(x) + |\{y \in \chi(t) | yx \in Q\}|$ for every $x \in \chi(t) \setminus \{v\}$
  $\mathrm{inn}(v) := |\{y \in \chi(t) | yv \in Q\}|$

If con is not irreflexive or $\mathrm{inn}(x) > q$ for some $x \in \chi(t)$, discard this branch. Otherwise, let $\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) := \max(\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}), \mathtt{new})$ where $\mathtt{new} = \mathcal{R}_{t'}(\mathrm{loc}', \mathrm{con}', \mathrm{inn}') + \sum_{ab \in Q} f_b(a)$. As before, $\bot$ is assumed to be a minimal element here.

Consider our final computed value of $\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn})$ for some $(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) \in S(t)$.

For correctness, assume that $\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) = \tau$ for some $\tau \neq \bot$ and is obtained from $(\mathrm{loc}', \mathrm{con}', \mathrm{inn}'), Q$ defined as above. Then $\mathcal{R}_{t'}(\mathrm{loc}', \mathrm{con}', \mathrm{inn}') = \tau - \sum_{ab \in Q} f_b(a)$. Construct a directed graph D from the witness D' of $\mathcal{R}_{t'}(\mathrm{loc}', \mathrm{con}', \mathrm{inn}')$ by adding the arcs specified in Q. As $\mathrm{con} = \mathtt{trcl}(\mathit{con}' \cup Q)$ is irreflexive and D' is a DAG, D is a DAG as well by 13. Moreover, $\mathrm{inn}(x) \leq q$ for every $x \in \chi(t)$ and the rest of vertices have in D the same parents as in D', so $D \in \mathcal{D}_t$. In particular, $(\mathrm{loc}, \mathrm{con}, \mathrm{inn})$ is a snapshot of D in t and D witnesses $\mathcal{R}_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) \geq \mathcal{R}_{t'}(\mathrm{loc}', \mathrm{con}', \mathrm{inn}') + \sum_{ab \in Q} f_b(a) = \tau$.

On the other hand, if $\mathcal{R}_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) = \tau$ for some $\tau \neq \bot$, then there must exist a directed acyclic graph $D = (\chi_t^{\downarrow}, A)$ in $\mathcal{D}_t$ that achieves a score of τ. Let Q be the restriction of A to arcs containing v, and let $D' = (\chi_t^{\downarrow} \setminus v, A \setminus Q)$, clearly $D' \in \mathcal{D}_{t'}$. Let $(\mathrm{loc}', \mathrm{con}', \mathrm{inn}')$ be the snapshot of D' at t'. Observe that $\mathrm{loc} = \mathrm{loc}' \cup Q$, $\mathrm{con} = \mathtt{trcl}(\mathrm{con}' \cup Q)$, inn differs from inn' by the numbers of incoming arcs in Q and the score of D' is precisely equal to the score τ of D minus $\sum_{(a,b) \in Q} f_b(a)$. Therefore $\mathcal{R}_{t'}(\mathrm{loc}', \mathrm{con}', \mathrm{inn}') \geq \tau - \sum_{(a,b) \in Q} f_b(a)$ and in the algorithm $\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) \geq \mathcal{R}_{t'}(\mathrm{loc}', \mathrm{con}', \mathrm{inn}') + \sum_{(a,b) \in Q} f_b(a) \geq \tau$. Equality then follows from the previous direction of the correctness argument.

Hence, at the end of our procedure we can correctly set $\mathcal{R}_t = \mathcal{R}^0_t$.

t **is a join node.** Let t_1, t_2 be the two children of t in $\mathcal{T}$, recall that $\chi(t_1) = \chi(t_2) = \chi(t)$. By the well-known separation property of tree-decompositions, $\chi_{t_1}^{\downarrow} \cap \chi_{t_2}^{\downarrow} = \chi(t)$ [DF13, CFK+15]. We initiate by setting $\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) := \bot$ for each $(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) \in S(t)$.

Let us branch over each $\mathrm{loc}, \mathrm{con}_1, \mathrm{con}_2 \subseteq A_{\chi(t)}$ and $\mathrm{inn}_1, \mathrm{inn}_2 : \chi(t) \to [q]_0$. For every $b \in \chi(t)$ set $\mathrm{inn}(b) = \mathrm{inn}_1(b) + \mathrm{inn}_2(b) - |\{a | ab \in \mathrm{loc}\}|$. If:

- $\mathtt{trcl}(\mathrm{con}_1 \cup \mathrm{con}_2)$ is not irreflexive and/or
- $\mathcal{R}_{t_1}(\mathrm{loc}, \mathrm{con}_1, \mathrm{inn}_1) = \bot$, and/or
- $\mathcal{R}_{t_2}(\mathrm{loc}, \mathrm{con}_2, \mathrm{inn}_2) = \bot$, and/or

- $\mathrm{inn}(b) > q$ for some $b \in \chi(t)$

then discard this branch. Otherwise, set $\mathrm{con} = \texttt{trcl}(\mathrm{con}_1 \cup \mathrm{con}_2)$, $\texttt{doublecount} = \sum_{ab \in \mathrm{loc}} f_b(a)$ and $\texttt{new} = \mathcal{R}_{t_1}(\mathrm{loc}, \mathrm{con}_1) + \mathcal{R}_{t_2}(\mathrm{loc}, \mathrm{con}_2) - \texttt{doublecount}$. We then set $\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) := \max(\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}), \texttt{new})$ where $\perp$ is once again assumed to be a minimal element.

At the end of this procedure, we set $\mathcal{R}_t = \mathcal{R}^0_t$.

For correctness, assume that $\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) = \tau \neq \perp$ is obtained from loc,con_1, con_2, inn_1, inn_2 as above. Let $D_1 = (\chi^{\downarrow}_{t_1}, A_1)$ and $D_2 = (\chi^{\downarrow}_{t_2}, A_2)$ be DAGs witnessing $\mathcal{R}_{t_1}(\mathrm{loc}, \mathrm{con}_1, \mathrm{inn}_1)$ and $\mathcal{R}_{t_2}(\mathrm{loc}, \mathrm{con}_2, \mathrm{inn}_2)$ correspondingly. Note that common vertices of D_1 and D_2 are precisely $\chi(t)$. In particular, if D_1 and D_2 share an arc ab, then $a, b \in \chi(t)$ and therefore $ab \in \mathrm{loc}$. On the other hand, $\mathrm{loc} \subseteq A_1$, $\mathrm{loc} \subseteq A_2$, so $loc = A_1 \cap A_2$. Hence inn specifies the number of parents of every $b \in \chi(T)$ in $D = D_1 \cup D_2$. Rest of vertices $v \in V(D) \setminus \chi(t)$ belong to precisely one of D_i and their parents in D are the same as in this D_i. As $\texttt{trcl}(\mathrm{con}_1 \cup \mathrm{con}_2)$ is irreflexive, D is a DAG by Lemma 13, so $D \in \mathcal{D}_t$. The snapshot of D in t is $(\mathrm{loc}, \mathrm{con}, \mathrm{inn})$ and $\texttt{score}(D) = \sum_{ab \in A(D)} f_b(a) = \sum_{ab \in A_1} f_b(a) + \sum_{ab \in A_2} f_b(a) - \sum_{ab \in loc} f_b(a) = \texttt{score}(D_1) + \texttt{score}(D_2) - \texttt{doublecount} = \mathcal{R}_{t_1}(\mathrm{loc}, \mathrm{con}_1, \mathrm{inn}_1) + \mathcal{R}_{t_2}(\mathrm{loc}, \mathrm{con}_2, \mathrm{inn}_2) - \texttt{doublecount} = \tau$. So D witnesses that $\mathcal{R}_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) \geq \tau$.

For the converse, assume that $\mathcal{R}_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) = \tau \neq \perp$ and D is a DAG witnessing this. Let D_1 and D_2 be restrictions of D to $\chi^{\downarrow}_{t_1}$ and $\chi^{\downarrow}_{t_2}$ correspondingly, then by the same arguments as above $A(D_1) \cap A(D_2) = \mathrm{loc}$, in particular $D = D_1 \cup D_2$. Let $(\mathrm{loc}, \mathrm{con}_i, \mathrm{inn}_i)$ be the snapshot of D_i in t_i, $i = 1, 2$, then $\mathcal{R}_{t_i}(\mathrm{loc}, \mathrm{con}_i, \mathrm{inn}_i) \geq \texttt{score}(D_i)$. By the procedure of our algorithm, $\mathcal{R}^0_t(\mathrm{loc}, \mathrm{con}, \mathrm{inn}) \geq \mathcal{R}_{t_1}(\mathrm{loc}, \mathrm{con}_1, \mathrm{inn}_1) + \mathcal{R}_{t_2}(\mathrm{loc}, \mathrm{con}_2, \mathrm{inn}_2) - \texttt{doublecount} \geq \texttt{score}(D_1) + \texttt{score}(D_2) - \sum_{ab \in \mathrm{loc}} f_b(a) = \texttt{score}(D) = \tau$.

Hence the resulting record $\mathcal{R}_t$ is correct, which concludes the correctness proof of the algorithm.

Since the nice tree-decomposition $\mathcal{T}$ has $\mathcal{O}(n)$ nodes, the runtime of the algorithm is upper-bounded by $\mathcal{O}(n)$ times the maximum time required to process each node. This is dominated by the time required to process join nodes, for which there are at most $(2^{k^2})^3((q+1)^k)^2 = 8^{k^2} \cdot (q+1)^{2k}$ branches corresponding to different choices of $\mathrm{loc}, \mathrm{con}_1, \mathrm{con}_2, \mathrm{inn}_1, \mathrm{inn}_2$. Constructing $\texttt{trcl}(\mathrm{con}_1 \cup \mathrm{con}_2)$ and verifying that it is irreflexive can be done in time $\mathcal{O}(k^3)$. Computing $\texttt{doublecount}$ and inn takes time at most $\mathcal{O}(k^2)$. So the record for a join node can be computed in time $2^{\mathcal{O}(k^2)} \cdot q^{\mathcal{O}(k)}$. Hence, after we have computed a width-optimal tree-decomposition for instance by Bodlaender's algorithm [Bod96], the total runtime of the algorithm is upper-bounded by $2^{\mathcal{O}(k^2)} \cdot q^{\mathcal{O}(k)} \cdot n$.

Finally, to obtain the desired result for BNSL$^+$, we can simply adapt the above algorithm by disregarding the entry inn and disregard all explicit bounds on the in-degrees (e.g., in the definition of $\mathcal{D}_t$). The runtime for this dynamic programming procedure is then $2^{\mathcal{O}(k^2)} \cdot n$. □

CHAPTER 8

Conclusion

In this thesis we investigated the problem of Bayesian network structure learning. Bayesian networks are among the most prominent models for representing conditional dependencies between random variables. They constitute acyclic digraphs where vertices correspond to the variables with dependencies represented by arcs. Constructing Bayesian networks of optimal structure has a significant meaning for making conclusions and predictions based on incomplete information, such as weather forecasts or disease recognition.

There are different ways to associate to every network a score measuring how precisely a probability distribution depicted by the acyclic digraph agrees with the set of observation. We focus on a special case when the score function decomposes on local scores of single variables, given as a part of an input. Even under this assumption Bayesian Network Structure Learning is NP-hard. This fact along with crucial practical meaning gave rise to extensive research on parameterized complexity of the problem.

While previous works provided nearly exhaustive classification of complexity for parameters based on vertex deletion distances, we concentrated on the ones related to edge deletion distances. We showed that parameterization by a localised version of feedback edge number makes the problem fixed parameter tractable. Moreover, we designed a reduction procedure that allows to shrink given instances to equivalent ones with number of variables linear in feedback edge number. The results also hold for a closely related problem of Polytree Learning, where the constructed network is in addition required to be a polytree.

Local feedback edge number, introduced in this work, is a natural generalisation of feedback edge number. It might be interesting to study deeper its connections with another parameters, such as tree-cut width. We also believe that tractability of many different graph problems, known to be FPT by feedback edge number, can be lifted to lfen. Another related task that arises is to provide an effective procedure for constructing the spanning tree T of a given graph G with $\mathrm{lfen}(G, T) = \mathcal{O}(\mathrm{lfen}(G))$.

Previous hardness results and some practical implications motivated us to consider the special case of additive score functions, defined as scores per arc. Additive representation guarantees that the total input size is at most quadratic in number of vertices. In contrast to standard non-zero representation, this variant of the problem becomes FPT even if parameterized by the treewidth of the superstructure alone. One of possible directions for the further research would be to consider local score functions with properties weaker then additivity, such as monotone ones. Indeed, considerable part of score functions used in practice can be decomposed into a monotone function and some penalty depending on parent set size only.

Bibliography

[BDD+16] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.

[Bod96] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[CFK+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Chi96] David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Learning from data (Fort Lauderdale, FL, 1995)*, volume 112 of *Lecture Notes in Statist.*, pages 121–130. Springer Verlag, 1996.

[Das99] Sanjoy Dasgupta. Learning polytrees. In Kathryn B. Laskey and Henri Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 134–141. Morgan Kaufmann, 1999.

[DF13] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[EG08] Gal Elidan and Stephen Gould. Learning bounded treewidth bayesian networks. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 417–424. Curran Associates, Inc., 2008.

[EGKS19] Eduard Eiben, Robert Ganian, Iyad Kanj, and Stefan Szeider. The parameterized complexity of cascading portfolio scheduling. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox,

and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7666–7676, 2019.

[GK20] Niels Grüttemeier and Christian Komusiewicz. Learning bayesian networks under sparsity constraints: A parameterized complexity analysis. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4245–4251. ijcai.org, 2020.

[GKL+15] Serge Gaspers, Mikko Koivisto, Mathieu Liedloff, Sebastian Ordyniak, and Stefan Szeider. On finding optimal polytrees. *Theor. Comput. Sci.*, 592:49–58, 2015.

[GKM21] Niels Grüttemeier, Christian Komusiewicz, and Nils Morawietz. On the parameterized complexity of polytree learning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI-21)*, 2021. To appear; preprint available at https://arxiv.org/abs/2105.09675.

[GKO21] Robert Ganian, Fabian Klute, and Sebastian Ordyniak. On structural parameterizations of the bounded-degree vertex deletion problem. *Algorithmica*, 83(1):297–336, 2021.

[GKOS18] Robert Ganian, Iyad A. Kanj, Sebastian Ordyniak, and Stefan Szeider. Parameterized algorithms for the matrix completion problem. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1642–1651. PMLR, 2018.

[GO18] Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artif. Intell.*, 257:61–71, 2018.

[Klo94] T. Kloks. *Treewidth: Computations and Approximations*. Springer Verlag, Berlin, 1994.

[KP13] 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 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, volume 31 of *JMLR Workshop and Conference Proceedings*, pages 370–378. JMLR.org, 2013.

[KP15] Janne H. Korhonen and Pekka Parviainen. Tractable bayesian network structure learning with bounded vertex cover number. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual*

Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pages 622–630, 2015.

[Mee01] Christopher Meek. Finding a path is harder than finding a tree. *J. Artif. Intell. Res.*, 15:383–389, 2001.

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford, 2006.

[OS13] Sebastian Ordyniak and Stefan Szeider. Parameterized complexity results for exact bayesian network structure learning. *J. Artif. Intell. Res.*, 46:263–302, 2013.

[SCdCZ16] Mauro Scanagatta, Giorgio Corani, Cassio P. de Campos, and Marco Zaffalon. Learning treewidth-bounded bayesian networks with thousands of variables. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1462–1470, 2016.

[SdCCZ15] Mauro Scanagatta, Cassio P. de Campos, Giorgio Corani, and Marco Zaffalon. Learning bayesian networks with thousands of variables. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1864–1872, 2015.

[SFGP19] Kirill Simonov, Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. Refined complexity of PCA with outliers. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5818–5826. PMLR, 2019.

[SMS13] Javad Safaei, Ján Manuch, and Ladislav Stacho. Learning polytrees with constant number of roots from data. In Stephen Cranefield and Abhaya C. Nayak, editors, *AI 2013: Advances in Artificial Intelligence - 26th Australasian Joint Conference, Dunedin, New Zealand, December 1-6, 2013. Proceedings*, volume 8272 of *Lecture Notes in Computer Science*, pages 447–452. Springer, 2013.