Philipps Universität Marburg

Parameterized Algorithmics for Graph-Based Data Analysis

DISSERTATION
FOR THE DEGREE OF DOCTOR OF NATURAL SCIENCES
(DR. RER. NAT.)

Submitted by
Niels Grüttemeier, M.Sc.
born September 03, 1990 in Detmold, Germany.

Referees:
Prof. Dr. Christian Komusiewicz, Philipps-Universität Marburg, Germany
Dr. Sebastian Ordyniak, Lecturer, University of Leeds, United Kingdom
Prof. Dr. Charis Papadopoulos, University of Ioannina, Greece

Examination Committee:
Prof. Dr. Hajo Holzmann (Chairman)
Prof. Dr. Christian Komusiewicz
Dr. Sebastian Ordyniak
Prof. Dr. Gabriele Taentzer
Prof. Dr. Bernhard Seeger

Submitted on November 25, 2021
Defended on February 2, 2022
Philipps-Universität Marburg (1180)
Fachbereich 12 – Mathematik und Informatik
Marburg, Germany, 2022

Grüttemeier, Niels
Parameterized Algorithmics for Graph-Based Data Analysis
Dissertation, Philipps-Universität Marburg, 2021.

Curriculum vitae

- Since October 2017: Member of the *Algorithmics* group
  Philipps-Universität Marburg, Germany
- October 2014 – September 2017: M.Sc. in *Mathematics*
  Friedrich-Schiller-Universität Jena, Germany.
- October 2011 – September 2014: B.Sc. in *Mathematics*
  Friedrich-Schiller-Universität Jena, Germany.

Preface

This thesis is a summary of my work on two families of NP-hard graph problems that find application in social network analysis and in the field of artificial intelligence. The research for this work was performed from June 2017 to September 2021. From June 2017 until September 2017, I worked as student research assistant in the group of Christian Komusiewicz at Friedrich-Schiller-Universität Jena. In October 2017, Christian Komusiewicz moved from Friedrich-Schiller-Universität Jena to Philipps-Universität Marburg and I became a research assistant in his group in Marburg.

First and foremost, I want to thank my doctoral supervisor Christian Komusiewicz for giving me the opportunity to work in his group, for proposing interesting problems to study, for his support, and for many fruitful discussions helping me to improve my understanding of the field significantly. Furthermore, I want to thank my colleagues and friends Nils Morawietz, Frank Sommer, and Jaroslav Garvardt for creating a wonderful working atmosphere, and for the pleasant and productive cooperation. I also want to express my gratitude to my coauthors Laurent Bulteau, Nils Jakob Eckstein, Christian Komusiewicz, Nils Morawietz, Jannik Schestag, Frank Sommer, and Manuel Sorge. Finally, I want to thank my parents Ralf Grüttemeier and Andrea Grüttemeier, whose emotional and financial support made it possible for me to study at the university.

The chapters of this thesis are based on journal and conference publications which were created in close collaboration with coauthors. Below, I provide an overview on which publication contributed to which chapter. I will also specify the contributions of all coauthors to these publications. Besides the publications that are part of this thesis, I have contributed to papers about edge deletion problems in edge-colored graphs [82, 48], graph problems that aim to prevent small cuts in a graph [81, 134, 135], and on string factorization problems [80].

Part II: Strong Triadic Closure. Chapter 2 is based on parts of the publication "On the Relation of Strong Triadic Closure and Cluster Deletion", which appeared in *Algorithmica* [77]. A preliminary version of this publication appeared in *Proceedings of the 44th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '18)* [75]. Christian Komusiewicz proposed to study the two problems Strong Triadic Closure and Cluster Deletion. Both coauthors participated in the development of the dichotomy on H-free graphs, the fixed-parameter algorithms for the parameter ℓ and the kernel lower bound. I worked out the details and prepared the draft of the dichotomy and the kernel lower bound. Christian Komusiewicz prepared the draft of the fixed-parameter algorithms.

Chapter 3 and Chapter 4 are based on the publication "Your rugby mates don't need to know your colleagues: Triadic closure with edge colors", which appeared in *Journal of Computer and System Sciences* [20]. A preliminary version of this publication appeared in *Proceedings of the 11th International Conference on Algorithms and Complexity (CIAC '19)* [19]. I proposed to study Multi-STC and its generalizations at the 7th annual research retreat of the Algorithmics and Computational Complexity group of TU Berlin, Darlingerode, Germany, March 18th–23rd, 2018. The results about the classic and the fine-grained complexity were developed jointly by all coauthors. Christian Komusiewicz worked out the details of the $3^m \cdot n^{\mathcal{O}(1)}$ time algorithm. I prepared the draft and worked out the technical details of the ETH-based lower bound. The fixed-parameter algorithm and the problem kernelization for k_1 were developed jointly by Christian Komusiewicz and me. I prepared the draft and worked out the technical details. Furthermore, I showed the W[1]-hardness for VL-Multi-STC.

Chapter 5 is based on parts of the publication "Maximum Edge-Colorable Subgraph and Strong Triadic Closure Parameterized by Distance to Low-Degree Graphs", which appeared in *Proceedings of the 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT '20)* [78]; a full version of this publication is planned. Nils Morawietz and I developed the problem kernelization for ECS parameterized by ξ_{c-1}. The problem kernelization for ECS parameterized by λ_c and the problem kernelization for Multi-STC parameterized by $\xi_{\lfloor \frac{c}{2} \rfloor + 1}$ were jointly developed by all coauthors. I prepared the draft and worked out the technical details of all results contained in this chapter.

Part III Bayesian Network Structure Learning. Parts of Chapter 6 are based on the publication "Learning Bayesian Networks Under Sparsity Constraints: A Parameterized Complexity Analysis", which appeared in *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)* [76]; a full version of this publication is currently under review. Christian Komusiewicz proposed to study Bayesian Network Structure Learning. The XP algorithms for $(\Pi_0 + v)$-Skeleton BNSL, $(\Pi_1 + v)$-Moral BNSL, the FPT algorithm for $(\Pi_0 + e)$-Skeleton BNSL, and the hardness results from Section 6.4 were jointly developed by both coauthors. The W[2]-hardness for $(\Pi_0 + v)$-Skeleton BNSL was discovered in a discussion by Nils Morawietz and me. I developed the kernel lower bound for Vanilla-BNSL and the W[1]-hardness results for $(\Pi_0 + e)$-Moral BNSL and $(\Pi_F + e)$-Moral BNSL. I prepared the draft for all results in this chapter.

Chapter 7 is based on the publication "On the Parameterized Complexity of Polytree Learning", which appeared in *Proceedings of the 30th International Joint*

Conference on Artificial Intelligence (IJCAI '21) [83]; a full version is planned. I proposed to study Polytree Learning. The FPT algorithm for parameterization by the number n of vertices was jointly developed by all coauthors. Nils Morawietz and I developed the kernel lower bound for n and the W[1]-hardness for parameterization by the number d of dependent vertices. Nils Morawietz worked out the technical details and prepared the original draft of all of these results. Christian Komusiewicz proposed to consider the technique of computing representative sets in a matroid as an approach to solve Polytree Learning. I developed the FPT algorithm for $d+p$ and the problem kernel and prepared the draft for these two results.

Chapter 8 is based on the publication "Efficient Bayesian Network Structure Learning via Parameterized Local Search on Topological Orderings", which appeared in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '21)* [79]. Christian Komusiewicz proposed to study ordering-based local search. He also implemented a local search algorithm and performed the preliminary experiments. All theoretical results were jointly developed by all coauthors. Nils Morawietz worked out the technical details and prepared the draft of the polynomial-time algorithm for W-BNSL when the directed superstructure is a DAG and the W[1]-hardness proofs for Insert-Local W-BNSL and Swap-Local W-BNSL. I worked out the technical details and prepared the draft of the FPT algorithms for Inv-Local W-BNSL and InvWin-Local W-BNSL.

Abstract

In this thesis we investigate the computational complexity of two families of graph problems with applications in social network analysis and artificial intelligence. We analyze the classic, fine-grained, and parameterized complexity of the considered problems.

Social networks can be modeled with the help of undirected graphs, in which the vertices correspond to the agents in the network and an edge between two vertices corresponds to a relationship or an interaction between two agents. One task in social network analysis is to classify the relationships into *strong* and *weak* relationships, if only the graph structure of the social network is known. In the computational problem Strong Triadic Closure (STC) we are given an undirected graph G and an integer k and aim to label the edges of G as *strong* or *weak* such that at most k edges are weak and G contains no induced P_3 with two strong edges. We investigate the classic and parameterized complexity of STC. We also study a version of STC with multiple strong relationship types (Multi-STC) and introduce further generalizations where—for example—one may use vertex lists (VL-Multi-STC) to restrict the set of possible strong relationship types incident with each vertex. We show that under the Exponential Time Hypothesis (ETH), VL-Multi-STC cannot be solved in time $2^{o(|V|^2)}$. We then proceed with a parameterized complexity analysis of Multi-STC and its generalizations. For example, we provide a problem kernel for Multi-STC parameterized by an edge deletion distance to low-degree graphs.

A *Bayesian network structure* is a directed acyclic graph, where the vertices correspond to random variables and the arcs correspond to conditional dependencies. We study the algorithmic task of learning an optimal network structure from observed data using a score-based approach. Extending previous work, we analyze the parameterized complexity of learning an optimal network structure when additional constraints are posed on the network structure or on its moralized graph. For example, we show that learning an optimal network whose moralized graph has dissociation number at most k can be done in polynomial time for constant k. Furthermore, we analyze the parameterized complexity of learning a good network structure where the underlying undirected graph is acyclic. This problem variant is known as Polytree Learning. Finally, we study parameterized local search algorithms for learning a network structure. We consider the ordering-based approach where a network structure is represented by a topological ordering. For a given integer r and a pre-defined distance function on the space of orderings, we aim to find an optimal ordering that has distance at most r to a given ordering. We analyze the parameterized complexity for r with regard to four natural distance functions.

Zusammenfassung

In dieser Dissertation untersuchen wir die Berechnungskomplexität zweier Familien von Graphproblemen, die Anwendung im Bereich der Analyse sozialer Netzwerke und im Bereich der künstlichen Intelligenz finden. Wir analysieren die klassische, fine-grained und parametrisierte Komplexität.

Soziale Netzwerke können mit Hilfe ungerichteter Graphen modelliert werden, wobei die Knoten den Agenten und die Kanten den Beziehungen oder Interaktionen entsprechen. Eine Aufgabe in der Analyse sozialer Netzwerke ist es, Beziehungen zwischen Agenten als *stark* oder *schwach* zu klassifizieren, wenn nur die Netzwerkstruktur gegeben ist. In Strong Triadic Closure (STC) ist die Eingabe ein Graph G und eine ganze Zahl k. Das Ziel ist es, die Kanten in G als *stark* oder *schwach* zu klassifizieren, sodass G keinen induzierten P_3 mit zwei starken Kanten enthält. Wir untersuchen die klassische und die parametrisierte Komplexität von STC und einer STC-Version mit mehreren starken Verbindungstypen (Multi-STC) und führen weitere Verallgemeinerungen des Problems ein, in denen etwa Knotenlisten benutzt werden können um mögliche starke Verbindungstypen an einem Knoten einzuschränken (VL-Multi-STC). Wir zeigen, dass VL-Multi-STC unter der Annahme der Exponential Time Hypothesis (ETH) nicht in $2^{o(|V|^2)}$ Zeit gelöst werden kann. Bezüglich der parametrisierten Komplexität liefern wir etwa einen Problemkern für Multi-STC parametrisiert durch einen Kantenlöschungsparameter, der die Distanz von G zu einem Graphen mit niedrigem Maximalgrad misst.

Eine *bayessche Netzwerkstruktur* ist ein gerichteter Graph, wobei die Knoten Zufallsvariablen und die Kanten bedingten Abhängigkeiten entsprechen. Wir untersuchen die algorithmische Aufgabe eine optimale Netzwerkstruktur aus gemessenen Daten mit Hilfe eines score-basierten Ansatzes zu lernen. In Anknüpfung an bestehende Forschung untersuchen wir die parametrisierte Komplexität des Lernens einer Netzwerkstruktur, wenn zusätzliche Bedingungen an das Netzwerk oder an den moralisierten Graph gestellt werden. Wir zeigen beispielsweise, dass das Lernen einer Netzwerkstruktur, deren Moralgraph eine Dissociation Number von höchstens k hat, in polynomialzeit lösbar ist, falls k eine Konstante ist. Weiterhin analysieren wir die parametrisierte Komplexität vom Lernen einer Netzwerkstruktur, deren unterliegender, ungerichteter Graph azyklisch ist. Diese Problemversion ist auch als Polytree Learning bekannt. Schließlich betrachten wir parametrisierte lokale Suche für das Lernen von bayesschen Netzwerkstrukturen. Wir verwenden dabei einen ordnungsbasierten Ansatz, in dem eine Netzwerkstruktur durch ihre topologische Sortierung repräsentiert wird. Für eine gegebene Zahl r und eine vordefinierte Distanzfunktion auf dem Raum der topologischen Sortierungen geht es darum, eine optimale Sortierung mit Distanz höchstens r zur gegebenen Sortierung zu finden. Wir analysieren die parametrisierte Komplexität für den Parameter r für vier natürliche Distanzfunktionen.

Contents

Part I

Introduction

Many complex systems can be modeled as *graphs*, which are mathematical structures to model relations between objects. Roughly speaking, a graph is a collection of objects (called *vertices*) together with a collection of pairwise relations (called *edges* or *arcs*) between the vertices. With this simple structure at hand, we are able to describe complex systems with thousands or millions of objects. Graphs can model biological processes [130], links between web pages [169], friendships between agents in a social network [72], or mathematical relations like conditional dependencies of random variables [145]. It is thus well-motivated to study computational problems that aim to extract informations from given graphs or to find graph structures representing a given set of data. We often face the algorithmic challenge that many such problems are NP-hard, which indicates that these problems are presumably not solvable within polynomial time in the size of the given input [67]. Algorithms that solve these problems have a running time that is superpolynomial in the size of the input. For large instances, a superpolynomial—for example exponential—running time in the input size might become impractical.

To deal with the NP-hardness we follow the approach of *parameterized algorithmics* [38, 46, 58, 139]. In this approach we aim to compute an exact solution in a running time that is strongly influenced by some structural parameters of the input instance. In other words, we analyze the running time of algorithms with a function depending on multiple parameters instead of only the total input size. Some of these parameters have a crucial influence on the running time if the running time—for example—contains a factor that grows exponentially in the size of the parameter. For other parameters, the influence might be only polynomial. Given an instance where parameters that have a crucial influence on the running time are small, the proposed algorithms might be an efficient approach to solve this instance. In this work, we study parameterized algorithmics for two families of computational problems concerning graph-based data analysis.

Social networks are graphs that represent relationships between agents. For example, every profile of an online social network can be modeled as a vertex and two vertices are connected by an edge if the corresponding users are friends with each other. Modern social networks may have a high number of agents and analyzing these networks requires efficient algorithms. Strong Triadic Closure is the task of determining the strength of relationships in a social network [164] by using a structural notion that was established in sociological work [72, 73]. In the first part of this work, we analyze the classic and parameterized complexity of Strong Triadic Closure and multiple generalizations of this problem.

In the second part of this work, we study the task of learning the structure of *Bayesian networks* [143]. Bayesian networks are a versatile tool in artificial intelligence that enable machines to reason probabilistically [114]. A Bayesian network is a compact representation of a multivariate probability distribution. One important part of a Bayesian network is its so-called *network structure*, which is a graph representing conditional dependencies between random variables. This graph can be built from observational data. More precisely, given a data set containing multiple instantiations of the random variables, one aims to compute a network structure that has a good trade-off between matching the data set and providing querying efficiency [160]. In this work, we study multiple variants of the task of finding good network structures.

The aim of this work is to provide a (parameterized) complexity overview for these two families of practically motivated NP-hard problems. In this overview, we provide negative and positive results for the studied problems. The negative results are intractability results that rely on standard complexity theoretic assumptions. These lower bounds can be seen as a guide in the algorithm design process. The positive results are concrete efficient algorithms to solve the problems or reduction rules for an efficient pre-processing of input instances. While the presented results are purely theoretic, we believe that some positive results have a practical potential. Experimental evaluations are left open for future work.

Chapter 1

Preliminaries

In this chapter, we give an overview of the central definitions, notations, and concepts that we use in this work.

Given a set S, we let 2^S denote the power set of S. We let $\mathbb{N}$ denote the set of positive integers and we let $\mathbb{N}_0$ denote the set of non-negative integers. Given a mapping $f : A \to B$ and a subset $A' \subseteq A$, we let $f|_{A'} : A' \to B$ with $f|_{A'}(a) := f(a)$ denote the *restriction of f to A'*. Given a set S, we call a family $\mathcal{S} := \{S_1, \ldots, S_t\}$ of subsets of S a *partition of S* if $S_i \cap S_j = \emptyset$ for $i \neq j$ and $\bigcup_{i \in \{1,\ldots t\}} S_i = S$. A *finite sequence of length* r is an r-tuple $A := (a_1, \ldots, a_r)$ of specific elements. For given $j \in \{1, \ldots, r\}$, we refer to the jth element on A as $A(j)$.

1.1 Graph Theory

We consider undirected and directed graphs. In the following, we describe the main concepts and notations of graphs that we use in this work.

1.1.1 Undirected Graphs

A *simple undirected graph* is a tuple $G := (V, E)$ with a vertex set V and an edge set $E \subseteq \{\{u, v\} \subseteq V \mid u \neq v\}$. If not stated otherwise, we set $n := |V|$ and $m := |E|$. Two vertices $u \in V$ and $v \in V$ are *adjacent* if $\{u, v\} \in E$. A vertex $v \in V$ is *incident* with an edge $e \in E$ if $v \in e$. Two edges e_1 and e_2 are *incident* if $|e_1 \cap e_2| = 1$. Given a vertex $v \in V$, we define $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$ as the *open neighborhood of v* and $N_G[v] := N_G(v) \cup \{v\}$ as the *closed neighborhood of v*. Given a vertex set $V' \subseteq V$, we define $N_G(V') := \bigcup_{v \in V'} N_G(v)$. We define $\deg_G(v) := |N_G(v)|$ as the degree of v. We say G *has maximum degree* Δ for some $\Delta \in \mathbb{N}_0$ if $\deg_G(v) \leq \Delta$

for all $v \in V$. Given two vertex sets $V_1 \subseteq V$ and $V_2 \subseteq V$, we let $E_G(V_1, V_2) := \{\{v_1, v_2\} \in E \mid v_1 \in V_1 \text{ and } v_2 \in V_2\}$ denote the set of edges between V_1 and V_2. As a shorthand, we set $E_G(V_1) := E_G(V_1, V_1)$. For all notations we may omit the subscript G if the graph is clear from the context.

Given a set $V' \subseteq V$, we let $G[V'] := (V', E_G(V'))$ denote the *subgraph of G induced by V'* and $G - V' := G[V \setminus V']$ denote the graph obtained from G after *removing the vertices of V'*. A *clique* is a vertex set $K \subseteq V$ such that in $G[K]$ the vertices are pairwise adjacent. An *independent set* is a vertex set $I \subseteq V$ such that in $G[I]$ no two vertices are adjacent. A *vertex cover* is a vertex set $S \subseteq V$ such that $S \cap e \neq \emptyset$ for every edge $e \in E$. A graph is a *bounded-vc graph* for some given integer k if a minimum vertex cover has size at most k. A *matching* is an edge set $M \subseteq E$ where $e_1 \cap e_2 = \emptyset$ for every pair $e_1, e_2 \in M$ with $e_1 \neq e_2$. A matching M is *maximal* if there is no matching $M' \supsetneq M$ and *maximum* if its size $|M|$ is maximal among all matchings.

Two graphs $G_1 := (V_1, E_1)$ and $G_2 := (V_2, E_2)$ are *isomorphic* if there exists a bijective function $f : V_1 \to V_2$ such that $\{u, v\} \in E_1$ if and only if $\{f(u), f(v)\} \in E_2$ for every $u, v \in V_1$. Given another graph H, we say that G *contains H as an induced subgraph* if there is a vertex set $V' \subseteq V$ such that $G[V']$ and H are isomorphic. The graph G is called H-free if G does not contain H as an induced subgraph.

Let $G = (V, E)$ be an undirected graph. A *path (of length r) in G* is a finite sequence $P := (v_1, \dots, v_r)$ of vertices of G, where $\{v_i, v_{i+1}\} \in E$ for all $i \in \{1, \dots, r-1\}$. The path P is called *vertex simple* if no vertex appears twice on P and *edge simple* if there are no distinct $i, j \in \{1, \dots, r-1\}$ such that $\{v_i, v_{i+1}\} = \{v_j, v_{j+1}\}$. Given a path P we define the sets $V(P) := \{v_i \mid i \in \{1, \dots, r\}\}$ and $E(P) := \{\{v_i, v_{i+1}\} \mid j \in \{1, \dots, r-1\}\}$. Two vertices u and v are called *connected in G* if there exists a path P in G with $u \in V(P)$ and $v \in V(P)$. A *connected component of G* is a maximal subset $V' \subseteq V$ such that all vertices in V' are pairwise connected.

A *graph class* Π is a set of undirected graphs. For a graph class Π and $k \in \mathbb{N}_0$, we let $\Pi + kv := \{G = (V, E) \mid \exists V' \subseteq V : (|V'| \leq k \wedge G - V' \in \Pi)\}$ denote the class of graphs that can be transformed into a graph in Π by performing at most k vertex deletions. Analogously, we let $\Pi + ke := \{G = (V, E) \mid \exists E' \subseteq E : (|E'| \leq k \wedge (V, E \setminus E') \in \Pi)\}$ denote the class of graphs that can be transformed into a graph in Π by performing at most k edge deletions. We call Π *monotone* if Π is closed under edge deletions and vertex deletions. Note that Π being monotone implies that for every $k \in \mathbb{N}_0$, the graph classes $\Pi + kv$ and $\Pi + ke$ are monotone.

Throughout this work we refer to some specific small graphs with their commonly used names. An overview of the small graphs used in this work is shown in Figure 1.1.

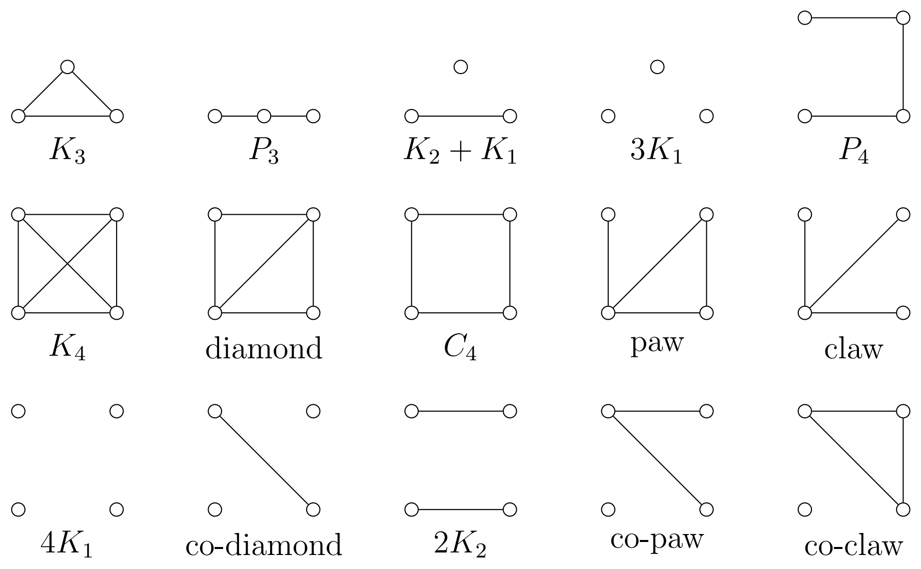


Figure 1.1: Some small graphs considered in this work.

1.1.2 Directed Graphs

A directed graph $D = (N, A)$ consists of a *vertex set* N and an *arc set* $A \subseteq N \times N$. If not stated otherwise, we set $n := |N|$. Given a set $P \subseteq N$ and a vertex $v \in N$, we define $P \times v := P \times \{v\}$ as a shorthand. The *skeleton of* D is the undirected graph $\mathcal{S}(D) := (N, E)$, with $E := \{\{u, v\} \mid (u, v) \in A\}$.

A *directed path* (of length r) is a finite sequence $P = (v_1, \dots, v_r)$ of vertices, where $(v_i, v_{i+1}) \in A$ for every $i \in \{1, \dots, r-1\}$. A directed path is called *directed cycle* if $v_1 = v_r$. If D does not contain directed cycles, then D is a *directed acyclic graph* (DAG). In a DAG D, we call a vertex v_1 an *ancestor of* v_r and v_r a *descendant of* v_1 if there is a directed path $(v_1, v_2, \dots, v_r)$ in D. A *topological ordering* of D is an n-tuple $(v_1, \dots, v_n)$ containing every vertex of D exactly once such that $i \leq j$ for every arc $(v_i, v_j) \in A$. A directed graph has a topological ordering if and only if it is a DAG.

An arc $(u, v) \in A$ is called *incoming arc into* v and *outgoing arc from* u. Given a vertex v, the number of incoming arcs into v is called the *in-degree of* v, and the number of outgoing arcs from v is the *out-degree of* v. A vertex without incoming arcs is a *source*. A vertex without outgoing arcs is a *sink*. The set $P_v^A := \{u \in N \mid (u, v) \in A\}$ is called *parent set of* v. The vertices in P_v^A are called *parents of* v and for every $u \in P_v^A$, the vertex v is called *child of* u.

1.2 Computational Complexity

We study computational problems that arise from practical motivations. In context of the practical motivation, these problems are optimization problems where one aims to find a feasible solution that minimizes or maximizes some target function. In a complexity-theoretic framework, problems are defined as decision problems via formal languages. In a decision problem, the task is to decide whether a given instance is a yes-instance or a no-instance of the problem. For every optimization problem, there is a corresponding decision problem, where the input consists of an additional integer k and the question is, whether there is a feasible solution under which the target function is at least (or at most) k. The optimization problem for an instance I can then be solved by finding the largest (or smallest) integer k, for which the tuple (I, k) is a yes-instance of the corresponding decision problem.

We use classic complexity, fine-grained complexity, and parameterized complexity. We use *classic complexity* as a framework which allows us to distinguish between problems that can be solved in polynomial time and problems that can not be solved in polynomial time when assuming the famous hypothesis P $\neq$ NP. The *fine-grained complexity* allows us to show even stronger lower bounds for problems which are based on a stronger hypothesis called *Exponential Time Hypothesis* (ETH) [96]. The *parameterized complexity* is a framework which allows us to analyze the complexity of a problem with regard to multiple parameters instead of only the input size.

Throughout this work, we analyze the running times of the algorithms using the *random-access machine* (RAM) model of computation as it is described in the textbook by Cormen et al. [35].

1.2.1 Classic Complexity

A *decision problem* is formally defined as a language $L \subseteq \Sigma^*$ where Σ is a finite alphabet. Given an instance $x \in \Sigma^*$, the task is to decide whether $x \in L$ or $x \notin L$. An instance x is a *yes-instance* if $x \in L$. Otherwise, x is a *no-instance.*

Given an algorithm that decides if $x \in L$, we measure the *running time* of the algorithm as a function of the input size $|x|$ for all $x \in \Sigma^*$. The algorithm *runs in polynomial time* if the corresponding function of the running time is a polynomial in $|x|$. A *complexity class* is a set of decision problems. The most important complexity classes are called P and NP. A decision problem L belongs to the complexity class P if there exists an algorithm that decides L and runs in polynomial time. A decision problem L belongs to the complexity class NP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time algorithm A such that for every $x \in \Sigma^*$ it holds that $x \in L$ if and only if there exists a *certificate* $u \in \Sigma^{p(|x|)}$ and A accepts the input tuple (x, u).

By the definition of P and NP it is easy to see that P $\subseteq$ NP. A widely believed conjecture in computer science is P $\neq$ NP. This conjecture means that there are problems in NP that can not be decided in polynomial time. Some of these problems that can presumably not be decided in polynomial time are identified via polynomial-time reductions. Given two problems $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$, a *polynomial-time reduction* from L_1 to L_2 is a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that $x \in L_1$ if and only if $f(x) \in L_2$ for every $x \in \Sigma^*$. A problem L is called NP-*hard* if there is a polynomial-time reduction from L' to L for every $L' \in$ NP. An NP-hard problem is NP-*complete* if it belongs to the class NP. Assuming P $\neq$ NP, an NP-hard problem L can not be solved in polynomial time since otherwise every $L' \in$ NP can be decided in polynomial time: given an instance x for L', first compute an equivalent instance $f(x)$ using the reduction from L' to L, and then use a polynomial-time algorithm to decide whether $f(x) \in L$. For a detailed introduction into the theory of NP-completeness we refer the reader to the standard textbook by Garey and Johnson [67].

In an *optimization problem*, for every instance I there is a set S_I of *feasible solutions* and a polynomial-time computable objective function that assigns a non-negative integer to every $s \in S_I$. Given an instance I, the task is to find a feasible solution that maximizes the objective function. Given a constant $c > 0$, a factor-c approximation is an algorithm that finds a feasible solution in polynomial time whose objective function value is at least c times the optimum (in case of maximization) or at most c times the optimum (in case of minimization). For a detailed introduction into the theory of approximation algorithms we refer the reader to the textbook by Vazirani [176]. The algorithms presented in this work are formulated to solve decision problems. However, the algorithms usually also solve the natural optimization problem that corresponds to the decision problem.

1.2.2 Parameterized Complexity

We study the parameterized complexity of NP-hard problems. In the following, we describe the main concepts and definitions of parameterized algorithmics. For a detailed introduction, we refer to the standard textbooks [38, 46, 58, 139].

A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}_0$ where Σ is a finite alphabet. Given an instance (x, k) we refer to k as the *parameter*. An instance $(x, k) \in \Sigma^* \times \mathbb{N}_0$ is a *yes-instance* of L if $(x, k) \in L$. Otherwise, (x, k) is a *no-instance*. With the next two definitions we define two classes in parameterized complexity that we consider in this work.

Definition 1.1. *A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}_0$ is called* slice-wise polynomial (XP) *if there exist computable functions $f : \mathbb{N}_0 \to \mathbb{N}_0$, $g : \mathbb{N}_0 \to \mathbb{N}_0$, and an algorithm that, given an instance (x, k) decides whether $(x, k) \in L$ in $f(k) \cdot |(x, k)|^{g(k)}$ time. The complexity class* XP *is the class of all parameterized problems that are slice-wise polynomial.*

Definition 1.2. *A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}_0$ is called* fixed-parameter tractable (FPT) *if there exists a computable function $f : \mathbb{N} \to \mathbb{N}_0$ and an algorithm that, given an instance (x, k) decides whether $(x, k) \in L$ in $f(k) \cdot |(x, k)|^{\mathcal{O}(1)}$ time. The complexity class* FPT *is the class of all parameterized problems that are fixed-parameter tractable.*

For constant values of the parameter k, problems in XP and problems in FPT can both be solved within polynomial time. For a problem that is XP, the degree of the polynomial depends on the value of k. In contrast, if the problem is FPT, the degree is independent from k. Proving that a parameterized problem is FPT by providing efficient algorithms with running time $f(k) \cdot |(x, k)|^{\mathcal{O}(1)}$ is arguably one of the major goals in the field parameterized complexity.

The W-Hierarchy. There are parameterized problems in XP that resist all attempts to find fixed-parameter algorithms. Like the widely believed assumption that P $\neq$ NP, there is a similar assumption about parameterized complexity classes. Downey and Fellows [44, 45, 46] developed a theory to show that some problems in XP are unlikely to be in FPT. For every integer $i \in \mathbb{N}$, they defined a complexity class W[i] and a widely believed assumption states that FPT $\subsetneq$ W[1] $\subsetneq$ W[2] $\subsetneq \cdots \subsetneq$ XP.

Let $i \in \mathbb{N}$. To show that a problem is presumably not fixed-parameter tractable one can show W[i]-hardness which is defined by parameterized reductions. Let $L_1 \subseteq \Sigma^* \times \mathbb{N}$ and $L_2 \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. A *parameterized reduction* from L_1 to L_2 is a computable function $f : \Sigma^* \times \mathbb{N}_0 \to \Sigma^* \times \mathbb{N}_0$ that maps an instance (x, k) of L_1 to an instance (x', k') of L_2 such that

- $(x, k) \in L_1$ if and only if $(x', k') \in L_2$,
- $k' \leq g(k)$ for some computable function g, and
- f can be computed in $h(k) \cdot |(x, k)|^{\mathcal{O}(1)}$ time for some computable function h.

A parameterized problem L is W[i]-hard for some $i \in \mathbb{N}$ if there is a polynomial reduction from L' to L for every $L' \in$ W[i]. If a problem is W[i]-hard it is assumed to be not fixed-parameter tractable. Note that a parameterized reduction runs in $h(k) \cdot |(x,k)|^{\mathcal{O}(1)}$ time by definition. However, all parameterized reductions contained in this work run in $|(x,k)|^{\mathcal{O}(1)}$ time and thus, we may conclude NP-hardness from our W[1]-hardness results.

Problem Kernelization. An important tool in the development of parameterized algorithms is problem kernelization, which is a polynomial-time preprocessing by data reduction rules yielding a problem kernel. Formally, this is defined as follows.

Definition 1.3. *A parameterized problem L* admits a (problem) kernel *if there is a polynomial-time algorithm that, given an instance (x,k) of L, computes an equivalent instance (x',k') of L such that $|x'| + k' \leq g(k)$ for some computable function g. The function g is called the* kernel size*. If g is a polynomial, we say that L* admits a polynomial kernel*.*

Problem kernels are often obtained by performing data reduction on an input instance. A data *reduction rule* is an algorithm that transforms an instance (x,k) of a problem L into an instance (x',k') of the same problem L such that (x,k) is a yes-instance if and only if (x',k') is a yes-instance. A set of data reduction rules has been *exhaustively applied* on an instance if no further application of one of the rules changes the instance. An instance on which the set of reduction rules are exhaustively applied is called a *reduced instance*.

Some parameterized problems are fixed-parameter tractable but presumably do not admit a polynomial kernel [38]. By using *polynomial parameter transformations* we can transfer these kernel lower bounds to other problems [17]. A polynomial parameter transformation maps any instance (x,k) of some parameterized problem L in polynomial time to an equivalent instance (x',k') of a parameterized problem L' such that $k' \leq p(k)$ for some polynomial p. All kernel lower bounds contained in this work rely on the widely believed assumption that NP $\not\subseteq$ coNP/poly.

Throughout this work we use the following problems as auxiliary problems to transfer such kernel lower bounds via polynomial parameter transformations.

Multicolored Clique
Input: A graph $G = (V,E)$ with a partition $(V_1, \ldots, V_t)$ of V such that each V_i is an independent set.
Question: Is there a multicolored clique in G, that is, a clique containing one vertex from each set V_i?

Multicolored Independent Set
Input: A graph $G = (V, E)$ with a partition $(V_1, \ldots, V_t)$ of V such that each V_i is an independent set.
Question: Is there a multicolored independent set in G, that is, an independent set containing one vertex from each set V_i?

Proposition 1.4. *The problems* Multicolored Clique *and* Multicolored Independent Set *do not admit a polynomial kernel when parameterized by* $\sum_{i=1}^{t-1} |V_i|$ *unless* NP $\subseteq$ coNP/poly.

Proof. We first show that the statement holds for Multicolored Clique by giving a polynomial parameter transformation from Clique. In Clique the input is a graph G and an integer t and the question is if there is a clique of size at least t in G. Clique does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly when parameterized by the size of a minimum vertex cover of the input graph [16]. We may assume that a minimum vertex cover S of G is provided as input.

Construction. Let (G, t) be an instance of Clique and let $S = \{v_1, \ldots, v_s\}$ be a minimum vertex cover in G. Furthermore, let $I := V \setminus S$ be the remaining independent set. Since I is an independent set, the maximal size of the clique is $s + 1$. Consequently, (G, t) is a trivial no-instance if $t > s + 1$. Hence, we may assume that $t \leq s + 1$. We construct an instance G' for Multicolored Clique as follows.

First, we define t disjoint sets $V_1, \ldots, V_t$ that form the partition of the vertex set of G'. For every vertex $v_i \in S$ there are t copies $v_{i,1}, \ldots, v_{i,t}$ in G' such that $v_{i,1} \in V_1, v_{i,2} \in V_2, \ldots, v_{i,t} \in V_t$. We also add all vertices of I to V_t. Now, the classes $V_1, \ldots, V_{t-1}$ contain exactly s elements and the class V_t contains $s + |I|$ elements. Note that $\sum_{i=1}^{t-1} |V_i| = (t - 1) \cdot s$. Thus, the parameter is polynomially bounded in s since $t \leq s + 1$.

Next, we describe which edges are present in G'. If two vertices v_i and v_j of S are adjacent in G, we connect all copies of v_i with all copies of v_j, except those that are in the same set of the partition. Moreover, for the first $(t - 1)$ classes $V_1, \ldots, V_{t-1}$ we do the following: For every edge $\{v_i, w\}$ with $v_i \in S$ and $w \in I$, we add edges $\{v_{i,j}, w\}$ for each copy $v_{i,j}$ of v_i. Observe that $V_1, \ldots, V_t$ are all independent sets in G'. Hence, G' is a feasible instance of Multicolored Clique.

Correctness: To prove that the transformation from (G, t) into G' is a polynomial parameter transformation, it remains to show that G has a clique of size at least t if and only if G' has a multicolored clique.

($\Rightarrow$) Let K be a clique of size t in G. Since I is an independent set we can assume that $t - 1$ vertices of K lie in S and one vertex u of K lies in $S \cup I$. Without loss of generality, we assume that $K = \{v_1, \dots, v_{t-1}, u\}$. If $u \in S$, then $u = v_i$ for some $i \geq t$. Then there is a copy $v_{i,t} \in V_t$ of v_i and the vertices $v_{1,1}, \dots, v_{t-1,t-1}, v_{i,t}$ form a multicolored clique in G'. If $u \in I$, then $u \in V_t$ and the vertices $v_{1,1}, \dots, v_{t-1,t-1}, u$ form a multicolored clique in G'.

($\Leftarrow$) Conversely, let K' be a multicolored clique in G'. Without loss of generality, we assume that $K' = \{v_{i_1,1}, v_{i_2,2}, \dots, v_{i_{(t-1)},t-1}, u\}$ with $u \in V_t$. Note that the indices $i_1, \dots, i_{(t-1)}$ are pairwise distinct by the construction of G'. If $u \in I$, then the vertices $v_{i_1}, v_{i_2}, \dots, v_{i_{(t-1)}}, u$ form a clique of size t in G. Otherwise, if $u \notin I$, then we can assume that $u = v_{i_t,t}$ for some i_t that is different from $i_1, \dots, i_{t-1}$. Then, the vertices $v_{i_1}, v_{i_2}, \dots, v_{i_{(t-1)}}, v_{i_t}$ form a clique of size t in G, which completes the correctness proof.

Multicolored Independent Set. By the above, we have shown that the statement holds for MULTICOLORED CLIQUE. We next conclude that it also holds for MULTICOLORED INDEPENDENT SET. To this end, we consider a simple polynomial parameter transformation. Let G be an instance of MULTICOLORED CLIQUE where $V_1, \dots, V_t$ is the partition of the vertex set of G. We then construct an instance G' of MULTICOLORED INDEPENDENT SET from G as follows: For every pair $u \in V_i$ and $w \in V_j$ with distinct indices i and j we remove the edge $\{u, w\}$ if u and w are adjacent in G or we add an edge $\{u, w\}$ otherwise. It is easy to see that G contains a multicolored clique if and only if G' contains a multicolored independent set. Moreover, the parameter is the same in both instances. Consequently, the transformation described above is a polynomial parameter transformation and thus, the statement holds for MULTICOLORED INDEPENDENT SET. □

Exponential Time Hypothesis (ETH). In classic complexity, the widely believed conjecture P $\neq$ NP provides a framework which allows us to distinguish between problems that can be solved in polynomial time and problems that presumably cannot be solved in polynomial time. However, there might be significant differences in superpolynomial running times. For example, an algorithm with a subexponential running time of $\mathcal{O}(2^{\sqrt[5]{n}})$ appears to be much more efficient than an algorithm with a superexponential running time of $\mathcal{O}(2^{(n^5)})$. The *fine-grained complexity* is a framework that allows us to provide more elaborate lower bounds.

In fine-grained complexity, the ETH [96] is the standard conjecture. The ETH is a conjecture about the complexity of 3-SAT. In 3-SAT the input is a logical formula ϕ in 3-conjunctive normal form (3-CNF). That is, ϕ is a conjunction of disjunctions of (at most) three literals. Let δ_3 be the infimum of the set of constants c for which there exists an algorithm solving 3-SAT in $2^{c \cdot n} \cdot n^{\mathcal{O}(1)}$ time where n denotes the number of variables in the input formula. The ETH then states that $\delta_3 > 0$, which implies that 3-SAT can not be solved in $2^{o(n)} \cdot n^{\mathcal{O}(1)}$ time. Together with the Sparsification Lemma [96], the ETH implies that 3-SAT cannot be solved in $2^{o(|\phi|)} \cdot |\phi|^{\mathcal{O}(1)}$ time, where $|\phi|$ is the size of the input formula.

We can transfer the ETH-based lower-bound for 3-SAT via reductions. Given a computable function f, we let f^{-1} denote the inverse function of f. If there exists a polynomial-time reduction from 3-SAT to a problem L that produces an instance x of size $\mathcal{O}(f(|\phi|))$ for some computable function f, then L can not be solved in $2^{o(f^{-1}(|x|))} \cdot |x|^{\mathcal{O}(1)}$ time unless the ETH fails. Furthermore, if there exists a polynomial-time parameterized reduction from 3-SAT parameterized by $|\phi|$ that produces an instance (I, k) of a parameterized problem L with $k \in \mathcal{O}(f(|\phi|))$ for some computable function f, then L can not be solved in $2^{o(f^{-1}(k))} \cdot |I|^{\mathcal{O}(1)}$ time unless the ETH fails.

Part II

Strong Triadic Closure

Consider your relations to other persons. Some of these relations are strong relations like close friendships, relationships to family members or to close colleagues. On the flipside, some relations are weak relations like casual acquaintances or colleagues you see once in a while at a conference. In the first part of this work we study a class of computational problems that are motivated by recovering the strength of relations when only the structure of a social network is known.

Social networks represent relations between individuals such as friendship or acquaintance in online social networks. These relations are modeled as an undirected graph where the vertices represent agents in the network and an edge represents an interaction or a friendship between two agents. One task in social network analysis is to determine the strength [73, 156, 164, 180] and type [42, 168, 181] of the relationship signified by each edge of the network. We study a class of computational problems called *strong triadic closure problems* which were introduced by Sintos and Tsaparas in 2014 [164]. In these problems one aims to infer the strength of relationships in the following sense: One wants to find out which edges represent strong relationships and which edges represent a weak relationship and label these edges accordingly.

Even though the computational problems were introduced in 2014, the basic idea of partitioning relationships of a social network into strong and weak relationships goes back to the notion of *strong and weak ties* by the sociologist Mark Granovetter [72, 73] who postulates that, if an agent has strong relations to two other agents, then these two agents should have at least a weak relation. This property is known as *strong triadic closure property*. Following this assertion, Sintos and Tsaparas [164] proposed to find strong ties in social networks by labeling the edges as weak or strong such that the strong triadic closure property is fulfilled and the number of strong edges is maximized. Figure II.1 shows an example of a social network with weak and strong relationships such that the strong triadic closure property is satisfied.

In context of sociological research, Granovetter's idea has been used extensively as a base for numerous works [105]. The strong and weak ties of an individual might have an influence on poverty, health, and its well-being [24]. Sociologists also studied the influence of weak ties on mental health and suicidal thoughts [10]. Weak ties are also associated with the accessibility of specific resources [127] and may help when one is looking for a job [74]. Strong ties might give instrumental and emotional support [39].

Besides sociological research, the notion of strong triadic closure received attention in data mining. In experimental studies, the strong triadic closure property has been used to identify strong relationships in social networks [164, 93, 156, 1]. The strong edges found in these experiments are considered as reasonable strong

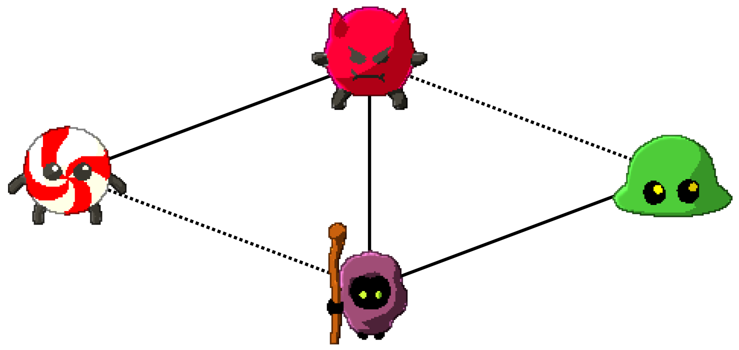


Figure II.1: An example of a social network. The vertices represent the four agents *Devil*, *Peppermint*, *Slime*, and *Witch*[1]. An edge between two agents represents a friendship. The dotted edges correspond to weak relationships and the remaining edges correspond to strong relationships. Observe that the strong triadic closure property is satisfied: Whenever one agent has strong relations with two other agents, these two agents are connected by an edge.

relationships for a given data set [156] and have been claimed to seem sensible in practice [164]. Adriaens et al. [1] experimentally evaluated relaxations of strong triadic closure problems. Their experiments show that that the strong triadic closure property is a reasonable tool to find strong relationships in social networks but one might fail to capture some strong ties that rely for example on community structures and external information. Other studies combine strong triadic closure and related approaches with external information like prior knowledge about specific strong relationships within the network [156, 155] to infer strong ties. The strong ties that were discovered using this prior knowledge are reasonable for the given data [156]. Moreover, it was observed that showing only the strong relationships that were discovered in these experiments significantly simplifies the network structure [156]. Furthermore, there are approaches of using strong triadic closure to recover dynamic information such as the order in which the relationships occurred in the network using a probabilistic factor graph model [52].

Apparently, inferring strong ties using the strong triadic closure property is a reasonable technique that finds application in the data mining community. In this

[1]I would like to thank Sebastian Lieb for allowing me to use the characters from his game *Evolings*. (`https://soerbgames.itch.io/evolings`)

work, we study strong triadic closure from an algorithmic point of view. We study the task of finding a maximum number of strong ties as a combinatorial optimization problem. The aim is to find efficient algorithms to solve this problem exactly, to outline the limits of tractability, and to improve our theoretical understanding of strong triadic closure as a graph property.

In the following, we give the formal problem definitions of the combinatorial optimization problems studied in this part and we state some basic observations about strong triadic closure and so-called Gallai graphs [64, 124, 167].

Problem Definitions. In the computational problem Strong Triadic Closure (STC) introduced by Sintos and Tsaparas [164] we are looking for an *STC-labeling*, which is defined as follows.

Definition II.1. A *labeling* $L = (S_L, W_L)$ of an undirected graph $G = (V, E)$ is a partition of the edge set E. The edges in S_L are called *strong* and the edges in W_L are called *weak*. A labeling $L = (S_L, W_L)$ is an *STC-labeling* if there exists no pair of strong edges $\{u, v\} \in S_L$ and $\{v, w\} \in S_L$ such that $\{u, w\} \notin E$.

For a vertex u and a strong edge $\{u, v\}$, we call v a *strong neighbor of* u. Analogously, if $\{u, v\}$ is weak, we call v a *weak neighbor of* u. The computational problem described informally above is now the following.

Strong Triadic Closure (STC)
Input: An undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}_0$.
Question: Is there an STC-labeling $L = (S_L, W_L)$ with $|W_L| \leq k$?

Sintos and Tsaparas [164] also introduced an extension of STC where agents may have c different types of strong relationships. In this model, the strong triadic closure property only applies to edges of the same strong type.

This extension is motivated by the observation that agents may very well have close relations to agents that do not know each other if these relations themselves arise in segregated contexts. For example, it is quite likely that one's rugby teammates do not know all of one's close colleagues. The edge labelings with up to c strong colors that model this variant of STC and the corresponding problem are defined as follows.

Definition II.2. A *c-labeling* $L = (S_L^1, \dots, S_L^c, W_L)$ of an undirected graph $G = (V, E)$ is a partition of the edge set E into $c + 1$ color classes. The edges in S_L^i, $i \in \{1, \dots, c\}$, are *strong* and the edges in W_L are *weak*. A c-labeling L is an *STC-labeling* if there exists no pair of edges $\{u, v\} \in S_L^i$ and $\{v, w\} \in S_L^i$ such that $\{u, w\} \notin E$ for any $i \in \{1, \dots, c\}$. We say that such a pair of edges *violates STC* for a c-labeling.

MULTI STRONG TRIADIC CLOSURE (MULTI-STC)
Input: An undirected graph $G = (V, E)$ and integers $c \in \mathbb{N}$ and $k \in \mathbb{N}_0$.
Question: Is there an STC-labeling $L = (S_L^1, \dots, S_L^c, W_L)$ with $|W_L| \leq k$?

Note that MULTI-STC is a generalization of STC and thus NP-hard [164].

We also study two generalizations of MULTI-STC. The first generalization deals with the case where one restricts the set of possible relations for some agents. Assume, for example, that strong edges correspond to family relations or professional relations. If one knows the profession of some agents, then this knowledge can be modeled by introducing different strong colors for each profession and constraining the sought edge labeling in such a way that each agent may receive only a strong edge corresponding to a familial relation or to his profession. In other words, for each agent we are given a list of allowed strong colors that may be assigned to incident relationships. Formally, we arrive at the following extension of STC-labelings.

Definition II.3. Let $G = (V, E)$ be a graph, $\Lambda : V \to 2^{\{1,2,\dots,c\}}$ a mapping for some $c \in \mathbb{N}$, and $L = (S_L^1, \dots, S_L^c, W_L)$ a c-colored STC-labeling. We say that an edge $\{v, w\} \in E$ *satisfies the* Λ*-list property under* L if $\{v, w\} \in W_L$ or $\{v, w\} \in S_L^\alpha$ for some $\alpha \in \Lambda(v) \cap \Lambda(w)$. We call a c-colored STC-labeling Λ-*satisfying* if every edge satisfies the Λ-list property under L.

VERTEX-LIST MULTI STRONG TRIADIC CLOSURE (VL-MULTI-STC)
Input: An undirected graph $G = (V, E)$, integers $c \in \mathbb{N}$ and $k \in \mathbb{N}_0$, and vertex lists $\Lambda : V \to 2^{\{1,2,\dots,c\}}$.
Question: Is there a Λ-satisfying STC-labeling L with $|W_L| \leq k$?

MULTI-STC is the special case where $\Lambda(v) = \{1, \dots, c\}$ for all $v \in V$.

The second generalization deals with the case where one restricts the set of possible relationship types for each relation. For example, if two rugby players live far apart, it is unlikely that they play rugby together. We might model this knowledge by restricting the number of relationship types for this specific relationship. In other words, for each relationship we are given a list of possible strong colors that may be assigned to this relationship. Observe that this is an actual generalization of VL-MULTI-STC: Consider three agents v_1, v_2, and v_3 that are pairwise related in the network. Assume the relationship between v_1 and v_2 and the relationship between v_1 and v_3 are both restricted to 'rugby' and 'colleagues'. If now the relationship between v_2 and v_3 is restricted to 'ballet class' and 'drinking buddies', this situation cannot be expressed with vertex lists. This more general constraint is formalized as follows.

Definition II.4. Let $G = (V, E)$ be a graph, $\Psi : E \to 2^{\{1,2,\dots,c\}}$ a mapping for some $c \in \mathbb{N}$ and $L = (S_L^1, \dots, S_L^c, W_L)$ a c-colored STC-labeling. We say that an edge $e \in E$ *satisfies the* Ψ*-list property under* L if $e \in W_L$ or $e \in S_L^\alpha$ for some $\alpha \in \Psi(e)$. We call a c-colored STC-labeling Ψ*-satisfying* if every edge satisfies the Ψ-list property under L.

This leads to the most general variant of STC studied in this work.

Edge-List Multi Strong Triadic Closure (EL-Multi-STC)
Input: An undirected graph $G = (V, E)$, integers $c \in \mathbb{N}$ and $k \in \mathbb{N}_0$ and edge lists $\Psi : E \to 2^{\{1,2,\dots,c\}}$.
Question: Is there a Ψ-satisfying STC-labeling L with $|W_L| \leq k$?

Gallai Graphs and their Relation to STC Problems. From a more abstract point of view, in Multi-STC we aim to label the edges in a way such that no two edges that form an induced P_3 receive the same strong color. Thus, whenever two edges form an induced P_3 in a graph, these two edges can be seen as a 'conflict pair' regarding the strong triadic closure property. The so-called Gallai graph [64, 124, 167] of a given graph models these conflict pairs. Formally, a Gallai graph is defined as follows.

Definition II.5. Given a graph $G = (V, E)$, the *Gallai graph* $\widetilde{G} := (\widetilde{V}, \widetilde{E})$ of G is defined by $\widetilde{V} := E$ and

$$\widetilde{E} := \{\{e_1, e_2\} \mid e_1 \text{ and } e_2 \text{ form an induced } P_3 \text{ in } G\}.$$

The Gallai graph of an n-vertex and m-edge graph has $\mathcal{O}(m)$ vertices and $\mathcal{O}(mn)$ edges. Gallai graphs do have restricted structure in the sense that not every graph is a Gallai graph of some other graph. For an example, consider a claw as given in Figure 1.1 for an example: The central vertex in a claw in $\widetilde{G}$ would correspond to an edge in G that forms an induced P_3 with three other edges that do not form an induced P_3 with each other. It is easy to see that this situation can not occur in a graph with four edges. However, for every graph H, there is a Gallai graph which contains H as induced subgraph [124]. For STC, the relation to Gallai graphs is as follows: A graph $G = (V, E)$ has an STC-labeling with at most k weak edges if and only if its Gallai graph has a vertex cover of size at most k [164]. Thus, STC can be solved in $\mathcal{O}(1.28^k + nm)$ time by using the current fastest algorithm for Vertex Cover [27]. More generally, a graph $G = (V, E)$ has a c-colored STC-labeling with at most k weak edges if and only if the Gallai graph of G has a properly c-colorable induced subgraph on $m - k$ vertices [164].

In the following, we extend this relation to EL-Multi-STC by considering list-colorings of the Gallai graph. The special cases VL-Multi-STC, Multi-STC, and STC nicely embed into the construction. First, let us define the problem that we need to solve in the Gallai graph formally. Given a graph $G = (V, E)$, we call a mapping $\chi : V \to \{0, 1, \ldots, c\}$ a *subgraph-c-coloring* if there is no edge $\{u, v\} \in E$ with $\chi(u) = \chi(v) \neq 0$. Vertices v with $\chi(v) = 0$ correspond to deleted vertices. The List-Colorable Subgraph problem is defined as follows.

List-Colorable Subgraph
Input: An undirected graph $G = (V, E)$ and integers $c \in \mathbb{N}$, $k \in \mathbb{N}_0$ and lists $\Gamma : V \to 2^{\{1,\ldots,c\}}$.
Question: Is there a subgraph-c-coloring $\chi : V \to \{0, 1, \ldots, c\}$ such that $|\{v \in V \mid \chi(v) = 0\}| \leq k$ and $\chi(w) \in \Gamma(w) \cup \{0\}$ for every $w \in V$?

EL-Multi-STC and List-Colorable Subgraph have the following relationship.

Proposition II.6. An instance (G, c, k, Ψ) of EL-Multi-STC is a yes-instance if and only if $(\widetilde{G}, c, k, \Psi)$ is a yes-instance of List-Colorable Subgraph, where $\widetilde{G}$ is the Gallai graph of G.

Proof. We first describe how to transform c-colored Ψ-satisfying labeling L for G into a coloring χ_L for $\widetilde{G}$ that satisfies $\chi_L(v) \in \Psi(v) \cup \{0\}$ for every vertex v of $\widetilde{G}$ and vice versa, such that the number of weak edges under L and the number of vertices that receive color 0 under χ_L are the same. We let L_χ denote the Ψ-satisfying labeling for G resulting from a coloring χ for $\widetilde{G}$.

Construction of χ_L and L_χ. For any c-colored labeling L we set $\chi_L(e) := i$ for each edge in S_L^i with $i \in \{1, \ldots, c\}$, and $\chi_L(e) = 0$ for each edge in W_L. By definition, the c-colored labeling χ is Ψ-satisfying if and only if χ_L satisfies the list constraints in the List-Colorable Subgraph instance, that is, $\chi_L(v) \in \Psi(v) \cup \{0\}$ for each vertex v. Moreover, the number of weak edges in L is precisely the number of vertices in $\widetilde{G}$ that receive color 0. By symmetric arguments, each subgraph-c-coloring χ that satisfies Ψ and has k vertices v such that $\chi(v) = 0$ defines a c-colored labeling L_χ of G that is Ψ-satisfying and has k weak edges.

Equivalence. We show that

a) If L is an STC-labeling for G, then χ_L is a subgraph-c-coloring for $\widetilde{G}$, and

b) If χ_L is a subgraph-c-coloring for $\widetilde{G}$, then L is an STC-labeling for G.

a) Let L be an STC-labeling. We show that for all adjacent vertices u and v in $\widetilde{G}$ either $\chi_L(u) \neq \chi_L(v)$ or $\chi_L(u) = 0$ or $\chi_L(v) = 0$. Assume that $\chi_L(u) \neq 0$ and $\chi_L(v) \neq 0$. Then, the corresponding edges u and v in G are colored with some strong colors S_L^i and S_L^j. Since u and v are adjacent in $\widetilde{G}$, u and v form a P_3 in G and since L is an STC-labeling, we have $i \neq j$. Therefore, $\chi(u) \neq \chi(v)$.

b) Let χ be a subgraph-c-coloring. We show that L_χ is an STC-labeling. Consider a pair of incident edges u and v that form a P_3 in G. If $\chi(u) = 0$ or $\chi(v) = 0$, then one of the two edges is weak in L_χ. Otherwise, we have $\chi(u) \neq \chi(v)$ because u and v are adjacent in $\widetilde{G}$. Thus, u and v have different strong colors under L_χ. Therefore, L_χ is an STC-labeling. □

The correspondence from Proposition II.6 means that we can solve EL-Multi-STC by solving List-Colorable Subgraph on the Gallai graph of the input graph. To this end we give a running time bound for List-Colorable Subgraph. The algorithm for obtaining this running time is a straightforward dynamic programming algorithm over subsets. Since we are not aware of any concrete result in the literature implying this running time bound, we provide a proof for the sake of completeness.

Proposition II.7. List-Colorable Subgraph can be solved in $\mathcal{O}(3^n \cdot c^2(n + m))$ time. EL-Multi-STC can be solved in $\mathcal{O}(3^m \cdot c^2mn)$ time.

Proof. We first describe an algorithm for List-Colorable Subgraph. Afterwards, we analyze the running time.

Algorithm. Let $(G = (V, E), c, k, \Gamma)$ be an instance of List-Colorable Subgraph. We define a dynamic programming table T with entries $T[S, i]$ where $S \subseteq V$ and $i \in \{1, \dots, c\}$. The aim is to fill T such that for all entries we have $T[S, i] =$ 'true' if there is a subgraph-c-coloring χ for $G[S]$ such that $\chi(v) \in \{1, \dots, i\} \cap \Gamma(v)$ for all $v \in S$ and $T[S, i] =$ 'false' otherwise.

The table is initialized for $i = 1$ and each $S \subseteq V$ by setting

$$T[S,1] := \begin{cases} \text{‘true’} & \text{if } S \text{ is an independent set } \wedge \forall v \in S : 1 \in \Gamma(v), \\ \text{‘false’} & \text{otherwise.} \end{cases}$$

For $i > 1$, the table entries are computed by the recurrence

$$T[S,i] := \begin{cases} \text{‘true’} & \text{if } \exists S' \subseteq S \text{ such that } S' \text{ is an independent set} \\ & \qquad\qquad\qquad\qquad \wedge\ \forall v \in S' : i \in \Gamma(v) \\ & \qquad\qquad\qquad\qquad \wedge\ T[S \setminus S', i-1] = \text{‘true’}, \\ \text{‘false’} & \text{otherwise.} \end{cases}$$

To check if the instance is a yes-instance we check if $T[S, c] =$ 'true' for some S such that $|S| \geq n - k$. The correctness proof is straightforward and thus omitted.

Running Time. For each $i \in \{1, \ldots, c\}$ an entry $T[S, i]$ can be computed in $\mathcal{O}(2^{|S|} \cdot c \cdot (m+n))$ time where the factor $c \cdot (m+n)$ corresponds to the time needed to check whether S' is an independent set and whether $i \in \Gamma(v)$. Consequently, all entries can be computed in $\mathcal{O}(c \cdot \sum_{j=1}^{c} \binom{n}{j} \cdot c \cdot (m+n)) = \mathcal{O}(3^n \cdot c^2 \cdot (m+n))$ time. The running time for EL-Multi-STC follows from Proposition II.6 and the fact that the Gallai graph of a graph G with n vertices and m edges has $\mathcal{O}(m)$ vertices and $\mathcal{O}(mn)$ edges. $\square$

Overview of Part II. In Chapter 2 we study STC—the problem variant with one strong color—and its relation to a problem called Cluster Deletion (CD). In CD, the input is a graph G and an integer k and the question is whether one can delete at most k edges such that every connected component of the remaining graph is a clique. Recall that in STC one aims to label one edge of every induced P_3 as weak. Therefore, as observed previously [110], STC is closely related to CD where one aims to destroy induced P_3s by edge deletions. First, we analyze the classic complexity of both problems on graph classes that can be described by one forbidden subgraph H of order at most 4. For every possible H, we then analyze if the solution structure of STC corresponds to a solution of CD. Afterwards, we study the parameterized complexity of both problems when parameterized by the number of strong edges (in case of STC) or non-deleted edges (in case of CD). We present FPT algorithms for STC and CD and show that both problems are unlikely to admit a polynomial kernel.

In Chapter 3, we consider the case where we have multiple strong colors. More precisely, we study Multi-STC and its list variants. We observe that known results for a classic edge coloring problem imply the NP-hardness of Multi-STC for every fixed number of strong colors. If the number of colors is at least 3, Multi-STC is NP-hard even if $k = 0$. The main result of this chapter is a lower bound for VL-Multi-STC: we show that the Exponential Time Hypothesis (ETH) implies that VL-Multi-STC cannot be solved in $2^{o(n^2)}$ time.

In Chapter 4, we initiate the study of the parameterized complexity of Multi-STC and its list variants. Since we observed in Chapter 3 that Multi-STC is NP-hard even if $k = 0$, we consider parameterization by the solution size k_1 of STC. On the positive side, we show that Multi-STC is FPT for k_1 and that the list versions are FPT for $k_1 + c$, where c denotes the number of strong colors. Moreover, we show that all three problems admit a problem kernel with at most $2^{c+1}k_1$ vertices. On the negative side, we show that VL-Multi-STC and EL-Multi-STC are W[1]-hard when parameterized by k_1 alone and unlikely to admit a polynomial kernel when parameterized by $k_1 + c$.

In Chapter 5, we study the relation between MULTI-STC and a problem called EDGE COLORABLE SUBGRAPH (ECS). In ECS one is given a graph G and two integers c and k. The task is to delete at most k edges from G such that the remaining edges can be labeled with colors 1 to c in a way that no incident edges receive the same color. Due to Vizing's Theorem [178] ECS and MULTI-STC are trivial on graphs with maximum-degree $c-1$. We use this fact as a base for a distance-from-triviality parameterizations. For every c, we provide polynomial problem kernels for ECS when parameterized by the edge-deletion distance ξ_{c-1} to graphs with maximum degree $c-1$ and the vertex-deletion distance λ_c to graphs with maximum component size c. The main technical contribution of this chapter is to lift the problem kernel for ξ_{c-1} to MULTI-STC for the cases where $c \in \{1, 2, 3, 4\}$.

Chapter 2

Strong Triadic Closure and Cluster Deletion

We study Strong Triadic Closure (STC) and a closely related problem called Cluster Deletion (CD). Both problems arise in social network analysis and data clustering. Recall that STC is an edge coloring problem with two colors (one color representing *weak* relationships and one color representing *strong* relationships) such that the strong triadic closure property is satisfied. Informally, the strong triadic closure property is the assumption that if one agent has strong relationships with two other agents, then these two other agents should have at least a weak relationship. The aim in the computational problem is then to label a maximum number of edges of the given social network as strong while fulfilling this requirement.

Formally, the problem asks for an STC-labeling as defined in the introduction of Part II. Recall that a *labeling* $L = (S_L, W_L)$ of an undirected graph $G = (V, E)$ is a partition of the edge set E and that the edges in S_L are called *strong* and the edges in W_L are called *weak*. A labeling $L = (S_L, W_L)$ is then called an *STC-labeling* if there exists no pair of strong edges $\{u, v\} \in S_L$ and $\{v, w\} \in S_L$ such that $\{u, w\} \notin E$. For a vertex u and a strong edge $\{u, v\}$, we call v a *strong neighbor of* u. Analogously, if $\{u, v\}$ is weak, we call v a *weak neighbor of* u. Recall that the computational problem STC is defined as follows.

Strong Triadic Closure (STC)
Input: An undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Is there an STC-labeling $L = (S_L, W_L)$ with $|W_L| \leq k$?

We call an STC-labeling L *optimal* for a graph G, if the number $|W_L|$ of weak edges is minimal. Furthermore, recall that the STC-labeling property can also be stated in terms of induced subgraphs: For every induced P_3 in G, at most one edge is labeled strong. Therefore, as observed previously [110], STC is closely related to the problem of destroying induced P_3s by edge deletions. Since the graphs without an induced P_3 are exactly the disjoint union of cliques, this problem is usually formulated as follows.

> Cluster Deletion (CD)
> **Input**: An undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
> **Question**: Can we transform G into a *cluster graph*, that is, a graph where every connected component is a clique, by at most k edge deletions?

Observe that any set $D \subseteq E$ of at most k edges where $(V, E \setminus D)$ is a cluster graph, directly implies an STC-labeling $(E \setminus D, D)$ with at most k weak edges. There are, however, graphs G where the minimum number of weak edges in an STC-labeling is strictly smaller than the number of edge deletions that are needed in order to transform G into a cluster graph [110]. Due to the close relation between the two problems, there are graph classes where any minimum-cardinality solution of CD directly implies an optimal STC-labeling. This is the case, for example, on graphs with maximum degree 3 [110].

In this chapter, we study the correspondence and the classic computational complexity of both problems in graph classes that can be described by one forbidden induced subgraph of order at most 4. Furthermore, we study the parameterized complexity of STC when parameterized by the number of strong edges and consider a similar parameterization for CD.

Related Work. Sintos and Tsarparas [164] introduced the computational problem STC and proved that it is NP-hard. The NP-hardness holds even when restricted to graphs with maximum degree 4 [110] or to split graphs [111]. In contrast, STC is solvable in polynomial time when the input graph is subcubic [110], a proper interval graph [111], or a cograph [110], that is, a graph with no induced P_4. STC can be solved in $\mathcal{O}(1.28^k + nm)$ time which is implied by a parameter-preserving reduction to Vertex Cover using the Gallai graph [164] and the current fastest algorithm for Vertex Cover [27]. The reduction to vertex cover also implies a factor-2 approximation. STC admits a problem kernel with $4k$ vertices [77][1], which has recently been improved to a kernel with $2k$ vertices [23].

[1]The $4k$ vertex kernel for STC is presented in the puplication on which Chapter 2 is based on [77]. In this work, it is not included in Chapter 2 since we provide a generalization of this problem kernelization in Chapter 4.

Table 2.1: Complexity dichotomy and correspondence of STC and CD on H-free graphs.

	STC	CD	correspondent
$H \in \{3K_1, K_4, 4K_1, C_4, 2K_2, \text{claw}, \text{co-claw}, \text{co-diamond}, \text{co-paw}\}$	NP-h	NP-h	NO
$H = \text{diamond}$	NP-h	NP-h	YES
$H \in \{K_3, P_3, K_2K_1, \text{paw}, P_4\}$	P	P	YES

There are variants of STC with additional edge insertions [164, 138]. Other variants of STC ask for a labeling in which some prespecified communities are connected via strong edges [156, 175, 93]. Golovach et al. [71] considered a generalization of STC where the aim is to label at most k edges weak such that each induced subgraph isomorphic to a fixed graph F has at least one weak edge. STC is then the special case where $F = P_3$.

CD is NP-hard [162], even when restricted to graphs with maximum degree 4 [109] or to $(2K_2, 3K_1)$-free graphs [66], and solvable in polynomial time on cographs [66] and in time $O(1.42^k + m)$ on general graphs [14]. CD admits a problem kernels with $4k$ vertices [85], which has very recently been improved to a kernel with $2k$ vertices [23].

Independent from our work, Golovach et al. [71] showed that STC parameterized by the number of strong edges $\ell := m - k$ has no polynomial kernel unless NP $\subseteq$ coNP/poly, even when the input graph is a split graph.

Our Results. First, we extend the line of research studying the complexity of CD [66] and STC [110] on H-free graphs where H is a graph of order at most 4. These results are shown in Table 2.1. We present a complexity dichotomy between polynomial-time solvable and NP-hard cases for all possibilities for H. Moreover, we show for all such graphs H whether STC and CD *correspond* on H-free graphs, that is, whether every STC-labeling with at most k weak edges implies the existence of a CD solution with at most k edge deletions.

Second, we initiate the study of the parameterized complexity of STC and CD with respect to the parameter $\ell := |E| - k$. Hence, in STC we are searching for an STC-labeling with at least ℓ strong edges and in CD we are searching for a cluster graph G' that is a subgraph of G and that has at least ℓ edges; we call these edges the *cluster edges* of G'. While we present fixed-parameter algorithms for both problems parameterized by ℓ, we also show that, somewhat surprisingly, both problems do not admit a polynomial kernel with respect to ℓ, unless NP $\subseteq$ coNP/poly.

2.1 STC and CD on H-free Graphs

First, we study the correspondence and the classic complexity of STC and CD on H-free graphs where H is a graph on at most four vertices. For an illustration of these small graphs we refer to Figure 1.1.

2.1.1 The Correspondence between STC and CD on H-Free Graphs

We study if an STC solution implies the existence of a CD solution of the same size. Recall that the opposite direction always holds: Any set $D \subseteq E$ of at most k edges such that $(V, E \setminus D)$ is a cluster graph directly gives an STC-labeling $(E \setminus D, D)$ with at most k weak edges. Thus, every solution of CD provides an STC-labeling. Throughout this work, we call such labeling a *cluster labeling*. A cluster labeling is not necessarily optimal, as shown in Figure 2.1. The figure provides two examples where a cluster labeling is not an optimal solution of STC. In the upper example—provided by Konstantinidis and Papadopoulos [111]—an optimal STC-labeling has eight strong edges, while the best cluster labeling has only seven cluster edges. In the second example, an optimal STC-labeling has seven strong edges, while the best cluster labeling has six cluster edges. It has recently been shown that the number of weak edges in a cluster labeling corresponding to an optimal CD solution is at most twice as large as the number of weak edges in an optimal STC labeling [177].

We define that the problems STC and CD *correspond* on a graph class Π if for every graph in Π we can find a cluster labeling that is an optimal STC-labeling. In this case we call the labeling an *optimal cluster labeling*.

With the examples shown in Figure 2.1 we can identify graph classes on which STC and CD do not correspond. The upper example is C_4-, $2K_2$-, and co-diamond-free. The lower example is $3K_1$-, K_4-, $4K_1$-, co-paw-, claw-, and co-claw free. With the next theorem, we show that STC and CD correspond on the class of H-free graphs on all remaining cases where H has three or four vertices.

Theorem 2.1. *Let H be a small graph on three or four vertices. Then,* STC *and* CD *correspond on the class of H-free graphs if and only if*

$$H \in \{K_3, P_3, K_2 + K_1, P_4, \textit{diamond}, \textit{paw}\}.$$

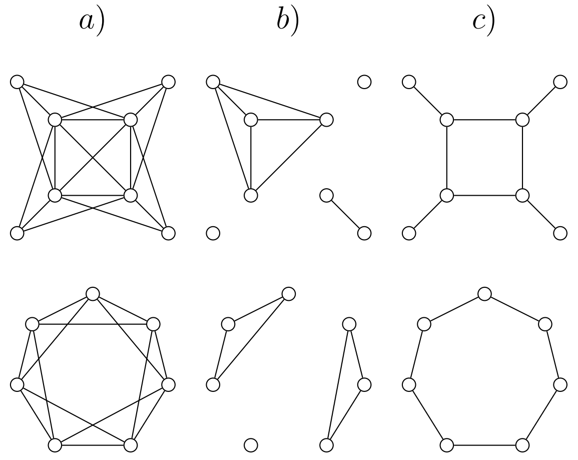


Figure 2.1: Two graphs where no cluster labeling is an optimal STC-labeling. Column a) shows the input graph, Column b) shows the strong edges of an optimal cluster labeling, and Column c) shows the strong edges in an optimal STC-labeling.

Proof. The examples in Figure 2.1 show that the problems do not correspond on the class of H-free graphs if $H \in \{3K_1, C_4, 2K_2, \text{co-paw}, \text{co-diamond}, K_4,\ 4K_1, \text{claw}, \text{co-claw}\}$. It remains to show that the problems correspond if $H \in \{K_3, P_3, K_2 + K_1, P_4, \text{ diamond, paw}\}$.

Case 1: $H = K_3$**.** On triangle-free graphs, the strong edges of an optimal STC-labeling correspond to the edges of a maximum matching, which is an optimal cluster labeling.

Case 2: $H = P_3$**.** On P_3-free graphs, every edge is labeled as strong in an optimal STC-labeling [164]. Since P_3-free graphs are cluster graphs, this labeling is a cluster labeling.

Case 3: $H = P_4$**.** On P_4-free graphs, there is an optimal cluster labeling [110].

Case 4: $H =$ paw**.** Then, each component of the input graph is triangle-free or complete multipartite [140]. Complete multipartite graphs are P_4-free. Thus, it follows by the Cases 1 and 3 that there is an optimal cluster labeling on paw-free graphs.

Case 5: $H = K_2 + K_1$**.** Then, since every $K_2 + K_1$-free graph is paw-free, there is an optimal cluster labeling on $K_2 + K_1$-free graphs due to Case 4.

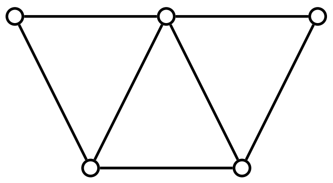


Figure 2.2: A graph that is contained as an induced subgraph in both examples given in Figure 2.1.

Case 6: $H =$ diamond**.** Let $G = (V, E)$ be a diamond-free graph. We prove that there is an optimal cluster labeling for G. The class of diamond-free graphs can be characterized as *strictly clique irreducible* graphs [150]. That is, every edge of a diamond-free graph lies in a unique maximal clique.

To show that there is an optimal cluster labeling, it is sufficient to prove that there is an optimal STC-labeling $L = (S_L, W_L)$ such that there is no triangle on vertices u_1, u_2, and u_3, with $\{u_1, u_2\} \in S_L$, $\{u_2, u_3\} \in S_L$, and $\{u_1, u_3\} \in W_L$.

Let $v \in V$ be a vertex of G. Since G is strictly clique irreducible, we can partition $N[v]$ into maximal cliques $K_1, K_2, \ldots, K_t$ such that $K_i \cap K_j = \{v\}$ for $i \neq j$. Let $L = (S_L, W_L)$ be an optimal STC-labeling for G such that v has a strong neighbor w_1 in K_1 under L. We prove that v does not have a strong neighbor in any of the other maximal cliques, which means $E(\{v\}, N(v) \setminus K_1) \subseteq W_L$. Assume towards a contradiction that v has a strong neighbor $w_j \in K_j$ for some $j \neq 1$. Then, there is an edge $\{w_1, w_j\} \in E$ since L is an STC-labeling. Consequently, there is a clique $K_+ \subseteq N[v]$ containing v, w_1, and w_j, which contradicts the fact that G is strictly clique irreducible.

Now assume towards a contradiction that there is a triangle on vertices u_1, u_2, and u_3, such that $\{u_1, u_2\} \in S_L$, $\{u_2, u_3\} \in S_L$, and $\{u_1, u_3\} \in W_L$. Since every vertex can only have strong neighbors in one maximal clique, u_1, u_2, and u_3 are elements of the same maximal clique K. Since u_1 and u_3 do not have any strong neighbors in $V \setminus K$, we do not produce a strong P_3 by adding $\{u_1, u_3\}$ to S_L, which contradicts the fact that L is an optimal STC-labeling. □

Finding examples of graphs where a cluster labeling is not an optimal STC-labeling increases our understanding of strong triadic closure as a property of edge-labeled graphs. It helps us to evaluate to which extent the inference of strong ties using the strong triadic closure property differs from clustering the social network.

With Theorem 2.1 we provide a first step towards a characterization of graph classes on which STC and CD correspond. A next step could be to classify the family $\mathcal{H}$ containing all graphs H such that STC and CD correspond on H-Free graphs. Theorem 2.1 shows that $\{K_3, P_3, K_2 + K_1, P_4, \text{diamond}, \text{paw}\} \subseteq \mathcal{H}$. Furthermore, note that each graph G where a cluster labeling is not an optimal STC-labeling contains every $H \in \mathcal{H}$ as an induced subgraph. In particular, every $H \in \mathcal{H}$ is contained as an induced subgraph in both examples given in Figure 2.1. As a consequence, every H in $\mathcal{H}$ consists of at most seven vertices and has a maximum clique of size at most 3. Figure 2.2 shows a small graph on five vertices that is called *gem*. Both examples shown in Figure 2.1 contain a gem as induced subgraph. Thus, it would be an interesting step to investigate if STC and CD correspond on gem-free graphs.

From a purely algorithmic point of view one might exploit the correspondence between STC and CD to determine the classic complexity of STC. Recall that Konstantinidis et al. [110] showed that the problems correspond on P_4-free graphs and used this correspondence to provide a polynomial-time algorithm. In the next section we also exploit the correspondence to obtain NP-hardness of STC on some restricted graph classes by using existing results for CD.

2.1.2 The Complexity of STC and CD on H-Free Graphs

We next study the classic complexity of STC and CD on H-free graphs, where H is a small graph on three or four vertices. We provide a dichotomy between polynomial-time solvable and NP-hard cases for all possibilities of H. For the case where $H \in \{C_4, \text{diamond}, K_4\}$ we exploit the correspondence between STC and CD on diamond-free graphs (Theorem 2.1) and an existing hardness result for CD [109]. We first identify the cases where both problems are solvable in polynomial time.

Lemma 2.2. *If $H \in \{K_3, P_3, K_2 + K_1, P_4, \text{paw}\}$, then* STC *and* CD *are solvable in polynomial time on H-free graphs.*

Proof. STC and CD are solvable in polynomial time on P_4-free graphs [110]. Moreover, STC can also be solved in polynomial time on $K_2 + K_1$-free and P_3-free graphs, since $K_2 + K_1$-free and P_3-free graphs are P_4-free. On triangle-free graphs, we can solve both problems by computing a maximal matching, which can be done in polynomial time [133]. Every component of a paw-free graph is triangle-free or complete multipartite and thus P_4-free [140]. Hence, we can use the polynomial-time algorithm on the P_4-free components and a polynomial-time algorithm to find a maximum matching on the triangle-free components. □

In the following, we show that for all other possible cases where H has three or four vertices, both problems remain NP-hard on H-free graphs. For the case where $H \in$

$\{3K_1, 4K_1, 2K_2, \text{claw}, \text{co-diamond}, \text{co-paw}\}$ we use a reduction from CLIQUE that has been previously used to show certain hardness results for CD [137]. When H is a co-claw we provide a slightly more technical reduction from 3-CLIQUE COVER. The remaining cases where $H \in \{C_4, \text{diamond}, K_4\}$ follow from previous results [109] combined with the correspondence on diamond-free graphs from Theorem 2.1.

For the reduction from CLIQUE we introduce the following definition.

Definition 2.3. *Let $G = (V, E)$ be a graph. The* expanded graph $\widehat{G}$ *of G is the graph obtained by adding a clique $\widehat{K} := \{v_1, \dots, v_{|V|^3}\}$ and edges such that every vertex in V is adjacent to all vertices in $\widehat{K}$.*

Obviously, we can construct $\widehat{G}$ from G in polynomial time. We use this construction to give a reduction from CLIQUE to STC and CD that also transfers certain H-freeness properties from G to $\widehat{G}$. With the following lemma, we state that a CLIQUE instance (G, t) is a yes-instance if and only if the STC-instance $(\widehat{G}, \binom{n^3}{2} + t \cdot n^3)$ is a yes-instance.

Lemma 2.4. *Let $(G = (V, E), t)$ be a* CLIQUE *instance.*

a) There is a clique of size at least t in G if and only if there is an STC-labeling $L = (S_L, W_L)$ for $\widehat{G}$ such that $|S_L| \geq \binom{n^3}{2} + t \cdot n^3$.

b) There is a clique of size at least t in G if and only if $\widehat{G}$ has a solution for CD *with at least $\binom{n^3}{2} + t \cdot n^3$ cluster edges.*

Proof. *a)* We separately prove the two directions of the statement.

$(\Rightarrow)$ Let $V' \subseteq V$ be a clique on t vertices in G. Then, we obtain an STC-labeling $L := (S_L, W_L)$ for $\widehat{G}$ with at least $\binom{n^3}{2} + t \cdot n^3$ strong edges by defining $S_L := E_{\widehat{G}}(V' \cup \widehat{K})$. Note that L is a cluster labeling, and thus, it is an STC-labeling. From $|\widehat{K}| = n^3$ we also get that $|S_L| = \binom{t}{2} + \binom{n^3}{2} + t \cdot n^3 \geq \binom{n^3}{2} + t \cdot n^3$.

$(\Leftarrow)$ Conversely, let there be an STC-labeling $L := (S_L, W_L)$ for $\widehat{G}$ with at least $\binom{n^3}{2} + t \cdot n^3$ strong edges. We show that there is a clique V' of size at least t in G.

To this end, we call two vertices v_1 and v_2 in $\widehat{K}$ *members of the same family* F, if v_1 and v_2 have the same strong neighbors in V. For each family $F \subseteq \widehat{K}$, the set of strong neighbors of F forms a clique. Otherwise, if some vertex $v \in F$ has two nonadjacent strong neighbors u and w, then the edges $\{u, v\}$ and $\{v, w\}$ form a strong P_3 under L, contradicting the fact that L is an STC-labeling.

Let $F_1, \dots, F_p$ be the families of vertices in $\widehat{K}$, and let K_{F_i} denote the set of strong neighbors of each family F_i. We show that $|K_{F_i}| \geq t$ for some family F_i. First, observe that

$$|S_L \setminus (E_{\widehat{G}}(\widehat{K}) \cup E)| = \sum_{i=1}^{p} |F_i| \cdot |K_{F_i}| \leq \max_i |K_{F_i}| \cdot \sum_{i=1}^{p} |F_i| = \max_i |K_{F_i}| \cdot n^3.$$

Next, observe that there are at least $t \cdot n^3$ edges in $S_L \setminus E_{\widehat{G}}(\widehat{K})$, since $|S_L| \geq \binom{n^3}{2} + t \cdot n^3$ and $|E_{\widehat{G}}(\widehat{K})| = \binom{n^3}{2}$. Together with the fact that $|E| \leq \binom{n}{2}$, this implies

$$|S_L \setminus (E_{\widehat{G}}(\widehat{K}) \cup E)| \geq t \cdot n^3 - \binom{n}{2} > (t-1) \cdot n^3.$$

Combining both inequalities, we have $(t-1) < \max_i |K_{F_i}|$. Hence, there is a clique of size at least t in G.

b) Let $V' \subseteq V$ be a clique on t vertices in G. Since the labeling L from the proof of Property *a*) is a cluster labeling, there is a solution of CD on $\widehat{G}$ with at least $\binom{n^3}{2} + t \cdot n^3$ cluster edges.

Now, let there be a solution of CD on $\widehat{G}$ such that there are at least $\binom{n^3}{2} + t \cdot n^3$ cluster edges. We define an STC-labeling $L = (S_L, W_L)$ for $\widehat{G}$ by defining S_L as the set of cluster edges in the solution. Then, there is an STC-labeling with at least $\binom{n^3}{2} + t \cdot n^3$ strong edges. It then follows by *a*), that G has a clique of size at least t. □

With the next lemma, we show that transforming a graph G into the expanded graph $\widehat{G}$ transfers certain H-free properties.

Lemma 2.5. *Let $H \in \{3K_1, 2K_2, \text{co-diamond}, \text{co-paw}, 4K_1\}$. If a graph G is H-free, then the expanded graph $\widehat{G}$ is H-free.*

Proof. Note that each $H \in \{3K_1, 2K_2, \text{co-diamond}, \text{co-paw}, 4K_1\}$ is disconnected. Assume towards a contradiction that $\widehat{G}$ has H as an induced subgraph. Since G is H-free and we do not add any edges between vertices of G during the construction of $\widehat{G}$, at least one of the vertices of this induced subgraph lies in $\widehat{K}$. By construction, each vertex in $\widehat{K}$ is adjacent to every other vertex in $\widehat{G}$, which contradicts the fact that the induced subgraph is disconnected. □

We next use Lemmas 2.4 and 2.5 to obtain NP-hardness results for STC and CD. Note that the NP-hardness for CD on $3K_1$-free graphs and $2K_2$-free graphs is already known [66]. Moreover, the NP-hardness of STC on $2K_2$-free graphs and C_4-free graphs is implied by the NP-hardness of STC on split graphs [111].

Lemma 2.6. STC *and* CD *remain NP-hard on H-free graphs if*

$$H \in \{3K_1, 2K_2, \text{co-diamond}, \text{co-paw}, 4K_1, \text{claw}, C_4, \text{diamond}, K_4\}.$$

Proof. We consider the following cases.

Case 1: $H \in \{3K_1, 2K_2, \text{co-diamond}, \text{co-paw}, 4K_1\}$. Clique remains NP-hard on $3K_1$-,$2K_2$-, co-diamond-, co-paw- and $4K_1$-free graphs since Independent Set is NP-hard on the complement graphs: K_3-, C_4-, diamond-, paw-, and K_4-free graphs [148]. By Lemma 2.4, $(G,k) \mapsto (\widehat{G}, m - (\binom{n^3}{2} + k \cdot n^3))$ is a polynomial-time reduction from Clique to STC and CD. By Lemma 2.2 we know that if G is H-free for $H \in \{3K_1, 2K_2, \text{co-diamond}, \text{co-paw}, 4K_1\}$, then $\widehat{G}$ is H-free as well. Thus, STC and CD remain NP-hard on H-free graphs.

Case 2: $H = \text{claw}$. Since both problems are NP-hard on $3K_1$-free graphs due to Case 1, it follows, that they are NP-hard on claw-free graphs.

Case 3: $H \in \{C_4, \text{diamond}, K_4\}$. There is a reduction from 3Sat to CD producing a C_4-, K_4-, and diamond-free CD instance [109]. By Theorem 2.1, there is an optimal cluster labeling for STC on diamond-free graphs, so the reduction works also for STC. Thus, STC and CD remain NP-hard on C_4-, K_4-, and diamond-free graphs. □

It remains to show NP-hardness on co-claw-free graphs. Since Independent Set can be solved in polynomial time on claw-free graphs [158], we can solve Clique on co-claw-free graphs in polynomial time. Thus, a reduction from Clique using the expanded graph as in the proof of Lemma 2.6 does not show NP-hardness in this case.

Lemma 2.7. STC *and* CD *remain NP-hard on co-claw-free graphs.*

Proof. We give a reduction from 3-Clique Cover. In 3-Clique Cover one is given a graph $G = (V,E)$ and the question is whether the vertex set V can be partitioned into three sets V_1, V_2, and V_3 such that each V_i is a clique in G. 3-Clique Cover is NP-hard even if the input graph is co-claw-free [90]. We first describe a reduction to STC. Afterwards, we argue why this is also a correct reduction to CD.

Construction. Let $G := (V,E)$ be a co-claw-free instance for 3-Clique Cover. We construct a co-claw-free STC instance $(G' := (V', E'), k)$ as follows.

We define three vertex sets K_1, K_2, and K_3. Every K_i consists of n^3 vertices $v_{1,i}, \dots, v_{n^3,i}$. We set $V' := V \cup K_1 \cup K_2 \cup K_3$. Moreover, we define edges from every vertex in $K_1 \cup K_2 \cup K_3$ to every vertex in V and edges of the form $\{v_{c,i}, v_{d,j}\}$, where $c \neq d$. We define the edge set E' as the union of those edges and E. Note that each K_i is a clique of size n^3 in G'. We complete the construction by setting $k := |E'| - (3 \cdot \binom{n^3}{2} + n^4)$. Throughout this proof, let $\mathcal{K} := K_1 \cup K_2 \cup K_3$ denote the union of the three cliques.

G' is co-claw-free. Before we prove the correctness of the reduction, we show that G' is co-claw-free. A co-claw consists of a triangle and an isolated vertex. Assume towards a contradiction that G' has a co-claw as an induced subgraph. Then, there is a triangle on some vertices u_1, u_2, and u_3, and there is a vertex w that is not adjacent to any vertex from the triangle on u_1, u_2, and u_3. We consider the following cases.

Case 1: $w \in \mathcal{K}$. Without loss of generality, let $w = v_{p,1}$ for some $p = 1, \dots, n^3$. By the construction of G', the vertex w has edges to every other vertex of G' except $v_{p,2}$ and $v_{p,3}$. Hence, there cannot be three vertices in G' which are not adjacent to w. A contradiction.

Case 2: $w \in V$. Since G has no induced co-claw, one of the triangle vertices belongs to $\mathcal{K}$. Without loss of generality, let $u_1 \in \mathcal{K}$. Then, by construction of G', there is an edge $\{w, u_1\} \in E'$. A contradiction.

Correctness. We next show that the reduction is correct. That is, we prove that G has a clique cover of size 3 if and only if there is an STC-labeling $L := (S_L, W_L)$ for G' with $|S_L| \geq 3 \cdot \binom{n^3}{2} + n^4$.

($\Rightarrow$) Let G have a clique cover of size 3. Then, there are three disjoint cliques V_1, V_2, and V_3 in G such that $V_1 \cup V_2 \cup V_3 = V$. We define an STC-labeling $L := (S_L, W_L)$ for G' with at least $(3 \cdot \binom{n^3}{2} + n^4)$ strong edges by setting $S_L := \bigcup_{i=1}^{3} E_{G'}(K_i \cup V_i)$. Since all $K_i \cup V_i$ are disjoint cliques, L is a cluster labeling. Thus, L is an STC-labeling. Moreover, there are at least $(3 \cdot \binom{n^3}{2} + n^4)$ edges in S_L since

$$\begin{aligned}\sum_{i=1}^{3} |E_{G'}(V_i \cup K_i)| &= \sum_{i=1}^{3} \left(\binom{|V_i|}{2} + \binom{n^3}{2} + |V_i| \cdot n^3 \right) \\ &\geq 3 \cdot \binom{n^3}{2} + n^3 \sum_{i=1}^{3} |V_i| \\ &= 3 \cdot \binom{n^3}{2} + n^4.\end{aligned}$$

($\Leftarrow$) Conversely, let $L = (W_L, S_L)$ be an STC-labeling for G' with $|S_L| \geq 3 \cdot \binom{n^3}{2} + n^4$. Assume towards a contradiction that G does not have a clique cover of size at most 3. We show that this is a contradiction to the fact that there are at least $3 \cdot \binom{n^3}{2} + n^4$ strong edges. We first provide an upper bound for the number of strong edges in $E_{G'}(\mathcal{K})$.

Claim 1. *There are at most* $3 \cdot \binom{n^3}{2}$ *strong edges in* $E_{G'}(\mathcal{K})$*.*

Proof. To prove the claim, we first show that each vertex $v \in \mathcal{K}$ has at most $n^3 - 1$ strong neighbors in $\mathcal{K}$. Without loss of generality assume that $v = v_{1,1} \in K_1$. By the construction of G', the vertex $v_{1,1}$ is adjacent to all of $\mathcal{K}$ except $v_{1,2}$, $v_{1,3}$, and itself. It thus has $n^3 - 1$ neighbors in each of the cliques K_1, K_2, and K_3. Assuming $v_{1,1}$ has more than $n^3 - 1$ strong neighbors, it follows by the pigeonhole principle that there is a number $d = 2, \dots, n^3$ such that two vertices $v_{d,i}$ and $v_{d,j}$ with $i \neq j$ are strong neighbors of $v_{1,1}$. Since $\{v_{d,i}, v_{d,j}\} \notin E'$, the edges $\{v_{d,i}, v_{1,1}\}$ and $\{v_{1,1}, v_{d,j}\}$ form a strong P_3 under L, which contradicts the fact that L satisfies STC.

Since $|\mathcal{K}| = 3 \cdot n^3$ and each vertex in $\mathcal{K}$ has at most $n^3 - 1$ strong neighbors in $\mathcal{K}$, there are at most $\frac{3n^3 \cdot (n^3-1)}{2} = 3 \cdot \binom{n^3}{2}$ strong edges between vertices in $E_{G'}(\mathcal{K})$. ◊

We next give an upper bound on the number of strong edges in $E_{G'}(\mathcal{K}, V)$ by using the assumption that G does not have a clique cover of size at most 3.

Claim 2. *There are at most* $n^4 - n^3$ *strong edges in* $E_{G'}(\mathcal{K}, V)$*.*

Proof. For $v_{c,i} \in \mathcal{K}$, let $\mathcal{N}(v_{c,i}) := \{w \in V \mid \{v_{c,i}, w\} \in S_L\}$ denote the set of strong neighbors of $v_{c,i}$ that lie in V. Obviously, each $\mathcal{N}(v_{c,i})$ forms a clique, since otherwise there is a strong P_3 under L.

Consider a triple of vertices $v_{c,1}, v_{c,2}, v_{c,3} \in V$ for some fixed $c = 1, \dots, n^3$. By the construction of G', those three vertices are pairwise nonadjacent. It follows that $\mathcal{N}(v_{c,1})$, $\mathcal{N}(v_{c,2})$ and $\mathcal{N}(v_{c,3})$ are pairwise disjoint. Otherwise, if there is a vertex $w \in \mathcal{N}(v_{c,1}) \cap \mathcal{N}(v_{c,2})$, then the edges $\{v_{c,1}, w\}$ and $\{w, v_{c,2}\}$ form a strong P_3 under L.

Since $\mathcal{N}(v_{c,1})$, $\mathcal{N}(v_{c,2})$, and $\mathcal{N}(v_{c,3})$ are disjoint cliques in V, the assumption that there is no clique cover of size 3 implies that for each triple $v_{c,1}$, $v_{c,2}$, $v_{c,3}$ we can find a vertex $w \in V$ such that $w \notin \mathcal{N}(v_{c,1}) \cup \mathcal{N}(v_{c,2}) \cup \mathcal{N}(v_{c,3})$. Consequently, there are at most $n - 1$ strong edges in $E_{G'}(V, \{v_{c,1}, v_{c,2}, v_{c,3}\})$. Since $\mathcal{K}$ consists of n^3 such triples, there are at most $n^3 \cdot (n-1) = n^4 - n^3$ strong edges in $E_{G'}(\mathcal{K}, V)$. ◊

Claims 1 and 2 together with the fact that $|E_{G'}(V)| = |E| = \binom{n}{2} < n^3$ give us the following inequality:

$$\begin{aligned}
|S_L| &= |S_L \cap E_{G'}(\mathcal{K})| + |S_L \cap E_{G'}(\mathcal{K}, V)| + |S_L \cap E_{G'}(V)| \\
&\leq 3 \cdot \binom{n^3}{2} + (n^4 - n^3) + \binom{n}{2} \\
&< 3 \cdot \binom{n^3}{2} + n^4.
\end{aligned}$$

This inequality contradicts the fact that $|S_L| \geq 3 \cdot \binom{n^3}{2} + n^4$. Consequently, G has a clique cover of size 3 or less, which proves the correctness of the reduction.

Correctness for CD. We next argue that the reduction described above is also a correct reduction to CD. The STC-labeling defined in the forward direction of the proof is a cluster labeling. This implies the existence of a solution for CD with at most k edge deletions. The backwards direction follows from the fact that a solution for CD with at most k edge deletions implies an STC-labeling with at most k weak edges which then implies that the instance of 3-CLIQUE COVER is a yes-instance. □

From Lemmas 2.2, 2.6, and 2.7 we obtain the following complexity dichotomy for STC and CD on H-free graphs.

Theorem 2.8. *The problems* STC *and* CD *are solvable in polynomial time on H-free graphs, if $H \in \{K_3, P_3, K_2{+}K_1, P_4, \text{paw}\}$. Both problems are NP-hard on H-free graphs, if $H \in \{3K_1, K_4, 4K_1, C_4, 2K_2, \text{diamond}, \text{co-diamond}, \text{claw}, \text{co-claw}, \text{co-paw}\}$.*

Due to Theorem 2.8, the classic complexity of STC and CD is the same on H-free graphs for all cases of H that were studied in this work. The most interesting open question might be: Is there a graph class on which one of the two problems is solvable in polynomial time while the other problem is NP-hard? The existence of such graph class might give new insights into the relation of these two problems and it would be interesting to compare the solution structure (the clustering and the graph spanned by the strong edges) on graphs belonging to this class.

The clique reductions using the expanded graph, the reduction behind Lemma 2.7, and the reduction provided by Komusiewicz and Uhlmann [109] are reductions to graphs where a cluster labeling is an optimal STC-labeling. Thus, the reductions used in this work might help to easily identify graph classes on which STC and CD are both NP-hard. Note that a necessary condition for STC and CD to have different classic complexity on a graph class Π is that STC and CD do not correspond on Π. Thus, if one considers H-free graphs, a good starting point are the graphs H that are not induced subgraphs of graphs on which a cluster labeling is not an optimal STC-labeling, like the ones given in Figure 2.1.

2.2 Parameterization by the Number of Strong Edges

We now study the parameterized complexity of STC and CD. In case of STC, the number k of weak edges and the number $\ell := m - k$ of strong edges are arguably the most natural parameterizations. Analogously, for CD, these two natural parameters are the number of edge deletions and the number of cluster edges.

Both problems are known to be fixed-parameter tractable when parameterized by k [14, 164]. Furthermore, both problems admit problem kernels with a linear number of vertices [77, 85, 23]. We revisit the solution size of STC in Chapter 4, where we study the parameterized complexity of STC with multiple relationship types—a generalization of STC—regarding this parameter.

In the following, we study STC and CD parameterized by ℓ. Since parameterization by ℓ is arguably natural, it is somehow surprising that it has—to the best of our knowledge—not been studied for CD so far. On the positive side, we show that both problems are fixed-parameter tractable for ℓ. On the negative side, we show that both problems do not admit a polynomial kernel for ℓ unless NP $\subseteq$ coNP/poly. Recall that independent from our work, Golovach et al. [71] also showed that STC parameterized by ℓ has no polynomial kernel unless NP $\subseteq$ coNP/poly.

2.2.1 Fixed-Parameter Algorithms

For CD, we obtain a fixed-parameter algorithm by a simple dynamic programming algorithm.

Theorem 2.9. CD *can be solved in* $\mathcal{O}(9^\ell \cdot \ell^2 n)$ *time.*

Proof. We first describe the algorithm and afterwards, we analyze its running time.

Algorithm. The first step of the algorithm is to compute a maximal matching M in G. If $|M| \geq \ell$, then answer yes. Otherwise, since M is maximal, the endpoints of M are a vertex cover of size less than 2ℓ. Let C denote this vertex cover and let $I := V \setminus C$ denote the independent set consisting of the vertices that are not endpoints of edges in M. We now decide if there is a cluster subgraph with at least ℓ cluster edges using dynamic programming over subsets of C. In the following, we assume that $I := \{1, \dots, n-|C|\}$. The dynamic programming table T has entries of the type $T[i, C']$ for all $i \in \{0, 1, \dots, n - |C|\}$ and all $C' \subseteq C$. Each entry stores the maximum number of cluster edges in a clustering of $G[C' \cup \{1, \dots, i\}]$. The entries are computed for increasing values of i and subsets C' of increasing size. Note that the entry for $i = 0$ corresponds to the clusterings that contain no vertices of I. For $i = 0$ and $C' = \emptyset$, we set $T[0, \emptyset] := 0$. The recurrence to compute an entry for $i = 0$ and $C' \neq \emptyset$ is

$$T[0, C'] = \max_{C'' \subseteq C' : C'' \text{is a clique}} T[0, C' \setminus C''] + \binom{|C''|}{2}.$$

The recurrence to compute an entry for $i \geq 1$ is

$$T[i, C'] = \max_{C'' \subseteq C' : C'' \cup \{i\} \text{is a clique}} T[i-1, C' \setminus C''] + \binom{|C''|+1}{2}.$$

After filling this table completely, we have a yes-instance if $T[n-|C|, C] \geq \ell$ and a no-instance otherwise. The corresponding set of edge-deletions can be found via traceback. The correctness follows from the observation that we consider all cases for the clique containing i since i is not adjacent to any vertex $j < i$.

Running Time. The running time of the algorithm can be seen as follows. A maximal matching can be computed in linear time. If the matching has size less than ℓ, we fill the dynamic programming table as defined above. For each $i \in \{0, 1, \ldots, n-|C|\}$ an entry $T[i, C']$ can be computed in $\mathcal{O}(2^{|C'|} \cdot \ell^2)$ time where the factor ℓ^2 corresponds to the time needed to determine whether $C'' \cup \{i\}$ is a clique. Consequently, all entries can be computed in $\mathcal{O}(n \cdot \sum_{j=0}^{2\ell} \binom{2\ell}{j} \cdot \ell^2) = \mathcal{O}(3^{2\ell}\ell^2 n)$ time. □

For STC, we combine a branching on the graph that is induced by a maximal matching with a dynamic programming over the vertex sets of this graph.

Theorem 2.10. STC *can be solved in* $\ell^{\mathcal{O}(\ell)} \cdot n^2$ *time.*

Proof. We first describe the algorithm and afterwards, we analyze its running time.

Algorithm. The initial step of the algorithm is to compute a maximal matching M in G. If $|M| \geq \ell$, then answer yes. Otherwise, the endpoints of M are a vertex cover of size less than 2ℓ since M is maximal. Let C denote this vertex cover and let $I := V \setminus C$ denote the independent set consisting of the vertices that are not endpoints of edges in M. The algorithm now has two further main steps. First, try all STC-labelings of $G[C]$ with at most ℓ strong edges. If there is one STC-labeling with ℓ strong edges, then answer yes. Otherwise, compute for each STC-labeling of $G[C]$ with fewer than ℓ edges, whether it can be extended to an STC-labeling of G with ℓ strong edges by labeling sufficiently many edges of $E(C, I)$ as strong.

Observe that $G[C]$ has $\ell^{\mathcal{O}(\ell)}$ STC-labelings with at most ℓ strong edges and that they can be enumerated in $\ell^{\mathcal{O}(\ell)}$ time: The graph $G[C]$ has less than $\binom{2\ell}{2} = \mathcal{O}(\ell^2)$ edges and we enumerate all subsets of the edge set of $G[C]$ that have size at most ℓ. Consider one such set S_C. In $\mathcal{O}(\ell^2)$ time, we can check whether $(S_C, E(C) \setminus S_C)$ is a valid STC-labeling. If this is not the case, then discard the current set. Otherwise, compute whether this labeling can be extended into a labeling of G with at least ℓ strong edges by using dynamic programming over subsets of C. In the following, we assume that $I := \{1, \ldots, n-|C|\}$. The dynamic programming table T has entries of the type $T[i, C']$ for all $i \in \{0, 1, \dots, n - |C|\}$ and all $C' \subseteq C$. Each entry stores the maximum number of strong edges in an STC-labeling of $G[C \cup \{1, \dots, i\}]$ in which the strong edges in $E(C)$ are exactly those of S_C and in which the set of strong neighbors of the vertices in $\{1, \dots, i\}$ is a subset of C'. For any STC-labeling, the set of strong neighbors $\mathcal{N}(i)$ of each vertex i has to fulfill three properties:

- $\mathcal{N}(i)$ is a clique.
- No vertex of $\mathcal{N}(i)$ has a strong neighbor in $C \setminus N(i)$.
- No vertex of $\mathcal{N}(i)$ has a strong neighbor in $I \setminus \{i\}$.

We call a set that fulfills the first two properties *valid* for i. We ensure the third property by the recurrence in the dynamic programming.

The entries are computed for increasing values of i and subsets C' of increasing size. For $i = 0$ and $C' = \emptyset$, we set $T[0, \emptyset] := |S_C|$. The recurrence to compute an entry for $i \geq 1$ is

$$T[i, C'] = \max_{C'' \subseteq C' : C'' \text{is valid for } i} T[i-1, C' \setminus C''] + |C''|.$$

After filling this table completely, we have a yes-instance if $T[n - |C|, C] \geq \ell$. Otherwise, the current STC-labeling for $G[C]$ cannot be extended to an STC-labeling for G with at least ℓ strong edges. If $T[n - |C|, C] < \ell$ for every choice of an STC-labeling for $G[C]$, then we have a no-instance. The correctness follows from the observation that we consider all valid sets for strong neighbors and that in the optimal solution for $G[i - 1, C' \setminus C'']$ no vertex from $\{1, \dots, i - 1\}$ has strong neighbors in C''.

Running Time. The running time of the algorithm can be seen as follows. A maximal matching can be computed greedily in linear time. If the matching has size less than ℓ, we fill the dynamic programming table as defined above. The number of partial labelings S_C is $\ell^{\mathcal{O}(\ell)}$. Given a partial labeling, for each $i \in \{0, \dots, n - |C|\}$ an entry $T[C', i]$ can be computed in $\mathcal{O}(2^{|C'|} \cdot \ell \cdot n)$ time where the factor $\ell \cdot n$ corresponds to the time needed to compute the subsets of C' that are valid for i. Consequently, the time needed to fill T for one partial labeling is $\mathcal{O}(n \cdot \sum_{j=0}^{2\ell} \binom{2\ell}{j} \cdot \ell n) = \mathcal{O}(3^{2\ell} \cdot \ell n^2)$. Thus, the overall running time is $\ell^{\mathcal{O}(\ell)} \cdot n^2$. $\square$

An algorithm with running time $\ell^{\mathcal{O}(\ell)} \cdot n^2$ is too slow to be practically relevant. A natural open question is thus, if the factor $\ell^{\mathcal{O}(\ell)}$ can be replaced by a factor of $2^{\mathcal{O}(\ell)}$. Note that for every partial labeling S_C, the dynamic programming table T can be filled in $\mathcal{O}(3^{2\ell} \cdot \ell n^2)$ time. Thus, the bottleneck of the running time in the algorithm behind Theorem 2.10 is to iterate over all partial labelings S_C. Therefore, if one can think of a way to decrease the number of partial labelings that can be extended to a solution, one might obtain a faster algorithm for STC parameterized by ℓ.

2.2.2 A Kernel Lower Bound for the Parameter ℓ

In the following, we provide a kernel lower bound for STC and CD parameterized by ℓ. More precisely, we provide a polynomial parameter transformation that implies that both problems do not admit a polynomial kernel when parameterized by ℓ unless NP $\subseteq$ coNP/poly. To prove the kernel lower bound we give a polynomial parameter transformation from Multicolored Clique. Recall that in Multicolored Clique one is given a graph $G = (V, E)$ together with a partition $(V_1, \ldots, V_t)$ of V, where every set V_r is an independent set. The question is if there exists a multicolored clique in G, that is, a clique containing one vertex from each set V_r. Multicolored Clique does not admit a polynomial kernel when parameterized by $\sum_{r=1}^{t-1} |V_r|$ due to Proposition 1.4.

Theorem 2.11. STC *parameterized by the number of strong edges ℓ does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly.

Proof. Construction. Let $G = (V, E)$ together with a partition $(V_1, \ldots, V_t)$ be an instance of Multicolored Clique. Without loss of generality we assume that all V_r except V_t have the same size z: if this is not the case we add $\max_{i \in \{1,\ldots,t\}} |V_i| - |V_r|$ isolated vertices to each class V_r, which only causes a polynomial blow up of the parameter $\sum_{r=1}^{t-1} |V_r|$. Moreover, let $v_{1,r}, v_{2,r}, \ldots, v_{z,r}$ be the vertices in V_r. We describe how to construct an equivalent instance $(G' = (V', E'), k)$ in polynomial time such that ℓ is polynomially bounded in $\sum_{r=1}^{t-1} |V_r|$.

For each of the class V_r with $r \in \{1, \ldots, t-1\}$, we define a family $\mathcal{K}_r$ of $z-1$ vertex sets $K_{1,r}, K_{2,r}, \ldots, K_{z-1,r}$, each of size t, and we add edges such that each $K \in \mathcal{K}_r$ becomes a clique. Throughout this proof, we refer to these vertex sets as *attached cliques of* V_r. For every $i \in \{1, \ldots, z-1\}$ we also add edges $\{u, v\}$ from all $u \in K_{i,r}$ to all $v \in V_r$. Figure 2.3 shows an example of this construction. We complete the construction by setting $k := |E'| - \left(\binom{t}{2} + (t-1)(z-1)\binom{t+1}{2}\right)$. Consequently, $\ell = |E'| - k = \binom{t}{2} + (t-1) \cdot (z-1)\binom{t+1}{2}$, which is polynomially bounded in $\sum_{r=1}^{t-1} |V_r|$.

Intuition. Before we prove the correctness of the polynomial parameter transformation, we provide some intuition. Every vertex in a class V_r with $r \in \{1, \ldots, t-1\}$ can be incident with up to t strong edges. This is the case if and only if it forms a strong clique with one of the attached cliques of V_r. Since $|V_r| = z$ and there are only $z-1$ attached cliques, one vertex v_r of each class V_r does not form such a strong clique with an attached clique. All these vertices v_r together with one vertex from V_t then correspond to a multicolored clique in G.

Correctness. We show that G is a yes-instance of Multicolored Clique if and only if (G', k) is a yes-instance of STC.

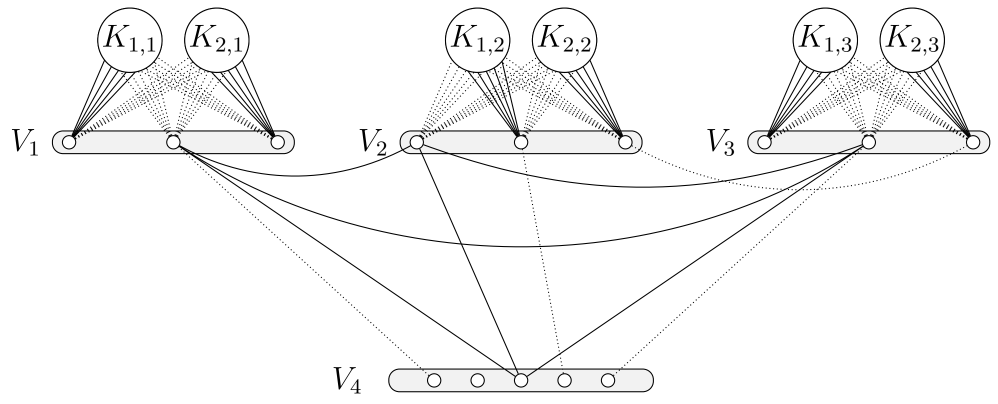


Figure 2.3: An example for the construction of G' described in the proof of Theorem 2.11 with $t = 4$ color classes. The upper part shows color classes V_1, V_2, and V_3 of size $z = 3$ and their attached cliques. The lower part shows color class V_4. The edges between the color classes are the edges from G. The dotted edges correspond to the weak edges of an optimal STC-labeling for G'.

($\Rightarrow$) Let R be a multicolored clique in G'. Without loss of generality we assume that $R = \{v_{z,1}, \ldots, v_{z,t-1}, u\}$ with $u \in V_t$. We define the labeling $L := (S_L, W_L)$ for G' by setting $S_L := E_R \cup E_{\mathcal{K}} \cup E_V$, where E_R is the set of all edges between vertices of R, $E_{\mathcal{K}}$ contains all edges between the vertices of the attached cliques, and E_V contains all edges between vertices $v_{i,r}$ with $i \leq z-1$ and $r \leq t-1$ and the vertices in the corresponding attached cliques $K_{i,r}$. Formally, this is

$$\begin{aligned}
E_R &:= E_{G'}(R),\\
E_{\mathcal{K}} &:= \bigcup_{\substack{i=1,\ldots,z-1\\ r=1,\ldots,t-1}} E_{G'}(K_{i,r}), \text{ and}\\
E_V &:= \bigcup_{\substack{i=1,\ldots,z-1\\ r=1,\ldots,t-1}} E_{G'}(\{v_{i,r}\}, K_{i,r}).
\end{aligned}$$

It remains to show that L is an STC-labeling with $|S_L| \geq \binom{t}{2} + (t-1)(z-1)\binom{t+1}{2}$.

The size of S_L is

$$\begin{aligned}|S_L| &= |E_R| + |E_{\mathcal{K}}| + |E_V| \\ &= \binom{t}{2} + (t-1)(z-1)\binom{t}{2} + (t-1)(z-1)t \\ &= \binom{t}{2} + (t-1)(z-1)\binom{t+1}{2}.\end{aligned}$$

We next show that there is no strong P_3 under L. We prove the slightly stronger statement that L is a cluster labeling. To this end, we consider two strong edges $e_1 \in S_L$ and $e_2 \in S_L$. We show that these edges are part of a strong triangle if they share an endpoint. Consider the following case distinction.

Case 1: $e_1, e_2 \in E_R$ or $e_1, e_2 \in E_{\mathcal{K}}$**.** Then, if e_1 and e_2 share one endpoint, all endpoints of e_1 and e_2 belong to the same strong clique.

Case 2: $e_1, e_2 \in E_V$**.** Then, if e_1 and e_2 share exactly one endpoint, they have the form $e_1 = \{v_{i,r}, w_1\}$ and $e_2 = \{v_{i,r}, w_2\}$ with $w_1, w_2 \in K_{i,r}$ for some $i \in \{1, \dots, z-1\}$ and $r \in \{1, \dots, t-1\}$ by the definition of E_V. Then, there exists a strong edge $\{w_1, w_2\} \in E_{\mathcal{K}}$.

Case 3: $(e_1 \in E_R$ and $e_2 \in E_{\mathcal{K}})$ or $(e_1 \in E_R$ and $e_2 \in E_V)$**.** Then, e_1 and e_2 do not share an endpoint by the construction of G'.

Case 4: $e_1 \in E_{\mathcal{K}}$ and $e_2 \in E_V$**.** Then, if e_1 and e_2 share exactly one endpoint, they have the form $e_1 = \{v_{i,r}, w_1\}$ and $e_2 = \{w_1, w_2\}$ with $w_1, w_2 \in K_{i,r}$ for some $i \in \{1, \dots, z-1\}$ and $r \in \{1, \dots, t-1\}$. Then, there exists a strong edge $\{v_{i,r}, w_2\}$.

We have thus shown that L is a cluster labeling. Consequently, L is an STC-labeling with $|S_L| = \binom{t}{2} + (t-1)(z-1)\binom{t+1}{2}$.

$(\Leftarrow)$ Conversely, let $L := (S_L, W_L)$ be an optimal STC-labeling with $|S_L| \geq \binom{t}{2} + (t-1)(z-1)\binom{t+1}{2}$. We show that G contains a multicolored clique. To this end, we use the following claim to show that we may make some assumptions about L regarding the edges between the classes V_r and their attached cliques.

Claim 1. *There exists an optimal STC-labeling $L^* := (S_{L^*}, W_{L^*})$ that satisfies the following properties for every tuple (r, K) with $r \in \{1, \dots, t-1\}$ and every $K \in \mathcal{K}_r$.*

a) There is at most one $v_{j,r} \in V_r$ that has a strong neighbor in K under L^*.

b) If there is a strong edge $\{v_{j,r}, w\} \in E_{G'}(V_r, K)$, then $E_{G'}(\{v_{j,r}\}, K) \subseteq S_{L^*}$

Proof. Recall that L is an optimal STC-labeling. If L satisfies *a)* and *b)* for all r and $K \in \mathcal{K}_r$, nothing more needs to be shown. Otherwise, let $r \in \{1, \dots, t-1\}$ such that there exists some $K \in \mathcal{K}_r$ where Properties *a)* or *b)* do not hold for the tuple (r, K). Then, there exists some $v \in V_r$ that has a strong neighbor $w \in K$. We define a new labeling $L' := (S_{L'}, W_{L'})$ by setting

$$S_{L'} := (S_L \setminus E_{G'}(V_r, K)) \cup E_{G'}(\{v\}, K)).$$

Then, L' satisfies $a)$ and $b)$ for K. It remains to show that L' is an optimal STC-labeling for G'.

We first show that L' is an STC-labeling. Let $\{v, u\} \in E_{G'}(\{v\}, K)$, and let $e \in E'$ be an edge that forms a P_3 with $\{v, u\}$. Then, $e \notin E_{G'}(K \cup \{v\})$, since $K \cup \{v\}$ is a clique in G'. We consider the following cases to show that e is weak under L'.

Case 1: $e = \{v', u\}$ with $v' \in V_r \setminus \{v\}$**.** Then, e is weak by the definition of L'.

Case 2: $e = \{v, y\}$ with $y \notin K$**.** Then, since $N_{G'}[u] = N_{G'}[w]$, the edge e also forms a P_3 with $\{v, w\}$ and thus, e is weak under L. Then, the edge e is also weak under L'.

Consequently, L' is an STC-labeling. We next show that L' is optimal. Observe that there are at most t edges in $S_L \cap E_{G'}(V_r, K)$, since otherwise—by pigeonhole principle—there is a vertex $w \in K$ that has two strong neighbors in V_r. Since V_r is an independent set, this contradicts the fact that L is an STC-labeling. Together with the fact that $|E_{G'}(\{v\}, K)| = t$, we have $|S_{L'}| \geq |S_L|$. Thus, L' is optimal.

Observe that by transforming L into L', we only changed the labels of edges in $E_{G'}(K, V_r)$. Thus, we can apply this transformation subsequently for every $r \in \{1, \ldots, t-1\}$ and $K \in \mathcal{K}_r$ so that we obtain an optimal STC-labeling L^* fulfilling Properties $a)$ and $b)$ for every tuple (r, K) with $r \in \{1, \ldots, t-1\}$ and $K \in \mathcal{K}_r$. $\Diamond$

Throughout the rest of this proof, we assume without loss of generality that L satisfies Properties $a)$ and $b)$ from Claim 1. We use this assumption to prove the next claim. In this claim we show that in every class V_r with $r \in \{1, \ldots, t-1\}$, there is exactly one vertex that has no strong neighbor in one of the attached cliques. We later use these vertices to show that there is a multicolored clique in G.

Claim 2. *In every class V_r with $r \in \{1, \ldots, t-1\}$, there are exactly $z-1$ vertices that form a strong clique under L of size $t+1$ with one of the attached cliques $K \in \mathcal{K}_r$.*

Proof. Let $r \in \{1, \ldots, t-1\}$. Observe that all edges inside the attached cliques are not part of any P_3 in G', and thus, they are strong under L. Note that since $|V_r| = z$ and $|\mathcal{K}_r| = z-1$, there are at most $z-1$ vertices in V_r that form a strong clique with one of the attached cliques. It remains to show that there are at least $z-1$ such vertices.

Assume towards a contradiction that there are two vertices $v, w \in V_r$ that do not form a strong clique with some $K \in \mathcal{K}_r$. From Claim 1 $b)$, we conclude that v and w

do not have any strong neighbor in $\bigcup_{K \in \mathcal{K}_r} K$. Furthermore, Claim 1 *a*) implies that there is at least one $K \in \mathcal{K}_r$ such that the vertices in K have no strong neighbors in V_r. Then, we can define a new labeling $L^+ = (S_{L^+}, W_{L^+})$ by

$$S_{L^+} := (S_L \setminus E_{G'}(\{v\}, \bigcup_{\substack{j=1,\dots,t \\ j \neq r}} V_j)) \cup E_{G'}(\{v\}, K).$$

Note that there are at most $t-1$ strong edges between v and the other vertices of G, since every V_j is an independent set. Together with the fact that $|E_{G'}(\{v\}, K)| = t$, we have $|S_{L^+}| > |S_L|$. Since every vertex in K has only weak neighbors in $V_r \setminus \{v\}$ and v has only weak neighbors in $V' \setminus K$ under L^+, the labeling L^+ is an STC-labeling. This contradicts the fact that L is an optimal STC-labeling. $\diamondsuit$

Observe that if a vertex $v \in V_r$ with $r \in \{1, \dots, t-1\}$ forms a strong clique with one of the attached cliques K, it has no strong neighbor outside K. From Claim 2 we know that there are $(t-1) \cdot (z-1) \cdot \binom{t+1}{2}$ strong edges in $E_{G'}(V' \setminus V_t)$. Since $|S_L| \geq \binom{t}{2} + (t-1)(z-1)\binom{t+1}{2}$, there are at least $\binom{t}{2}$ further edges that are strong under L. In the following, we describe how we can find a multicolored clique in G' using these $\binom{t}{2}$ strong edges.

By Claim 2, every class V_r with $r \in \{1, \dots, t-1\}$ contains a unique vertex that has no strong neighbor in one of the attached cliques. Without loss of generality, let $R := \{v_{1,1}, v_{1,2}, \dots, v_{1,t-1}\} \subseteq V_1 \cup \ldots \cup V_{t-1}$ be the set of these vertices. Since V_t is an independent set, each $v \in R$ has at most one strong neighbor in V_t. Therefore, there are at most $t-1$ strong edges in $E_{G'}(R, V_t)$. Since $\binom{t}{2} - (t-1) = \binom{t-1}{2}$, there are $\binom{t-1}{2}$ strong edges between the vertices of R. Consequently, $G'[R]$ is a complete subgraph and R is a strong clique under L.

Now let $U \subseteq V_t$ be the subset of vertices in V_t that have a strong neighbor in R. The set U is not empty since $|S_L| \geq \binom{t}{2} + (t-1)(z-1)\binom{t+1}{2}$. Thus, let $u \in R$. Then, u has edges to each $v \in R$. Otherwise, if $\{u, v\} \in E'$ and $\{u, w\} \notin E'$ for some $v, w \in R$, the edges $\{u, v\}$ and $\{w, v\}$ form a strong P_3 under L. Therefore, $R \cup \{u\}$ is a multicolored clique in G'. $\square$

The proof of Theorem 2.11 also implies that CD has no polynomial kernel with respect to the parameter ℓ: The strong edges in the STC-labeling obtained in the forward direction of the proof form a disjoint union of cliques and the converse direction follows from the fact that a cluster subgraph with at least ℓ cluster edges implies an STC-labeling with at least ℓ strong edges which then implies that the Multicolored Clique instance is a yes-instance. Thus, we obtain the following kernel lower bound for CD.

Corollary 2.12. CD *parameterized by the number of cluster edges* $\ell := |E| - k$ *does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly.

2.3 Concluding Remarks

We provided a study on the correspondence and classic complexity of STC and CD on some restricted graph classes. Moreover, we initiated the study of the parameterized complexity of STC and CD for the parameter ℓ.

This chapter is based on parts of the publication "On the relation of strong triadic closure and cluster deletion" which appeared in *Algorithmica* [77]. Besides the results in this chapter, the original publication provides the first linear-vertex kernel of order $4k$ for STC, where k is the number of weak edges. In Chapter 4 of this work, we study the solution size of STC as parameterization for Multi-STC and its list variants. Among other results, we provide a more general problem kernelization that implies a $4k$-vertex kernel for STC.

Open Questions. In the beginning of this chapter, we studied the correspondence of STC and CD on restricted graph classes. Analyzing the correspondence between STC and CD increases our understanding of strong triadic closure as a property of edge-labeled graphs and helps us to evaluate to which extent the inference of strong ties using the strong triadic closure property differs from clustering the social network. A natural question is to ask for further graph classes on which the two problems correspond, as we discussed at the end of Section 2.1.1. An even bigger goal is to achieve a complete characterization of the graphs in which an optimal solution of CD does not correspond to an optimal solution of STC.

A related open question concerns a variant of STC that is called Strong Triadic Closure with Edge Insertions (STC+). In STC+, one also aims to find an STC-labeling with at most k weak edges, but—in contrast to STC—one is allowed to connect two vertices with a weak edge that were nonadjacent in the input graph. Formally, given a graph $G = (V, E)$ and an integer k, one aims to find an STC-labeling $L = (S_L, W_L \cup E^+)$ of the graph $G^+ = (V, E \cup E^+)$ such that $|W_L \cup E^+| \leq k$. This problem is closely related to the well-studied graph clustering problem Cluster Editing (CE) [182, 12, 14, 13, 109]. In CE, one is given a graph G and an integer k and the question is if G can be transformed into a cluster graph with at most k edge modifications. Here, an edge modification is either an insertion or a deletion of an edge. Analogously to STC and CD, the problems STC+ and CE are related in the following sense: A solution of CE with k edge modifications implies the existence of a solution of STC+ with $|W_L \cup E^+| = k$. Intuitively, the edge deletions in CE correspond to labeling edges as weak, and the edge insertions in CE correspond to adding weak edges to the graph. We would like to repeat one open question posed by Neuendorf [138]: Does a solution for STC+ imply the existence of a solution for CE of the same size? In other words, is there an example of an instance where an optimal solution for STC+ is strictly smaller than an optimal solution for CE? In the two examples given in Figure 2.1 a solution of STC+ corresponds to an optimal CE solution. Thus, the examples of non-correspondence for STC and CD do not work for STC+ and CE. A correspondence between STC+ and CE might give new insights into CLUSTER EDITING: If an edge deletion corresponds to a weak edge labeling, the solution size is not affected by P_3s that arise from deleting one edge in a triangle. This observation might lead to new algorithmic approaches for the well-studied CLUSTER EDITING problem like new ILP formulations or new FPT algorithms.

We also provided a complexity dichotomy of the classic complexity on H-free graphs for all graphs H that have at most four vertices. For each choice of H, either both problems are NP-hard or solvable in polynomial time. Recall that an interesting open question might be if one could find a graph class on which one problem is polynomial-time solvable while the other one is NP-hard. The question appears to be interesting since the existence of such graph class might give new insights into the relation of STC and CD and it would be interesting to compare the solution structure on these instances. Furthermore, one could investigate if there exist further graph classes on which STC or CD are solvable in polynomial time. Besides extending our knowledge of the classic complexity of these problems, identifying such graph class Π might motivate the study of new parameterizations for STC and CD. For example: Is STC (or CD) FPT when parameterized by the vertex deletion distance to a graph that belongs to Π? These parameterizations are also known as *distance-from-triviality parameterizations.* In Chapter 5 of this work, we study a distance-from-triviality parameterization for MULTI-STC and a related edge coloring problem.

Concerning the parameterized complexity, a natural open question is to ask if there are more efficient algorithms for STC when parameterized by the number ℓ of strong edges. The algorithm behind Theorem 2.10 has running time $\ell^{\mathcal{O}(\ell)} \cdot n$. Can this be improved to $2^{\mathcal{O}(\ell)} \cdot n^{\mathcal{O}(1)}$ time? Recall that the running time bottleneck of the presented algorithm is to enumerate all labelings of edges between the endpoints of a maximal matching in the input graph. If one could efficiently identify a small subset $\mathcal{L}$ of these labelings such that it is sufficient to check whether one labeling in $\mathcal{L}$ can be extended to a solution, there would be a running time improvement. For parameterization by the number k of weak edges, it is open whether we can solve STC faster than in $\mathcal{O}(1.28^k + nm)$ time, the running time that is implied by the parameter-preserving reduction to Vertex Cover [164]. It seems that any faster algorithm would need to use new insights into STC.

Recall that an STC instance is a yes-instance if the input graph has a maximal matching M of size at least ℓ. We exploit this relationship between ℓ and $|M|$ in the algorithm behind Theorem 2.10 since we iterate over every partial labeling of edges connecting the at most 2ℓ endpoints of M. Concerning the relationship between the size of a maximal matching M and the number of strong edges ℓ, we would like to restate the following open question posed by Golovach et al. [71]: is there an FPT algorithm for STC when parameterized by $\ell - |M|$? Golovach et al. proved that STC is FPT for $\ell - |M|$ if the maximum degree of the input graph is 4. Thus, a further step towards answering this question is to determine if STC is FPT when parameterized by $\ell - |M| + \Delta$, where Δ denotes the maximum degree of the input graph.

Consider problem kernelization for STC. Since the vertex cover number of a graph is never larger than $2|M|$, Theorem 2.11 implies that STC does not admit a polynomial kernel for vc unless NP $\subseteq$ coNP/poly. This excludes the existence of polynomial kernels for many structural graph parameters. Furthermore, the relationship between ℓ and $|M|$ implies that STC has a simple polynomial kernel when parameterized by $\ell + \Delta$, where Δ denotes the maximum degree of the input graph. Is it possible to replace ℓ by a smaller parameter like $\ell - |M|$? Recall that it is not yet known whether STC is FPT when parameterized by $\ell - |M| + \Delta$.

Concerning approximability, a factor-2 approximation for minimizing the number of weak edges in STC is implied by the reduction to Vertex Cover [164]. It is open whether there is a polynomial-time approximation algorithm with an approximation factor smaller than 2.

Finally, we would like to state a practical question in context of STC. Given a real-world social network $G = (V, E)$ and an optimal STC-labeling $L = (S_L, W_L)$, what do the strong components look like? A *strong component* is a connected component in (V, S_L). Rozenshtein et al. [156] studied a variant of STC where one additionally has pre-labeled strong cliques in a graph. Their experimental evaluations demonstrated that showing only the strong edges significantly simplifies the graph. Maybe there are real world applications in the field of social network analysis, where the following framework delivers practically reasonable results:

1. Compute the strong edges in the network, and

2. solve a problem on the simplified network, where only the strong edges are shown.

However, one has to be careful, as it is possible that the network becomes too simple when only considering the strong edges: If—for example—all strong components were cliques in most of the real-world instances, there would be no actual difference between STC and CD. In this case, there are apparently no benefits in using the strong triadic closure property instead of a classic graph clustering approach.

Chapter 3

Hardness of Multicolored Strong Triadic Closure Problems

We study the classic and fine-grained complexity of triadic closure problems with multiple strong colors and its list variants as defined in the introduction of Part II. Recall that Sintos and Tsaparas [164] introduced the extension of Strong Triadic Closure where agents may have c different types of strong relationships. In this model, the strong triadic closure property only applies to edges of the same strong type.

Recall that, given an undirected graph $G = (V, E)$, a *c-labeling* as given in Definition II.2 is a partition $L = (S_L^1, \dots, S_L^c, W_L)$ of the edge set into $c + 1$ color classes. Such a labeling is an *STC-labeling* if there is no pair of edges $\{u, v\}$ and $\{v, w\}$ with $\{u, w\} \notin E$ belonging to the same strong color class S_L^i. The computational problem is then defined as follows.

Multi Strong Triadic Closure (Multi-STC)
Input: An undirected graph $G = (V, E)$ and integers $c \in \mathbb{N}$ and $k \in \mathbb{N}_0$.
Question: Is there a c-colored STC-labeling L with $|W_L| \leq k$?

Besides Multi-STC, we also study the list variants VL Multi-STC and EL-Multi-STC in this chapter. Recall that in VL-Multi-STC one is given a graph G, two integers c and k, and vertex lists Λ (Definition II.3). Analogously, in EL-Multi-STC one is given G, c, k, and edge lists Ψ (Definition II.4). A vertex list $\Lambda(v)$ for some vertex v is the set of strong colors that can be assigned to edges incident with v, and an edge list $\Psi(e)$ for some edge e is the set of strong colors that can be assigned to the edge e. In both problems, the question is whether there exists an STC-labeling with c strong colors and at most k weak edges that satisfies the additional restrictions on strong colors imposed by the lists.

Our Results. We study the classic and fine-grained complexity of Multi-STC and its two generalizations.

First, we observe that previous results on the Edge Coloring problem give, for every fixed c, a dichotomy of Multi-STC into NP-hard and polynomial-time solvable instances with respect to the maximum degree of the input graph. In particular, for all $c \geq 3$, Multi-STC is NP-hard even if $k = 0$. The NP-hardness of Multi-STC even if $k = 0$ implies that for all three problems, there is presumably no polynomial-time approximation algorithm.

Second, we show that the Exponential Time Hypothesis (ETH) implies a strong lower bound for VL-Multi-STC and EL-Multi-STC. More precisely, we show that, assuming the ETH, there is no $2^{o(n^2)}$-time algorithm for VL-Multi-STC and EL-Multi-STC even if $k = 0$ and $c \in \mathcal{O}(n)$. This result is achieved by a compression of 3-CNF formulas ϕ where each variable occurs in a constant number of clauses into graphs with $\mathcal{O}(\sqrt{|\phi|})$ vertices. Since the lower bound holds on instances with $k = 0$, it is not possible to compute an approximation within $2^{o(n^2)}$ time for VL-Multi-STC and EL-Multi-STC unless ETH fails.

3.1 Classic Complexity of Multi-STC

It was claimed that Multi-STC is NP-hard for every fixed c since in the Gallai graph this is exactly the NP-hard problem Odd Cycle Transversal (in case of $c = 2$) or Vertex c-Coloring (in case of $c \geq 3$) [164]. It is not known, however, whether these problems are NP-hard on Gallai graphs. Instead, the NP-hardness can be observed from hardness results for Edge Coloring. In Edge Coloring, one is given a graph G and a number of colors c and the question is whether the edges can be colored with colors $1, \dots, c$ such that no two incident edges receive the same color.

Recall that from a more abstract point of view, in Multi-STC we aim to label the edges in a way such that no two edges that form an induced P_3 receive the same strong color. In triangle-free graphs every pair of incident edges forms an induced P_3. Consequently, on instances with a triangle-free graph and $k = 0$, Multi-STC is equivalent to Edge Coloring. Using known results for Edge Coloring, this gives the following dichotomy of the complexity of Multi-STC on bounded-degree graphs.

Theorem 3.1. Multi-STC *exhibits the following complexity dichotomy on bounded-degree graphs:*

a) For $c = 1$, Multi-STC *is NP-hard on graphs with maximum degree at least 4 and solvable in polynomial time when the maximum degree is at most 3.*

b) For $c = 2$, Multi-STC *is NP-hard on graphs with maximum degree at least 3 and solvable in polynomial time when the maximum degree is at most 2.*

c) For every $c \geq 3$, Multi-STC *is NP-hard on instances with maximum degree at least c even if $k = 0$ and it can be solved in polynomial time if the maximum degree is at most $c - 1$.*

Proof. Statement *a*) is a known result for STC [110]. For Statement *c*), the NP-hardness follows from the classic result that Edge Coloring is NP-hard for every fixed $c \geq 3$ even if the input graph is triangle-free [126]. Furthermore, due to Vizing's Theorem [178], every graph with maximum degree Δ can be edge-colored with $\Delta+1$ colors such that no two incident edges recieve the same color. Thus, instances with maximum degree at most $c - 1$ are trivial yes-instances for Multi-STC. To show Statement *b*), we reduce from Edge Coloring, which is NP-hard even if $c = 3$ and the input graph is cubic and triangle-free [90].

Construction: Let $I := (G, c)$ be an instance of Edge Coloring where $c = 3$ and G is a cubic and triangle-free graph. Note that G has an even number of vertices. We define $J := (G, 2, k)$ with $k := \frac{n}{2}$.

Correctness: We show that I is a yes-instance of Edge Coloring if and only if J is a yes-instance of Multi-STC.

($\Rightarrow$) Let I be a yes-instance of Edge Coloring. Hence, one can assign the colors 1, 2, and 3 to the edges of G in a way that no two incident edges receive the same color. Then, the fact that G is cubic implies that each color class is a perfect matching in G. We let $L := (S^1_L, S^2_L, W_L)$ be a labeling where S^1_L contains the edges that are assigned color 1, S^2_L contains the edges that are assigned color 2, and W_L contains the edges that are assigned color 3. Note that $|W_L| = \frac{n}{2} = k$. Furthermore, L is an STC-labeling since no two incident edges in G received the same color. Consequently, J is a yes-instance of Multi-STC.

($\Leftarrow$) Let J be a yes-instance of Multi-STC. Then, there exists an STC-labeling $L := (S^1_L, S^2_L, W_L)$ for G such that $|W_L| \leq \frac{n}{2}$. Note that the strong color classes S^1_L and S^2_L each form a matching in G, since G is triangle-free. We next show that the edges in W_L also form a matching. Assume towards a contradiction that there are two edges in W_L that share an endpoint. Then, $|W_L| \leq \frac{n}{2}$ implies that there is one vertex v that is not incident with some edge in W_L. Since G is cubic, this implies that v is incident with two edges of the same strong color. This contradicts the fact that S^1_L and S^2_L are matchings in G. Consequently, W_L is a matching. Then, since the edges of G can be partitioned into three pairwise disjoint matchings, I is a yes-instance of Edge Coloring. □

Note that, for $c = 2$, Multi-STC parameterized by k is FPT since it is sufficient to solve Odd Cycle Transversal in the Gallai graph which is FPT with respect to k [152]. Thus, the dichotomy from Theorem 3.1 is also tight with regard to the complexity for instances with constant k.

As mentioned above, the proof of Theorem 3.1 is based on known results for Edge Coloring. In Chapter 5 of this work, we revisit the close relation between Multi-STC and Edge Coloring and study a structural parameterization that is based on Vizing's Theorem [178].

3.2 Fine-Grained Complexity of Multi-STC with Lists

Next, we provide a stronger hardness result for VL-Multi-STC and EL-Multi-STC: we show that they are unlikely to admit a single-exponential-time algorithm with respect to the number n of vertices. Thus, the simple algorithm behind Proposition II.7 is optimal in the sense that m cannot be replaced by n in dense graphs.

We remark that for List-Edge Coloring, an ETH-based lower bound of $2^{o(|V|^2)}$ has been shown recently [118]. In List-Edge Coloring one is given a graph G, a number c of colors, and a list $\Psi(e)$ of possible colors for each edge e. The question is whether the colors $1, 2, \dots, c$ can be assigned to the edges of G such that no two incident edges receive the same color and each edge receives a color from its list. Note that List-Edge Coloring and EL-Multi-STC with $k = 0$ correspond if the input graph is triangle-free, and that there is a natural reduction that transforms an instance of List-Edge Coloring into an equivalent instance where the input graph is triangle-free: Replace all edges with the gadget shown in Figure 3.1. However, this reduction adds $\Omega(cm)$ vertices and therefore does not imply the desired lower bound since the ETH reduction for List-Edge Coloring outputs a graph where the number of edges is quadratic in the number of vertices [118].

We provide a lower bound for VL-Multi-STC. The construction behind the $2^{o(|V|^2)}$ lower bound for List-Edge Coloring contains triangles with edge lists that cannot be easily modeled with vertex lists. In summary, we are not aware of any direct reduction from List-Edge Coloring to VL-Multi-STC that would transfers the desired lower bound to VL-Multi-STC.

Our lower bound for VL-Multi-STC is based on a reduction from 3-SAT. This reduction is inspired by a reduction used to show that Rainbow Coloring

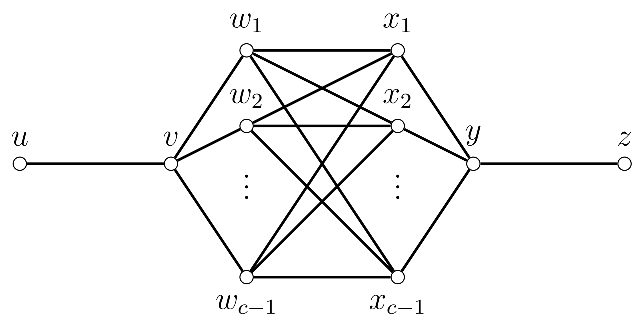


Figure 3.1: A graph consisting of edges $\{u, v\}$, $\{y, z\}$, $\{v, w_i\}$, $\{y, x_i\}$, and $\{w_i, x_j\}$ for every $i, j \in \{1, \ldots, c-1\}$. It is easy to see that in every proper edge-labeling with colors $1, \ldots, c$, the edges $\{u, v\}$ and $\{y, z\}$ must receive the same color. The graph can thus be used as a gadget to transform an instance of Edge-List Coloring into an equivalent triangle-free instance: we replace every edge e with list $\Psi(e)$ by such gadget, set $\Psi(\{u, v\}) := \Psi(\{y, z\}) := \Psi(e)$, and assign full lists to the other edges of the gadget.

cannot be solved in $2^{o(n^{3/2})}$ time under the ETH [117]. Rainbow Coloring is a mildly related problem where the input is a graph and an integer k and the question is, whether the edges can be colored with k distinct colors such that every pair of vertices is connected by a rainbow path, that is, a path where all edges on the path have distinct colors.

As it is the case in most 3-SAT reductions, our constructions has a variable part and a clause part. The compression of the variable part in our reduction works mostly analogously to the reduction to Rainbow Coloring [117]. However, in VL-Multi-STC we have vertex lists that need to be defined carefully. For the clause part of the reduction, we use equitable colorings [87, 102] to achieve an even stronger compression and thus a lower bound with a quadratic function in the exponent for VL-Multi-STC. Intuitively, an equitable coloring is a vertex coloring such that all color classes have roughly the same size.

Theorem 3.2. *If the ETH is true, then* VL-Multi-STC *cannot be solved in* $2^{o(|V|^2)}$ *time even if restricted to instances with* $k = 0$*.*

Proof. We give a reduction from 3-SAT to VL-Multi-STC such that the resulting graph has $\mathcal{O}(\sqrt{|\phi|})$ vertices, where ϕ is the input formula and $|\phi|$ is the number of variables plus the number of clauses. By the Sparsification Lemma [96], a $2^{o(|\phi|)}$-time algorithm for 3-SAT defeats the ETH and, hence, a $2^{o(|V|^2)}$-time algorithm for VL-Multi-STC defeats the ETH as well.

Below, we use n for the number of variables in ϕ. We can furthermore assume that, in the formula ϕ, each variable occurs in at most four clauses, since arbitrary 3-CNF formulas can be transformed in polynomial time to equivalent 3-CNF formulas fulfilling this restriction while only increasing the formula length by a constant factor [170]. Observe that in such instances the number of clauses in ϕ is at most $\frac{4}{3}n$.

Let ϕ be a 3-CNF formula with a set $X = \{x_1, \dots, x_n\}$ of n variables and a set $\mathcal{C} := \{C_1, \dots, C_m\}$ of $m \leq \frac{4}{3}n$ clauses. Let C_j be a clause and let x_i be a variable occurring in C_j. We define the *occurrence number* $\Omega(C_j, x_i)$ as the number of clauses in $\{C_1, C_2, \dots, C_j\}$ that contain x_i. Note that $\Omega(C_j, x_i)$ is only defined if x_i occurs in C_j. Intuitively, $\Omega(C_j, x_i) = r$ means that the rth occurrence of variable x_i is the occurrence in clause C_j. Since each variable occurs in at most four clauses, we have $\Omega(C_j, x_i) \in \{1, 2, 3, 4\}$.

We describe in three steps how to construct an equivalent instance

$$(G = (V, E), c = 9n + 4, k = 0, \Lambda)$$

for VL-Multi-STC such that $|V| \in \mathcal{O}(\sqrt{n})$. First, we describe the variable gadget. Second, we describe the clause gadget. In a third step, we describe how these two gadgets are connected. Before we present the formal construction, we give some intuition.

Intuition. The strong colors $1, \dots, 8n$ represent the truth assignments of the occurrences of the variables. We refer to these strong colors as T_i^r, F_i^r with $i \in \{1, \dots, n\}$ and $r \in \{1, 2, 3, 4\}$. The idea is that a strong color T_i^r represents a 'true'-assignment and F_i^r represents a 'false'-assignment of the rth occurrence of a variable $x_i \in X$. The strong colors $8n + 1, \dots, 9n + 4$ are auxiliary strong colors which we need for the correctness of our construction. We refer to these strong colors as $R_1, \dots, R_n$ and Z_1, Z_2, Z_3, Z_4. In the variable gadget, there are four distinct edges e_1, e_2, e_3, e_4 for each variable x_i representing the (at most) four occurrences of the variable x_i. We define vertex lists that ensure that every such edge e_r can only be labeled with the strong colors T_i^r and F_i^r. The coloring of these edges represents a truth assignment to the variable x_i. In the clause gadget, there are m distinct edges such that the coloring of these edges represents a choice of literals that satisfies ϕ. The edges between the two gadgets make the values of the literals from the clause part consistent with the assignment of the variable part. The construction consists of five layers of vertices. In the variable gadget we have an upper layer, a middle layer, and a down layer (U^X,M^X and D^X). In the clause gadget we have an upper and a down layer ($U^{\mathcal{C}}$ and $D^{\mathcal{C}}$). Figure 3.2 shows a sketch of the construction.

The Variable Gadget. The vertex set of the variable gadget consist of an upper layer, a middle layer, and a down layer. The vertices in the middle layer and the

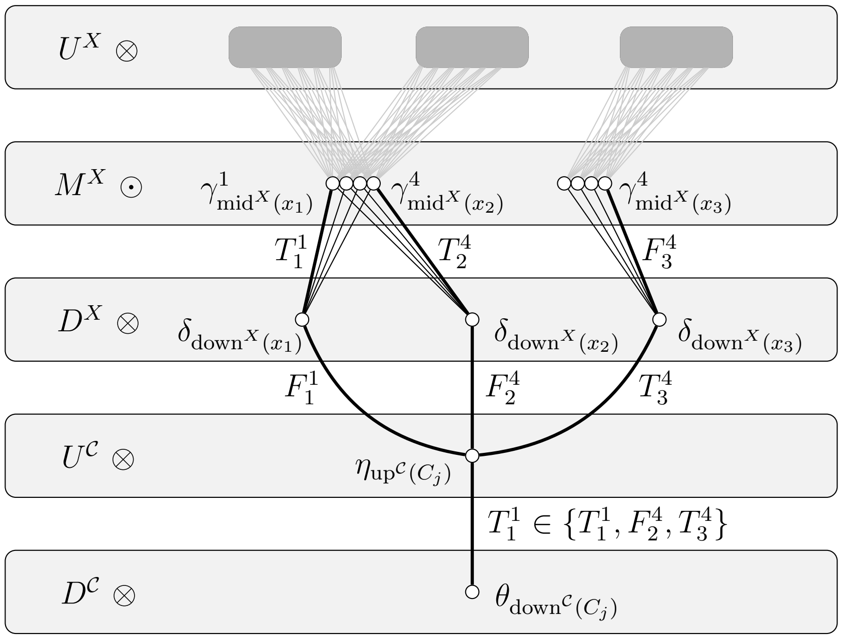

$U^X \otimes$
$M^X \odot$
$\gamma^1_{\text{mid}^X(x_1)}$
$\gamma^4_{\text{mid}^X(x_2)}$
$\gamma^4_{\text{mid}^X(x_3)}$
T^1_1
T^4_2
F^4_3
$D^X \otimes$
$\delta_{\text{down}^X(x_1)}$
$\delta_{\text{down}^X(x_2)}$
$\delta_{\text{down}^X(x_3)}$
F^1_1
F^4_2
T^4_3
$U^{\mathcal{C}} \otimes$
$\eta_{\text{up}^{\mathcal{C}}(C_j)}$
$T^1_1 \in \{T^1_1, F^4_2, T^4_3\}$
$D^{\mathcal{C}} \otimes$
$\theta_{\text{down}^{\mathcal{C}}(C_j)}$

Figure 3.2: An example of the construction. The grey rectangles correspond to the layers of the construction. The dark rectangles in layer U^X represent vertices $\alpha_t^{(r,r')}$ with the same value of t, $\otimes$ a clique, and $\odot$ an independent set. The edge $\{\eta_{\text{up}^{\mathcal{C}}(C_j)}, \theta_{\text{down}^{\mathcal{C}}(C_j)}\}$ represents a clause $C_j = (x_1 \vee \overline{x_2} \vee x_3)$ with $\Omega(C_j, x_1) = 1$ and $\Omega(C_j, x_2) = \Omega(C_j, x_3) = 4$. The edge $\{\gamma^1_{\text{mid}^X(x_1)}, \delta_{\text{down}^X(x_1)}\}$ has strong color T^1_1 which models an assignment where x_1 is true, which satisfies C_j. Due to the compression, we may have $\text{mid}(x_1) = \text{mid}(x_2)$ and therefore x_1 and x_2 may share the four middle vertices.

down layer form a *variable-representation gadget*, where each edge between the two parts represents one occurrence of a variable. The vertices in the upper layer form a *variable-soundness gadget*, which we need to ensure that for each variable either all occurrences are assigned ‘true’ or all occurrences are assigned ‘false’. For an illustration of the variable-representation and the variable-soundness gadget for some variable x_i see Figure 3.3.

We start by describing the variable-representation gadget. Let

$$M^X := \{\gamma_t^r \mid t \in \{1, \ldots, \lceil \sqrt{n} \rceil\}, r \in \{1,2,3,4\}\}$$

be the set of middle vertices, and let

$$D^X := \{\delta_t \mid t \in \{1, \ldots, \lceil \sqrt{n} \rceil + 9\}\}$$

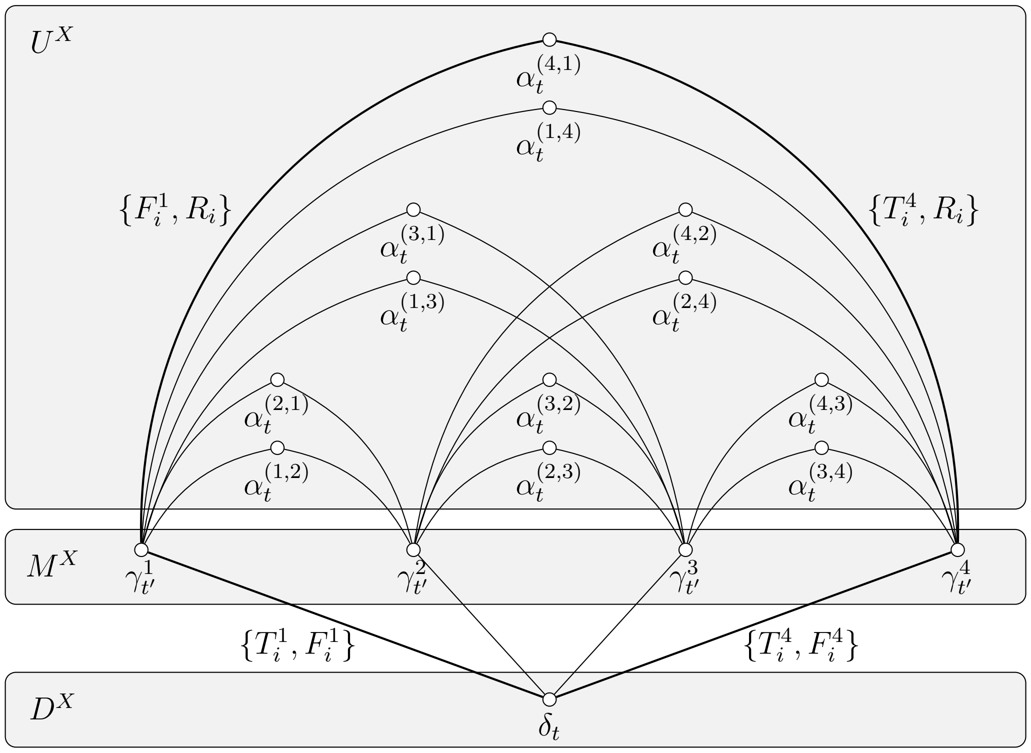


Figure 3.3: The variable-representation and the variable-soundness gadget for one variable $x_i \in X$ such that $\mathrm{down}^X(x_i) = \mathrm{up}^X(x_i) = t$ and $\mathrm{mid}^X(x_i) = t'$ with the possible colors for the edges $\{\delta_t, \gamma_{t'}^1\}$, $\{\delta_t, \gamma_{t'}^4\}$, $\{\alpha_t^{(4,1)}, \gamma_{t'}^1\}$, and $\{\alpha_t^{(4,1)}, \gamma_{t'}^4\}$. Note that labeling $\{\delta_t, \gamma_{t'}^1\}$ with the strong color F_i^1 and labeling $\{\delta_t, \gamma_{t'}^4\}$ with the strong color T_i^4 causes a P_3 with some strong color. The grey rectangles correspond to the layers U^X, M^X, and D^X of the construction.

be the set of down vertices.

We add edges such that D^X becomes a clique in G. To specify the correspondence between the variables in X and the edges in the variable-representation gadget, we define below the two mappings $\mathrm{mid}^X : X \to \{1, \dots, \lceil\sqrt{n}\,\rceil\}$ and $\mathrm{down}^X : X \to \{1, \dots, \lceil\sqrt{n}\,\rceil + 9\}$. Then, for each $x_i \in X$ we add four edges $\{\gamma^r_{\mathrm{mid}^X(x_i)}, \delta_{\mathrm{down}^X(x_i)}\}$ for $r \in \{1, 2, 3, 4\}$. We carefully define the two mappings mid^X, down^X and the vertex lists $\Lambda(v)$ for every $v \in M^X \cup D^X$ of the variable-representation gadget.

Intuitively, the chosen truth assignment for each variable will be transmitted to a clause by edges between the variable and clause gadgets. To ensure that each such transmitter edge is used for exactly one occurrence of one variable, we first define the *variable-conflict graph* $H_\phi^X := (X, \mathrm{Confl}^X)$ by $\mathrm{Confl}^X := \{\{x_i, x_j\} \mid x_i \text{ and } x_j \text{ occur in the same clause } C \in \mathcal{C}\}$, which we use to define mid^X and down^X. Since every variable of ϕ occurs in at most four clauses, the maximum degree of H_ϕ^X is at most 8. Hence, there is a proper vertex 9-coloring $\chi : X \to \{1, 2, \dots, 9\}$ for H_ϕ^X which we compute in polynomial time by a folklore greedy algorithm. We end up with nine color classes $\chi^{-1}(1), \dots, \chi^{-1}(9)$. Then, we partition each color class $\chi^{-1}(i)$ into $\left\lceil \frac{|\chi^{-1}(i)|}{\lceil \sqrt{n} \rceil} \right\rceil$ groups arbitrarily such that each group has size at most $\lceil \sqrt{n} \rceil$. Let s be the overall number of such groups and let $\mathcal{S} := \{S_1, S_2, \dots, S_s\}$ be the family of all such groups of vertices in H_ϕ^X (each corresponding to a pair of a color $i \in \{1, \dots, 9\}$ and a group in $\chi^{-1}(i)$). Consider the following claim about the sizes of the groups and the total number of groups.

Claim 1. *For the family $\mathcal{S} := \{S_1, S_2, \dots, S_s\}$ of groups of vertices in H_ϕ^X, it holds that*

a) $|S_i| \leq \lceil \sqrt{n} \rceil$ *for each $i \in \{1, \dots, s\}$, and*

b) $s \leq \lceil \sqrt{n} \rceil + 9$.

Proof. Inequality *a*) holds due to the definition of the groups S_i for $i \in \{1, \dots, s\}$. Inequality *b*) can be shown as follows:

$$s = \sum_{j=1}^{9} \left\lceil \frac{|\chi^{-1}(j)|}{\lceil \sqrt{n} \rceil} \right\rceil \leq 9 + \frac{\sum_{j=1}^{9} |\chi^{-1}(j)|}{\lceil \sqrt{n} \rceil} = 9 + \frac{n}{\lceil \sqrt{n} \rceil} \leq 9 + \lceil \sqrt{n} \rceil.$$

$\Diamond$

For any given $x_i \in X$ we define $\mathrm{down}^X(x_i) := j$ as the index of the group S_j that contains x_i. The mapping is well-defined since $\mathcal{S}$ forms a partition of the set of variables.

Claim 2. *If $x_i, x_j \in X$ occur in the same clause $C \in \mathcal{C}$, then* $\mathrm{down}^X(x_i) \neq \mathrm{down}^X(x_j)$.

Proof. By definition, x_i and x_j are adjacent in H_ϕ^X. Hence, x_i and x_j are in different color classes and therefore elements of different groups of $\mathcal{S}$. $\Diamond$

Next, we define the mapping $\mathrm{mid}^X : X \to \{1, \dots, \lceil \sqrt{n} \rceil\}$. To this end, consider the sequence $\mathrm{Seq}^n := (\mathrm{down}^X(x_1), \mathrm{down}^X(x_2), \dots, \mathrm{down}^X(x_n)) \in \{1, \dots, \lceil \sqrt{n} \rceil + 9\}^n$. We define $\mathrm{mid}^X(x_i)$ as the number of occurrences of $\mathrm{down}^X(x_i)$ in the partial sequence $\mathrm{Seq}^i := (\mathrm{down}^X(x_1), \dots, \mathrm{down}^X(x_i))$. From Claim 1 *a*) we conclude that $\mathrm{mid}^X(x_i) \in \{1, 2, \dots, \lceil \sqrt{n} \rceil\}$ for every $x_i \in X$.

Claim 3. *Let $x_i, x_j \in X$ and $r \in \{1,2,3,4\}$. If $x_i \neq x_j$, then*

$$\{\gamma^r_{\mathrm{mid}^X(x_i)}, \delta_{\mathrm{down}^X(x_i)}\} \neq \{\gamma^r_{\mathrm{mid}^X(x_j)}, \delta_{\mathrm{down}^X(x_j)}\}.$$

Proof. Without loss of generality, assume $i < j$. Obviously, the claim holds if $\mathrm{down}^X(x_i) \neq \mathrm{down}^X(x_j)$. Let $\mathrm{down}^X(x_i) = \mathrm{down}^X(x_j)$. Then, there is at least one more occurrence of $\mathrm{down}^X(x_i)$ in the partial sequence Seq_1^j compared to Seq_1^i. Therefore, $\mathrm{mid}^X(x_i) \neq \mathrm{mid}^X(x_j)$. ♢

Thus, we assigned a unique edge in $E(M^X, D^X)$ to each occurrence of a variable in X. Furthermore, the assigned edges of variables that occur in the same clause do not share an endpoint in D^X due to Claim 2.

We complete the description of the variable-representation gadget by defining the vertex list $\Lambda(v)$ for every $v \in M^X \cup D^X$. We set

$$\Lambda(\gamma^r_t) := \bigcup_{\substack{x_i \in X \\ \mathrm{mid}^X(x_i)=t}} \{T^r_i, F^r_i, R_i\} \qquad \text{for every } \gamma^r_t \in M^X, \text{ and}$$

$$\Lambda(\delta_t) := \bigcup_{\substack{x_i \in X \\ \mathrm{down}^X(x_i)=t}} \{T^1_i, T^2_i, T^3_i, T^4_i, F^1_i, F^2_i, F^3_i, F^4_i, Z_2\} \qquad \text{for every } \delta_t \in D^X.$$

Claim 4. *Let $x_i \in X$ and $r \in \{1,2,3,4\}$. Then, $\Lambda(\gamma^r_{\mathrm{mid}^X(x_i)}) \cap \Lambda(\delta_{\mathrm{down}^X(x_i)}) = \{T^r_i, F^r_i\}$.*

Proof. Let $\Lambda(i,r) := \Lambda(\gamma^r_{\mathrm{mid}^X(x_i)}) \cap \Lambda(\delta_{\mathrm{down}^X(x_i)})$. Obviously, T^r_i and F^r_i are elements of $\Lambda(i,r)$. It remains to show that there is no other strong color $Y \in \Lambda(i,r)$.

Case 1: $Y = Z_2$**.** Then, $Z_2 \notin \Lambda(\gamma^r_t)$ and it follows that $Y \notin \Lambda(i,r)$.

Case 2: $Y = R_j$ with $j \in \{1,\dots,n\}$**.** Then, $R_j \notin \Lambda(\delta_t)$ and it follows that $Y \notin \Lambda(i,r)$.

Case 3: $Y = T^{r'}_j$ or $Y = F^{r'}_j$ with $r' \neq r$ and $j \in \{1,\dots,n\}$**.** Then, $Y \notin \Lambda(\gamma^r_{\mathrm{mid}^X(x_i)})$ and it follows that $Y \notin \Lambda(i,r)$.

Case 4: $Y = T^r_{i'}$ or $Y = F^r_{i'}$ with $i' \neq i$**.** Assuming $T^r_{i'} \in \Lambda(i,r)$ it follows from the definition of Λ that there is some variable $x_{i'} \neq x_i$ such that $\mathrm{down}^X(x_{i'}) = \mathrm{down}^X(x_i)$ and $\mathrm{mid}^X(x_{i'}) = \mathrm{mid}^X(x_i)$, which contradicts Claim 3. Hence, $Y \notin \Lambda(i,r)$. ♢

For each variable x_i there are four edges $\{\{\gamma^r_{\mathrm{mid}^X(x_i)}, \delta_{\mathrm{down}^X(x_i)}\} \mid r \in \{1,2,3,4\}\}$ that can only be colored with the strong colors T^r_i and F^r_i representing the truth assignments of the four occurrences of variable x_i due to Claim 4. We need to ensure that there is no variable x_i, where, for example, the first occurrence is set to 'true' (T_i^1) and the second occurrence is set to 'false' (F_i^2) in a Λ-satisfying STC-labeling with no weak edges. To this end, we use a variable-soundness gadget, which we describe in the following.

We define

$$U^X := \{\alpha_t^{(r,r')} \mid t \in \{1, \dots, \lceil\sqrt{n}\rceil + 9\}, (r, r') \in \{1, 2, 3, 4\}^2, r \neq r'\}$$

to be the set of upper vertices. We add edges such that the vertices in U^X form a clique in G. To specify the correspondence between the variables and the edges in the variable-soundness gadget, we define below a mapping $\mathrm{up}^X : X \to \{1, 2, \dots, \lceil\sqrt{n}\rceil + 9\}$. The main idea of the variable-soundness gadget is that for each variable $x_i \in X$ and each pair $\{r, r'\} \subseteq \{1, 2, 3, 4\}$ there are four edges between the vertices γ_i^r, $\gamma_i^{r'}$ and the vertices $\alpha_t^{(r,r')}$, $\alpha_t^{(r',r)}$ of U^X which cannot all be strong in a Λ-satisfying STC-labeling if $\{\gamma^r_{\mathrm{mid}^X(x_i)}, \delta_{\mathrm{down}^X(x_i)}\}$ receives strong color T_i^r and $\{\gamma^{r'}_{\mathrm{mid}^X(x_i)}, \delta_{\mathrm{down}^X(x_i)}\}$ receives strong color $F_i^{r'}$. (Recall that we do not allow weak edges.) To this end, we assign a set of twelve endpoints in U^X to each variable x_i. We need to ensure in particular that two variables x_i and x_j with $\mathrm{mid}^X(x_i) = \mathrm{mid}^X(x_j)$ do not use the same endpoints in U^X. We define $\mathrm{up}^X(x_i) := \mathrm{down}^X(x_i)$. The following is directly implied by Claim 3.

Claim 5. *Let $x_i, x_j \in X$ with $x_i \neq x_j$. If $\mathrm{mid}^X(x_i) = \mathrm{mid}^X(x_j)$, then $\mathrm{up}^X(x_i) \neq \mathrm{up}^X(x_i)$.*

We add the following edges between the vertices of M^X and U^X: For every variable x_i, every $r \in \{1, 2, 3, 4\}$, and every $r' \in \{1, 2, 3, 4\} \setminus \{r\}$ we add the edges $\{\alpha^{(r,r')}_{\mathrm{up}^X(x_i)}, \gamma^r_{\mathrm{mid}^X(x_i)}\}$, and $\{\alpha^{(r,r')}_{\mathrm{up}^X(x_i)}, \gamma^{r'}_{\mathrm{mid}^X(x_i)}\}$.

We complete the description of the variable-soundness gadget by defining the vertex lists $\Lambda(v)$ for each $v \in U^X$. We set

$$\Lambda(\alpha_t^{(r,r')}) := \bigcup_{\substack{x_i \in X \\ \mathrm{up}^X(x_i) = t}} \{T_i^r, F_i^{r'}, R_i, Z_1\} \qquad \text{for every } \alpha_t^{(r,r')} \in U^X.$$

Claim 6. *Let $x_i \in X$, let $r \in \{1, 2, 3, 4\}$, and let $r' \in \{1, 2, 3, 4\} \setminus \{r\}$. Then*

a) $\Lambda(\alpha^{(r,r')}_{\mathrm{up}^X(x_i)}) \cap \Lambda(\gamma^r_{\mathrm{mid}^X(x_i)}) = \{T_i^r, R_i\}$, *and*

b) $\Lambda(\alpha^{(r,r')}_{\mathrm{up}^X(x_i)}) \cap \Lambda(\gamma^{r'}_{\mathrm{mid}^X(x_i)}) = \{F_i^{r'}, R_i\}$.

Proof. Recall that

$$\Lambda(\gamma_t^r) := \bigcup_{\substack{x_i \in X \\ \operatorname{mid}^X(x_i)=t}} \{T_i^r, F_i^r, R_i\}.$$

We first prove Statement a). Let $\Lambda(i,r,r') := \Lambda(\alpha^{(r,r')}_{\operatorname{up}^X(x_i)}) \cap \Lambda(\gamma^r_{\operatorname{mid}^X(x_i)})$. Clearly, $\Lambda(i,r,r')$ contains T_i^r and R_i. It remains to show that there is no other strong color $Y \in \Lambda(i,r,r')$. In the following case distinction we consider every possible strong color $Y \in \Lambda(\gamma^r_{\operatorname{mid}^X(x_i)})$.

Case 1: $Y = R_j$ or $Y = T_j^r$ for some $j \neq i$. Then, there is a variable $x_j \neq x_i$ with $\operatorname{mid}^X(x_j) = \operatorname{mid}^X(x_i)$. It follows by Claim 5 that $\operatorname{up}^X(x_j) \neq \operatorname{up}^X(x_i)$ and therefore $R_j, T_j^r \notin \Lambda(\alpha^{(r,r')}_{\operatorname{up}^X(x_i)})$. Hence, $Y \notin \Lambda(i,r,r')$.

Case 2: $Y = F_j^r$. Then, since

$$\{F_t^p \mid p \in \{1,2,3,4\}, t \in \{1,\dots,n\}\} \cap \Lambda(\alpha^{(r,r')}_{\operatorname{up}^X(x_i)}) \subseteq \{F_1^{r'}, F_2^{r'}, \dots, F_n^{r'}\}$$

and we have $r' \neq r$, we conclude $F_j^r \notin \Lambda(\alpha^{(r,r')}_{\operatorname{up}^X(x_i)})$. Hence, $Y \notin \Lambda(i,r,r')$.

We prove Statement *b*) with analogous arguments. Let $\overline{\Lambda}(i,r,r') := \Lambda(\alpha^{(r,r')}_{\operatorname{up}^X(x_i)}) \cap \Lambda(\gamma^{r'}_{\operatorname{mid}^X(x_i)})$. Clearly, $\{F_i^{r'}, R_i\} \subseteq \Lambda(i,r,r')$. It remains to show that there is no other color $Y \in \overline{\Lambda}(i,r,r')$.

Case 1: $Y = R_j$ or $Y = F_j^{r'}$ for some $j \neq i$. Then, analogously to the proof of Statement *a*) we conclude that $Y \notin \overline{\Lambda}(i,r,r')$.

Case 2: $Y = T_j^{r'}$. Then, since

$$\{T_t^p \mid p \in \{1,2,3,4\}, t \in \{1,\dots,n\}\} \cap \Lambda(\alpha^{(r,r')}_{\operatorname{up}^X(x_i)}) \subseteq \{T_1^r, T_2^r, \dots, T_n^r\}$$

and we have $r' \neq r$ we conclude $T_j^{r'} \notin \Lambda(\alpha^{(r,r')}_{\operatorname{up}^X(x_i)})$. Hence, $Y \notin \overline{\Lambda}(i,r,r')$. ◊

This completes the description of the variable gadget. We continue with the description of the clause gadget.

The Clause Gadget. The clause gadget consists of an upper part and a lower part. Let

$$U^{\mathcal{C}} := \{\eta_i \mid i \in \{1,\dots,12\lceil\sqrt{n}\,\rceil + 1\}\}$$

be the set of upper vertices and let

$$D^{\mathcal{C}} := \{\theta_i \mid i \in \{1,\dots,\lceil\sqrt{n}\,\rceil\}\}$$

be the set of lower vertices. We add edges such that $U^{\mathcal{C}}$ and $D^{\mathcal{C}}$ each form cliques in G.

Recall that for some clause $C_j \in \mathcal{C}$ and a variable x_i occurring in C_j the occurrence number $\Omega(C_j, x_i)$ is defined as the number of clauses in $\{C_1, C_2, \dots, C_j\}$ that contain x_i. Below we define two mappings $\mathrm{up}^{\mathcal{C}} : \mathcal{C} \to \{1, 2, \dots, 12\lceil\sqrt{n}\,\rceil + 1\}$, $\mathrm{down}^{\mathcal{C}} : \mathcal{C} \to \{1, 2, \dots, \lceil\sqrt{n}\,\rceil\}$, and vertex lists $\Lambda : V \to 2^{\{1,\dots,c\}}$. Then, for each clause $C_j \in \mathcal{C}$, we add an edge $\{\eta_{\mathrm{up}^{\mathcal{C}}(C_j)}, \theta_{\mathrm{down}^{\mathcal{C}}(C_j)}\}$. Next, we ensure that this edge can only be labeled with the strong colors that match the literals in C_j. This means, for example, if $C_j = (x_1 \vee \overline{x_2} \vee x_3)$ we have $\Lambda(\eta_{\mathrm{up}^{\mathcal{C}}(C_j)}) \cap \Lambda(\theta_{\mathrm{down}^{\mathcal{C}}(C_j)}) = \{T_1^{\Omega(C_j,x_1)}, F_2^{\Omega(C_j,x_2)}, T_3^{\Omega(C_j,x_3)}\}$.

As before, we need to ensure that each variable occurring in a clause has a unique edge between the clause and variable gadgets which transmits the variable's truth assignment to the clause. To achieve this, we define the *clause-conflict graph* $H_\phi^{\mathcal{C}} := (\mathcal{C}, \mathrm{Confl}^{\mathcal{C}})$ by

$$\begin{aligned}\mathrm{Confl}^{\mathcal{C}} := \{\{C_i, C_j\} \mid\ & C_i \text{ contains a variable } x_i \text{ and}\\ & C_j \text{ contains a variable } x_j \text{ such that}\\ & \mathrm{down}^X(x_i) = \mathrm{down}^X(x_j)\}.\end{aligned}$$

Clauses that share a variable are one example for adjacent vertices in $H_\phi^{\mathcal{C}}$. Furthermore, due to the compression there may be distinct variables that are mapped to the same value under down^X. Two distinct clauses containing these variables are another example for adjacent vertices in $H_\phi^{\mathcal{C}}$.

From the fact that each variable occurs in at most four clauses in combination with Claim 1 *a*), it follows that the maximum degree of $H_\phi^{\mathcal{C}}$ is at most $12\cdot\lceil\sqrt{n}\,\rceil$. Thus, there exists a proper vertex coloring $\chi : \mathcal{C} \to \{1, 2, \dots, 12\cdot\lceil\sqrt{n}\,\rceil + 1\}$ such that each color class $\chi^{-1}(i)$, $i \in \{1, \dots, 12\cdot\lceil\sqrt{n}\,\rceil + 1\}$, contains at most $\lceil\frac{m}{12\cdot\lceil\sqrt{n}\,\rceil+1}\rceil + 1 \leq \lceil\sqrt{n}\,\rceil$ clauses [87]. Such coloring is known as *equitable coloring* and since it has $\mathcal{O}(\sqrt{n})$ colors, it can be computed in polynomial time [102].

For a given clause $C_i \in \mathcal{C}$, we define $\mathrm{up}^{\mathcal{C}}(C_i) := j$ as the index of the color class $\chi^{-1}(j)$ that contains C_i. The following claim provides a useful property for the clause gadget. In the proof we use a similar argument as in the proof of Claim 2.

Claim 7. *If a clause $C_{j_1} \in \mathcal{C}$ contains a variable x_{i_1} and a clause $C_{j_2} \in \mathcal{C}$ contains a variable x_{i_2} such that* $\mathrm{down}^X(x_{i_1}) = \mathrm{down}^X(x_{i_2})$, *then* $\mathrm{up}^{\mathcal{C}}(C_{j_1}) \neq \mathrm{up}^{\mathcal{C}}(C_{j_2})$.

Proof. By definition, C_{j_1} and C_{j_2} are adjacent in $H_\phi^{\mathcal{C}}$. Hence, C_{j_1} and C_{j_2} are elements of different color classes and therefore $\mathrm{up}^{\mathcal{C}}(C_{j_1}) \neq \mathrm{up}^{\mathcal{C}}(C_{j_2})$. $\Diamond$

Next, we define $\mathrm{down}^{\mathcal{C}}$ analogously to up^X. To this end, consider the finite sequence $\mathrm{Seq}^m = (\mathrm{up}^{\mathcal{C}}(C_1), \mathrm{up}^{\mathcal{C}}(C_2), \dots, \mathrm{up}^{\mathcal{C}}(C_m))$ and define $\mathrm{down}^{\mathcal{C}}(C_j)$ as the number of occurrences of $\mathrm{up}^{\mathcal{C}}(C_j)$ in the finite sequence $\mathrm{Seq}^j := (\mathrm{up}^{\mathcal{C}}(C_1), \dots, \mathrm{up}^{\mathcal{C}}(C_j))$.

The fact that each color class contains at most $\lceil \sqrt{n}\,\rceil$ elements implies $\mathrm{down}^{\mathcal{C}}(C_j) \leq \lceil \sqrt{n}\,\rceil$. Intuitively, the following claim guarantees that distinct clauses correspond to distinct edges, which is an analogous statement to Claim 3.

Claim 8. *Let $C_i, C_j \in \mathcal{C}$ such that $C_i \neq C_j$. Then,*

$$\{\eta_{\mathrm{up}^{\mathcal{C}}(C_i)}, \theta_{\mathrm{down}^{\mathcal{C}}(C_j)}\} \neq \{\eta_{\mathrm{up}^{\mathcal{C}}(C_j)}, \theta_{\mathrm{down}^{\mathcal{C}}(C_j)}\}.$$

Proof. Without loss of generality, let $i < j$. The claim obviously holds if $\mathrm{up}^{\mathcal{C}}(C_i) \neq \mathrm{up}^{\mathcal{C}}(C_j)$, so let $\mathrm{up}^{\mathcal{C}}(C_i) = \mathrm{up}^{\mathcal{C}}(C_j)$. Then, there is at least one more occurrence of $\mathrm{up}^{\mathcal{C}}(C_i)$ in the partial sequence Seq_1^j compared to Seq_1^i. Therefore, $\mathrm{down}^{\mathcal{C}}(C_i) \neq \mathrm{down}^{\mathcal{C}}(C_j)$. $\Diamond$

We complete the description of the clause gadget by defining the vertex lists $\Lambda(v)$ for every $v \in U^{\mathcal{C}} \cup D^{\mathcal{C}}$. To this end, we introduce two color sets for each clause. Given a clause $C_j \in \mathcal{C}$, we define the *color set* $\mathfrak{X}(C_j)$ and the *literal color set* $\mathfrak{L}(C_j)$ of C_j by

$$\begin{aligned}\mathfrak{X}(C_j) &:= \{T_i^{\Omega(C_j,x_i)}, F_i^{\Omega(C_j,x_i)} \mid x_i \text{ occurs in } C_j\}, \text{ and}\\ \mathfrak{L}(C_j) &:= \{T_i^{\Omega(C_j,x_i)} \mid x_i \text{ occurs as a positive literal in } C_j\} \cup\\ &\qquad \{F_i^{\Omega(C_j,x_i)} \mid x_i \text{ occurs as a negative literal in } C_j\}.\end{aligned}$$

Note that $\mathfrak{L}(C_j) \subseteq \mathfrak{X}(C_j)$. The vertex lists for the vertices in $U^{\mathcal{C}} \cup D^{\mathcal{C}}$ are defined as

$$\begin{aligned}\Lambda(\eta_t) &:= \bigcup_{\substack{C_j \in \mathcal{C}\\ \mathrm{up}^{\mathcal{C}}(C_j)=t}} \mathfrak{X}(C_j) \cup \{Z_3\} && \text{for every } \eta_t \in U^{\mathcal{C}}, \text{ and}\\ \Lambda(\theta_t) &:= \bigcup_{\substack{C_j \in \mathcal{C}\\ \mathrm{down}^{\mathcal{C}}(C_j)=t}} \mathfrak{L}(C_j) \cup \{Z_4\} && \text{for every } \theta_t \in D^{\mathcal{C}}.\end{aligned}$$

Claim 9. *Let $C_j \in \mathcal{C}$. Then, $\Lambda(\eta_{\mathrm{up}^{\mathcal{C}}(C_j)}) \cap \Lambda(\theta_{\mathrm{down}^{\mathcal{C}}(C_j)}) = \mathfrak{L}(C_j)$.*

Proof. Let $\Lambda(j) := \Lambda(\eta_{\mathrm{up}^{\mathcal{C}}(C_j)}) \cap \Lambda(\theta_{\mathrm{down}^{\mathcal{C}}(C_j)})$. Since $\mathfrak{L}(C_j) \subseteq \mathfrak{X}(C_j)$ it holds that $\mathfrak{L}(C_j) \subseteq \Lambda(j)$. It remains to show that there is no other strong color $Y \in \Lambda(j) \setminus \mathfrak{L}(C_j)$.

Case 1: $Y \in \{Z_3, Z_4\}$. Then, since $Z_3 \notin \Lambda(\theta_{\mathrm{down}^{\mathcal{C}}(C_j)})$ and $Z_4 \notin \Lambda(\eta_{\mathrm{up}^{\mathcal{C}}(C_j)})$, it follows that $Y \notin \Lambda(j)$.

Case 2: $Y \notin \{Z_3, Z_4\}$. Assume towards a contradiction that $Y \in \Lambda(j)$. From $Y \in \Lambda(\theta_{\mathrm{down}^{\mathcal{C}}(C_j)})$ it follows that there is a clause C_{j_1} with $\mathrm{down}^{\mathcal{C}}(C_{j_1}) =$

$\text{down}^{\mathcal{C}}(C_j)$ and $Y \in \mathfrak{L}(C_{j_1})$. It holds that $C_{j_1} \neq C_j$, since otherwise $Y \in \mathfrak{L}(C_j)$, which contradicts the fact that $Y \in \Lambda(j) \setminus \mathfrak{L}(C_j)$. From $Y \in \Lambda(\eta_{up^{\mathcal{C}}(C_j)})$ it follows that there is a clause C_{j_2} with $\text{up}^{\mathcal{C}}(C_{j_2}) = \text{up}^{\mathcal{C}}(C_j)$ and $Y \in \mathfrak{X}(C_{j_2})$. By the definition of $\mathfrak{X}$ and $\mathfrak{L}$, there exists a variable x_i that occurs in C_{j_1} and C_{j_2} such that $Y = T_i^{\Omega(C_{j_1},x_i)} = T_i^{\Omega(C_{j_2},x_i)}$ or $Y = F_i^{\Omega(C_{j_1},x_i)} = F_i^{\Omega(C_{j_2},x_i)}$. We conclude $\Omega(C_{j_1}, x_i) = \Omega(C_{j_2}, x_i)$ and therefore $C_{j_2} = C_{j_1} \neq C_j$. Then, the fact that $\text{up}^{\mathcal{C}}(C_{j_1}) = \text{up}^{\mathcal{C}}(C_j)$ and $\text{down}^{\mathcal{C}}(C_{j_2}) = \text{down}^{\mathcal{C}}(C_j)$ contradicts Claim 8 and therefore $Y \notin \Lambda(j)$. $\Diamond$

Connecting the Gadgets. We complete the construction of G by describing how the vertices of the variable gadget and the vertices of the clause gadget are connected. The idea is to define edges between the vertices in D^X and $U^{\mathcal{C}}$ that model the occurrences of variables in the clauses.

For each $C_j \in \mathcal{C}$ we do the following: Let x_{i_1}, x_{i_2}, and x_{i_3} be the variables that occur in C_j. We add the three edges $\{\delta_{\text{down}^X(x_{i_1})}, \eta_{\text{up}^{\mathcal{C}}(C_j)}\}$, $\{\delta_{\text{down}^X(x_{i_2})}, \eta_{\text{up}^{\mathcal{C}}(C_j)}\}$, and $\{\delta_{\text{down}^X(x_{i_3})}, \eta_{\text{up}^{\mathcal{C}}(C_j)}\}$. The intuitive idea is that an edge $\{\delta_{\text{down}^X(x_i)}, \eta_{\text{up}^{\mathcal{C}}(C_j)}\}$ transmits the truth value of a variable x_i to a clause C_j, where x_i occurs as a positive or negative literal. The following claim states that the possible strong colors for such an edge are only $T_i^{\Omega(C_j,x_i)}$ and $F_i^{\Omega(C_j,x_i)}$, which correspond to the negated truth assignment of the $\Omega(C_j, x_i)$th occurrence of x_i.

Claim 10. *Let $C_j \in \mathcal{C}$ be a clause and let $x_i \in X$ be some variable that occurs in C_j. Then, $\Lambda(\delta_{\text{down}^X(x_i)}) \cap \Lambda(\eta_{\text{up}^{\mathcal{C}}(C_j)}) = \{T_i^{\Omega(C_j,x_i)}, F_i^{\Omega(C_j,x_i)}\}$.*

Proof. Let $\Lambda(i,j) := \Lambda(\delta_{\text{down}^X(x_i)}) \cap \Lambda(\eta_{\text{up}^{\mathcal{C}}(C_j)})$. Recall that

$$\Lambda(\delta_t) := \bigcup_{\substack{x_i \in X \\ \text{down}^X(x_i)=t}} \{T_i^1, T_i^2, T_i^3, T_i^4, F_i^1, F_i^2, F_i^3, F_i^4, Z_2\} \text{ and}$$

$$\Lambda(\eta_t) := \bigcup_{\substack{C_j \in \mathcal{C} \\ \text{up}^{\mathcal{C}}(C_j)=t}} \mathfrak{X}(C_j) \cup \{Z_3\}.$$

Obviously, $\{T_i^{\Omega(C_j,x_i)}, F_i^{\Omega(C_j,x_i)}\} \subseteq \Lambda(i,j)$. It remains to show that there is no strong color $Y \in \Lambda(i,j) \setminus \{T_i^{\Omega(C_j,x_i)}, F_i^{\Omega(C_j,x_i)}\}$.

Case 1: $Y = Z_3$ or $Y = Z_2$**.** Since $Z_3 \notin \Lambda(\delta_{\text{down}^X(x_i)})$ and $Z_2 \notin \Lambda(\eta_{\text{up}^{\mathcal{C}}(C_j)})$, we have $Y \notin \Lambda(i,j)$.

Case 2: $Y = T_t^r$ or $Y = F_t^r$ with $t \neq i$ and $r \in \{1,2,3,4\}$**.** If $Y \notin \Lambda(\eta_{\text{up}^{\mathcal{C}}(C_j)})$, then obviously $Y \notin \Lambda(i,j)$. Thus, let $Y \in \Lambda(\eta_{\text{up}^{\mathcal{C}}(C_j)})$. Then, by the definition of $\mathfrak{X}$, there is a clause $C_{j'}$ containing a variable $x_t \neq x_i$ with $\text{up}^{\mathcal{C}}(C_j) = \text{up}^{\mathcal{C}}(C_{j'})$. If $C_{j'} =$

C_j, then Claim 2 implies $\text{down}^X(x_i) \neq \text{down}^X(x_t)$ and thus $Y \notin \Lambda(\delta_{\text{down}^X(x_i)})$. Otherwise, if $C_{j'} \neq C_j$, then Claim 7 together with the fact that $\text{up}^{\mathcal{C}}(C_j) = \text{up}^{\mathcal{C}}(C_{j'})$ imply that $\text{down}^X(x_i) \neq \text{down}^X(x_t)$. Consequently, $Y \notin \Lambda(\delta_{\text{down}^X(x_i)}) \supseteq \Lambda(i,j)$.

Case 3: $Y = T_i^r$ or $Y = F_i^r$ with $r \neq \Omega(C_j, x_i)$**.** Obviously, $Y \in \Lambda(\delta_{\text{down}^X(x_i)})$. Assume towards a contradiction that $Y \in \Lambda(\eta_{\text{up}^{\mathcal{C}}(C_j)})$. Then, by the definition of the color set $\mathfrak{X}(\cdot)$, there is a clause $C_{j'}$ containing x_i such that $\text{up}^{\mathcal{C}}(C_{j'}) = \text{up}^{\mathcal{C}}(C_j)$ and $\Omega(C_{j'}, x_i) = r \neq \Omega(C_j, x_i)$. It follows that $C_{j'} \neq C_j$ which contradicts Claim 7. Hence, $Y \notin \Lambda(\eta_{\text{up}^{\mathcal{C}}(C_j)}) \supseteq \Lambda(i,j)$. $\Diamond$

This completes the description of the construction and basic properties of the VL-Multi-STC instance $(G, 9n+4, 0, \Lambda)$. Note that G has $\mathcal{O}(\sqrt{n})$ vertices. It remains to show the correctness of the reduction.

Correctness. We show that there is a satisfying assignment for ϕ if and only if there is a $(9n+4)$-colored Λ-satisfying STC-labeling L for G with strong color classes

$$S_L^{T_t^r}, S_L^{F_t^r}, S_L^{R_t}, S_L^{Z_r} \qquad \text{for all } t \in \{1,\dots,n\} \text{ and } r \in \{1,2,3,4\},$$

and $W_L = \emptyset$.

($\Rightarrow$) Let $A : X \to \{\text{'true'}, \text{'false'}\}$ be a satisfying assignment for ϕ. We describe to which strong color classes we add the edges of G so that we obtain a Λ-satisfying STC-labeling.

First, we describe to which strong color classes we add the edges of the variable gadget. Formally, these are the edges in in $E(U^X \cup M^X \cup D^X)$. Let $e := \{\delta_{\text{down}^X(x_i)}, \gamma^r_{\text{mid}^X(x_i)}\}$ be an edge of the variable-representation gadget for some $x_i \in X$ and $r \in \{1,2,3,4\}$. We add e to $S_L^{T_i^r}$ if $A(x_i) = $ 'true' or to $S_L^{F_i^r}$ if $A(x_i) = $ 'false'. In both cases, e satisfies the Λ-list property by Claim 4. Next, let $e_1 := \{\gamma^r_{\text{mid}^X(x_i)}, \alpha^{(r,r')}_{\text{up}^X(x_i)}\}$ and let $e_2 := \{\gamma^{r'}_{\text{mid}^X(x_i)}, \alpha^{(r,r')}_{\text{up}^X(x_i)}\}$ be two edges of the variable-soundness gadget for some $x_i \in X$, $r \in \{1,2,3,4\}$, and $r' \in \{1,2,3,4\} \setminus \{r\}$. We add e_1 to $S_L^{R_i}$ if $A(x_i) = $ 'true' or to $S_L^{T_i^r}$ if $A(x_i) = $ 'false'. Further, we add e_2 to $S_L^{F_i^{r'}}$ if $A(x_i) = $ 'true' or to $S_L^{R_i}$ if $A(x_i) = $ 'false'. In each case, e_1 and e_2 satisfy the Λ-list property by Claim 6. For the remaining edges of the variable-gadget we do the following: We add all edges of $E(U^X)$ to $S_L^{Z_1}$ and all edges of $E(D_X)$ to $S_L^{Z_2}$. Obviously, this does not violate the Λ-list property.

Second, we describe to which strong color classes we add the edges of the clause gadget. Formally, these are the edges in $E(U^{\mathcal{C}} \cup D^{\mathcal{C}})$. Let $C_j \in \mathcal{C}$ be a clause. Since A satisfies ϕ, there is some variable x_i occurring in C_j, such that the assignment $A(x_i)$ satisfies the clause C_j. Let $r := \Omega(C_j, x_i)$. We add the edge $\{\eta_{\text{up}^{\mathcal{C}}(C_j)}, \theta_{\text{down}^{\mathcal{C}}(C_j)}\}$ to $S_L^{T_i^r}$ if $A(x_i) = $ 'true' or to $S_L^{F_i^r}$ if $A(x_i) = $ 'false'. In both cases, the edge satisfies the Λ-list property by Claim 9. For the remaining edges of the clause gadget, we do the following: We add all edges of $E(U^{\mathcal{C}})$ to $S_L^{Z_3}$ and all edges of $E(D^{\mathcal{C}})$ to $S_L^{Z_4}$. Obviously, this does not violate the Λ-list property.

Finally, we describe to which strong color classes we add the edges between the two gadgets. Formally, these are the edges in $E(D^X, U^{\mathcal{C}})$. Let $C_j \in \mathcal{C}$ be a clause and let x_i be some variable occurring in C_j. Let $r := \Omega(C_j, x_i)$. We add the edge $\{\delta_{\text{down}^X(x_i)}, \eta_{\text{up}^{\mathcal{C}}(C_j)}\}$ to $S_L^{F_i^r}$ if $A(x_i) =$ 'true' or to $S_L^{T_i^r}$ if $A(x_i) =$ 'false'. This does not violate the Λ-list property by Claim 10.

We have now added every edge of G to exactly one strong color class of L such that L is Λ-satisfying. It remains to show that there is no induced P_3 containing two edges $\{u, v\}$ and $\{v, w\}$ from the same strong color class. In the following case distinction, we consider every possible induced P_3 on vertices u, v, and w where v is the central vertex. The case distinction is organized as follows: We consider all possible positions of the central vertex v, that is $v \in Q$ for $Q \in \{U^X, M^X, D^X, U^{\mathcal{C}}, D^{\mathcal{C}}\}$ (cases). For each such position, we consider all possible positions of u and w (subcases).

Case 1: $v \in U^X$**.** Then, $v = \alpha_t^{(r,r')}$ for some $t \in \{1, \dots, \lceil\sqrt{n}\,\rceil + 9\}$, $r \in \{1, 2, 3, 4\}$ and $r' \in \{1, 2, 3, 4\} \setminus \{r\}$. Note that the vertices in U^X are not adjacent to vertices in D^X, $U^{\mathcal{C}}$ and $D^{\mathcal{C}}$. Thus, it suffices to consider the following subcases.

Case 1.1: $u \in U^X$**.** Then, $\{u, v\} \in S_L^{Z_1}$. If $w \in U^X$, then the vertices u, v, and w do not form an induced P_3, since U^X is a clique in G. If $w \notin U^X$, then $\{v, w\} \notin S_L^{Z_1}$. Hence, there is no STC-violation.

Case 1.2: $u, w \in M^X$**.** Then, there are variables x_i and x_j with $\text{up}^X(x_i) = \text{up}^X(x_j) = t$ and $u = \gamma^p_{\text{mid}^X(x_i)}$, $w = \gamma^q_{\text{mid}^X(x_j)}$ for some $p, q \in \{r, r'\}$. We need to consider the following.

Case 1.2.1: $x_i \neq x_j$**.** Then $i \neq j$. By Claim 6 it holds without loss of generality that $\Lambda(u) \cap \Lambda(v) \subseteq \{T_i^r, F_i^r, T_i^{r'}, F_i^{r'}, R_i\}$ and $\Lambda(w) \cap \Lambda(v) \subseteq \{T_j^r, F_j^r, T_j^{r'}, F_j^{r'}, R_j\}$. Since L is Λ-satisfying, the edges $\{u, v\}$ and $\{v, w\}$ are elements of different strong color classes. Thus, there is no STC-violation.

Case 1.2.2: $x_i = x_j$**.** Then, $p \neq q$, since otherwise $u = v$. Without loss of generality, we have $u = \gamma^r_{\text{mid}^X(x_i)}$ and $w = \gamma^{r'}_{\text{mid}^X(x_i)}$. If $A(x_i) =$ 'true', it follows that $\{u, v\} \in S_L^{R_i}$ and $\{v, w\} \in S_L^{F_i^{r'}}$. Otherwise, if $A(x_i) =$ 'false', it follows that $\{u, v\} \in S_L^{T_i^{r'}}$ and $\{v, w\} \in S_L^{R_i}$. In both cases, the edges $\{u, v\}$ and $\{v, w\}$ are elements of different strong color classes. Thus, there is no STC-violation.

Case 2: $v \in M^X$**.** Then, $v = \gamma_t^r$ for some $t \in \{1, \dots, \lceil\sqrt{n}\,\rceil\}$ and $r \in \{1, 2, 3, 4\}$. Note that the vertices in M^X are not adjacent to vertices in $U^{\mathcal{C}}$, $D^{\mathcal{C}}$, and M^X. Thus, it suffices to consider the following subcases.

Case 2.1: $u, w \in U^X$ or $u, w \in D^X$**.** Then, since U^X and D^X are cliques in G, the vertices u, v, and w do not form an induced P_3 in G. Hence, there is no STC-violation.

Case 2.2: $u \in U^X$ and $w \in D^X$**.** Then, there are variables x_i and x_j with $\text{mid}^X(x_i) = \text{mid}^X(x_j) = t$ and $u \in \{\alpha^{(r,r')}_{\text{up}^X(x_i)}, \alpha^{(r',r)}_{\text{up}^X(x_i)}\}$, $w = \delta_{\text{down}^X(x_j)}$ for some $r' \neq r$. We need to consider the following subcases.

Case 2.2.1: $x_i \neq x_j$**.** Then, $i \neq j$. Without loss of generality it holds by Claim 6 that $\Lambda(u) \cap \Lambda(v) \subseteq \{T^r_i, F^r_i, T^{r'}_i, F^{r'}_i, R_i\}$ for some $r' \neq r$ and by Claim 4 that $\Lambda(v) \cap \Lambda(w) = \{T^r_j, F^r_j\}$. Since L is Λ-satisfying, the edges $\{u, v\}$ and $\{v, w\}$ are elements of different strong color classes. Thus, there is no STC-violation.

Case 2.2.2: $x_i = x_j$**.** Then, if $A(x_i) =$ 'true' it follows that $\{u, v\} \in S^{R_i}_L \cup S^{F^r_i}_L$ and $\{v, w\} \in S^{T^r_i}_L$. If $A(x_i) =$ 'false' it follows that $\{u, v\} \in S^{R_i}_L \cup S^{T^r_i}_L$ and $\{v, w\} \in S^{F^r_i}_L$. In both cases the edges $\{u, v\}$ and $\{v, w\}$ are elements of different strong color classes. Thus, there is no STC-violation.

Case 3: $v \in D^X$**.** Then, $v = \delta_t$ for some $t \in \{1, \ldots, \lceil\sqrt{n}\,\rceil + 9\}$. Note that the vertices in D^X are not adjacent to vertices in U^X and $D^{\mathcal{C}}$. Thus, it suffices to consider the following subcases.

Case 3.1: $u \in D^X$**.** Then, $\{u, v\} \in S^{Z_2}_L$. If $w \in D^X$, then the vertices u, v, and w do not form an induced P_3, since D^X is a clique in G. If $w \notin D^X$, then $\{v, w\} \notin S^{Z_2}_L$. Hence, there is no STC-violation.

Case 3.2: $u, w \in U^{\mathcal{C}}$**.** Then, the vertices u, v, and w do not form an induced P_3, since $U^{\mathcal{C}}$ forms a clique.

Case 3.3: $u, w \in M^X$**.** By Claim 4, all edges $\{v, y\} \in E(\{v\}, M^X)$ have distinct possible strong colors in $\Lambda(v) \cap \Lambda(y)$. Since L is Λ-satisfying, the edges $\{u, v\}$ and $\{v, w\}$ are elements of different strong color classes.

Case 3.4: $u \in M^X$ and $w \in U^{\mathcal{C}}$**.** Then, $u = \gamma^r_{\text{mid}^X(x_i)}$ for some $x_i \in X$ with $\text{down}^X(x_i) = t$ and $r \in \{1, 2, 3, 4\}$. Moreover, $w = \eta_{\text{up}^{\mathcal{C}}(C_j)}$ for some clause C_j containing a variable $x_{i'}$ with $\text{down}^X(x_{i'}) = t$. We need to consider the following.

Case 3.4.1: $x_i \neq x_{i'}$**.** Then, $i \neq i'$ and by Claim 4 we have $\Lambda(u) \cap \Lambda(v) = \{T^r_i, F^r_i\}$ and by Claim 10 we have $\Lambda(v) \cap \Lambda(w) = \{T^{r'}_{i'}, F^{r'}_{i'}\}$ with $r' = \Omega(C_j, x_{i'})$. Then, since L is Λ-satisfying, $\{u, v\}$ and $\{v, w\}$ are not elements of the same strong color class.

Case 3.4.2: $x_i = x_{i'}$**.** Then, if $A(x_i) =$ 'true' it follows that $\{u, v\} \in S^{T^r_i}_L$ and $\{v, w\} \in S^{F^{r'}_i}_L$ for some $r' \in \{1, 2, 3, 4\}$. If $A(x_i) =$ 'false', then it follows that $\{u, v\} \in S^{F^r_i}_L$ and $\{v, w\} \in S^{T^{r'}_i}_L$ for some $r' \in \{1, 2, 3, 4\}$. In both cases $\{u, v\}$ and $\{v, w\}$ are elements of different strong color classes.

Case 4: $v \in U^{\mathcal{C}}$**.** Then, $v = \eta_t$ for some $t \in \{1, \ldots, 12\lceil\sqrt{n}\,\rceil + 1\}$. Note that the vertices in $U^{\mathcal{C}}$ are not adjacent to vertices in U^X and M^X. Thus, it suffices to consider the following subcases.

Case 4.1: $u \in U^{\mathcal{C}}$**.** Then, $\{u,v\} \in S_L^{Z_3}$. If $w \in U^{\mathcal{C}}$, then the vertices u, v, and w do not form an induced P_3 since U^X is a clique in G. If $w \notin U^{\mathcal{C}}$ it follows that $\{v,w\} \notin S_L^{Z_3}$. Hence, there is no STC-violation.

Case 4.2: $u, w \in D^X$ or $u, w \in D^{\mathcal{C}}$**.** Then, the vertices u, v, and w do not form an induced P_3, since D^X and $D^{\mathcal{C}}$ form cliques in G.

Case 4.3: $u \in D^X$ and $w \in D^{\mathcal{C}}$**.** Then, there is a clause C_j with $\text{up}^{\mathcal{C}}(C_j) = t$ and a clause $C_{j'}$ containing a variable x_i with $\text{up}^{\mathcal{C}}(C_{j'}) = t$ and $u = \delta_{\text{down}^X(x_i)}$, $w = \theta_{\text{down}^{\mathcal{C}}(C_j)}$. We consider the following.

Case 4.3.1: $C_j \neq C_{j'}$**.** Then, since $\text{up}^{\mathcal{C}}(C_j) = \text{up}^{\mathcal{C}}(C_{j'})$ it follows by Claim 7 that C_j and $C_{j'}$ do not share a variable. Hence, x_i does not occur in C_j and therefore $T_i^{\Omega(C_{j'},x_i)}, F_i^{\Omega(C_{j'},x_i)} \notin \mathfrak{L}(C_j)$. Thus, by Claims 9 and 10 and the fact that L is Λ-satisfying, the edges $\{u,v\}$ and $\{v,w\}$ are elements of different strong color classes.

Case 4.3.2: $C_j = C_{j'}$**.** Let $r := \Omega(C_j, x_i)$. If $\{v,w\} \notin S_L^{T_i^r} \cup S_L^{F_i^r}$, the edges $\{u,v\}$ and $\{v,w\}$ are elements of different color classes. Thus, there is no STC-violation. If $\{v,w\} \in S_L^{T_i^r} \cup S_L^{F_i^r}$, then it follows by the construction of L that C_j is satisfied by the assignment $A(x_i)$. Without loss of generality assume that x_i occurs as a positive literal in C_j. Then, $A(x_i) =$ 'true'. This implies $\{v,w\} \in S_L^{T_i^r}$ and $\{u,v\} \in S_L^{F_i^r}$. Hence, $\{u,v\}$ and $\{v,w\}$ are elements of different strong color classes.

Case 5: $v \in D^{\mathcal{C}}$**.** Then, v is not adjacent with any vertices in U^X, M^X, or D^X. Hence, we need to consider the following cases.

Case 5.1: $u, w \in U^{\mathcal{C}}$ or $u, w \in D^{\mathcal{C}}$**.** Then, the vertices u, v, and w do not form an induced P_3 since $U^{\mathcal{C}}$ and $D^{\mathcal{C}}$ are cliques in G.

Case 5.2: $u \in D^{\mathcal{C}}$ and $w \in U^{\mathcal{C}}$**.** Then, $\{u,v\} \in S_L^{Z_4}$ and $\{v,w\} \notin S_L^{Z_4}$. Hence, there is no STC-violation.

This proves that L is a Λ-satisfying STC-labeling for G with no weak edges.

($\Leftarrow$) Conversely, let L be a $(9n+4)$-colored Λ-satisfying STC-labeling for G with $W_L = \emptyset$. We show that ϕ is satisfiable. We define an assignment $A : C \to \{\text{'true'}, \text{'false'}\}$ by

$$A(x_i) := \begin{cases} \text{'true'} & \text{if } \{\delta_{\text{down}^X(x_i)}, \gamma^1_{\text{mid}^X(x_i)}\} \in S_L^{T_i^1}, \text{ and} \\ \text{'false'} & \text{if } \{\delta_{\text{down}^X(x_i)}, \gamma^1_{\text{mid}^X(x_i)}\} \in S_L^{F_i^1}. \end{cases}$$

The assignment is well-defined due to Claim 4. The following claim states that, if one occurrence $r \in \{1,2,3,4\}$ of some variable x_i that is assigned 'true' (or 'false', respectively), then so is the first occurrence of x_i. We obtain this statement by using the variable-soundness gadget.

Claim 11. *Let $x_i \in X$ and let $r \in \{2, 3, 4\}$.*

a) If $\{\delta_{\mathrm{down}^X(x_i)}, \gamma^r_{\mathrm{mid}^X(x_i)}\} \in S_L^{T_i^r}$, then $\{\delta_{\mathrm{down}^X(x_i)}, \gamma^1_{\mathrm{mid}^X(x_i)}\} \in S_L^{T_i^1}$.

b) If $\{\delta_{\mathrm{down}^X(x_i)}, \gamma^r_{\mathrm{mid}^X(x_i)}\} \in S_L^{F_i^r}$, then $\{\delta_{\mathrm{down}^X(x_i)}, \gamma^1_{\mathrm{mid}^X(x_i)}\} \in S_L^{F_i^1}$.

Proof. We show *a*). Let $\{\delta_{\mathrm{down}^X(x_i)}, \gamma^r_{\mathrm{mid}^X(x_i)}\} \in S_L^{T_i^r}$. Consider the vertex $\alpha^{(r,1)}_{\mathrm{up}^X(x_i)}$. By Claim 6 we have

$$\Lambda(\alpha^{(r,1)}_{\mathrm{up}^X(x_i)}) \cap \Lambda(\gamma^r_{\mathrm{mid}^X(x_i)}) = \{T_i^r, R_i\}, \text{ and}$$
$$\Lambda(\alpha^{(r,1)}_{\mathrm{up}^X(x_i)}) \cap \Lambda(\gamma^1_{\mathrm{mid}^X(x_i)}) = \{F_i^1, R_i\}.$$

Note that the vertices $\delta_{\mathrm{down}^X(x_i)}, \gamma^r_{\mathrm{mid}^X(x_i)}, \alpha^{(r,1)}_{\mathrm{up}^X(x_i)}$ form an induced P_3 in G. Since L is a Λ-satisfying STC-labeling with no weak edges, we have $\{\gamma^r_{\mathrm{mid}^X(x_i)}, \alpha^{(r,1)}_{\mathrm{up}^X(x_i)}\} \in S_L^{R_i}$. Then, since the vertices $\gamma^r_{\mathrm{mid}^X(x_i)}$, $\alpha^{(r,1)}_{\mathrm{up}^X(x_i)}$, and $\gamma^1_{\mathrm{mid}^X(x_i)}$ form an induced P_3, we have $\{\alpha^{(r,1)}_{\mathrm{up}^X(x_i)}, \gamma^1_{\mathrm{mid}^X(x_i)}\} \in S_L^{F_i^1}$. Then, since $\Lambda(\delta_{\mathrm{down}^X(x_i)}) \cap \Lambda(\gamma^1_{\mathrm{mid}^X(x_i)}) = \{T_i^1, F_i^1\}$ due to Claim 4 and the vertices $\delta_{\mathrm{down}^X(x_i)}$, $\gamma^1_{\mathrm{mid}^X(x_i)}$, $\alpha^{(r,1)}_{\mathrm{up}^X(x_i)}$ form an induced P_3 we conclude that $\{\delta_{\mathrm{down}^X(x_i)}, \gamma^1_{\mathrm{mid}^X(x_i)}\} \in S_L^{T_i^1}$ as claimed.

Statement *b*) can be shown with the same arguments by considering the vertex $\alpha^{(1,r)}_{\mathrm{up}^X(x_i)}$ instead of $\alpha^{(r,1)}_{\mathrm{up}^X(x_i)}$. ◊

Next we use Claim 11 to show that every clause is satisfied by A. Let $C_j \in \mathcal{C}$ be a clause. Then, there is an edge $e_1 := \{\eta_{\mathrm{up}^C(C_j)}, \theta_{\mathrm{down}^C(C_j)}\} \in E$. By Claim 9 we have $\Lambda(\eta_{\mathrm{up}^C(C_j)}) \cap \Lambda(\theta_{\mathrm{down}^C(C_j)}) = \mathfrak{L}(C_j)$. Since L is Λ-satisfying, it follows that $e_1 \in S_L^Y$ for some $Y \in \mathfrak{L}(C_j)$.

Consider the case $Y = T_i^r$ for some variable x_i that occurs positively in C_j and $r = \Omega(C_j, x_i)$. We show that $A(x_i) =$ 'true'. Since x_i occurs in C_j there is an edge $e_2 := \{\delta_{\mathrm{down}^X(x_i)}, \eta_{\mathrm{up}^C(C_j)}\} \in E$ which can only be an element of the strong classes $S_L^{T_i^r}$ or $S_L^{F_i^r}$ due to Claim 10. Since e_1 and e_2 form an induced P_3 and L is an STC-labeling we have $e_2 \in S_L^{F_i^r}$. The edge $e_3 := \{\delta_{\mathrm{down}^X(x_i)}, \gamma^r_{\mathrm{mid}^X(x_i)}\}$ forms an induced P_3 with e_2 and can only be an element of the strong classes $S_L^{T_i^r}$ or $S_L^{F_i^r}$ by Claim 4. Hence, $e_3 \in S_L^{T_i^r}$. By Claim 11 we may conclude that $\{\delta_{\mathrm{down}^X(x_i)}, \gamma^1_{\mathrm{mid}^X(x_i)}\} \in S_L^{T_i^1}$ and therefore $A(x_i) =$ 'true'. Hence, C_j is satisfied by A.

For the case $Y = F_i^r$ we can use the same arguments to conclude $A(x_i) =$ 'false'. Consequently, A satisfies every clause of ϕ. □

Note that in the instance constructed in the proof of Theorem 3.2, every edge has at most three possible strong colors and $c \in \mathcal{O}(n)$ where n is the number of variables in ϕ. This implies the following.

Corollary 3.3. *If the ETH is true, then*

a) EL-Multi-STC *cannot be solved in* $2^{o(|V|^2)}$ *time even if restricted to instances where* $k = 0$ *and* $\max_{e \in E} |\Psi(e)| = 3$.

b) VL-Multi-STC *cannot be solved in* $c^{o(|V|^2/\log|V|)}$ *time even if restricted to instances where* $k = 0$.

Since the lower bound holds for instances with $k = 0$, we also obtain a lower bound for approximation algorithms for VL-Multi-STC and EL-Multi-STC.

Corollary 3.4. *If the ETH is true, then there exists no approximation algorithm for* VL-Multi-STC *that runs in time* $2^{o(n^2)}$.

Recall that our reduction is inspired by a reduction used to show that Rainbow Coloring cannot be solved in $2^{o(n^{3/2})}$ time under the ETH [117]. We remark that for Rainbow Coloring another ETH-based lower bound of $2^{o(m)}n^{\mathcal{O}(1)}$ has been shown recently [2]. Note that a lower bound of $2^{o(m)}n^{\mathcal{O}(1)}$ is not implied by the $2^{o(n^{3/2})}$ lower bound since m might be quadratic in n. In case of VL-Multi-STC, the lower bound from Theorem 3.2 directly implies that VL-Multi-STC can not be solved in $2^{o(m)}$ time, since $m \in \mathcal{O}(n^2)$. Consequently, the simple $3^m \cdot n^{\mathcal{O}(1)}$-time algorithm for EL-Multi-STC behind Proposition II.7 is optimal in the sense that 3^m can not be replaced by a factor that is subexponential in m.

3.3 Concluding Remarks

We have provided a first study of the classic complexity of Multi-STC and the fine-grained complexity of two of its generalizations. We also stated the correspondence between EL-Multi-STC and a vertex-coloring problem in the Gallai graph. Furthermore, we discussed the relationship between Multi-STC and edge coloring problems. In context of this work, the negative results presented in this chapter motivate the study of the parameterized complexity that we consider in the next two chapters.

Open Questions. There are many interesting research questions that can be pursued in future work. Most importantly, it is open whether Multi-STC admits an algorithm with running time $2^{\mathcal{O}(n)}$. Even for instances with $k = 0$ it is unknown whether such an algorithm exists. We would like to remark that an algorithm with such a running time is also open for Edge Coloring: so far, it is only known that List-Edge Coloring, where a color list is given for each edge, admits no $2^{o(n^2)}$-time algorithm under the ETH [118]. A first step to prove a lower bound for Edge Coloring could be to prove that a vertex list version of Edge Coloring admits no $2^{o(n^2)}$-time algorithm as well.

Analogously to the complexity dichotomy presented in Chapter 2, it might be interesting to analyze the classic complexity of Multis-STC and its generalizations on graph classes that can be defined via forbidden induced subgraphs. Recall that Edge Coloring is NP-hard on triangle-free graphs [126] and therefore, Multi-STC is NP-hard on H-free graphs for every H containing a triangle. Since STC is solvable in polynomial time on P_4-free graphs [110] it might be interesting to investigate if this result can be lifted to Multi-STC. Note that even on $(K_2 + K_1)$-free graphs it is not known if Multi-STC can be solved in polynomial time.

In Chapter 2 we also analyzed the correspondence between STC and CD. The question on correspondence of STC and CD can also be formulated as follows: Is there an STC solution in which every strong component is a clique? A strong component is a connected component in the graph where only the strong edges of the solution are present. A similar question regarding the solution structure can be asked for Multi-STC: How do the connected components look when only the edges of a fixed strong color are present? In which cases do these components form a clique? An example where not all strong components are cliques is an instance where the input graph is a diamond: Consider a 2-colored STC-labeling of a diamond such that there are no weak edges. The strong components are either two K_2s and a P_4, or a triangle and a P_3. Thus, in every possible solution, there are strong components that do not form a clique. Is this also the case if the input graph is diamond-free?

It could also be interesting to study a version of Multi-STC with the additional constraint that there is at most one (non trivial) strong component for every strong color. This might fit into the context of social network analysis when one associates one strong color with one community in the network. Observe that for this version of Multi-STC, we cannot conclude NP-hardness from existing results for Edge Coloring.

By the NP-hardness dichotomy from Theorem 3.1, Multi-STC with $c \geq 3$ is presumably not FPT for k. However, Multi-STC with $c = 2$ is FPT, since there exists a parameter preserving reduction to Odd Cycle Transversal [164]. This reduction also implies that Multi-STC admits a polynomial kernel for k: Given an instance of Multi-STC, apply the parameter preserving reduction to Odd Cycle Transversal. Afterwards, use an existing polynomial kernelization for Odd Cycle Transversal [120] and convert the resulting instance back into an instance of Multi-STC in polynomial time. The last step is possible since Multi-STC is NP-hard and Odd Cycle Transversal belongs to NP and thus, there exists such a polynomial-time reduction. Note that the size of the resulting instance of Multi-STC is polynomially bounded in k. However, the degree of the polynomial is not known. An interesting open question would thus be, if one could think of any direct problem kernelization for the parameter k or if one could find the reduction from Odd Cycle Transversal to Mulit-STC. The problem kernel for Odd Cycle Transversal is based on a randomized technique [120]. It might be interesting to study whether this idea can be adapted for Multi-STC, or if there is a simpler deterministic kernelization. One could also aim to find efficient FPT algorithms for k.

Chapter 4

Multicolored STC Parameterized by STC Solution Size

Motivated by the strong hardness results from the previous chapter we now study the parameterized complexity of MULTI-STC and its two generalizations VL-MULTI-STC and EL-MULTI-STC. Recall that MULTI-STC with $c = 1$—in other words STC—is FPT when parameterized by k [164] and admits a problem kernel with at most $2k$ vertices [23]. Due to Theorem 3.1, all variants of MULTI-STC are NP-hard even if $k = 0$. Thus, it is presumably impossible that MULTI-STC or its generalizations are FPT when parameterized by k.

Even though, fixed-parameter tractability for k is unlikely, we analyze how the positive results for STC can be lifted to the more general problem variants. To this end, we introduce a new structural parameter denoted by k_1. Informally, k_1 is the solution size of STC. Formally, we define k_1 as follows.

Definition 4.1. *Let $G = (V, E)$ be a graph and let $L = (S_L, W_L)$ be a 1-colored labeling for G such that*

- *L is an STC-labeling, and*
- *there is no 1-colored STC-labeling $L' = (S_{L'}, W_{L'})$ for G with $|W_{L'}| < |W_L|$.*

We set $k_1 := k_1(G) := |W_L|$.

Observe that given an instance of MULTI-STC we can compute k_1 in $\mathcal{O}(1.28^{k_1} + nm)$ time as discussed in Chapter 2 and immediately accept if $k \geq k_1$; in this sense one may assume $k \leq k_1$ for MULTI-STC. For VL-MULTI-STC and EL-MULTI-STC this is not necessarily true since some lists might be empty which enforces weak edges.

Table 4.1: An overview of the parameterized complexity results for the parameters number k of weak edges, number c of colors, and number k_1 of weak edges for $c = 1$.

	Multi-STC	VL-Multi-STC	EL-Multi-STC
k	FPT if $c \leq 2$ [164], NP-hard for $k = 0$ for all $c \geq 3$ (Thm 3.1)		
k_1	$(k_1 + 1)^{k_1} \cdot n^{\mathcal{O}(1)}$ time (Thm 4.4) $4k_1$-vertex kernel (Cor 4.11)	W[1]-hard (Thm 4.13)	
$k_1 + c$	$\mathcal{O}((c+1)^{k_1} \cdot (cm + nm))$ time (Thm 4.2)		
	$4k_1$-vertex kernel (Cor 4.11)	no polynomial kernel (Cor 4.14) $2^{c+1}k_1$-vertex kernel (Cor 4.11)	

The parameter k_1 is relevant for two reasons: First, it allows us to determine to which extent the FPT algorithms for STC carry over to Multi-STC, VL-Multi-STC, and EL-Multi-STC. Second, k_1 has a structural interpretation: it is the vertex cover number of the Gallai graph of the input graph G [164]. We believe that this parameterization is of independent interest and might be useful for other problems.

Our Results. Table 4.1 shows an overview of the parameterized complexity results presented in this chapter.

First, we provide an $\mathcal{O}((c+1)^{k_1} \cdot (c|E| + |V| \cdot |E|))$-time algorithm for the most general problem, EL-Multi-STC. We then use this algorithm to show that Multi-STC is FPT when parameterized by k_1 alone.

Second, we present a $2^{c+1} \cdot k_1$-vertex kernel for VL-Multi-STC and EL-Multi-STC by extending the $4k$-vertex kernelization [77] for STC. In the case of Multi-STC the kernelization gives a kernel with at most $4k_1$ vertices, thus extending the linear-vertex kernel from $c = 1$ to arbitrary values of c.

Third, we outline the limits of parameterization by k_1. We show that VL-Multi-STC and EL-Multi-STC parameterized by k_1 alone are W[1]-hard. Moreover, both problems are unlikely to admit a kernel that is polynomial in $c + k_1$, which complements the $2^{c+1} \cdot k_1$-vertex kernel.

4.1 A Fixed-Parameter Algorithm

We provide a simple FPT algorithm for EL-Multi-STC parameterized by (c, k_1), which is the most general of the three problems. The main idea of the algorithm is to solve List-Colorable Subgraph on the Gallai graph of the input graph which is equivalent due to Proposition II.6.

Recall that the *Gallai graph* $\widetilde{G} := (\widetilde{V}, \widetilde{E})$ of a given graph $G = (V, E)$ is a graph with vertex set $\widetilde{V} := E$ and two vertices are adjacent in $\widetilde{G}$ if and only if the corresponding edges of G form an induced P_3 in G. Moreover, we recall the definition of List Colorable Subgraph. Given a graph $G = (V, E)$, we call a mapping $\chi : V \to \{0, 1, \dots, c\}$ a *subgraph-c-coloring* if there is no edge $\{u, v\} \in E$ with $\chi(u) = \chi(v) \neq 0$. Vertices v with $\chi(v) = 0$ correspond to deleted vertices. The List-Colorable Subgraph problem is defined as follows.

List-Colorable Subgraph
Input: An undirected graph $G = (V, E)$, integers $c \in \mathbb{N}$, $k \in \mathbb{N}_0$ and lists $\Gamma : V \to 2^{\{1,\dots,c\}}$.
Question: Is there a subgraph-c-coloring $\chi : V \to \{0, 1, \dots, c\}$ such that $|\{v \in V \mid \chi(v) = 0\}| \leq k$ and $\chi(w) \in \Gamma(w) \cup \{0\}$ for every $w \in V$?

Theorem 4.2. EL-Multi-STC *can be solved in* $\mathcal{O}((c + 1)^{k_1} \cdot (cm + nm))$ *time.*

Proof. We first describe the algorithm and analyze its running time. Afterwards, we show that the algorithm is correct.

Algorithm. Let (G, c, k, Ψ) be an instance of EL-Multi-STC. The first step is to compute the Gallai graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ of G which has m vertices and at most nm edges. Observe that (G, c, k, Ψ) is equivalent to the instance $(\widetilde{G}, c, k, \Psi)$ of List-Colorable Subgraph due to Proposition II.6. We describe an algorithm that solves $(\widetilde{G}, c, k, \Psi)$ in $\mathcal{O}((c + 1)^s \cdot (|\widetilde{V}| \cdot c + |\widetilde{E}|))$ time, where $s = k_1$ denotes the size of a minimum vertex cover of $\widetilde{G}$.

Let $S \subseteq \widetilde{V}$ be a size-s vertex cover of $\widetilde{G}$, which can be computed in $\mathcal{O}(1.28^s + sn)$ time [27]. Let $I := \widetilde{V} \setminus S$ denote the remaining independent set. We now compute whether $\widetilde{G}$ has a subgraph-c-coloring $a : \widetilde{V} \to \{0, 1, \dots, c\}$ with $|\{v \in \widetilde{V} \mid a(v) = 0\}| \leq k$.

We enumerate all possible mappings $a_S : S \to \{0, 1, \dots, c\}$. Observe that there are $(c + 1)^s$ such mappings. For each a_S we check whether $a_S(v) \in \Psi(v) \cup \{0\}$ for all $v \in S$. Furthermore, we check in $\mathcal{O}(|\widetilde{V}| \cdot c + |\widetilde{E}|)$ time whether a_S is a subgraph-c-coloring for $\widetilde{G}[S]$. If this is not the case, then discard the current a_S. For every other choice of a_S we proceed as follows:

We check whether it is possible to extend a_S to a mapping $a : \widetilde{V} \to \{0, 1, \dots, c\}$ that is a subgraph-c-coloring for $\widetilde{G}$: For each vertex $v \in I$, we check whether $P_v := \Psi(v) \setminus \{a_S(w) \mid w \in N_{\widetilde{G}}(v)\}$ is empty. If $P_v = \emptyset$ we set $a(v) = 0$. Otherwise, we set $a(v) = p$ for some arbitrary $p \in P_v$. This can be done in $\mathcal{O}(|\widetilde{V}| \cdot c + |\widetilde{E}|)$ time. The resulting mapping $a : V \to \{0, 1, \ldots, c\}$ is a subgraph-c-coloring for $\widetilde{G}$, since a_S is a subgraph-c-coloring for $G[S]$ and every $v \in I$ has a color $a(v)$ distinct from all vertices in $N(v) \subseteq S$. Finally, we check whether the total number of vertices with $a(v) = 0$ is at most k. If so, then we accept and stop. Otherwise, we continue with the next mapping a_S. If we did not accept for any of the enumerated mappings a_S, then we reject.

Running Time. The overall running time of the algorithm is $\mathcal{O}((c+1)^s \cdot (nc+m))$. Recall that $k_1 = s$, $|\widetilde{V}| = m$ and $|\widetilde{E}| \leq nm$. Therefore, the overall running time to solve EL-Multi-STC is $\mathcal{O}((c+1)^{k_1} \cdot (cm + nm))$.

Correctness. To see that the above algorithm is correct, observe first that it only accepts if it has found a subgraph-c-coloring for $\widetilde{G}$ with $|\{v \in \widetilde{V} \mid a(v) = 0\}| \leq k$. For the other direction, assume that there is a subgraph-c-coloring $a^\star \colon \widetilde{V} \to \{0, 1, \ldots, c\}$ with $|\{v \in \widetilde{V} \mid a^\star(v) = 0\}| \leq k$. One of the enumerated mappings a_S satisfies $a_S = a^\star|_S$. For this mapping a_S the algorithm sets $a(v) = 0$ for some vertex $v \in I$ if and only if $\Psi(v) \setminus \{a^\star(w) \mid w \in N_{\widetilde{G}}(v)\}$ is empty. Thus, the number of vertices $v \in \widetilde{V}$ with $a(v) = 0$ is at most $|\{v \in \widetilde{V} \mid a^\star(v) = 0\}| \leq k$, as required. □

Next, we conclude that Multi-STC parameterized by k_1 alone is fixed-parameter tractable. To this end, we observe the following relationship between c and k_1.

Lemma 4.3. *Let $I := (G = (V, E), c, k)$ be an instance of* Multi-STC*. If $c > k_1(G)$, then I is a yes-instance.*

Proof. Let $c > k_1$. There exists an STC-labeling $L = (S_L, W_L)$ for G with one strong color and $|W_L| = k_1$. Let $e_1, e_2, \ldots, e_{k_1}$ be the weak edges of L. We define a c-colored labeling $L^+ := (S^1_{L^+}, \ldots, S^c_{L^+}, W_{L^+})$ by

$$W_{L^+} := \emptyset \text{ and } S^i_{L^+} := \begin{cases} \{e_i\} & \text{for } i \in \{1, \ldots, k_1\}, \\ S_L & \text{for } i = k_1 + 1, \\ \emptyset & \text{for } i \in \{k + 2, \ldots, c\}. \end{cases}$$

Since $c > k_1$, every edge of G is labeled by L^+. Because L is an STC-labeling, there is no induced P_3 containing two edges from $S^{k_1+1}_{L^+} = S_L$. Furthermore, since $|S^i_{L^+}| \leq 1$ for $i \neq k_1 + 1$, the labeling L^+ is an STC-labeling. Since $|W_{L^+}| = 0$, it holds that (G, c, k) is a yes-instance of Multi-STC for every integer k. □

Lemma 4.3 implies an FPT algorithm for Multi-STC parameterized by k_1 alone: Let (G, c, k) be an instance of Multi-STC. If $c > k_1$, the given instance is a yes-instance by Lemma 4.3. We thus only need to consider instances with $c \leq k_1$. Replacing c by k_1 in the running time bound of Theorem 4.2 then gives the following.

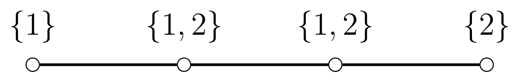


Figure 4.1: A graph G and vertex lists Λ of an instance $I := (G, c = 2, k = 0, \Lambda)$ of VL-Multi-STC. It is easy to see that I is a no-instance while $k_1 = 1 < 2 = c$.

Theorem 4.4. Multi-STC *can be solved in* $\mathcal{O}((k_1 + 1)^{k_1} \cdot (k_1 m + nm))$ *time.*

The fixed-parameter tractability of Multi-STC parameterized by k_1 alone relies on the relationship between c and k_1 from Lemma 4.3. Unfortunately, Lemma 4.3 does not hold for VL-Multi-STC; Figure 4.1 shows an example.

4.2 A Problem Kernel for EL-Multi-STC

We now show that EL-Multi-STC admits a $2^{c+1} \cdot k_1$-vertex kernel. That is, a linear-vertex kernel for every fixed value of c. The kernelization described in this section generalizes a $4k$-vertex kernel for STC [77].

To obtain a $2^{c+1} \cdot k_1$-vertex kernel, we introduce a new parameter τ. Let $I := (G, c, k, \Psi)$ be an instance of EL-Multi-STC. Then, $\tau := |\Psi(E) \setminus \{\emptyset\}|$ is defined as the number of different non-empty edge lists occurring in the instance I. We then show that EL-Multi-STC admits a kernel with at most $(\tau + 1) \cdot 2k_1$ vertices. Since $\tau \leq 2^c - 1$, this gives us the desired $2^{c+1} \cdot k_1$-vertex kernel.

For this kernelization we use the notion of *critical cliques* and *critical clique graphs* [151]. These concepts were also used to obtain linear-vertex kernels for graph clustering problems parameterized by the number of edge modifications [85, 28] and for STC [77] parameterized by the number of weak edges k. The kernelization described here lifts this linear-vertex kernel for STC to the more general EL-Multi-STC.

Definition 4.5. *A* critical clique *of a graph G is a clique K where the vertices of K all have the same neighbors in $V \setminus K$ and K is maximal under this property. Given a graph $G = (V, E)$, let $\mathcal{K}$ be the collection of its critical cliques. The* critical clique graph *$\mathcal{C}$ of G is the graph $(\mathcal{K}, E_{\mathcal{C}})$ with $\{K_i, K_j\} \in E_{\mathcal{C}}$ if and only if $\{u, v\} \in E$ for all $u \in K_i$ and $v \in K_j$.*

For a critical clique K we let $\mathcal{N}(K) := \bigcup_{K' \in N_{\mathcal{C}}(K)} K'$ denote the union of its neighbor cliques in the critical clique graph and we let $\mathcal{N}^2(K) := \bigcup_{K' \in N^2_{\mathcal{C}}(K)} K'$ denote the union of the critical cliques at distance exactly two from K. Given a graph G, the critical clique graph of G can be constructed in $\mathcal{O}(n + m)$ time [91].

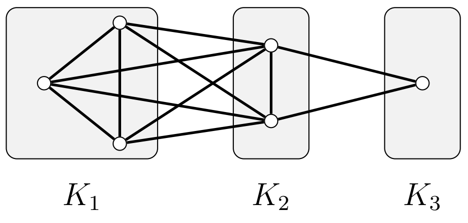


Figure 4.2: A graph where the vertex set is partitioned into critical cliques K_1, K_2, and K_3. Note that K_2 is an open critical clique while K_1 and K_3 are closed critical cliques.

Critical cliques are an important tool for EL-Multi-STC, because every edge contained in some critical clique is not part of any induced P_3 in G. Hence, each such edge e is strong under any STC-labeling unless $\Psi(e) = \emptyset$. In the following, we distinguish between two types of critical cliques. We say that a critical clique K is *closed* if $\mathcal{N}(K)$ forms a clique in G and that K is *open* otherwise. We will see that the number of vertices in open critical cliques is at most $2k_1$. The reduction rule leading to our kernel deals with large closed critical cliques. An example of the open and closed critical cliques of a graph is shown in Figure 4.2.

Before we state the reduction rule leading to the problem kernel, we provide some informal description. Let K be a critical clique and let $v \in \mathcal{N}(K)$. Furthermore, let there be a subset $K_v^\psi \subseteq K$ such that all edges between v and K_v^ψ have the same edge list $\psi \neq \emptyset$ under Ψ. The key idea is that all edges in $E(\{v\}, K_v^\psi)$ may receive the same color under an optimal c-colored STC-labeling. Our reduction rule then removes vertices from K_v^ψ if $|K_v^\psi|$ exceeds some size. This is done by considering each vertex $v \in \mathcal{N}(K)$ and each color list, and taking some edges between v and K with that color list and marking the other end of that edge. Afterwards, we remove all unmarked vertices in K. These steps are formally described in Algorithm 1.

Reduction Rule 4.1. *If G has a closed critical clique K with*

$$|K| > \tau \cdot |E(\mathcal{N}(K), \mathcal{N}^2(K))|,$$

then apply Algorithm 1 on G and K.

Proposition 4.6. *Rule 4.1 is safe and can be applied in polynomial time.*

Proof. Let $I := (G = (V, E), c, k, \Psi)$ be an instance for EL-Multi-STC and let K be a closed critical clique with $|K| > \tau \cdot |E(\mathcal{N}(K), \mathcal{N}^2(K))|$. Furthermore, let $I' := (G' = (V', E'), c, k', \Psi')$ be the instance we obtain after applying Algorithm 1 on

Algorithm 1 EL-Multi-STC kernelization

```
1: **Input:** A graph $G = (V, E)$ and a closed critical clique $K \subseteq V$ in $G$
2: for each $v \in \mathcal{N}(K)$ do
3:     for each $\psi \in \{\Psi(e) \neq \emptyset \mid e \in E(\{v\}, K)\}$ do
4:         $i := 0$
5:         for each $w \in N(v) \cap K$ do
6:             if $\Psi(\{v, w\}) = \psi$ then
7:                 Mark $w$ as *important*
8:                 $i := i + 1$
9:             if $i = |E(\{v\}, \mathcal{N}^2(K))|$ then
10:                break
11: Delete all vertices $u \in K$ which are not marked as important from $G$
12: Decrease the value of $k$ by the number of edges $e$ such that $e$ is incident with a
    deleted vertex $u$ and $\Psi(e) = \emptyset$.
```

G and K. Since $|\mathcal{N}(K)|, |N(v) \cap K| \leq n$ and $|\{\Psi(e) \neq \emptyset \mid e \in E(\{v\}, K)\}| \leq |K| \leq n$, the given algorithm clearly runs in $\mathcal{O}(n^3)$ time.

It remains to show that the produced instance I is a yes-instance if and only if I' is a yes-instance. To this end, let $D_V \subseteq V$ be the set of vertices that were deleted by Algorithm 1, let D_E be the set of edges that are incident with some $v \in D_V$, and let $D_E^{\emptyset} \subseteq D_E$ be the set of edges $e \in D_E$ with $\Psi(e) = \emptyset$. We have

$$G' = (V \setminus D_V, E \setminus D_E),\ k' = k - |D_E^{\emptyset}|, \text{ and } \Psi' = \Psi|_{E \setminus D_E}.$$

We also define $K' := K \setminus D_V$ as the modified critical clique in G'.

($\Rightarrow$) Let $L = (S_L^1, S_L^2, \ldots, S_L^c, W_L)$ be a Ψ-satisfying STC-labeling for G such that $|W_L| \leq k$. We define a labeling $\widehat{L} = (S_{\widehat{L}}^1, \ldots, S_{\widehat{L}}^c, W_{\widehat{L}})$ by $W_{\widehat{L}} := W_L \setminus D_E$ and $S_{\widehat{L}}^i := S_L^i \setminus D_E$ for each $i \in \{1, \ldots, c\}$. The fact that L is Ψ-satisfying implies that $\widehat{L}$ is Ψ'-satisfying. It also holds that

$$|W_{\widehat{L}}| = |W_L \setminus D_E| = |W_L| - |W_L \cap D_E| \leq k - |D_E^{\emptyset}| = k',$$

since $D_E^{\emptyset} \subseteq W_L \cap D_E$. It remains to prove that $\widehat{L}$ does not violate STC. Assume towards a contradiction that there is an induced P_3 on vertices u, v, and w in V' with edges $\{u, v\}, \{v, w\} \in S_{\widehat{L}}^i$ for some strong color $i \in \{1, \ldots, c\}$. Then $\{u, w\} \in D_E$, since L is an STC-labeling. Thus, by the definition of D_E, at least one of the vertices u or w was deleted by the algorithm. This contradicts the fact that u and w

are elements of $V' = V \setminus D_V$. Consequently, $\widehat{L}$ is a Ψ'-satisfying STC-labeling for G' with at most k' weak edges.

($\Leftarrow$) Conversely, let $\widehat{L} = (S^1_{\widehat{L}}, \dots, S^c_{\widehat{L}}, W_{\widehat{L}})$ be an optimal Ψ'-satisfying STC-labeling for G' such that $|W_{\widehat{L}}| \leq k - |D^{\emptyset}_E|$. We define a Ψ-satisfying STC-labeling L for G, with $|W_L| \leq k$. To this end, we show that we may make an assumption regarding the strong edges in $E_{G'}(K', \mathcal{N}(K'))$. Given a vertex $v \in \mathcal{N}(K')$ and an edge list $\psi \neq \emptyset$, we let $K^{\psi}_v \subseteq K'$ denote the inclusion maximal subset of vertices in K' where all edges in $E_{G'}(\{v\}, K^{\psi}_v)$ have list ψ under Ψ.

Claim 1. *There exists an optimal c-colored STC-labeling L^* for G' such that for every combination of a vertex $v \in \mathcal{N}(K')$ and a list $\psi \neq \emptyset$ with $|K^{\psi}_v| \geq |E_{G'}(\{v\}, \mathcal{N}^2(K'))|$ at least one edge in $E_{G'}(\{v\}, K^{\psi}_v)$ is strong under L^*.*

Proof. Recall that $\widehat{L}$ is an optimal c-colored STC-labeling for G'. If $\widehat{L}$ satisfies the property stated in the claim, nothing more needs to be shown. Otherwise, there exists some $v \in \mathcal{N}(K')$ and some K^{ψ}_v with $|K^{\psi}_v| \geq |E_{G'}(\{v\}, \mathcal{N}^2(K'))|$ such that $E_{G'}(\{v\}, K^{\psi}_v) \subseteq W_{\widehat{L}}$. Note that, whenever an edge $e \in E_{G'}(\{v\}, K_v)$ forms an induced P_3 with another edge e', we have $e' \in E_{G'}(\{v\}, \mathcal{N}^2(K'))$ since K' is closed. Since $\psi \neq \emptyset$, there exists some $i \in \psi$. We define a new labeling $L' := (S^1_{L'}, \dots, S^c_{L'}, W_{L'})$ for G' by

$$\begin{aligned}
S^i_{L'} &:= S^i_{\widehat{L}} \cup E_{G'}(\{v\}, K^{\psi}_v) \setminus E_{G'}(\{v\}, \mathcal{N}^2(K')),\\
W_{L'} &:= W_{\widehat{L}} \setminus E_{G'}(\{v\}, K^{\psi}_v) \cup (S^i_{\widehat{L}} \cap E_{G'}(\{v\}, \mathcal{N}^2(K'))), \text{ and}\\
S^j_{L'} &:= S^j_{\widehat{L}} \text{ for all } j \neq i.
\end{aligned}$$

Since $|K^{\psi}_v| \geq |E_{G'}(\{v\}, \mathcal{N}^2(K'))|$ we conclude

$$\begin{aligned}
|W_{L'}| &= |W_{\widehat{L}}| - |E_{G'}(\{v\}, K_v)| + |S^i_{\widehat{L}} \cap E_{G'}(\{v\}, \mathcal{N}^2(K'))|\\
&\leq |W_{\widehat{L}}| - |K_v| + |E_{G'}(\{v\}, \mathcal{N}^2(K'))|\\
&\leq |W_{\widehat{L}}|.
\end{aligned}$$

Moreover, L' clearly is Ψ'-satisfying. It remains to show that L' is an STC-labeling. To this end, we show that there is no induced P_3 with an edge $e \in E_{G'}(\{v\}, K_v) \subseteq S^i_{L'}$ and another edge $e' \in S^i_{L'}$. As mentioned above, the edges in $E_{G'}(\{v\}, K_v)$ only form an induced P_3 with edges in $E_{G'}(\{v\}, \mathcal{N}^2(K'))$. By the construction of L', no edge in $E_{G'}(\{v\}, \mathcal{N}^2(K'))$ belongs to $S^i_{L'}$. Hence, L' is an optimal c-colored STC-labeling for G' such that $E_{G'}(\{v\}, K^{\psi}_v)$ contains strong edges.

Observe that by transforming $\widehat{L}$ into L', we only changed the labels of edges in $E_{G'}(\{v\}, K_v^{\psi}) \cup E_{G'}(\{v\}, \mathcal{N}^2(K))$. Thus, we can apply this transformation subsequently for every combination of $v \in \mathcal{N}(K')$ and $\psi \neq \emptyset$ which results in an optimal c-colored STC-labeling L^* that satisfies the property stated in the claim. $\Diamond$

Throughout the rest of this proof, we assume without loss of generality that $\widehat{L}$ satisfies the property from Claim 1. We define the labeling L for G by extending $\widehat{L}$. To this end, we describe to which strong color classes of $\widehat{L}$ we add the edges in D_E. First, we label all edges in $D_E^{\emptyset}$ as weak. Let $W_L := W_{\widehat{L}} \cup D_E^{\emptyset}$ be the resulting set of weak edges. Since $|W_{\widehat{L}}| \leq k - |D_E^{\emptyset}|$ we have $|W_L| \leq k$. It remains to label all edges in $D_E \setminus D_E^{\emptyset}$. Let u be some fixed vertex in D_V and $v \in N(u)$ such that $\{u, v\} \notin D_E^{\emptyset}$. We consider the following cases.

Case 1: $v \in K$**.** Then, the edge $\{u, v\}$ is an edge between two vertices of a critical clique. Since $\{u, v\} \notin D_E^{\emptyset}$, there is some $i \in \Psi(\{u, v\})$. We add $\{u, v\}$ to $S^i_{\widehat{L}}$. Clearly, $\{u, v\}$ satisfies the Ψ-list property. Since $\{u, v\}$ is not part of any induced P_3 this does not violate STC.

Case 2: $v \in \mathcal{N}(K)$**.** Then, there is a set $Y \subseteq K'$ which contains at least $|E(\{v\}, \mathcal{N}^2(K))|$ vertices distinct from u such that $\Psi(\{v, y\}) = \Psi(\{u, v\})$ for every $y \in Y$. Otherwise, u would have been marked as *important* by Algorithm 1, which contradicts the fact that $u \in D_V$. Since $\widehat{L}$ satisfies the property of Claim 1, we know that $E_{G'}(\{v\}, Y)$ contains an edge $\{v, y\}$ that belongs to a strong color class $S^i_{\widehat{L}}$ for some $i \in \Psi(\{u, v\})$. We then add $\{u, v\}$ to $S^i_{\widehat{L}}$. Clearly, $\{u, v\}$ satisfies the Ψ-list property. Moreover, adding $\{u, v\}$ to $S^i_{\widehat{L}}$ does not violate STC: The only edges that form an induced P_3 with $\{u, v\}$ are the edges in $E(\{v\}, \mathcal{N}^2(K))$. Now, since $\{v, y\} \in S^i_{\widehat{L}}$ and since $\widehat{L}$ is an STC-labeling for G', none of these edges is contained in $S^i_{\widehat{L}}$.

Consequently, L is a Ψ-satisfying STC-labeling for G with $|W_L| \leq k$. $\square$

We next show that applying Reduction Rule 4.1 exhaustively leads to a problem kernel containing at most $(\tau + 1) \cdot 2k_1$ vertices. To this end, we provide two lemmas that we need to show the size bound. The first lemma gives a bound on the size of a closed critical clique in a reduced instance. The second lemma is a more general statement declaring that there is no edge connecting vertices of distinct closed critical cliques.

Lemma 4.7. *Let* (G, c, k, Ψ) *be a reduced instance of* EL-Multi-STC*. Then,* $|K| \leq \tau \cdot |E(\mathcal{N}(K), \mathcal{N}^2(K))|$ *for every closed critical clique* K *in* G*.*

Proof. We prove the lemma by having a closer look at the vertices that were not deleted by Algorithm 1. The algorithm is applied on every closed critical clique K

with $|K| > \tau \cdot |E(\mathcal{N}(K), \mathcal{N}^2(K))|$. Every vertex that was not marked as *important* in Line 7 of the algorithm is deleted from G. Note that there are at most τ possible images $\psi \neq \emptyset$ of $\Psi : E \to 2^{\{1,\dots,c\}}$. By Lines 7, 9, and 10 it holds that for every $v \in \mathcal{N}(v)$ the algorithm marks at most $\tau \cdot |E(\{v\}, \mathcal{N}^2(K))|$ vertices of K. Consequently, there are at most

$$\tau \cdot \sum_{v \in \mathcal{N}(K)} |E(\{v\}, \mathcal{N}^2(K))| = \tau \cdot |E(\mathcal{N}(K), \mathcal{N}^2(K))|$$

marked vertices in K, since $\{E(\{v\}, \mathcal{N}^2(K)) \mid v \in \mathcal{N}(v)\}$ forms a partition of the set $E(\mathcal{N}(K), \mathcal{N}^2(K))$. Hence, $|K| \leq \tau \cdot |E(\mathcal{N}(K), \mathcal{N}^2(K))|$ for every closed critical clique K in G. □

Lemma 4.8. *Let K_1 and K_2 be closed critical cliques in a graph G. Then,*

$$E_G(K_1, K_2) = \emptyset.$$

Proof. Assume towards a contradiction that $E_G(K_1, K_2) \neq \emptyset$. Then, since K_1 and K_2 are critical cliques, we have $\{v_1, v_2\} \in E$ for every $v_1 \in K_1$ and $v_2 \in K_2$. We consider the following cases.

Case 1: $\mathcal{N}(K_1) \setminus K_2 = \mathcal{N}(K_2) \setminus K_1$. Then, all vertices in $K_1 \cup K_2$ have the same closed neighborhood in G, which is a contradiction to the maximality of K_1 and K_2 since $K_1 \cup K_2$ forms a bigger critical clique.

Case 2: $\mathcal{N}(K_1) \setminus K_2 \neq \mathcal{N}(K_2) \setminus K_1$. Without loss of generality, assume that there exists a vertex $v \in \mathcal{N}(K_1) \setminus K_2$ with $v \notin \mathcal{N}(K_2) \setminus K_1$. Then, for any $w \in K_2$, the vertices v and w are contained in $\mathcal{N}(K_1)$ but not adjacent in G. This is a contradiction to the fact that K_1 is closed. □

We now prove the kernel result for EL-Multi-STC.

Theorem 4.9. EL-Multi-STC *admits a problem kernel with at most $(\tau + 1) \cdot 2k_1$ vertices.*

Proof. Due to Proposition 4.6, Reduction Rule 4.1 can be applied in polynomial time. Since every application of Reduction Rule 4.1 deletes some vertices, the rule can be exhaustively applied in polynomial time. Next, let $(G = (V, E), c, k, \Psi)$ be an instance that is reduced regarding Reduction Rule 4.1. We show that $|V| \leq (\tau + 1) \cdot 2k_1$. To this end, let $L := (S_L, W_L)$ be an optimal 1-colored STC-labeling for G. Recall that $|W_L| = k_1$.

We first show that the overall number of vertices in open critical cliques is bounded by $2k_1$. Let K be an open critical clique. Since $\mathcal{N}(K)$ does not form a clique in G, there are two vertices u and w in $\mathcal{N}(K)$ with $\{u, w\} \notin E$. So, for every vertex $v \in K$, the edges $\{u, v\}$ and $\{v, w\}$ form an induced P_3. Thus, each vertex in any open critical clique has at least one weak neighbor under L. Consequently, the overall number of vertices in open critical cliques is at most $2k_1$.

Let $\mathbb{K}$ denote the set of all vertices in closed critical cliques. We next show that $|\mathbb{K}| \leq 2 \cdot \tau \cdot k_1$. Intuitively, we show that there is a correspondence between the weak edges of L and all vertices in $\mathbb{K}$ such that for every weak edge under L there are at most $2 \cdot \tau$ distinct vertices in $\mathbb{K}$. Formally, we give a mapping $\Phi : \mathbb{K} \to W_L$ such that for each $e \in W_L$ we have $|\Phi^{-1}(e)| \leq 2 \cdot \tau$, where $\Phi^{-1}(e) := \{v \in \mathbb{K} \mid \Phi(v) = e\} \subseteq \mathbb{K}$. Since $|W_L| = k_1$, this implies

$$|\mathbb{K}| \leq \sum_{e \in W_L} |\Phi^{-1}(e)| \leq 2 \cdot \tau \cdot k_1.$$

It remains to show the existence of such a mapping Φ. First, let $v \in \mathbb{K}$ such that v is incident with an edge $\{v, w\} \in W_L$. We then set $\Phi(v) := \{v, w\}$. Next, let K be a closed critical clique and consider the vertices of K that are not incident with weak edges. Let $v \in K$. We define $\Phi(v) := \{w, u\}$ with $w \in \mathcal{N}(K)$ and $u \in \mathcal{N}^2(K)$. Since $|K| \leq \tau \cdot |E(\mathcal{N}(K), \mathcal{N}^2(K))|$ due to Lemma 4.7, we can do this in a way such that $\Phi^{-1}(\{w, u\})$ contains at most τ distinct vertices from K for every $\{w, u\} \in E(\mathcal{N}(K), \mathcal{N}^2(K))$. Note that for each such v with $\Phi(v) = \{w, u\}$, the edges $\{v, w\}$ and $\{w, u\}$ form a P_3 in G. Thus, since v is not incident with weak edges, we have $\{w, u\} \in W_L$. Therefore, Φ is well-defined. We next show that $|\Phi^{-1}(e)| \leq 2 \cdot \tau$ for every $e \in W_L$. Consider the following case distinction.

Case 1: $e = \{w, u\}$ such that one endpoint w lies in $\mathbb{K}$**.** Let K be the closed critical clique containing w. Due to Lemma 4.8 we have $u \notin \mathbb{K}$. Thus, any vertex $v \in \mathbb{K} \setminus \{w\}$ with $\Phi(v) = \{w, u\}$ has only strong neighbors under L.

Assume towards a contradiction that $|\Phi^{-1}(\{w, u\})| > 2 \cdot \tau$. Then, there exist at least two closed critical cliques K_1 and K_2 distinct from K that contain vertices $v_1 \in K_1$ and $v_2 \in K_2$ with $\Phi(v_1) = \Phi(v_2) = \{w, u\}$. Since $w \in K$ and vertices from distinct closed critical cliques are not adjacent by Lemma 4.8 we have $\{v_1, u\} \in E$ and $\{v_2, u\} \in E$. Furthermore, Lemma 4.8 also implies $\{v_1, v_2\} \notin E$. Then, $\{v_1, u\}$ and $\{v_2, u\}$ form a strong P_3 under L contradicting the fact that L is an STC-labeling. Consequently, $|\Phi^{-1}(\{w, u\})| \leq 2 \cdot \tau$.

Case 2: $e = \{w, u\}$ such that w and u do not belong to $\mathbb{K}$**.** Then, every $v \in \mathbb{K}$ with $\Phi(v) = \{w, u\}$ has only strong neighbors under L.

Assume towards a contradiction that $|\Phi^{-1}(\{w, u\})| > 2 \cdot \tau$. Then, there exist three distinct closed critical cliques K_1, K_2, and K_3 containing vertices $v_1 \in K_1$, $v_2 \in K_2$, and $v_3 \in K_3$ with $\Phi(v_1) = \Phi(v_2) = \Phi(v_3) = \{w, u\}$. By pigeonhole principle we may assume that v_1 and v_2 are both adjacent to u. Due to Lemma 4.8 we have $\{v_1, v_2\} \notin E$. Since v_1 and v_2 only have strong neighbors under L, the edges $\{v_1, u\}$ and $\{v_2, u\}$ form a strong P_3 under L. This contradicts the fact that L is an STC-labeling. Consequently, $|\Phi^{-1}(\{w, u\})| \leq 2 \cdot \tau$.

We thus have shown that $|\mathbb{K}| \leq 2 \cdot \tau \cdot k_1$. Therefore, G has at most $2k_1 + 2\tau k_1 = (\tau + 1) \cdot 2k_1$ vertices. □

Given an instance of EL-Multi-STC, Algorithm 1 removes vertices and their incident edges, but does not apply changes on the edge lists. Thus, the kernelization described above also implies a kernel for VL-Multi-STC: Given an instance (G, c, k, Λ), we covert the vertex lists into edge lists by setting $\Psi(\{u, v\}) := \Lambda(u) \cap \Lambda(v)$ for every edge $\{u, v\}$ of G. Afterwards, we reduce the resulting instance of EL-Multi-STC exhaustively regarding Reduction Rule 4.1 and obtain $(G' = (V', E'), c, k', \Psi|_{E'})$ with $|V'| \leq (\tau + 1) \cdot 2k_1$. We then return the VL-Multi-STC instance $(G', c, k', \Lambda|_{V'})$, where $\Lambda|_{V'}(v)$ is the union of all edge lists $\Psi|_{E'}(e)$ of edges e incident with v. It is easy to see that these two instances are equivalent. Therefore, the following holds.

Corollary 4.10. VL-Multi-STC *admits a problem kernel with at most* $(\tau+1)\cdot 2k_1$ *vertices, where* $\tau := |\{\Lambda(u) \cap \Lambda(v) \mid \{u, v\} \in E\} \setminus \{\emptyset\}|$.

Recall that for any EL-Multi-STC instance (G, c, k, Ψ) we have $\tau \leq 2^c - 1$. Also, Multi-STC is the special case of EL-Multi-STC where every edge has the list $\{1, 2, \ldots, c\}$, and thus $\tau = 1$. Thus, we obtain the following.

Corollary 4.11. *The following kernel results hold for strong triadic closure problems.*

a) EL-Multi-STC *and* VL-Multi-STC *admit a problem kernel with at most* $2^{c+1}k_1$ *vertices.*

b) Multi-STC *admits a problem kernel with at most* $4k_1$ *vertices.*

Recall that a problem kernel with at most $2k$ vertices for STC has been shown recently [23].

4.3 Limits of Parameterization by k_1

We next complement the positive results from Sections 4.1 and 4.2 by studying the limits of parameterization by k_1 and $c + k_1$.

Recall that the FPT algorithm for Multi-STC parameterized by k_1 alone behind Theorem 4.4 relies on the relationship between c and k_1 from Lemma 4.3.

Furthermore, recall that Figure 4.1 shows an example that this relationship does not hold for VL-Multi-STC. In the following, we show that VL-Multi-STC and EL-Multi-STC are presumably not fixed-parameter tractable when parameterized by k_1 alone. More precisely, we show that VL-Multi-STC is W[1]-hard for k_1 by giving a parameterized reduction from Set Cover which is defined as follows.

Set Cover
Input: A finite universe $U \subseteq \mathbb{N}$, a family $\mathcal{F} \subseteq 2^U$, and an integer $t \in \mathbb{N}$.
Question: Is there a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| \leq t$ and $\bigcup_{F \in \mathcal{F}'} F = U$?

We reduce from Set Cover parameterized by the dual parameter $|\mathcal{F}| - t$. The W[1]-hardness of Set Cover for this parameterization follows from a classic reduction from Independent Set [99]. We provide it here for the sake of completeness.

Proposition 4.12. Set Cover *parameterized by* $|\mathcal{F}| - t$ *is* W[1]*-hard.*

Proof. In Independent Set one is given a graph $G = (V, E)$ and an integer s and the question is whether there is a subset $V' \subseteq V$ of pairwise nonadjacent vertices such that $|V'| \geq s$. Independent Set is W[1]-hard when parameterized by s [46].

Let $(G = (V, E), s)$ be an instance of Independent Set. We construct a Set Cover-instance $(U, \mathcal{F}, t)$ as follows. Set $U := E$, $\mathcal{F} := \{F_v \mid v \in V\}$ with $F_v := \{\{v, u\} \mid u \in N(v)\}$ and $t := |V| - s$. Note that $|\mathcal{F}| = |V|$, hence $|\mathcal{F}| - t = |V| - (|V| - s) = s$. □

Theorem 4.13. VL-Multi-STC *parameterized by* k_1 *is* W[1]*-hard, even if* $k = 0$.

Proof. We give a parameterized reduction from Set Cover parameterized by $|\mathcal{F}| - t$ which is W[1]-hard due to Proposition 4.12.

Construction. Let $(U, \mathcal{F}, t)$ be an instance of Set Cover. We describe how to construct an equivalent instance $(G = (V, E), c, k, \Lambda)$ of VL-Multi-STC with $k_1 \leq |\mathcal{F}| - t$ and $k = 0$. Let $\mathcal{F} = \{F_1, \ldots, F_{|\mathcal{F}|}\}$. We define the vertex set V of G by $V := U \cup Z \cup \{a\}$ where a is a new vertex and $Z := \{z_i \mid i \in \{t+1, t+2, \ldots, |\mathcal{F}|\}\}$. Furthermore, we define the edge set of G by $E := E_U \cup E_{Ua} \cup E_{Za}$ with

$$\begin{aligned} E_U &:= \{\{v, w\} \mid v, w \in U\}, \\ E_{Ua} &:= \{\{u, a\} \mid u \in U\}, \text{ and} \\ E_{Za} &:= \{\{z, a\} \mid z \in Z\}. \end{aligned}$$

Note that $|E_{Za}| = |Z| = |\mathcal{F}| - t$ and that U is a clique. We set $c := |\mathcal{F}| + 1$ and define the lists Λ as

$$\Lambda(v) := \begin{cases} \{i \mid v \in F_i\} \cup \{|\mathcal{F}| + 1\} & \text{if } v \in U, \\ \{1, 2, \ldots, |\mathcal{F}|\} & \text{if } v \notin U. \end{cases}$$

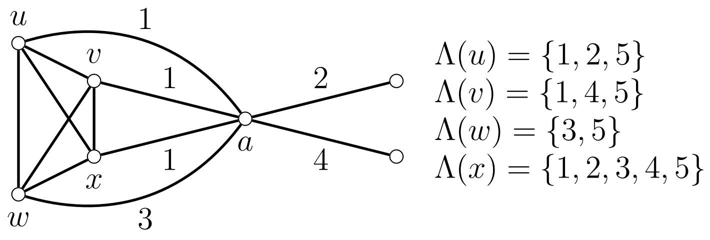

u
v
x
w
a
1
1
1
3
2
4
$\Lambda(u) = \{1, 2, 5\}$
$\Lambda(v) = \{1, 4, 5\}$
$\Lambda(w) = \{3, 5\}$
$\Lambda(x) = \{1, 2, 3, 4, 5\}$

Figure 4.3: An example of the construction in the proof of Theorem 4.13 for the Set Cover-instance $(U, \{F_1, F_2, F_3, F_4\}, 2)$ with $U = \{u, v, w, x\}$, $F_1 = \{u, v, x\}$, $F_2 = \{u, x\}$, $F_3 = \{w, x\}$, and $F_4 = \{v, x\}$ together with a 5-colored STC-labeling. The edges between u, v, w, and x are labeled with strong color 5. Note that $\{F_1, F_3\}$ is a set cover of size 2 corresponding to the strong colors of edges in E_{Ua}.

Figure 4.3 shows an example of the construction.

Intuition. Before we prove the correctness, we provide some intuition. The strong colors $1, \dots, |\mathcal{F}|$ correspond to the sets $F_1, \dots, F_{|\mathcal{F}|}$. The vertex a selects sets from $\mathcal{F}$ by labeling the edges in E_{Ua} with the corresponding color. The edges in E_{Za} ensure that there are exactly t different strong colors left for the edges in E_{Ua}.

Parameter Transformation. We first show that $k_1 \leq |\mathcal{F}| - t$. Recall that k_1 is the number of weak edges in an optimal 1-colored STC-labeling for G. Let $e_1, e_2 \in E$ be the edges of an induced P_3 in G. Since $U \cup \{a\}$ is a clique by construction, at least one of the edges e_1 or e_2 belongs to E_{Za}. Since every P_3 in G contains at least one edge from E_{Za} we obtain a 1-colored STC-labeling $L := (S_L, W_L)$ by setting $W_L := E_{Z_a}$. Then, since $|E_{Za}| = |\mathcal{F}| - t$, we have $k_1 \leq |\mathcal{F}| - t$.

Correctness. It remains to show that $(U, \mathcal{F}, t)$ has a solution $\mathcal{F}'$ of size t if and only if G has a Λ-satisfying STC-labeling $L = (S^1_L, \dots, S^{|\mathcal{F}|+1}_L, W_L)$ with $W_L = \emptyset$.

($\Rightarrow$) Let $\mathcal{F}' \subseteq \mathcal{F}$ be a set cover of U with $|\mathcal{F}'| = t$. Without loss of generality (by renaming) let $\mathcal{F}' = \{F_1, F_2, \dots, F_t\}$. We define an STC-labeling $L = (S^1_L, \dots, S^{|\mathcal{F}|+1}_L, \emptyset)$ as follows. We start by defining the classes S^i_L for each $i \in \{t+1, t+2, \dots, |\mathcal{F}|+1\}$. We set

$$S^i_L := \{\{a, z_i\}\} \text{ for every } i \in \{t+1, t+2, \dots, |\mathcal{F}|\} \text{ and } S^{|\mathcal{F}|+1}_L := E_U.$$

Note that $S^{t+1}_L \cup \dots \cup S^{|\mathcal{F}|+1}_L = E_U \cup E_{Za}$, so by defining the strong color classes $S^{t+1}_L, \dots, S^{|\mathcal{F}|+1}_L$ we have labeled all edges in $E_U \cup E_{Za}$. Before we continue with the definition of L, we show that the definition of the strong classes $S^{t+1}_L, \dots, S^{|\mathcal{F}|+1}_L$ does not violate the STC property and every edge in $E_U \cup E_{Za}$ satisfies the Λ-list property. Since U is a clique by construction, there is no induced P_3 containing two edges from $S_L^{|\mathcal{F}|+1}$ violating STC in E_U. Moreover, since all sets $S_L^{t+1}, \dots, S^{|\mathcal{F}|}$ contain exactly one edge, there is obviously no STC violation in E_{Za}. For every vertex $u \in U$ it holds that $|\mathcal{F}| + 1 \in \Lambda(u)$, hence the Λ-list property is satisfied for every $e \in E_U$. Since $\{1, 2, \dots, |\mathcal{F}|\} = \Lambda(a) = \Lambda(z_{t+1}) = \dots = \Lambda(z_{|\mathcal{F}|})$, the edges in E_{Za} also satisfy the Λ-list property.

We now label the edges in E_{Ua} by defining the sets $S_L^1, \dots, S_L^t$. Recall that $\mathcal{F}' = \{F_1, \dots, F_t\}$ is a set cover of size t. We set

$$S_L^1 := \{\{u, a\} \mid u \in F_1\} \text{ and } S_L^i := \{\{u, a\} \mid u \in F_i \setminus (F_1 \cup \dots \cup F_{i-1})\}$$

for each $i \in \{2, 3, \dots, t\}$. Obviously, each edge of E_{Ua} is an element of at most one of the sets $S_L^1, \dots, S_L^t$. Since $\mathcal{F}'$ is a set cover, we know that $F_1 \cup \dots \cup F_t = U$. Hence, $S_L^1, \dots, S_L^t$ is a partition of E_{Ua}. Since $U \cup \{a\}$ forms a clique, no pair of edges in E_{Ua} violates the STC property. Moreover, since the edges in E_{Ua} receive different colors from the edges in E_{Za}, no pair of edges from these two sets violates the definition of STC-labelings. From the definition of Λ we know that $\Lambda(a) = \{1, \dots, |\mathcal{F}|\}$ and for every $u \in U$ it holds that $i \in \Lambda(u)$ if $u \in F_i$. Hence, every edge in E_{Ua} satisfies the Λ-list property. Consequently, L is a c-colored STC-labeling with $W_L = \emptyset$ such that every edge satisfies the Λ-list property under L.

($\Leftarrow$) Conversely, let $L = (S_L^1, \dots, S_L^{|\mathcal{F}|+1}, \emptyset)$ be a c-colored STC-labeling for G such that every edge of G satisfies the Λ-list property. We construct a set cover $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| \leq t$. We focus on the vertex a and its incident edges. Those are exactly the edges of $E_{Ua} \cup E_{Za}$. Since there are no weak edges, we know that all those edges are elements of strong color classes S_L^i. Since L is an STC-labeling and every pair of edges $e, e' \in E_{Za}$ forms a P_3, it follows by $|E_{Za}| = |\mathcal{F}| - t$ that the edges in E_{Za} are elements of $|\mathcal{F}| - t$ distinct color classes. Because there is no edge between the vertices of U and Z, it also holds that there is no $e \in E_{Ua}$ that is an element of the same strong color class as some $e' \in E_{Za}$. Otherwise, e and e' form a P_3 with the same strong color which contradicts the fact that L is an STC-labeling. Thus, the edges in E_{Ua} are elements of at most t distinct strong color classes, since $|\Lambda(a)| = |\mathcal{F}|$ and every edge of G satisfies the Λ-list property under L. Without loss of generality (by renaming) we can assume that those strong color classes are $S_L^1, \dots, S_L^t$.

We define $\mathcal{F}' := \{F_1, F_2, \dots, F_t\}$ (recall that $\mathcal{F} = \{F_1, F_2, \dots, F_{|\mathcal{F}|}\}$). Obviously, $|\mathcal{F}'| = t$. We next show that $\mathcal{F}' \subseteq \mathcal{F}$ is a set cover of U. The fact that all edges of G satisfy the Λ-list property under L implies that for every $u \in U$ there is a $j \in \{1, \dots, t\}$ such that $j \in \Lambda(u)$. Since $\Lambda(u) = \{i \mid u \in F_i\} \cup \{|\mathcal{F}| + 1\}$ for all $u \in U$ by construction, every $u \in U$ is an element of one of the sets $F_1, F_2, \dots, F_t$. Hence, $U = F_1 \cup F_2 \cup \dots \cup F_t$ and, thus, $\mathcal{F}'$ is a set cover of size t. $\square$

A closer look at the instance (G, c, k, Λ) for VL-Multi-STC constructed from the Set Cover instance $(U, \mathcal{F}, t)$ in the proof of Theorem 4.13 reveals that $c = |\mathcal{F}| + 1$ and $k_1 \leq |\mathcal{F}| - t$. Thus, $c + k_1 \leq 2|\mathcal{F}| + 1$ and the construction is a polynomial parameter transformation from Set Cover parameterized by $|\mathcal{F}|$ to VL-Multi-STC parameterized by $c + k_1$. Now, since Set Cover parameterized by $|\mathcal{F}|$ does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly [43] we obtain the following.

Corollary 4.14. VL-Multi-STC *parameterized by* $c + k_1$ *does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly.

Corollary 4.14 complements the $2^{c+1} \cdot k_1$-vertex kernel from Corollary 4.11 in the sense that it is unlikely to replace 2^{c+1} by a factor that is polynomial in $c + k_1$.

4.4 Concluding Remarks

We have provided the first study of the parameterized complexity of Multi-STC and its list versions. More precisely, we studied how the positive results for STC parameterized by solution size can be extended to the more general problems. We also outlined the limits of such an extension. In context of this work, studying the parameterized complexity of Multi-STC and its list versions is motivated by the strong hardness results presented in Chapter 3. The STC solution size k_1 is one of two parameterization for Multi-STC that we consider in this work. In the next chapter we introduce parameterization by a structural parameter, which is motivated by the close connection between Multi-STC and edge coloring problems.

Open Questions. We have shown that the list variants of Multi-STC can be solved in $(c+1)^{k_1} \cdot n^{\mathcal{O}(1)}$ time and that there is presumably no FPT algorithm for parameterization by k_1 alone due to W[1]-hardness. Note that the algorithm with running time $(c+1)^{k_1} \cdot n^{\mathcal{O}(1)}$ is not an XP algorithm since the value of c might be exponentially large in the total input size. Thus, one open question is, if VL-Multi-STC and EL-Multi-STC are XP for k_1. To answer this question, it might be a good starting point to consider instances where c is superpolynomial in the input size. Maybe it is possible to find a polynomial-time reduction that decreases the number of colors in such an instance.

By Theorem 3.1, Multi-STC is NP-hard even if the number k of allowed weak edges is 0. Analogously to STC it might be interesting to study the number of strong edges $\ell := m - k$ as a possible parameterization for Multi-STC and its list variants. A Multi-STC instance is a trivial yes-instance if the input graph has a maximal matching of size at least ℓ. In this sense, the parameter ℓ is closely related to the size of a maximal matching in the input graph and appears to be a promising parameter to obtain FPT algorithms for Multi-STC. However, for VL-Multi-STC and EL-Multi-STC there might be edges that may not receive any strong color due to the list constraints. Thus, the relation between ℓ and the matching size might not hold for these problem variants.

In case of EL-Multi-STC, it is even possible to observe W[1]-hardness for ℓ from the literature: Sintos and Tsaparas [164] introduced a problem called Max-Ego-STC where one is given a Graph G, a vertex v of G, and an integer ℓ. The question is whether one can label at least ℓ edges incident with v as strong such that the strong triadic closure property is satisfied for these edges. Sintos and Tsaparas proved NP-hardness of this problem by giving a parameter-preserving reduction from Clique parameterized by solution size which is known to be W[1]-hard [46]. Defining the edge lists of edges incident with v as $\{1\}$ and the remaining edge lists as $\emptyset$, Max-Ego-STC is a special case of EL-Multi-STC. Consequently, EL-Multi-STC is W[1]-hard for the number of strong edges. Note that it is not clear how to model Max-Ego-STC with vertex lists. Thus, one open question is whether VL-Multi-STC is FPT for ℓ. In case of EL-Multi-STC, one might consider a more restricted problem version, where none of the edge lists is empty and ask whether this restricted version is FPT for ℓ.

We introduced the parameter k_1 as STC solution size. In other words, k_1 is the number of weak edges in an STC-labeling with one strong color. Analogously to k_1, one may define a parameter $\ell_1 := m - k_1$ where ℓ_1 corresponds to the number of strong edges in an STC-labeling with one strong color. In case of Multi-STC, we intuitively have $\ell_1 \leq \ell$, since the number of strong edges in an STC-labeling with multiple strong colors should not be smaller than the number of strong edges in a 1-colored STC-labeling. Thus, if it turns out that Multi-STC is FPT for ℓ, it might be interesting to investigate whether it is also FPT for ℓ_1. However, recall that the current-best algorithm to compute a 1-colored STC labeling has running time ${\ell_1}^{\mathcal{O}(\ell_1)} \cdot n^{\mathcal{O}(1)}$ (Theorem 2.10). Thus, to obtain an FPT algorithm for Multi-STC parameterized by ℓ_1 that could be practically relevant, a first step might be to revisit STC parameterized by ℓ. In case of VL-Multi-STC and EL-Multi-STC, ℓ_1 is unbounded by the minimum number of strong edges in a solution, since there might be empty lists. Furthermore, observe that the reduction from Clique to Max-Ego-STC described above, the value of ℓ_1 is unbounded in the parameter. It is thus open, whether the list variants of Multi-STC are FPT for ℓ_1.

Recall that k_1 also has a structural interpretation as the vertex cover number of the Gallai graph. It would be interesting to study if k_1 or other structural parameters of the Gallai graph are useful for further problems. Furthermore, from an abstract point of view, the Gallai graph models the conflicts between edges with regard to the strong triadic closure property: Intuitively, every induced P_3 forms an STC conflict in the sense that the two edges of the P_3 may not receive the same strong color. Since each edge in the Gallai graph corresponds to two edges forming a P_3 in the original graph, the edges model the corresponding STC conflicts. Golovach et al. [71] considered a generalization of STC where the aim is to color at most k edges weak such that each induced subgraph isomorphic to a fixed graph F has at least one weak edge. The corresponding problem is called Strong F Closure. In case of $F = 2K_2$, one could think of a conflict graph that is defined analogously to the Gallai graph: There are vertices corresponding to the edges of the input graph and two such vertices are adjacent if their corresponding edges form a $2K_2$. It is easy to see that a minimum vertex cover of this auxiliary graph corresponds to the weak edges of a solution of Strong $2K_2$ Closure. Moreover, an algorithm similar to the algorithm from the proof of Theorem 4.2 could work for a multicolored version of Strong $2K_2$ Closure. It might be interesting to study whether a multicolored version of Strong $2K_2$ Closure has a polynomial kernel when parameterized by the vertex cover number of the auxiliary graph. Considering Strong F Closure for some F that contains more than two edges, it is not clear how a conflict graph can look like. For example, let $F = K_3$. Then, one might considers a conflict graph where an edge corresponds to two edges that are in one induced K_3 of the input graph. Such auxiliary graph is also known as the *anti-Gallai graph* [124]. However, note that a minimal vertex cover of the anti-Gallai graph does not correspond to a solution of Strong K_3 Closure.

Another open question is if the $2^{c+1} \cdot k_1$-vertex kernel for EL-Multi-STC can be improved. Recall that for STC the kernel size has recently been improved from $4k$ vertices to $2k$ vertices [23]. This kernel is obtained by removing vertices v where $N[v]$ is a clique and the degree of v is bigger than the number of edges between $N[v]$ and the rest of the graph. It is unknown if this approach can also be used to obtain better kernelizations for Multi-STC and its list versions.

Chapter 5

Edge Coloring Problems Parameterized by Distance to Low-Degree Graphs

As discussed in Chapter 3, the MULTI-STC problem has a strong relation to the classic EDGE COLORING where one aims to label the edges of a given graph G with a given number of colors such that no incident edges are labeled with the same color. Such a labeling is called a *proper (edge) labeling* of G. The existence of a proper labeling with c colors implies the existence of an STC-labeling with c strong colors and without weak edges. We used the relation between the two problems to give an NP-hardness dichotomy for MULTI-STC (Theorem 3.1). In this chapter, we revisit the relation between MULTI-STC and EDGE COLORING and we study parameterization by structural parameters that are motivated by this correspondence.

EDGE COLORING and its many variants form a fundamental problem family in algorithmic graph theory [33, 88, 90, 97]. By Vizing's famous theorem, the number of necessary colors for a proper edge coloring of a graph G is closely related to the degree of G.

Theorem 5.1 (Vizing's Theorem [178]). *For every graph G with maximum degree Δ there exists a proper edge labeling with $\Delta + 1$ colors.*

Since the proof of Vizing's theorem provides an algorithm that computes a proper edge labeling with $\Delta + 1$ colors, it is an early example of an additive approximation algorithm predating the theory of NP-completeness. Later it was shown that EDGE COLORING is NP-hard for $c = 3$ [90], and in light of Vizing's result it is clear that the hard instances for $c = 3$ are exactly the subcubic graphs. Not surprisingly, the NP-hardness extends to every fixed $c \geq 3$ [126].

In this chapter we study the more general Edge-Colorable Subgraph (ECS) problem and its relation to Multi-STC. Intuitively, we want to decide whether we can remove a certain number of edges from the input graph such that the remaining graph has a proper edge labeling with a given number of colors. Feige et al. [55] mention that ECS has applications in call admittance in telecommunication networks. The edge labelings that model this variant of Edge Coloring and the corresponding problem are defined as follows.

Definition 5.2. *A* c-labeling $L = (S_L^1, \dots, S_L^c, W_L)$ *is a* proper labeling *if there exists no pair of edges* $e_1, e_2 \in S_L^i$ *for some strong color* i*, such that* $e_1 \cap e_2 \neq \emptyset$*.*

Edge-Colorable Subgraph (ECS)
Input: An undirected graph $G = (V, E)$ and integers $c \in \mathbb{N}$ and $k \in \mathbb{N}_0$.
Question: Is there a c-colored proper labeling L with $|W_L| \leq k$?

Recall that ECS and Multi-STC are NP-hard even for constant c and k [90]. Therefore, both problems are presumably not fixed-parameter tractable for $k + c$. Instead of the parameter $k + c$, we consider the parameter ξ_{c-1} which we define as the minimum number of edges that need to be deleted in the input graph to obtain a graph with maximum degree $c - 1$. This is a distance-from-triviality parameterization [86]: Due to Vizing's Theorem, the answer is always yes if the input graph has maximum degree $c - 1$. We parameterize by the edge-deletion distance to this trivial case.

For another view on the parameter ξ_{c-1}, consider Edge Coloring instead of ECS. In Edge Coloring, the instances with maximum degree larger than c are trivial no-instances. Moreover, in non-trivial instances, the number of vertices with degree c is at most $2\xi_{c-1}$. Thus, in non-trivial instances, the parameter ξ_{c-1} is essentially the same as the number of vertices that have degree c. This is, arguably, one of the most natural parameterizations for Edge Coloring.

Recall that ξ_{c-1} is the edge-deletion distance to trivial yes-instances. We also study of ECS parameterized by a vertex-deletion distance to another case that is trivial due to Vizing's theorem. The *c-component order connectivity* is the size λ_c of a smallest vertex set D such that deleting D from G results in a graph where each connected component has at most c vertices. The parameter λ_c presents a different distance-from-triviality parameterization, since a graph with connected components of order at most c can trivially be colored with c edge colors. In case of $c = 1$, λ_c is the vertex cover number of the input graph. Moreover, observe that λ_c is non-increasing for increasing values of c. Thus, λ_c is never larger than the vertex cover number which is a popular structural parameter.

Table 5.1: A summary of our results for the two problems. Here, ξ_{c-1} denotes the edge-deletion distance to graphs with maximum degree at most $c-1$, and λ_c denotes the vertex-deletion distance to graphs where every connected component has order at most c.

	ECS	Multi-STC
$\xi_{c-1}+c$	$\mathcal{O}(\xi_{c-1}c)$-vertex kernel (Thm 5.8)	$\mathcal{O}(\xi_{c-1})$-edge kernel if $c \leq 4$, (Thm 5.28)
λ_c+c	$\mathcal{O}(c^3 \cdot \lambda_c)$-vertex kernel (Thm 5.13)	No poly Kernel, even for $c=1$ (Prop 5.14)

Related Work. As discussed in Chapter 2, there are many results regarding the classic complexity of Multi-STC for the case where $c=1$.

ECS is NP-hard for $c=2$ [55] and it has received a considerable amount of interest for small constant values of c such as $c=2$ [55, 115], $c=3$ [115, 116, 153], and $c \leq 7$ [98]. Regarding the parameterized complexity, there are FPT algorithms for the number $\ell := m-k$ of strong edges and the vertex cover number vc of the input graph [3]. It admits an $\mathcal{O}(\ell \cdot c)$-vertex kernel and an $\mathcal{O}(\text{vc} \cdot c)$-vertex kernel [3]. On the negative side, ECS does not admit a problem kernel of size $\mathcal{O}(x^{1-\varepsilon} \cdot f(c))$ for any $\varepsilon > 0$ and computable function f unless NP $\subseteq$ coNP/poly [3]. In the case of $c=2$, ECS is FPT when parameterized by structural width parameters [5].

Our Results. Table 5.1 shows a summary of the results of this chapter. We study problem kernelizations for ECS and Multi-STC parameterized by $\xi_{c-1}+c$ and λ_c+c.

First, we show that ECS admits a problem kernel with at most $4\xi_{c-1} \cdot c$ vertices and $\mathcal{O}(\xi_{c-1} \cdot c^2)$ edges that can be computed in $\mathcal{O}(n+m)$ time. Observe that this is a linear-size kernel for every fixed value of c. Second, we show that ECS admits a problem kernel with $\mathcal{O}(\lambda_c \cdot c^3)$ vertices.

Afterwards, we consider Multi-STC. We first observe that Multi-STC does not admit a polynomial kernel when parameterized by λ_c. Second, we study problem kernelization for the parameter ξ_{c-1}. Even though the core technique we used to obtain the kernel for ECS does not work for Multi-STC, we succeed to transfer the kernelization result from ECS to Multi-STC for $c \leq 4$. In fact, our result for $c \geq 3$ can be extended to a more general problem kernel with $\mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor+1} \cdot c)$ vertices and $\mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor+1} \cdot c^2)$ edges that can be computed in $\mathcal{O}(n+m)$ time. Here, $\xi_{\lfloor \frac{c}{2} \rfloor+1}$ denotes the number of edge-deletions needed to obtain maximum degree $\lfloor \frac{c}{2} \rfloor + 1$. For $c=5$, this gives a linear-size kernel for the parameter ξ_3, for $c=6$, a linear-size kernel for the parameter ξ_4 and so on. Our techniques to obtain this kernel are very loosely inspired by the proof of Vizing's Theorem but in the context of MULTI-STC several obstacles need to be overcome. As a result, the proof differs quite substantially from the one for ECS.

Notation and Parameter Definitions. For technical reasons, we let the first vertex on a path P have index 0 throughout this chapter. More precisely, a path of length r is a sequence $P := (v_0, \ldots, v_{r-1})$ and given an index $j \in \{0, \ldots, r-1\}$ we refer to the vertex v_j as $P(j)$.

In many proofs in this chapter we compare two labelings that are equal on a subset of edges and may differ on other edges. To this end, we introduce the following notation. Let $L := (S_L^1, \ldots, S_L^c, W_L)$ be a labeling for some graph $G := (V, E)$ and let $E' \subseteq E$. Then, $L|_{E'} := (S_L^1 \cap E', \ldots, S_L^c \cap E', W_L \cap E')$ denotes the *restriction of* L *to* E'. Let L' be another labeling for G. The labelings L and L' are are called *partially equal with respect to* E' if $L|_{E'} = L'|_{E'}$.

We also consider the colors of edges that are consecutive edges on a path in the graph. Given a path $P = (P(0), \ldots, P(r-1))$ we define the *color sequence* Q_L^P of P under L as the finite sequence $Q_L^P = (q_0, q_1, \ldots, q_{r-2})$ of elements in $\{0, \ldots, c\}$, such that $\{P(i), P(i+1)\} \in S_L^{q_i}$ if $q_i \geq 1$ and $\{P(i), P(i+1)\} \in W_L$ if $q_i = 0$.

We next define the parameter ξ_t. For a given graph $G = (V, E)$ and a constant $t \in \mathbb{N}_0$, we call $D_t \subseteq E$ an *edge-deletion set of* G *and* t if the graph $(V, E \setminus D_t)$ has maximum degree t. We define the parameter ξ_t as the size of the minimum edge-deletion set of G and t. Note that an edge-deletion set of G and t of size ξ_t can be computed in polynomial time [63]. More importantly for our applications, we can compute a factor-2 approximation D'_t for an edge-deletion set of size ξ_t in linear time as follows: Add for each vertex v of degree at least $t+1$ an arbitrary set of $\deg(v) - t$ incident edges to D'_t. Then $|D'_t| \leq \sum_{v \in V} \max(\deg(v) - t, 0)$. This implies that $|D'_t|$ is a 2-approximation for ξ_t since $\sum_{v \in V} \max(\deg(v) - t, 0) \leq 2\xi_t$ as every edge deletion decreases the degree of at most two vertices.

Proposition 5.3. *Given a graph* G *and* $t \in \mathbb{N}_0$*, a factor-2 approximation of an edge-deletion set of* G *and* t *can be computed in* $\mathcal{O}(n + m)$ *time.*

A given edge-deletion set D_t induces the following important partition of the vertex set V of a graph.

Definition 5.4. *Let* $t \in \mathbb{N}_0$*, let* $G = (V, E)$ *be a graph, and let* $D_t \subseteq E$ *be an edge-deletion set of* G *and* t*. We call* $\mathscr{C} = \mathscr{C}(D_t) := \{v \in V \mid \exists e \in D_t : v \in e\}$ *the set of* core vertices *and* $\mathscr{P} = \mathscr{P}(D_t) := V \setminus \mathscr{C}$ *the set of* periphery vertices *of* G*.*

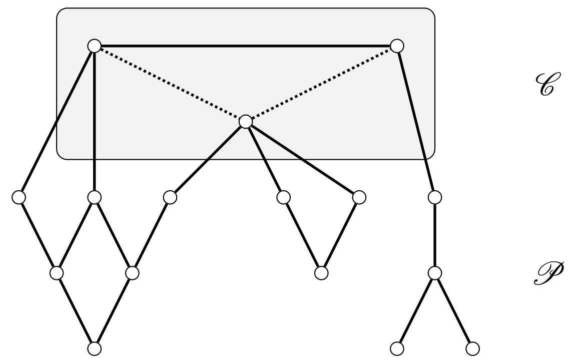


Figure 5.1: An example of a partition into core and periphery vertices. The dotted edges correspond to an edge-deletion set D_3. The upper part shows the core vertices and the lower part shows the periphery vertices.

Note that for arbitrary $t \in \mathbb{N}_0$ and G we have $|\mathscr{C}| \leq 2|D_t|$ and for every $v \in \mathscr{P}$ it holds that $\deg_G(v) \leq t$. Moreover, every vertex in $\mathscr{C}$ is incident with at most t edges in $E \setminus D_t$. Figure 5.1 shows an example of a graph where the vertex set is partitioned into core vertices and periphery vertices. In context of ECS and Multi-STC, for a given instance (G, c, k) we consider some fixed edge deletion set D_t of the input graph G and some integer t which depends on the value of c.

Finally, we define the parameter λ_t. For a given graph $G = (V, E)$ and a constant $t \in \mathbb{N}_0$, we call $D \subseteq V$ an *order-t component cover* if every connected component in $G - D$ contains at most t vertices. Then, we define the *component order connectivity* λ_t to be the size of a minimum order-t component cover. In context of ECS we study parameterization by λ_t, where $t = c$ is the number of colors. A factor-$(c + 1)$ approximation of the minimal order-c-component cover can be computed in polynomial time [122].

The parameters ξ_{c-1} and λ_c are incomparable in the following sense: In a path P_n the parameter λ_c can be arbitrarily large when n increases while $\xi_{c-1} = 0$ for all $c \geq 3$. In a star S_n the parameter ξ_{c-1} can be arbitrary large when n increases while $\lambda_c = 1$.

5.1 Kernelization for Edge-Colorable Subgraph

We first provide problem kernels for ECS parameterized by the edge deletion distance ξ_{c-1} to graphs with maximum degree $c - 1$, and the size λ_c of a minimum order-c component cover. We first show that ECS admits a kernel with $\mathcal{O}(\xi_{c-1} \cdot c)$ vertices and $\mathcal{O}(\xi_{c-1} \cdot c^2)$ edges that can be computed in $\mathcal{O}(n + m)$ time. Afterwards, we consider λ_c and show that ECS admits a problem kernel with $\mathcal{O}(c^3\lambda_c)$ vertices, which is a linear vertex kernel for every fixed value of c. Note that if $c = 1$, solving ECS is equivalent to computing a maximum matching, which can be done in polynomial time. Hence, we assume $c \geq 2$ for the rest of this section. In this case the problem is NP-hard [55].

5.1.1 ECS Kernel for $\xi_{c-1} + c$

The kernelization presented here is based on Vizing's theorem [178]. Note that Vizing's Theorem implies that an ECS instance (G, c, k) is always a yes-instance if $\xi_{c-1} = 0$. Thus, parameterization by ξ_{c-1} is a distance-from-triviality parameterization.

Our kernelization relies on the following lemma. This lemma is a reformulation of a known fact about edge colorings [166, Theorem 2.3] which, in turn, is based on the so-called *Vizing fan equation* [166, Theorem 2.1].

Lemma 5.5 ([166, Theorem 2.3]). *Let $G = (V, E)$ be a graph and let $e := \{u, v\} \in E$. Moreover, let there be a be a proper Δ_G-colored labeling L for the graph $(V, E \setminus \{e\})$ such that $W_L = \emptyset$. If*

$$\sum_{z\in Z}(\deg_G(z) + 1 - \Delta_G) < 2$$

for all $Z \subseteq N_G(u)$ with $|Z| \geq 2$ and $v \in Z$, then there exists a proper Δ_G-colored labeling L' for G such that $W_{L'} = \emptyset$.

Note that the sum $\sum_{z\in Z}(\deg_G(z) + 1 - \Delta_G)$ essentially counts the vertices with degree Δ_G in the neighborhood of u. Since we require that $v \in Z$, an intuitive understanding of the property of Lemma 5.5 is: The smaller the degree of v, the more vertices in $N_G(u)$ may have high degree.

As mentioned above, Lemma 5.5 is a consequence of the so-called Vizing fan equation [166, Theorem 2.1]. Intuitively, a *maximal Vizing fan* is a tool in context of graph coloring that is used to transform a proper edge labeling into another labeling with the same number of colors, when an additional edge is inserted. Roughly speaking, the maximal fan is a sequence of incident edges with one of the endpoints of the inserted edge. The set Z from Lemma 5.5 corresponds to the endpoints of the edges in this sequence. The process of transforming the coloring does not work if the endpoints of the edges of the maximal Vizing fan satisfy the inequality $\sum_{z\in Z}(\deg_G(z) + 1 - \Delta_G) \geq 2$. Intuitively, if we can show that for every possible choice of Z, this inequality is not satisfied, then there exists a maximal Vizing fan that can be used to transform the coloring into a new coloring of the graph with one additional edge.

Lemma 5.5 is a statement about proper labelings without weak edges. We use it for ECS to prove the next lemma which is the main tool that we need for our kernelization. To handle proper labelings that contain weak edges, we exploit the fact that, given any proper labeling L for a graph $G = (V, E)$, the labeling $(S_L^1, \dots, S_L^c, \emptyset)$ is a proper labeling for the graph $(V, E \setminus W_L)$.

Lemma 5.6. *Let $L := (S_L^1, S_L^2, \dots, S_L^c, W_L)$ be a proper labeling with $|W_L| = k$ for a graph $G := (V, E)$. Moreover, let $e \subseteq V$ such that $e \notin E$ and let $G' := (V, E \cup \{e\})$ be obtained from G by adding e. If for each endpoint u of e it holds that every vertex $w \in N_{G'}[u]$ has degree at most $c-1$ in G', then there exists a proper labeling L' for G' with $|W_{L'}| = k$.*

Proof. Consider the auxiliary graph $G_{\text{aux}} := (V, E \setminus W_L)$. Since L is a proper labeling for G, the labeling $L_{\text{aux}} := (S_L^1, \dots, S_L^c, \emptyset)$ is a proper labeling for G_{aux}. Let $H_{\text{aux}} := (V, E_H)$ where $E_H := (E \setminus W_L) \cup \{e\}$. In order to prove the lemma, we show that there exists a proper labeling L'_{aux} for H_{aux} such that $W_{L'_{\text{aux}}} = \emptyset$.

To this end, we first consider the maximum degree of H_{aux}. We have $\deg_{H_{\text{aux}}}(w) \leq \deg_{G'}(w)$ for all $w \in V$. Hence, the property that $\deg_{G'}(w) \leq c-1$ for all $w \in N_{G'}[u]$ implies $\Delta_{H_{\text{aux}}} = \max(\Delta_{G_{\text{aux}}}, c-1)$. Since L_{aux} is a proper c-colored labeling for G_{aux} we know that $\Delta_{G_{\text{aux}}} \leq c$ and therefore we have $\Delta_{H_{\text{aux}}} \leq c$. So, to find a proper c-colored labeling without weak edges for H_{aux} it suffices to consider the following cases.

Case 1: $\Delta_{H_{\text{aux}}} \leq c - 1$. Then, there exists a proper labeling L'_{aux} for H_{aux} such that $W_{L'_{\text{aux}}} = \emptyset$ due to Vizing's Theorem.

Case 2: $\Delta_{H_{\text{aux}}} = c$. In this case we can apply Lemma 5.5: Observe that $(V, E_H \setminus \{e\}) = G_{\text{aux}}$ and L_{aux} is a proper labeling for G_{aux} such that $W_{L_{\text{aux}}} = \emptyset$. Consider an arbitrary $Z \subseteq N_{H_{\text{aux}}}(u)$ with $|Z| \geq 2$ and $v \in Z$. Note that $Z \subseteq N_{H_{\text{aux}}}(u) \subseteq N_{G'}(u)$ and therefore $\deg_{H_{\text{aux}}}(z) \leq c - 1$ for all $z \in Z$. Thus,

$$\sum_{z \in Z} (\underbrace{\deg_{H_{\text{aux}}}(z)}_{\leq c-1} + 1 - \underbrace{\Delta_{H_{\text{aux}}}}_{=c}) < 2.$$

Since Z was arbitrary, there exists a proper labeling L'_{aux} for H_{aux} with $W_{L'_{\text{aux}}} = \emptyset$ due to Lemma 5.5.

We now define $L' := (S^1_{L'_{\text{aux}}}, S^2_{L'_{\text{aux}}}, \dots S^c_{L'_{\text{aux}}}, W_L)$. Note that the edge set $E \cup \{e\}$ of G' can be partitioned into W_L and the edges of G'_{aux}. Together with the fact that L'_{aux} is a labeling for G'_{aux} it follows that every edge of G' belongs to exactly one color class of L'. Moreover, it obviously holds that $|W_{L'}| = |W_L| = k$. Since there is no vertex with two incident edges in the same strong color class $S^i_{L'_{\text{aux}}}$, the labeling L' is a proper labeling for G'. $\square$

We now introduce the reduction rule leading to the problem kernelization. Intuitively, Lemma 5.6 guarantees that when we have a proper c-labeling of a graph, we may insert a new edge e into the graph such that the resulting graph still has a proper c-labeling if one endpoint of e has only neighbors with a low degree. Recall that $\mathscr{C}$ is the set of vertices that are incident with at least one of the ξ_{c-1} edge-deletions that transform G into a graph with maximum degree $c-1$. We make use of the fact that edges that have at least one endpoint u that is not in $\mathscr{C} \cup N(\mathscr{C})$ satisfy $\deg(w) \le c-1$ for all $w \in N[u]$. Lemma 5.6 guarantees that these edges are not important to decide whether an instance of ECS is a yes-instance.

Reduction Rule 5.1. *Remove all vertices in $V \setminus (\mathscr{C} \cup N(\mathscr{C}))$ from G.*

Proposition 5.7. *Rule 5.1 is safe.*

Proof. Let $(G' = (V', E'), c, k)$ be the reduced instance after applying Rule 5.1. We prove the safeness of Rule 5.1 by showing that there is a proper labeling with at most k weak edges for G if and only if there is a proper labeling with k weak edges for G'.

($\Rightarrow$) Let $L = (S^1_L, S^2_L, \dots, S^c_L, W_L)$ be a proper labeling with $|W_L| \le k$ for G. Then, obviously $L' := L|_{E'}$ is a proper labeling for G' with $|W_{L'}| \le |W_L| \le k$.

($\Leftarrow$) Conversely, let $L' = (S^1_{L'}, S^2_{L'}, \dots, S^c_{L'}, W_{L'})$ be a proper labeling with $|W_{L'}| \le k$ for G'. Let $E \setminus E' = \{e_1, e_2, \dots, e_p\}$. We define $p+1$ graphs $G_0, G_1, G_2, \dots, G_p$ by $G_0 := (V, E')$, and $G_i := (V, E' \cup \{e_1, \dots, e_i\})$ for $i \in \{1, \dots, p\}$. Note that $G_p = G$, $\deg_{G_i}(v) \le \deg_G(v)$, and $N_{G_i}(v) \subseteq N_G(v)$ for every $i \in \{0, 1, \dots, p\}$ and $v \in V$. We prove by induction over i that all G_i have a proper labeling with at most k weak edges.

Base Case: $i = 0$. Then, since G_0 and G' have the exact same edges, L' is a proper labeling for G_0 with at most k weak edges.

Inductive Step: $0 < i \le p$. Then, by the inductive hypothesis, there exists a proper labeling L_{i-1} for $G_{i-1} = (V, E' \cup \{e_1, \dots, e_{i-1}\})$ with at most k weak edges. From $E' = E(\mathscr{C} \cup N(\mathscr{C}))$ we conclude $e_i \in E \setminus E(\mathscr{C} \cup N(\mathscr{C})) = E(\mathscr{P}) \setminus E(N(\mathscr{C}))$. Thus, we have $N_G[u] \subseteq \mathscr{P}$ for at least one of the endpoints u of e. Therefore, $\deg_G(w) \le c-1$ for all $w \in N_G[u]$. Together with the facts that $\deg_{G_i}(w) \le \deg_G(w)$ and $N_{G_i}(w) \subseteq N_G(w)$ we conclude $\deg_{G_i}(w) \le c-1$ for all $w \in N_{G_i}[u]$. Then, by Lemma 5.6, there exists a proper labeling L_i for G_i such that $|W_{L_i}| = |W_{L_{i-1}}| \le k$. $\square$

It remains to show that after applying Reduction Rule 5.1 the input graph consists of $\mathcal{O}(\xi_{c-1} \cdot c)$ vertices.

Theorem 5.8. ECS *admits a problem kernel with at most $4\xi_{c-1} \cdot c$ vertices and $\mathcal{O}(\xi_{c-1} \cdot c^2)$ edges that can be computed in $\mathcal{O}(n+m)$ time.*

Proof. Let (G, c, k) be an instance of ECS. We apply Rule 5.1 on (G, c, k) as follows: First, we compute a 2-approximation D'_{c-1} of the smallest possible edge-deletion set D_{c-1} in $\mathcal{O}(n+m)$ time using the algorithm behind Proposition 5.3. Let $\mathscr{C} := \mathscr{C}(D'_{c-1})$.We then remove all vertices in $V \setminus (\mathscr{C} \cup N_G(\mathscr{C}))$ from G which can also be done in $\mathcal{O}(n+m)$ time. Hence, applying Rule 5.1 can be done in $\mathcal{O}(n+m)$ time.

We next show that after this application of Rule 5.1 the graph consists of at most $4\xi_{c-1} \cdot c$ vertices and $\mathcal{O}(\xi_{c-1} \cdot c^2)$ edges. Since D'_{c-1} is a 2-approximation of the smallest possible edge-deletion set we have $|\mathscr{C}| \leq 4\xi_{c-1}$. Since every vertex in $\mathscr{C}$ has at most $c-1$ neighbors in $V \setminus \mathscr{C}$, we conclude $|\mathscr{C} \cup N(\mathscr{C})| \leq 4\xi_{c-1} \cdot c$. In $E(\mathscr{C} \cup N(\mathscr{C}))$ there are obviously the at most $2\xi_{c-1}$ edges of D'_{c-1}. Moreover, each of the at most $4\xi_{c-1} \cdot c$ vertices has at most $c-1$ further incident edges. Hence, after applying Rule 5.1, the reduced instance has $\mathcal{O}(\xi_{c-1} \cdot c^2)$ edges. □

If we consider Edge Coloring instead of ECS, we can immediately reject if one vertex has degree more than c. Then, since there are at most $|\mathscr{C}| \leq 2\xi_{c-1}$ vertices that have a degree of at least c, Theorem 5.8 implies the following.

Corollary 5.9. *Let h_c be the number of vertices with degree c.* Edge Coloring *admits a problem kernel with $\mathcal{O}(h_c \cdot c)$ vertices and $\mathcal{O}(h_c \cdot c^2)$ edges that can be computed in $\mathcal{O}(n+m)$ time.*

5.1.2 ECS Kernel for $\lambda_c + c$

We now present a problem kernel for ECS parameterized by the number of strong colors c and the component order connectivity λ_c. We prove that ECS admits a problem kernel with $\mathcal{O}(c^3 \cdot \lambda_c)$ vertices, which is a linear vertex kernel for every fixed value of c. Our kernelization is based on the Expansion Lemma [154], a generalization of the Crown Rule [31]. We use the formulation given by Cygan et al. [38].

Lemma 5.10 (Expansion Lemma)**.** *Let q be a positive integer and let G be a bipartite graph with partite sets A and B such that $|B| \geq q|A|$ and there are no isolated vertices in B. Then there exist nonempty vertex sets $X \subseteq A$ and $Y \subseteq B$ with $N(Y) \subseteq X$. Moreover, there exist edges $M \subseteq E(X, Y)$ such that*

a) every vertex of X is incident with exactly q edges of M, and

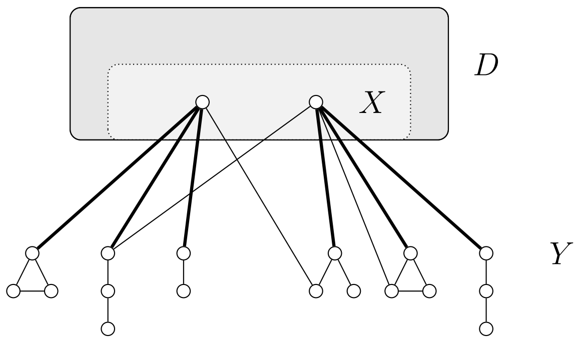


Figure 5.2: A visualization of Proposition 5.11. The upper part shows an order-3 component cover D and the subset $X \subseteq D$. The lower part shows the vertices in $Y \subseteq I$. The thick edges correspond to the set $M \subseteq E(X, Y)$.

b) $q \cdot |X|$ vertices in Y are endpoints of edges in M.

The sets X and Y can be found in $(n + q)^{\mathcal{O}(1)}$ time.

Before we apply Lemma 5.10 we consider the following simple reduction rule.

Reduction Rule 5.2. *If there exists a connected component J of size at most c in G, then remove all vertices in J from G.*

Rule 5.2 is safe since $|J| \leq c$ and therefore the graph $G[J]$ has maximum degree $c-1$ and can be labeled by Vizing's theorem with c colors. For the rest of this section we assume that (G, c, k) is reduced regarding Rule 5.2. The following proposition is a direct consequence of Lemma 5.10. A visualization of the proposition is shown in Figure 5.2.

Proposition 5.11. *Let $(G = (V, E), c, k)$ be an instance of* ECS *that is reduced regarding Rule 5.2, let D be an order-c component cover of G, and let $I := V \setminus D$. If $|I| \geq c^2 \cdot |D|$, then there exist nonempty sets $X \subseteq D$ and $Y \subseteq I$ with $N(Y) \subseteq X \cup Y$. Moreover, there exists a set $M \subseteq E(X, Y)$ such that*

a) every vertex of X is incident with exactly c edges of M, and

b) $c \cdot |X|$ vertices in Y are endpoints of edges in M and every connected component in $G[Y]$ contains at most one such vertex.

The sets X and Y can be computed in polynomial time.

Proof. We prove the proposition by applying Lemma 5.10. To this end, we define an equivalence relation $\sim$ on the vertices of I: Two vertices $v, u \in I$ are equivalent, denoted $u \sim v$, if and only if u and v belong to the same connected component in $G[I]$. Obviously, $\sim$ is an equivalence relation. For a given vertex $u \in I$, let $[u] := \{v \in I \mid v \sim u\}$ denote the equivalence class of u. Note that $|[u]| \leq c$ since D is an order-c component cover.

We next define an auxiliary graph G_{aux}, on which we will apply Lemma 5.10. Intuitively, we obtain G_{aux} from G by deleting all edges in $E_G(D)$ and merging the at most c vertices in every equivalence class in I. Formally, $G_{\text{aux}} := (D \cup I^*, E_{\text{aux}})$ with $I^* := \{[u] \mid u \in I\}$ and

$$E_{\text{aux}} := \{\{[u], v\} \mid [u] \in I^*, v \in \bigcup_{w \in [u]} (N_G(w) \setminus I)\}.$$

Note that G_{aux} can be computed from G in polynomial time and that $|I| \geq |I^*| \geq \frac{1}{c}|I|$.

Observe that G_{aux} is bipartite with partite sets D and I^*. Since G is reduced regarding Rule 5.2, every $[u] \in I^*$ is adjacent to some $v \in D$ in G_{aux}. Consequently, G_{aux} is a bipartite graph without isolated vertices in I^*. Moreover, since $|I| \geq c^2|D|$ and $|I^*| \geq \frac{1}{c}|I|$ we have $|I^*| \geq c \cdot |D|$. By applying Lemma 5.10 on G_{aux} and $c \in \mathbb{N}$, there exist nonempty vertex sets $X' \subseteq D$ and $Y' \subseteq I^*$ with $N_{G_{\text{aux}}}(Y') \subseteq X'$ that can be computed in polynomial time such that there exists a set $M' \subseteq E_{G_{\text{aux}}}(X', Y')$ of edges, such that every vertex of X' is incident with exactly c edges of M', and $c \cdot |X'|$ vertices in Y' are endpoints of edges in M'.

We now describe how to construct the sets X, Y, and M from X', Y', and M'. We set $X := X' \subseteq D$ and $Y := \bigcup_{[u] \in Y'} [u] \subseteq I$. We prove that $N_G(Y) \subseteq X \cup Y$. Let $y \in Y$. Note that all neighbors of y in I are elements of Y by the definition of the equivalence relation $\sim$ and therefore

$$N_G(y) \subseteq N_{G_{\text{aux}}}([y]) \cup Y \subseteq X' \cup Y = X \cup Y.$$

Next, we construct $M \subseteq E_G(X, Y)$ from M'. To this end we define a mapping $\pi : M' \to E_G(X, Y)$. For every edge $\{[u], v\} \in M'$ with $[u] \in Y'$ and $v \in X'$ we define $\pi(\{[u], v\}) := \{w, v\}$, where w is some fixed vertex in $[u]$. We set $M := \{\pi(e') \mid e' \in M'\}$. It remains to show that Properties *a*) and *b*) hold for M.

a) Observe that $\pi(\{[u_1], v_1\}) = \pi(\{[u_2], v_2\})$ implies $[u_1] = [u_2]$ and $v_1 = v_2$. Thus, π is injective. We conclude $|M| = |M'|$. Moreover, observe that the edges of M have the same endpoints in X as the edges of M'. Thus, since every vertex of X' is incident with exactly c edges of M', the Property *a*) holds for M.

b) By the Properties *a*) and *b*) of the Expansion Lemma, no two edges in M' have a common endpoint in Y'. Hence, in every connected component in $G[Y]$ there is at most one vertex incident with an edge in M. Moreover, since $|M| = |M'|$ and there are exactly $c \cdot |X'|$ vertices in Y' that are endpoints of edges in M', the Property *b*) holds for M. □

The following rule is the key rule for our kernelization.

Reduction Rule 5.3. *If $|I| \geq c^2 \cdot |D|$, then compute the sets X and Y from Proposition 5.11, delete all vertices in $X \cup Y$ from G, and decrease k by $|E_G(X, V)| - c \cdot |X|$.*

Proposition 5.12. *Reduction Rule 5.3 is safe.*

Proof. Let $G' = (V', E') := G - (X \cup Y)$ be the graph after applying Rule 5.3. Then, $V' = V \setminus (X \cup Y)$ and $E' = E \setminus (E_G(X \cup Y, V))$. Moreover, let $k' := k - |E_G(X, V)| + c \cdot |X|$. We show that there exists a proper labeling L with $|W_L| \leq k$ for G if and only if there is a proper labeling L' with $|W_{L'}| \leq k'$ for G'.

($\Rightarrow$) Let L be a proper labeling for G with $|W_L| \leq k$. We define a labeling L' for G' by $L' := L|_{E'}$. Obviously, no vertex in V is incident with two edges of the same strong color under L, and therefore no vertex in $V' \subseteq V$ is incident with two edges of the same strong color under L'. Hence, L' is a proper labeling for G'. It remains to show that $|W_{L'}| \leq k'$. Obviously, every vertex $x \in X$ is incident with at most c edges of distinct strong colors under L, since L is a proper labeling. Hence, the maximum number of strong edges in $E_G(X, V)$ is $c \cdot |X|$. Thus, we have $|W_L \cap E_G(X, V)| \geq |E_G(X, V)| - c \cdot |X|$. Therefore,

$$\begin{aligned} |W_{L'}| &= |W_L \cap E'| = |W_L| - |W_L \cap E_G(X \cup Y, V)| \\ &\leq |W_L| - |W_L \cap E_G(X, V)| \leq k - |E_G(X, V)| + c \cdot |X| = k'. \end{aligned}$$

($\Leftarrow$) Conversely, let L' be a proper labeling for G' with $|W_{L'}| \leq k'$. We now describe how to construct a labeling L for G with $|W_L| \leq k$ from L'. We set $W_L := W_{L'} \cup (E_G(X, V) \setminus M)$, where $M \subseteq E_G(X, Y)$ is the set of edges satisfying Properties *a*) and *b*) from Proposition 5.11. This implies

$$|W_L| = |W_{L'}| + |E_G(X, V)| - |M| \leq k' + |E_G(X, V)| - c|X| = k.$$

Next, we describe to which strong color classes of L we add the remaining edges of G. Since $N_G(Y) \subseteq X \cup Y$ it remains to label all edges in $E' \setminus W_{L'} \cup E_G(Y) \cup M$.

First, consider the edges in $E' \setminus W_{L'}$. Every edge $e \in E' \setminus W_{L'}$ has a strong color i under L'. We then add e to S^i_L. Note that this implies $L|_{E'} = L'|_{E'}$.

Second, consider the edges in M. For each $x \in X$ we define a set $B_x^M := \{e \in M \mid x \in e\} \subseteq M$. By Proposition 5.11 *a*), every $x \in X$ is incident with exactly c edges in M. Thus, $|B_x^M| = c$ and we let $e_x^1, e_x^2, \ldots, e_x^c$ denote the elements of B_x^M. For every $x \in X$ we add e_x^i to the strong color class S_L^i. Note that by Proposition 5.11, there are $c|X|$ vertices in Y that are incident with edges in M. Hence, the family $\{B_x^M \subseteq M \mid x \in X\}$ forms a partition of M and therefore every edge in M belongs to exactly one strong color class of L.

Finally, consider the edges in $E_G(Y)$. Let $J \subseteq Y$ be a connected component in $G[Y]$. Note that $N_G(J) \subseteq J \cup X$ and observe that by Proposition 5.11 *b*) there is at most one vertex $v \in J$ that is an endpoint of some edge in M. Hence, there is at most one edge in $E_G(J, X)$ that belongs to some strong color class S_L^i. Since D is an order-c component cover, we know that $|J| \leq c$, and therefore $\Delta_{G[J]} \leq c - 1$. Consequently, there exists a proper labeling $L'' = (S_{L''}^1, \ldots, S_{L''}^c, W_{L''})$ for $G[J]$ due to Vizing's Theorem. Without loss of generality we can assume that v is not incident with an edge in $S_{L''}^i$: Since $\deg_{G[J]}(v) \leq c-1$, there exists one strong color class $S_{L''}^j$ that contains no edge incident with v and we can interchange the edges in $S_{L''}^i$ and $S_{L''}^j$. Then, for every $t \in \{1, \ldots, c\}$ we add all edges in $S_{L''}^t$ to the strong color class S_L^t.

It remains to show that L is a proper labeling. To this end, we show for every vertex $v \in V$, that v is not incident with two edges of the same strong color under L. Consider the following case distinction.

Case 1: $v \in V \setminus (X \cup Y)$**.** Then, since $E_G(\{v\}, Y) = \emptyset$, and $E_G(\{v\}, X) \subseteq W_L$, every strong edge incident with v has the same strong color under L as it has under L'. Since L' is a proper labeling, the vertex $v \in V \setminus (X \cup Y)$ is not incident with two edges of the same strong color under L.

Case 2: $v \in X$**.** Then, since $E_G(\{v\}, V \setminus Y) \subseteq W_L$, and $E_G(\{v\}, Y) \setminus B_v^M \subseteq W_L$, all strong edges incident with v are elements of B_v^M. Since $B_v^M = \{e_v^1, \ldots, e_v^c\}$, and every $e_v^i \in S_L^i$, the vertex v is not incident with two edges of the same strong color under L.

Case 3: $v \in Y$**.** Let $J \subseteq Y$ be the connected component in $G[Y]$ that contains v. Note that $N_G(v) \subseteq X \cup J$. First, consider the case, that v has no strong neighbors in X. Then, there are no ECS violations since $L|_{E_G(J)}$ is a proper labeling for $G[J]$ by construction. Second, consider the case that v has strong neighbors in X. Then, by Proposition 5.11, there is exactly one edge in $E_G(\{v\}, X)$ that belongs to M and therefore it is in some strong color class S_L^i. By the construction of L, all edges in $E_G(\{v\}, J)$ have pairwise distinct strong colors which are all distinct from i under L. Therefore, the vertex v is not incident with two edges of the same strong color under L. □

Rules 5.2 and 5.3 together with the fact that we can compute a $(c+1)$-approximation of the minimum order-c component cover in polynomial time [122] give us the following.

Theorem 5.13. ECS *admits a problem kernel with* $\mathcal{O}(c^3 \cdot \lambda_c)$ *vertices.*

Proof. We first describe how to compute the kernel. We compute a $(c+1)$-approximation D of the minimum oder-c component cover in polynomial time [122]. Afterwards, we apply Rules 5.2 and 5.3. Obviously, one application of Rule 5.2 can be done in polynomial time if D is known. Moreover, Rule 5.3 can also be applied in polynomial time due to Proposition 5.11. Since every application of one of these two rules removes some vertices, we can compute an instance that is reduced regarding Rules 5.2 and 5.3 from an arbitrary input instance of ECS in polynomial time.

We next consider the number of vertices in a reduced instance $(G = (V, E), c, k)$ of ECS regarding Rules 5.2 and 5.3. Recall that $D \subseteq V$ is a $(c+1)$-approximation of the minimum order-c component cover. Let $I := V \setminus D$. Since no further application of Rule 5.3 is possible, we have $|I| < c^2 \cdot |D|$. Therefore, we have $|V| = |I| + |D| < (c^2 + 1) \cdot |D| \leq (c^2 + 1) \cdot (c + 1) \cdot \lambda_c \in \mathcal{O}(c^3\lambda_c)$. □

Recall that the vertex deletion distance to component size c is a distance from triviality parameterization since connected components with size at most c can be colored with c edge colors due to Vizing's theorem. In this context, a further natural parameterization is the vertex deletion distance to maximum degree $c - 1$. Let d_{c-1} denote this parameter. Observe that d_{c-1} is never larger than λ_c. Let D be a minimum edge deletion set to maximum degree $c - 1$. Adapting the approach by the expansion lemma described above may result in a reduction rule producing an instance (G, c, k), where the number of connected components in $G - D$ is polynomially bounded in d_{c-1}. However, these components may be arbitrarily large. Thus, to find a polynomial kernel for ECS parameterized by $d_{c-1} + c$, a good starting point might be to investigate if there is a reduction rule that decreases the size of large connected components in $G - D$.

5.2 Kernelization for Multi-STC

We now provide a problem kernelization for MULTI-STC parameterized by $\xi_{c-1} + c$ for instances with $c \in \{1, 2, 3, 4\}$. To obtain the result for $c \in \{3, 4\}$ we provide a more general problem kernelization for all $c \geq 3$ by considering parameterization by $\xi_{\lfloor \frac{c}{2} \rfloor + 1} + c$. Before we describe the problem kernel, we briefly discuss parameterization by $\lambda_c + c$. STC does not admit a polynomial kernel if parameterized by the

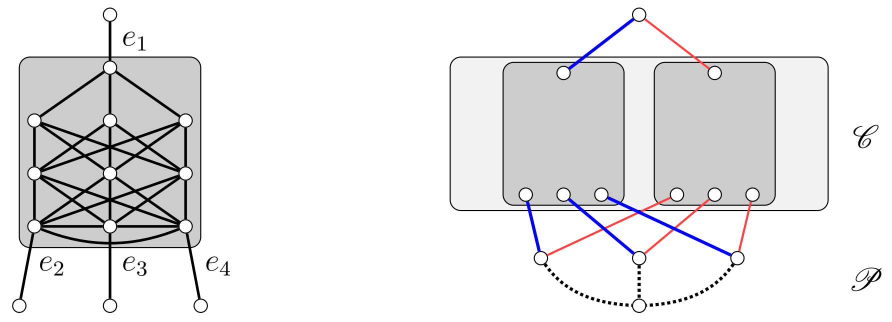


Figure 5.3: Left: A graph where in any STC-labeling with four strong colors and without weak edges, the edges e_1, e_2, e_3, and e_4 are part of the same strong color class. Right: A no-instance of Multi-STC with $c = 4$ and $k = 0$, where Rule 5.1 does not produce an equivalent instance: The inner rectangles correspond to two copies of the graph on the left. Observe that all blue edges must have a common strong color, and all red edges must have a common strong color distinct from the color of the blue edges. Hence, for any STC-labeling of $G[\mathscr{C} \cup N(\mathscr{C})]$ it is not possible to extend the labeling to the dotted edges without violating STC or using weak edges. However, Rule 5.1 converts this no-instance into a yes-instance.

number of strong edges ℓ unless NP $\subseteq$ coNP/poly due to Theorem 2.11. In nontrivial instances, the number of strong edges is bigger than the size of a maximal matching M. Since the vertex cover number is never larger than $2|M|$, this implies that Multi-STC has no polynomial kernel if parameterized by the vertex cover number unless NP $\subseteq$ coNP/poly. Since λ_c is never larger than the vertex cover number, we obtain the following.

Proposition 5.14. Multi-STC *parameterized by λ_c does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly, *even if* $c = 1$.

Next, consider parameterization by ξ_{c-1}. Observe that Rule 5.1 which gives a problem kernel for ECS does not work for Multi-STC; see Figure 5.3 for an example: Intuitively, a gadget containing triangles on core vertices propagates conflicts into the edges of the periphery. For Multi-STC we need a fundamentally new approach: For STC-labelings the maximum degree and the number of colors are not as closely related as in ECS, since—for example—all edges inside an arbitrarily large clique may be labeled with the same strong color. Therefore, Lemma 5.5 might not be helpful for Multi-STC. Moreover, in the proof of Lemma 5.6 we exploit that in ECS we may remove weak edges from the instance. This is not safe for Multi-STC since removing a weak edge may produce P_3s. However, as described above, the results for ECS parameterized by $\xi_{c-1} + c$ can be lifted to the seemingly harder MULTI-STC for $c \in \{1, 2, 3, 4\}$. We will first discuss the cases $c = 1$ and $c = 2$. For the cases $c \in \{3, 4\}$ we show the more general statement that MULTI-STC admits a problem kernel with $\mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor + 1} \cdot c)$ vertices. Recall that, given an edge deletion set D_{c-1}, the *core* $\mathscr{C}$ is the set of vertices incident with an edge in D_{c-1}, and the *periphery* $\mathscr{P}$ is the set of all vertices that are not in $\mathscr{C}$.

If $c = 1$, the parameter $\xi_{c-1} = \xi_0$ equals the number m of edges in G. Hence, MULTI-STC admits a trivial ξ_{c-1}-edge kernel in this case. If $c = 2$, any input graph consists of core vertices $\mathscr{C}$, periphery vertices in $N(\mathscr{C})$ and isolated vertices and edges. We can compute an equivalent instance in linear time by deleting these isolated components. The safeness of this rule is obvious. Afterwards, the graph contains at most $2\xi_{c-1}$ core vertices. Since each of these vertices has at most one neighbor outside $\mathscr{C}$, we have a total number of $4\xi_{c-1}$ vertices.

To extend this result to $c \in \{3, 4\}$, we now provide a problem kernel for MULTI-STC parameterized by $\xi_{\lfloor \frac{c}{2} \rfloor + 1} + c$. Throughout this section, let (G, c, k) be an instance of MULTI-STC with edge-deletion set $D := D_{\lfloor \frac{c}{2} \rfloor + 1}$, and let $\mathscr{C}$ and $\mathscr{P}$ be the core and periphery of G with regard to D. Roughly speaking, we obtain our kernelization by removing specific vertices from $\mathscr{P}$.

This section is organized as follows: In Section 5.2.1, we introduce the term *good periphery component* to describe vertex sets in $\mathscr{P}$ from which we can safely remove specific vertices. In Section 5.2.2, we consider the case where the number of strong colors c is odd, and provide a simple reduction rule leading to a problem kernel for this case. Finally, in Section 5.2.3 we provide a problem kernelization for even values of c.

5.2.1 Good Periphery Components

To be more precise when addressing vertex sets in $\mathscr{P}$, we introduce the following terms.

Definition 5.15. *A subset $A \subseteq \mathscr{P}$ is called* periphery component *if it is a connected component in $G[\mathscr{P}]$. Furthermore, for a periphery component $A \subseteq \mathscr{P}$ we define the subset $A^* \subseteq A$ of* close vertices *in A as $A^* := N(\mathscr{C}) \cap A$, that is, the set of vertices of A that are adjacent to core vertices.*

Figure 5.4 shows an example of periphery components in the case where $c = 4$. To obtain our problem kernel we describe rules to remove non-close vertices from some periphery components. With the next definition and the following proposition we identify a property of periphery components that makes it safe to remove non-close vertices.

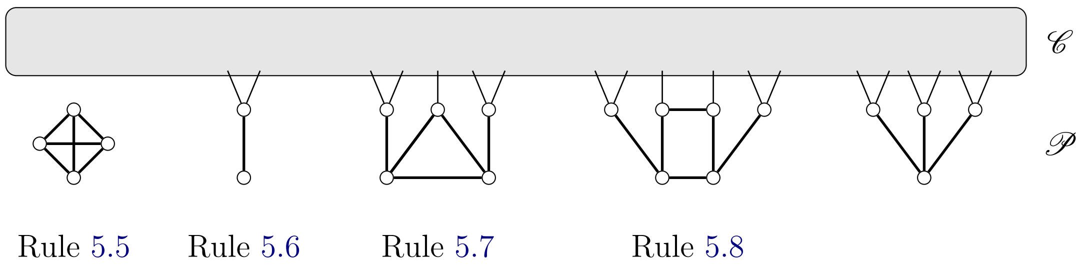


Figure 5.4: Five examples of periphery components for $c = 4$. The upper part corresponds to the set of core vertices. The thick edges correspond to the edges within the periphery components and the thin edges correspond to the edges between the core vertices and the periphery vertices. In our kernelization, the non-close vertices of the first four periphery components from the left are removed by the corresponding rules. The rightmost periphery component is not affected by our reduction rules. We will see that the number of non-close vertices in such periphery components is bounded by our parameter.

Definition 5.16. *Let (G, c, k) be an instance of* Multi-STC *with core vertices $\mathscr{C}$ and periphery vertices $\mathscr{P}$. A periphery component $A \subseteq \mathscr{P}$ is called* good*, if for every STC-labeling $L = (S_L^1, \dots, S_L^c, W_L)$ for G with $E(A) \cap W_L \neq \emptyset$ there exists an STC-labeling $L' = (S_{L'}^1, \dots, S_{L'}^c, W_{L'})$ for G such that*

1. *$L'|_{E \setminus E(A)} = L|_{E \setminus E(A)}$, and*
2. *$W_{L'} \cap E(A) = \emptyset$.*

Intuitively, a good periphery component A is a periphery component where the edges in $E(A)$ can always be added to some strong color classes of an STC-labeling, no matter how the other edges of G are labeled.

Proposition 5.17. *Let (G, c, k) be an instance of* Multi-STC *with core vertices $\mathscr{C}$ and periphery vertices $\mathscr{P}$. Furthermore, let $A \subseteq \mathscr{P}$ with $|A| \geq 2$ be a good periphery component. Then, (G, c, k) is a yes-instance if and only if $(G - (A \setminus A^*), c, k)$ is a yes-instance.*

Proof. Let $\widetilde{G} = (\widetilde{V}, \widetilde{E}) := (G - (A \setminus A^*), c, k)$. We show that G has a c-colored STC-labeling with at most k weak edges if and only if $\widetilde{G}$ has a c-colored STC-labeling with at most k weak edges.

($\Rightarrow$) Let $L = (S_L^1, \dots, S_L^c, W_L)$ be a c-colored STC-labeling for G such that $|W_L| \leq k$. Then, we define by $\widetilde{L} := L|_{\widetilde{E}}$ a c-colored labeling for $\widetilde{G}$. Obviously, $|W_L \cap \widetilde{E}| \leq k$.

Moreover, since $\widetilde{G}$ is an induced subgraph of G, $\widetilde{L}$ satisfies STC since L is an STC-labeling.

($\Leftarrow$) Conversely, let $\widetilde{L} = (S^1_{\widetilde{L}}, \dots, S^c_{\widetilde{L}}, W_{\widetilde{L}})$ be a c-colored STC-labeling for $\widetilde{G}$ such that $|W_{\widetilde{L}}| \leq k$. We define a c-colored labeling $L := (S^1_L, \dots, S^c_L, W_L)$ for G by $S^j_L := S^j_{\widetilde{L}} \setminus E(A)$ and $W_L := W_{\widetilde{L}} \cup E(A)$. Note that $L|_{E \setminus E(A)} = \widetilde{L}|_{E \setminus E(A)}$ and $E(A) \cap W_L \neq \emptyset$. Then, by the definition of good periphery components there exists an STC-labeling $L' = (S^1_{L'}, \dots, S^c_{L'}, W_{L'})$ for G such that $L'|_{E \setminus E(A)} = L|_{E \setminus E(A)} = \widetilde{L}|_{E \setminus E(A)}$, and $W_{L'} \cap E(A) = \emptyset$. We then have $W_{L'} \subseteq W_{\widetilde{L}}$ and thus, $|W_{L'}| \leq k$. Consequently, L' is an STC-labeling for G with at most k weak edges. □

5.2.2 Problem Kernelization when c is Odd

In the following, we show that for instances (G, c, k) where $c \geq 3$ is odd, we can compute an equivalent instance with $\mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor + 1} c)$ vertices. In this case, all periphery components are good as stated in the following proposition.

Proposition 5.18. *Let (G, c, k) be an instance of* Multi-STC*, where $c \geq 3$ is odd. Moreover, let $A \subseteq \mathscr{P}$ be a periphery component. Then, A is good.*

Proof. Let L be an arbitrary STC-labeling for G with $E(A) \subseteq W_L$. We prove that there is an STC-labeling which is partially equal to L with respect to $E \setminus E(A)$ and has no weak edges in $E(A)$.

Let L' be an STC-labeling for G with $L'|_{E \setminus E(A)} = L|_{E \setminus E(A)}$. If $W_{L'} \cap E(A) = \emptyset$, nothing more needs to be shown. Thus, suppose $E(A)$ contains a weak edge $\{u, v\}$. Then, $\deg(u) \leq \lfloor \frac{c}{2} \rfloor + 1$ and $\deg(v) \leq \lfloor \frac{c}{2} \rfloor + 1$ since $u, v \in A$. Since c is odd, the edge $\{u, v\}$ is incident with at most $2 \cdot \lfloor \frac{c}{2} \rfloor < c$ edges in G. Consequently, there exists a strong color $i \in \{1, \dots, c\}$, such that $\{u, v\}$ can be moved from $S^i_{L'}$ to $W_{L'}$ without producing any STC violations. This way, we transformed L' into an STC-labeling L'' such that $L''|_{E \setminus \{\{u,v\}\}} = L|_{E \setminus \{\{u,v\}\}}$ and $|W_{L''}| = |W_{L'}| - 1$. Applying this transformation subsequently for every weak edge in $E(A)$ results in an STC-labeling that has only strong edges in $E(A)$. Since L was arbitrary, the periphery component A is good. □

The Propositions 5.17 and 5.18 guarantee the safeness of the following rule.

Reduction Rule 5.4. *If c is odd, then remove $A \setminus A^*$ from all periphery components $A \subseteq \mathscr{P}$.*

Proposition 5.19. *Let $(G = (V, E), c, k)$ be an instance of* Multi-STC *where $c \geq 3$ is odd. Then, we can compute an instance $(G' = (V', E'), c, k)$ in $\mathcal{O}(n + m)$ time such that $|V'| \leq 4 \cdot \xi_{\lfloor \frac{c}{2} \rfloor + 1} \cdot (\lfloor \frac{c}{2} \rfloor + 1)$, and $|E'| \in \mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor + 1} \cdot c^2)$.*

Proof. We compute (G', c, k) as follows: First, we compute a 2-approximation $D' := D'_{\lfloor \frac{c}{2} \rfloor + 1}$ of the minimum edge-deletion set $D_{\lfloor \frac{c}{2} \rfloor + 1}$ in $\mathcal{O}(n+m)$ time using the algorithm behind Proposition 5.3. Let $\mathscr{C} := \mathscr{C}(D')$ and $\mathscr{P} := \mathscr{P}(D')$. We compute $(G' = (V', E'), c, k)$ from G by applying Rule 5.4 exhaustively. This can be done by computing $G[\mathscr{C} \cup N(\mathscr{C})]$ in $\mathcal{O}(n+m)$ time.

We next analyze the size of $\mathscr{C} \cup N(\mathscr{C})$. Since $|\mathscr{C}| \leq 4\xi_{\lfloor \frac{c}{2} \rfloor + 1}$, and every $v \in \mathscr{C}$ has at most $\lfloor \frac{c}{2} \rfloor + 1$ neighbors in $\mathscr{P}$, there are at most $4 \cdot \xi_{\lfloor \frac{c}{2} \rfloor + 1} \cdot (\lfloor \frac{c}{2} \rfloor + 1)$ vertices in V'. Since $|D'| \leq 2\xi_{\lfloor \frac{c}{2} \rfloor + 1}$ and each vertex is incident with at most $\lfloor \frac{c}{2} \rfloor + 1$ further edges, we have $|E'| \in \mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor + 1} \cdot c^2)$. □

5.2.3 Problem Kernelization when c is Even

It remains to consider instances where c is an even number and $c \geq 4$. In this case, not every periphery component is good (Figure 5.3 shows an example), so we need to identify good periphery components more carefully. In summary, we identify the following periphery components as good: isolated periphery components, periphery components with low-degree vertices, periphery components that contain a triangle, and periphery components that contain cycles with at least four vertices. Figure 5.4 gives an overview of these periphery components and the corresponding rules that remove the non-close vertices. We first introduce a rule that removes isolated periphery components.

Reduction Rule 5.5. *Remove periphery components $A \subseteq \mathscr{P}$ with $A^* = \emptyset$ from G.*

Rule 5.5 is safe since due to Vizing's theorem, there is an STC-labeling without weak edges for $G[A]$ for every isolated periphery component A. Thus, every isolated periphery component is good.

The proof of Vizing's theorem as presented in the book by Behzad and Chartrand [11, Section 15.2] relies on switching colors of so-called 'maximal alternating paths'. A *maximal alternating paths* is an edge-simple path P on which two strong colors q_1 and q_2 appear in an alternating manner such that P is maximal under this property. It is easy to see that—in case of MULTI-STC—switching the strong colors on a maximal alternating path might convert an STC-labeling into a labeling that is not an STC-labeling. However, with the next lemma, we provide a related technique of changing colors on paths in context of MULTI-STC. Intuitively, the small degree of vertices in periphery components can be used to 'move' weak edges inside periphery components. More precisely, if there is an edge-simple path in a periphery component that starts with a weak edge, we can either move the weak edge to the end of that path by keeping the same number of weak edges or find a labeling with fewer weak edges. We refer to this lemma as *moving lemma.* Recall that, given a path $P = (P(0), \dots, P(r-1))$, the *color sequence* Q_L^P of P under L as the finite sequence $Q_L^P = (q_0, q_1, \dots, q_{r-2})$ of elements in $\{0, \dots, c\}$, such that $\{P(i), P(i+1)\} \in S_L^{q_i}$ if $q_i \geq 1$ and $\{P(i), P(i+1)\} \in W_L$ if $q_i = 0$.

Lemma 5.20 (Moving Lemma)**.** *Let $A \subseteq \mathscr{P}$ be a periphery component, let L be an STC-labeling of G, and let $e := \{v_1, v_2\} \in W_L \cap E(A)$ be a weak edge in $E(A)$. Furthermore, let $P = (v_0, v_1, \dots, v_{r-1})$ be an edge-simple path in $G[A]$ with color sequence $Q_L^P = (q_0 = 0, q_1, q_2, \dots, q_{r-2})$ under L. Then, there exists an STC-labeling L' with $L'|_{E \setminus E(P)} = L|_{E \setminus E(P)}$ such that*

$$Q_{L'}^P = (q_1, q_2, \dots, q_{r-2}, 0) \text{ or } |W_{L'}| < |W_L|.$$

Proof. We prove the statement by induction over the length r of P.

Base Case: $r = 2$. Then, $P = (v_0, v_1)$ and $Q_L^P = (0)$. We can trivially define the labeling L' by setting $L' := L$.

Inductive Step: Let $P = (v_0, \dots, v_{r-1})$ be an edge-simple path with color sequence $Q_L^P = (0, q_1, \dots, q_{r-2})$ under L. Consider the edge-simple subpath $P' = (v_0, \dots, v_{r-2})$. By induction hypothesis, there exists an STC-labeling L'' for G with $L''|_{E \setminus E(P')} = L|_{E \setminus E(P')}$, such that $Q_{L''}^{P'} = (q_1, q_2, \dots, q_{r-3}, 0)$ or $|W_{L''}| < |W_L|$.

Case 1: $|W_{L''}| < |W_L|$**.** Then, we define L' by $L' := L''$.

Case 2: $|W_{L''}| \geq |W_L|$**.** Then, $Q_{L''}^{P'} = (q_1, q_2, \dots, q_{r-3}, 0)$. Since $Q_{L''}^{P'}$ contains the same elements as $Q_L^{P'}$ and $L''|_{E \setminus E(P')} = L|_{E \setminus E(P')}$, we have $|W_{L''}| = |W_L|$.

Case 2.1: There exists an edge $e \neq \{v_{r-2}, v_{r-1}\}$ with $e \in S_{L''}^{q_{r-2}}$ that is incident with $\{v_{r-3}, v_{r-2}\}$**.** Since $\deg(v_{r-3}) \leq \lfloor \frac{c}{2} \rfloor + 1$ and $\deg(v_{r-2}) \leq \lfloor \frac{c}{2} \rfloor + 1$, the edge $\{v_{r-3}, v_{r-2}\}$ is incident with at most c other edges of G. Since two of these incident edges have the same strong color q_{r-2} under L'', the edge $\{v_{r-3}, v_{r-2}\}$ is incident with at most $c-1$ edges of distinct strong colors under L''. Consequently, there exists a strong color $i \in \{1, \dots, c\}$ such that $\{v_{r-3}, v_{r-2}\}$ can safely be moved from $W_{L''}$ to the strong color class $S_{L''}^i$ without producing any strong P_3. This way, we transformed L'' into an STC-labeling L', such that $L'|_{E \setminus E(P')} = L|_{E \setminus E(P')}$ and $|W_{L'}| < |W_L|$.

Case 2.2: There is no edge $e \neq \{v_{r-2}, v_{r-1}\}$ with $e \in S_{L''}^{q_{r-2}}$ that is incident with $\{v_{r-3}, v_{r-2}\}$**.** We then define L' by moving $\{v_{r-3}, v_{r-2}\}$ from $W_{L''}$ to $S_{L''}^{q_{r-2}}$ and moving $\{v_{r-2}, v_{r-1}\}$ from $S_{L''}^{q_{r-2}}$ to $W_{L''}$. This way, we transformed L'' into a labeling L' such that $Q_{L'}^P = (q_1, q_2, \dots, q_{r-2}, 0)$ and $L'|_{E \setminus E(P)} = L|_{E \setminus E(P)}$. Moreover, since P is edge-simple, the edge $\{v_{r-2}, v_{r-1}\}$ does not lie on the path P'. Therefore, every edge has exactly one color under L'.

It remains to show that L' satisfies STC. It suffices to show that there is no edge $e \in S_{L'}^{q_{r-2}}$ forming a P_3 with $\{v_{r-3}, v_{r-2}\}$. By the condition of Case 2.2, there is no edge $e \neq \{v_{r-2}, v_{r-1}\}$ with $e \in S_{L''}^{q_{r-2}}$ that is incident with $\{v_{r-3}, v_{r-2}\}$. Then, since $L'|_{E\setminus\{\{v_{r-3},v_{r-2}\},\{v_{r-2},v_{r-1}\}\}} = L''|_{E\setminus\{\{v_{r-3},v_{r-2}\},\{v_{r-2},v_{r-1}\}\}}$, there is no edge $e \in S_{L'}^{q_{r-2}}$ forming a P_3 with $\{v_{r-3}, v_{r-2}\}$. □

We now use the moving lemma (Lemma 5.20) to show useful properties of periphery components. We first show that every periphery component contains at most one weak edge. The intuition behind the proof is the following: If there are two weak edges in one periphery component A, we move one of the edges along a path in A to make the two weak edges incident. This then helps us to define a new labeling that has fewer weak edges in A.

Proposition 5.21. *Let $A \subseteq \mathscr{P}$ be a periphery component and let L be an STC-labeling for G. Then, there exists an STC-labeling L' with $L'|_{E\setminus E(A)} = L|_{E\setminus E(A)}$ and $|W_{L'} \cap E(A)| \leq 1$.*

Proof. If $|W_L \cap E(A)| \leq 1$ the statement already holds for $L' = L$. So, assume there are two distinct edges $e_1, e_2 \in W_L \cap E(A)$. In this case, we construct an STC-labeling which is partially equal to L with respect to $E \setminus E(A)$ and has strictly fewer weak edges in $E(A)$ than L.

Since periphery components are connected components in $G[\mathscr{P}]$, there exists an edge-simple path $P = (v_0, \ldots, v_{r-1})$ in $G[A]$ such that $e_1 = \{v_0, v_1\}$ and $e_2 = \{v_{r-2}, v_{r-1}\}$. Applying the moving lemma (Lemma 5.20) on the edge-simple subpath $P' = (v_0, \ldots, v_{r-2})$ gives us an STC-labeling L' with $L'|_{E\setminus E(P)} = L|_{E\setminus E(P)}$ such that $|W_{L'}| < |W_L|$ or $Q_{L'}^{P'} = (q_1, q_2, \ldots, q_{r-3}, 0)$.

In case of $|W_{L'}| < |W_L|$, nothing more needs to be shown. So, assume $|W_{L'}| = |W_L|$. Then, $Q_{L'}^{P'} = (q_1, q_3, \ldots, q_{r-3}, 0)$ and therefore $Q_{L'}^{P} = (q_1, q_2, \ldots, q_{r-3}, 0, 0)$. Thus, the incident edges $\{v_{r-3}, v_{r-2}\}$ and e_2 are weak under L'. Since $\deg(v_{r-2}) \leq \lfloor \frac{c}{2} \rfloor + 1$ and $\deg(v_{r-1}) \leq \lfloor \frac{c}{2} \rfloor + 1$, the edge e_2 is incident with at most c edges. Since at least one of these incident edges is weak, e_2 is incident with at most $c - 1$ edges of distinct strong colors. Consequently, there exists a strong color color $i \in \{1, \ldots, c\}$ such that e_2 can be moved from $W_{L'}$ to $S_{L'}^i$ without violating STC. This way, we transformed L' into an STC-labeling L'' such that $L''|_{E\setminus E(A)} = L|_{E\setminus E(A)}$ and $|W_{L''} \cap E(A)| < |W_L \cap E(A)|$. Applying this transformation as long as there are at least two weak edges in $E(A)$ results in an STC-labeling that has only one weak edge in $E(A)$. □

Next, we use Proposition 5.21 to identify further good periphery components.

Proposition 5.22. *Let $A \subseteq \mathscr{P}$ be a periphery component where some edge $\{u, v\} \in E(A)$ forms an induced P_3 with at most $c - 1$ other edges in G. Then, A is good.*

Proof. Let L be an arbitrary STC-labeling for G with $E(A) \subseteq W_L$. We prove that there is an STC-labeling which is partially equal to L with respect to $E \setminus E(A)$ and has no weak edges in $E(A)$.

Let L' be an STC-labeling for G with $L'|_{E\setminus E(A)} = L|_{E\setminus E(A)}$. If $W_{L'} \cap E(A) = \emptyset$, nothing more needs to be shown. So, let $W_{L'} \cap E(A) \neq \emptyset$. By Proposition 5.21 we may assume that there is one unique edge $e \in W_{L'} \cap E(A)$. Since A is a connected component in $G[\mathscr{P}]$, there exists an edge-simple path $P = (v_0, \dots, v_{r-1})$ such that $\{v_0, v_1\} = e$, and $\{v_{r-2}, v_{r-1}\} = \{u, v\}$ with $Q^P_{L'} = (0, q_1, \dots, q_{r-2})$. By the moving lemma (Lemma 5.20), there exists an STC-labeling L'' with $L''|_{E\setminus E(A)} = L|_{E\setminus E(A)}$ such that $|W_{L''}| < |W_L|$ or $Q^P_{L''} = (q_1, \dots, q_{r-2}, 0)$. In case of $|W_{L''}| < |W_L|$, nothing more needs to be shown. Otherwise, the edge e is weak under L''. Since e is part of at most $c-1$ induced P_3s in G, there exists one strong color $i \in \{1, \dots, c\}$, such that e can safely be moved from $W_{L''}$ to $S^i_{L''}$ without violating STC. This way, we transform L'' into an STC-labeling L''' with $L'''|_{E\setminus E(A)} = L|_{E\setminus E(A)}$ and $W_{L'''} \cap E(A) = \emptyset$. Since L was arbitrary, the periphery component A is good. □

We use Proposition 5.22 to show that periphery components with low-degree vertices and periphery components that contain a triangle are good. Recall that Figure 5.4 shows examples of all types of periphery components that we consider in this section.

Proposition 5.23. *Let $A \subseteq \mathscr{P}$ be a periphery component such that there exists a vertex $v \in A$ with $\deg_G(v) < \lfloor \frac{c}{2} \rfloor + 1$. Then, A is good.*

Proof. If $|A| = 1$, then A is obviously good, since $E(A) = \emptyset$. Let $|A| \geq 2$. Since A contains at least two vertices and forms a connected component in $G[\mathscr{P}]$ there exists a vertex $u \in A$, such that $\{u, v\} \in E(A)$. Since $\deg_G(v) < \lfloor \frac{c}{2} \rfloor + 1$ and $\deg_G(u) \leq \lfloor \frac{c}{2} \rfloor + 1$, the edge $\{u, v\}$ forms induced P_3s with less than c other edges in G. Consequently, A is good by Proposition 5.22. □

Propositions 5.17 and 5.23 guarantee the safeness of the following rule.

Reduction Rule 5.6. *If there is a periphery component $A \subseteq \mathscr{P}$ with $A \setminus A^* \neq \emptyset$ such that there exists a vertex $v \in A$ with $\deg_G(v) < \lfloor \frac{c}{2} \rfloor + 1$, then delete $A \setminus A^*$ from G.*

Proposition 5.24. *Let $A \subseteq \mathscr{P}$ be a periphery component such that there exists an edge $\{u, v\} \in E(A)$ which is part of a triangle $G[\{u, v, w\}]$ in G. Then, A is good.*

Proof. Since $u, v \in A$, we know $\deg_G(u) \leq \lfloor \frac{c}{2} \rfloor + 1$ and $\deg_G(v) \leq \lfloor \frac{c}{2} \rfloor + 1$. Since u and v are part of a triangle in G, the edge $\{u, v\}$ forms an induced P_3 with less than c other edges in G. Consequently, A is good by Proposition 5.22. □

Propositions 5.17 and 5.24 guarantee the safeness of the following rule.

Reduction Rule 5.7. *If there is a periphery component $A \subseteq \mathscr{P}$ with $A \setminus A^* \neq \emptyset$ such that there exists an edge $\{u, v\} \in A$ which is part of a triangle $G[\{u, v, w\}]$ in G, then delete $A \setminus A^*$ from G.*

For the rest of this section we consider instances (G, c, k) for Multi-STC that are reduced regarding Rules 5.5–5.7. Observe that these instances only contain triangle-free periphery components A where every vertex $v \in A$ has $\deg_G(v) = \lfloor \frac{c}{2} \rfloor + 1$. Since ECS and Multi-STC are the same on triangle-free graphs one might get the impression that we can use Vizing's theorem to prove that all periphery components in G are good. Consider the example in Figure 5.3 to see that this is not necessarily the case, since there are might be triangles in the core.

We now continue with the description of the kernel for Multi-STC. Let (G, c, k) be an instance of Multi-STC that is reduced regarding Rules 5.5–5.7. We consider the periphery components of G that contain cycles with at least four vertices. In this context, a *cycle* (of length r) is an edge-simple path $P = (v_0, v_1, \dots, v_{r-1}, v_0)$ where the last vertex and the first vertex of P are the same, and all other vertices occur at most once on P. We will see that the number of vertices in acyclic periphery components—which are periphery components A where $G[A]$ is a tree—is already bounded in c and $\xi_{\lfloor \frac{c}{2} \rfloor + 1}$. To remove the other components, we show that periphery components with cycles are good. To this end we show two lemmas. The idea behind the second lemma (Lemma 5.26) is that, given a cycle P such that some edge in $E(P)$ is labeled with a strong color q, we can transform the labeling in a way that another edge $e' \in E(P)$ has the strong color q. Lemma 5.26 is a technical statement that we need for the proof of Lemma 5.26. The intuitive idea behind its proof is that we use our technique of moving weak edges along paths to rotate weak and strong edge-colors in the cycle.

Lemma 5.25. *Let $A \subseteq \mathscr{P}$ be a periphery component and let L be an STC-labeling for G. Moreover, let $P = (v_0, v_1, \dots, v_{r-1}, v_0)$ be a cycle in A such that $W_L \cap E(P) \neq \emptyset$ and let $Q_L^P = (q_0, q_1, \dots, q_{r-1})$ be the color sequence of P under L. Then, there exist STC-labelings $L_0, L_1, L_2, \dots, L_{r-1}$ for G such that $L_i|_{E \setminus E(P)} = L|_{E \setminus E(P)}$ and*

$$Q_{L_i}^P(j) = q_{(i+j) \bmod r} \text{ or } |W_{L_i}| < |W_L|$$

for all $i, j \in \{0, \dots, r-1\}$.

Proof. Without loss of generality, we assume that $\{v_0, v_1\} \in W_L$ and therefore $q_0 = 0$. We prove the existence of the labelings L_i with $i \in \{0, 1, \dots, r-1\}$ by induction over i.

Base Case: $i = 0$. In this case we set $L_0 := L$.

Inductive Step: By the inductive hypothesis, there is a labeling L_{i-1} such that $|W_{L_{i-1}}| < |W_L|$ or

$$Q^P_{L_{i-1}}(j) = q_{(i-1+j) \bmod r}.$$

If $|W_{L_{i-1}}| < |W_L|$, then we define L_i by $L_i := L_{i-1}$ and nothing more needs to be shown. Otherwise, let $P' := (v_{r-i+1}, v_{r-i+2}, \dots, v_{r-1}, v_0, v_1, \dots, v_{r-i+1})$. Note that P' describes the same cycle as P by rotating the vertices. More precisely,

$$P(j) = P'((j + i - 1) \bmod r).$$

Therefore, P' is edge-simple and has the color sequence $Q^{P'}_{L_{i-1}} = (q_0 = 0, q_1, \dots, q_{r-1})$. By the moving lemma (Lemma 5.20), there exists an STC-labeling L_i with $L_i|_{E\setminus E(P)} = L_{i-1}|_{E\setminus E(P)}$, such that $|W_{L_i}| < |W_{L_{i-1}}|$ or

$$Q^{P'}_{L_i}(j) = q_{(j+1) \bmod r}.$$

In case of $|W_{L_i}| < |W_{L_{i-1}}|$, nothing more needs to be shown. Otherwise, observe that

$$Q^P_{L_i}(j) = Q^{P'}_{L_i}((j + i - 1) \bmod r) = q_{(j+i) \bmod r}$$

which completes the inductive step. □

Lemma 5.26. *Let $A \subseteq \mathscr{P}$ be a periphery component and let L be an STC-labeling. Moreover, let $P = (v_0, v_1, \dots, v_{r-1}, v_0)$ be a cycle in A with $W_L \cap E(P) \neq \emptyset$ and let $e_1, e_2 \in E(P)$ with $e_2 \in S^q_L$ for some strong color $q \in \{1, \dots, c\}$. Then, there exists an STC-labeling L' with $L'|_{E\setminus E(P)} = L|_{E\setminus E(P)}$ such that $e_1 \in S^q_{L'}$ or $|W_{L'}| < |W_L|$.*

Proof. Let $Q^P_L := (q_0, q_1, \dots, q_{r-1})$. Without loss of generality, we assume that $\{v_0, v_1\} \in W_L$ and $e_2 = \{v_t, v_{t+1}\}$ for some $t \in \{1, \dots, r-1\}$. Then, $q_0 = 0$ and $q_t = q$. Furthermore, since $e_1 \in E(P)$ we have $e_1 = \{P(j), P(j+1)\}$ for some $j \in \{0, 1, \dots, r-1\}$.

Consider the STC-labelings $L_0, L_1, L_2, \dots L_{r-1}$ from Lemma 5.25. If $|W_{L_i}| < |W_L|$ for one such labeling L_i, then nothing more needs to be proven. Otherwise, set $i := (t - j) \bmod r$. We show that $e_1 \in S^{q_t}_{L_i}$ by proving $Q^P_{L_i}(j) = q_t$:

$$Q^P_{L_i}(j) = q_{(i+j) \bmod r} = q_{((t-j) \bmod r)+j) \bmod r} = q_{(t-j+j) \bmod r} = q_t.$$

□

We next use Lemma 5.26 to prove that periphery components with cycles are good.

Proposition 5.27. *Let $(G = (V, E), c, k)$ be a reduced instance of* Multi-STC *regarding Rules 5.5–5.7, where $c \geq 4$ is even. Let $A \subseteq \mathscr{P}$ be a periphery component in G such that there is a cycle $P := (v_0, v_1, \dots, v_{r-1}, v_0)$ in $G[A]$. Then, A is good.*

Proof. Without loss of generality, we assume that the cycle P has no chords. Otherwise, we replace P by the shorter cycle. Furthermore, P has length at least 4, since the instance is reduced regarding Rule 5.7. Let L be an arbitrary STC-labeling for G with $E(A) \subseteq W_L$. We prove that there is an STC-labeling which is partially equal to L with respect to $E \setminus E(A)$ and has no weak edges in $E(A)$.

Let L' be an STC-labeling for G with $L'|_{E\setminus E(A)} = L|_{E\setminus E(A)}$. If $W_{L'} \cap E(A) = \emptyset$, nothing more needs to be shown. So, let $W_{L'} \cap E(A) \neq \emptyset$. Then, by Proposition 5.21 we can assume that there exists one unique $e \in W_{L'} \cap E(A)$. Moreover, by the moving lemma (Lemma 5.20) we assume without loss of generality that $e = \{v_0, v_1\}$: If $e \neq \{v_0, v_1\}$ we apply the moving lemma on a path P, where the first edge on P is e and the last edge on P is $\{v_0, v_1\}$. Then, P is a cycle with $E(P) \cap W_{L'} \neq \emptyset$ in $G[A]$.

The idea is to use Lemma 5.26 to transform L' into an STC-labeling without weak edges in $E(A)$. To this end, we introduce some notation: For a vertex $v \in V(P)$, we let $\operatorname{out}(v) := \{i \in \{1, \dots, c\} \mid \exists e \in E \setminus E(P) : e \cap V(P) = \{v\} \wedge e \in S^i_{L'}\}$ denote the set of strong colors of incident edges of v that are not in $E(P)$. Note that $|\operatorname{out}(v)| \leq \frac{c}{2} - 1$ for every $v \in V(P)$. Consider the following case distinction.

Case 1: There exists an edge $\{v_j, v_{(j+1) \bmod r}\} \in E(P)$ that has a strong color $q \in \bigcup_{v \in V(P)} \operatorname{out}(v)$ under L**.** Then, let $v \in V(P)$ be a vertex with $q \in \operatorname{out}(v)$, and let $e \in E(P)$ be an edge incident with v. Since $\{v_j, v_{(j+1) \bmod r}\} \in S^q_{L'}$, Lemma 5.26 guarantees the existence of an STC-labeling L'' with $L''|_{E\setminus E(P)} = L'|_{E\setminus E(P)}$, such that $e \in S^q_{L''}$ or $|W_{L''}| < |W_{L'}|$.

Assume towards a contradiction that $e \in S^q_{L''}$. Then, since $L''|_{E\setminus E(P)} = L'|_{E\setminus E(P)}$ and $q \in \operatorname{out}(v)$, the vertex v has two incident edges with the same strong color. Furthermore, since G is reduced regarding Rule 5.7, no edge in $E(A)$ is part of a triangle in G. Hence, v is the central vertex of an induced P_3 where both edges have strong color q under L''. This contradicts the fact that L'' is an STC-labeling. We conclude $|W_{L''}| < |W_{L'}|$, which implies $E(A) \cap W_{L''} = \emptyset$. Since L' was arbitrary, the periphery component A is good.

Case 2: There is no edge in $E(P)$ that has a strong color $q \in \bigcup_{v \in V(P)} \operatorname{out}(v)$**.**

Case 2.1: There is some $j \in \{0, 1, \dots, r-1\}$ with $|\operatorname{out}(v_j) \cup \operatorname{out}(v_{(j+1) \bmod r})| < c-2$**.** In this case, consider the edge-simple subpath $P' = (v_0, v_1, \dots, v_j, v_{(j+1) \bmod r})$. Observe that $Q^{P'}_{L'}(0) = 0$, since $\{v_0, v_1\} \in W_{L'}$. By the moving lemma (Lemma 5.20), there exists an STC-labeling L'' with $L''|_{E\setminus E(P')} = L'|_{E\setminus E(P')}$ such that $|W_{L''}| < |W_{L'}|$ or $Q^{P'}_{L''}(j) = 0$. In case of $|W_{L''}| < |W_{L'}|$, nothing more needs to be shown.

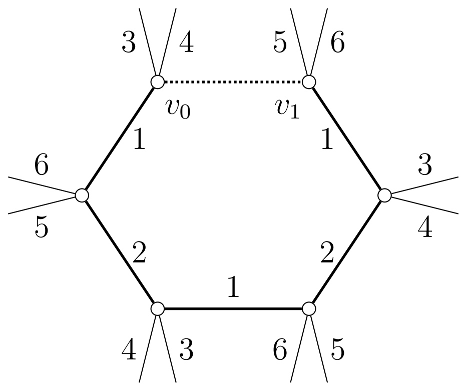


Figure 5.5: A sketch of Case 2.2 from the proof of Proposition 5.27, where $c = 6$. The thick edges correspond to the edges in $E(P)$. The edge labels correspond to the strong colors of the edges under L' and the edge $\{v_0, v_1\}$ is weak under L'. There are two out-sets $\{3, 4\}$ and $\{5, 6\}$ that appear in an alternating manner on the vertices on P. The edge $\{v_0, v_1\}$ is incident with edges of the strong colors 1, 3, 4, 5, and 6. Thus, it can be moved from $W_{L'}$ to $S^2_{L'}$ without producing any STC violation.

Otherwise, $Q^{P'}_{L''}(j) = 0$ implies $\{v_j, v_{(j+1) \bmod r}\} \in W_{L''}$. Note that there are at most $2 + |\mathrm{out}(v_j) \cup \mathrm{out}(v_{(j+1) \bmod r})| < c$ distinct strong colors of edges incident with $\{v_j, v_{(j+1) \bmod r}\}$. Consequently, we can transform L'' into an STC-labeling L''' with $L'''|_{E \setminus E(A)} = L''_{E \setminus E(A)}$ and $W_{L'''} \cap E(A) = \emptyset$. Since L' was arbitrary, the periphery component A is good.

Case 2.2: $|\mathrm{out}(v_j) \cup \mathrm{out}(v_{(j+1) \bmod r})| = c - 2$ for every $j \in \{0, \dots, r-1\}$**.** We first provide some intuition for this case. Our main goal is to show that the edges $\{v_1, v_2\}$ and $\{v_{r-1}, v_0\}$ both have the same strong color under L'. Then, the weak edge $\{v_0, v_1\}$ is incident with edges of at most $c - 1$ distinct strong colors and thus, it can safely be moved to a strong color class. An example of a labeling L' for this case is shown in Figure 5.5.

The proof of this case is structured into two claims. The first claim states some technical properties about out-sets that we use in the rest of this proof. With the second claim we show that there are only two possible out-sets that appear in an alternating manner on the vertices on P. We then use these two claims to derive a statement about the color sequence $Q^P_{L'}$ which then implies that $\{v_1, v_2\}$ and $\{v_{r-1}, v_0\}$ have the same strong color under L'.

Claim 1. *It holds that*

a) $|\mathrm{out}(v_j)| = \frac{c}{2} - 1$ *for every* $j \in \{0, \dots, r-1\}$*,* *b)* $\mathrm{out}(v_j) \cap \mathrm{out}(v_{(j+1) \bmod r}) = \emptyset$ *for every* $j \in \{0, \dots, r-1\}$*, and*

c) $|\bigcup_{v \in V(P)} \mathrm{out}(v)| \leq c - 2$.

Proof. Statements *a*) and *b*) hold since otherwise there exists some $j \in \{1, \dots, r-1\}$ with $|\mathrm{out}(v_j) \cup \mathrm{out}(v_{(j+1) \bmod r})| < c - 2$ contradicting the constraint of Case 2.2. It remains to show Statement *c*). Recall that no edge in $E(A)$ is part of a triangle, since(G, c, k) is reduced regarding Rule 5.7. Then, since L' satisfies STC, there are at least two different edges in $E(P)$ that are labeled with distinct strong colors under L'. Since no edge in $E(P)$ has a strong color in $\bigcup_{v \in V(P)} \mathrm{out}(v)$ by the condition of Case 2, we have $|\bigcup_{v \in V(P)} \mathrm{out}(v)| \leq c - 2$. ♢

We next use Claim 1 to show that, given a vertex on P, both of its neighbors on the cycle have the same out-set.

Claim 2. *Let* $j \in \{0, \dots, r-1\}$*. Then,* $\mathrm{out}(v_{(j-1) \bmod r}) = \mathrm{out}(v_{(j+1) \bmod r})$.

Proof. Assume towards a contradiction that $\mathrm{out}(v_{(j-1) \bmod r}) \neq \mathrm{out}(v_{(j+1) \bmod r})$. Then, the union $Z := \mathrm{out}(v_{(j-1) \bmod r}) \cup \mathrm{out}(v_{(j+1) \bmod r})$ contains more than $\frac{c}{2} - 1$ elements since $|\mathrm{out}(v_{(j-1) \bmod r})| = |\mathrm{out}(v_{(j+1) \bmod r})| = \frac{c}{2} - 1$ due to Claim 1 *a*). By Claim 1 *b*), $\mathrm{out}(v_j)$ is disjoint from Z. Consequently, we have

$$\mathrm{out}(v_j) \subseteq \left(\bigcup_{v \in V(P)} \mathrm{out}(v) \right) \setminus Z.$$

Then, Claim 1 *c*) implies

$$\begin{aligned} |\mathrm{out}(v_j)| &\leq |\bigcup_{v \in V(P)} \mathrm{out}(v)| - |Z| \\ &< (c-2) - (\frac{c}{2} - 1) \\ &= \frac{c}{2} - 1, \end{aligned}$$

which contradicts Claim 1 *a*). ♢

Claim 2 implies that the number $|V(P)|$ of vertices on $P = (v_0, \dots, v_{r-1}, v_0)$ is even, since otherwise $\mathrm{out}(v_0) = \mathrm{out}(v_2) = \mathrm{out}(v_4) = \cdots = \mathrm{out}(v_{r-1})$ contradicting the fact that $\mathrm{out}(v_{r-1}) \cap \mathrm{out}(v_0) = \emptyset$ due to Claim 1 *b*).

Next, by Claim 1 *a*) and *b*), the vertices v_0 and v_1 have two distinct out-sets, each of size $\frac{c}{2} - 1$. By the condition of Case 2, no edge of $E(P)$ has a strong color that appears in one of the out-sets and thus, there are at most two distinct strong colors q_1 and q_2 on edges in $E(P)$. Then, since the cycle P has no chords, the strong colors q_1 and q_2 appear in an alternating manner on the edges of P. Moreover, since $|V(P)|$ is even, we have $Q^P_{L'} = (0, q_1, q_2, q_1, \dots, q_1)$ or $Q^P_{L'} = (0, q_2, q_1, \dots, q_2)$. Consequently, the edges $\{v_1, v_2\}$ and $\{v_{r-1}, v_0\}$ both have the same strong color under L'. As discussed above, the weak edge $\{v_0, v_1\}$ is then incident with edges of at most $c-1$ distinct strong colors. Therefore, $\{v_0, v_1\}$ safely be moved from $W_{L'}$ to a strong color class. This way, we transformed L' into an STC-labeling L'' with $L'|_{E\setminus E(A)} = L''|_{E\setminus E(A)}$ and $W_{L''} \cap E(A) = \emptyset$. Since L' was arbitrary, the periphery component A is good. □

Propositions 5.27 and 5.17 imply the safeness of the final rule which together with Rules 5.5–5.7 gives the kernel.

Reduction Rule 5.8. *If there is a periphery component $A \subseteq \mathscr{P}$ such that $G[A]$ contains a cycle, then delete $A \setminus A^*$ from G.*

Theorem 5.28. MULTI-STC *restricted to instances with $c \geq 3$ admits a problem kernel with $\mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor+1} \cdot c)$ vertices and $\mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor+1} \cdot c^2)$ edges that can be computed in $\mathcal{O}(n+m)$ time.*

Proof. From Proposition 5.19 we know, that if $c \geq 3$ is odd, we can compute a problem kernel with at most $4 \cdot \xi_{\lfloor \frac{c}{2} \rfloor+1} \cdot (\lfloor \frac{c}{2} \rfloor + 1)$ vertices, and $\mathcal{O}(\xi \cdot c^2)$ in $\mathcal{O}(n+m)$ time. Let $(G = (V, E), c, k)$ be an instance of MULTI-STC, where $c \geq 4$ is an even number. Throughout this proof let $\xi \leq 2\xi_{\lfloor \frac{c}{2} \rfloor+1}$ denote the size of a 2-approximate set D' of the minimum edge-deletion set $D_{\lfloor \frac{c}{2} \rfloor+1}$. Recall that D' can be computed in $\mathcal{O}(n+m)$ time due to Proposition 5.3. Let $\mathscr{C} := \mathscr{C}(D')$ and $\mathscr{P} := \mathscr{P}(D')$. We compute an instance $(G' = (V', E'), c, k)$ as follows: We start by applying Rules 5.5 and 5.6 exhaustively. This can be done in $\mathcal{O}(n+m)$ time by computing all connected components of $G[\mathscr{P}]$ and checking whether they have close vertices or vertices of low degree. Afterwards, we apply Rules 5.7 and 5.8 exhaustively. This can also be done in $\mathcal{O}(n+m)$ time by testing if the connected components in $G[\mathscr{P}]$ with non-close vertices contain cycles. Note that at this point it is not important whether a cycle is a triangle or a cycle of length at least 4.

We next show $|V'| \leq (c+7) \cdot \xi$. Let $\mathscr{C}$ be the set of core vertices of G' and $\mathscr{P}$ be the set of periphery vertices of G'. Since $|\mathscr{C}| \leq 2\xi$, and every $v \in \mathscr{C}$ is incident with at most $\frac{c}{2}+1$ edges, there are at most $2\xi + 2\xi(\frac{c}{2}+1) = \xi c + 4\xi$ vertices in $\mathscr{C} \cup N(\mathscr{C})$. It remains to show that there are at most 3ξ non-close vertices in $\mathscr{P}$. Consider the following family of periphery components.

$$\mathcal{A} := \{A \subseteq \mathscr{P} \mid A \text{ is a periphery component with } A \setminus A^* \neq \emptyset\}$$

Since G' is reduced regarding Rules 5.6–5.8, every $G[A]$ with $A \in \mathcal{A}$ is a tree, where every vertex $v \in A$ has degree $\deg_G(v) = \frac{c}{2} + 1$ in G. We define a *leaf vertex* as a vertex $v \in \bigcup_{A\in\mathcal{A}} A$ with $\deg_{G[\mathscr{P}]}(v) = 1$. Note that these vertices are exactly the leaves of the tree $G[A]$ for some $A \in \mathcal{A}$, and all leaf vertices are close vertices in $\mathscr{P}$, since (G', c, k) is reduced regarding Rule 5.6. Let p be the number of leaf vertices. We show that $p \leq 3\xi$. Since (G', c, k) is reduced regarding Rule 5.6, every vertex $v \in \bigcup_{A\in\mathcal{A}} A$ has a degree of $\deg_G(v) = \frac{c}{2} + 1$. Hence, every leaf vertex has exactly $\frac{c}{2}$ neighbors in $\mathscr{C}$. We thus have

$$p \cdot \frac{c}{2} \leq |E(\mathscr{C}, N(\mathscr{C}))| \leq 2\xi \left(\frac{c}{2} + 1\right),$$

and therefore $p \leq 2\xi + \frac{4\xi}{c} \leq 3\xi$, since $c \geq 4$. Recall that every non-close vertex v in some tree $G[A]$ satisfies $\deg_{G[A]}(v) = \frac{c}{2} + 1 > 2$. Since a tree has at most as many vertices with degree at least three as it has leaves, we conclude $|(\bigcup_{A\in\mathcal{A}} A) \setminus (\bigcup_{A\in\mathcal{A}} A^*)| \leq 3\xi$. Consequently, there are at most 3ξ non-close vertices in $\mathscr{P}$. Then, G' contains of at most $(c + 7) \cdot \xi \in \mathcal{O}(\xi c)$ vertices, as claimed. Since each vertex is incident with at most $\frac{c}{2} + 1$ edges, G' has $\mathcal{O}(\xi c^2)$ edges. □

5.3 Concluding Remarks

We studied ECS parameterized by distance to low-degree graphs and analyzed the parameterized complexity of Multi-STC for similar parameterizations. Recall that in Chapters 3 and 4 we studied two generalizations of Multi-STC: In VL-Multi-STC every vertex has a list of strong colors that restrict the possible strong colors of its incident edges and in EL-Multi-STC, every edge has a list of possible strong colors. Analogously, we can define the generalizations VL-ECS and EL-ECS of ECS. Unfortunately, all list-variants of ECS and Multi-STC are NP-hard for every fixed $c \geq 3$ even if $\xi_3 = 0$ [78]. Therefore, $\xi_{c-1} + c$ is not a promising parameter for the list variants. However, if we consider ξ_2 instead of ξ_{c-1}, EL-ECS and EL-Multi-STC admit an $11\xi_2$-edge and $10\xi_2$-vertex kernel that can be computed in $\mathcal{O}(n^2)$-time for every c [78].

Open Questions. There are several ways of extending our results that seem interesting topics for future research. In the FPT results of this chapter the value of c is always part of the parameter and it would be very interesting to understand whether this is necessary. For example, is ECS fixed-parameter tractable with respect to ξ_{c-1} alone? Moreover, the results presented in this chapter all rely on problem kernelizations. One further interesting question is thus to ask for direct algorithms that solve Multi-STC and ECS in FPT time for the parameters studied in this chapter. In this context, we would like to emphasize that even though Multi-STC does not admit a polynomial kernel when parameterized by $\lambda_c + c$, it is likely that the problem is FPT for this parameter: Courcelle's Theorem [36] might imply that Multi-STC is FPT when parameterized by $\mathrm{tw} + c$, where tw denotes the so-called *treewidth* [15] of the input graph. Intuitively, the treewidth is a parameter that measures how tree-like a graph is. Since the treewidth is never bigger than $\lambda_c + \mathcal{O}(1)$, this would imply that Multi-STC is FPT when parameterized by $\lambda_c + c$.

Our kernel for ECS parameterized by $\xi_{c-1} + c$ implies an observation about degree-one vertices in a proper edge-colored graph. Let G be a graph with at least three degree-one vertices such that there is a proper edge labeling with c colors and without weak edges for G. The fact that Rule 5.1 is correct implies that there is a proper c-labeling L for G with $W_L = \emptyset$ such that not all edges incident with degree-one vertices receive the same strong color: if such a graph exists, one could use it to construct an example instance similar to the one shown in Figure 5.3 where Rule 5.1 fails. As a consequence, given a graph with maximum degree Δ, for every set S of three or more vertices with degree $\Delta - 1$, there exists a proper edge labeling such that not all vertices in S are incident with the same colors. However, it is not clear whether this observation can be exploited to obtain new algorithmic results for Edge Coloring or ECS.

In case of Multi-STC, the most obvious open question is if the fixed-parameter tractability for ξ_{c-1} can be extended to instances with $c > 4$. Even for $c = 5$ and ξ_4 we believe that one might need a fundamentally new idea since moving weak edges through paths of low-degree vertices—the core-technique used in Section 5.2—appears to be more complicated in this case.

Finally, in our parameterization for Multi-STC we use the fact that Multi-STC is polynomial-time solvable on graphs with maximum degree $c - 1$. This is a simple corollary of Vizing's theorem and the fact that every proper edge labeling is an STC-labeling. It would be nice to extend the class of tractable instances further in the following sense: For which superclasses of the graphs with maximum degree $c-1$ does Multi-STC remain polynomial-time solvable? Surely, our fixed-parameter algorithms give such superclasses but are there some that can be described without the use of parameters, for example via a characterization of forbidden induced subgraphs of size at most $f(c)$? Identifying further polynomial-time solvable cases of ECS and Multi-STC may motivate the study of new distance-from-triviality parameterizations for the two problems.

Another interesting direction of research is, to experimentally evaluate the practical impact of the reduction rules presented in this chapter. In real-world social networks, one may assume that there are relatively few vertices with a high degree, while most of the vertices have a relatively low degree. In terms of core and periphery as defined in this chapter, this might indicate that the core is relatively small in comparison to the periphery. Furthermore, the parameters ξ_{c-1} and $\xi_{\lfloor \frac{c}{2} \rfloor+1}$ are non-increasing for increasing values of c. Or—in the terms of core and periphery—the core might get smaller when the number of strong colors increases. However, recall that the problem kernel for ECS consists of $\mathcal{O}(\xi_{c-1} \cdot c)$ vertices and the problem kernel for Multi-STC consists of $\mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor+1} \cdot c)$ vertices. It is not clear for which values of c the kernels might be practically relevant.

To obtain the $\mathcal{O}(\xi_{c-1} \cdot c)$-vertex kernel for ECS and the $\mathcal{O}(\xi_{\lfloor \frac{c}{2} \rfloor+1} \cdot c)$-vertex kernel for Multi-STC, we essentially showed that (most of) the edges between low-degree vertices are always strong. Thus, one may ask how useful the approach of strong triadic closure with multiple strong colors is to identify weak relationships in a social network; especially in sparse parts of the network. However, note that besides the identification of weak relationships, the approach of multiple strong colors helps to classify the strong relationship types.

Part III

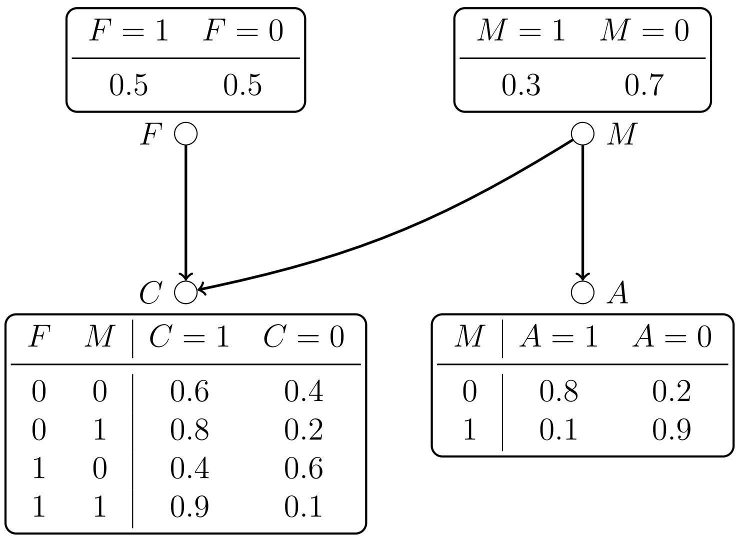
Bayesian Network Structure Learning

"In solving a problem of this sort, the grand thing is to be able to reason backwards." as noted by Sherlock Holmes in the novel *A study in scarlet* by Arthur Conan Doyle [47] outlines one of his important skills to solve crime cases. In a technique called *induction*, Sherlock Holmes derives the solution of a crime case from given evidence. As mentioned in "The Book of Why" by Judea Pearl and Dana Mackenzie [145], automating this process of reasoning from evidence to criminal act or—more generally—from effect to cause is an important direction of study in the field of artificial intelligence (AI). One of the basic tools to reason about causality are *Bayesian Networks* [144].

In 1985, Judea Pearl [143] introduced Bayesian Networks which were named after Thomas Bayes who is famous for his well-known theorem about conditional probabilities. Roughly speaking, Bayesian networks are a compact representation of multivariate probability distributions and provide a good trade-off between expressive power and querying efficiency [160]. Bayesian networks have many applications: Medical diagnosis systems build a hypothesis for a disease given the set of observed symptoms of a patient [103]; intelligent tutoring systems based on Bayesian networks emulate a human tutor [21]; Bayesian networks can be used as a tool for credit rating of companies [70]; and—matching the introductory quote of Sherlock Holmes—in forensic science Bayesian networks assist reasoning about evidence in legal settings [49]. For a detailed collection of various applications of Bayesian networks we refer to "Bayesian Networks: A Practical Guide to Applications" [149].

Formally, a *Bayesian Network* is a tuple (D, T) consisting of a directed acyclic graph (DAG) D and a set T of conditional probability tables. The vertices of D correspond to the random variables of the conditional probability distribution and the arcs between these vertices correspond to conditional dependencies. More precisely, if a vertex v has an incoming arc from a vertex u, then the random variable corresponding to v depends on the random variable corresponding to u. The set T contains one conditional probability table for every variable. These tables encode the distribution of the corresponding variable given the values of its parents in D.

Consider the toy example Bayesian network given in Figure III.1. The network models a crime story. The vertices of the DAG correspond to the random variables *being friends with the victim* (F), *having a conflict with the victim* (C), *being the murderer* (M), and *having a believable alibi* (A). In this particular setting all random variables can be either true (1) or false (0). The most interesting question in this context might be: What is the chance that a person is the murderer, if we know that this person had a conflict with the victim and can not produce a believable alibi? Even though this particular probability is not explicitly contained in one of the conditional probability tables, the information can be extracted from the Bayesian



$F=1$	$F=0$
0.5	0.5

$M=1$	$M=0$
0.3	0.7

F	M	$C=1$	$C=0$
0	0	0.6	0.4
0	1	0.8	0.2
1	0	0.4	0.6
1	1	0.9	0.1

M	$A=1$	$A=0$
0	0.8	0.2
1	0.1	0.9

Figure III.1: A Bayesian network.

network. This relies on the so-called *chain rule* of Bayesian networks [40] which tells us that for each instanciation $(F = f, C = c, M = m, A = a)$ of the random variables, the probability $\Pr(f, c, m, a)$ can be expressed as the product $\theta_f \cdot \theta_{c|f,m} \cdot \theta_m \cdot \theta_{m|a}$, where the θ-values correspond to entries in the conditional probability tables. The task of computing such probabilites from a Bayesian network is known as Bayesian Inference, which is an NP-hard computational problem [40].

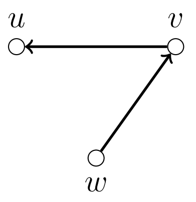
Before we may use a Bayesian network to answer such queries, we need to set up the Bayesian network itself. More precisely, we need to compute the DAG and the entries of the conditional probability tables from a given set of observed data. It turns out that the crucial part of this task is to construct the DAG and that the entries of the conditional probability tables can be computed in a straightforward manner once the DAG is known [40]. The task of constructing the DAG is called Bayesian Network Structure Learning (BNSL) which is the topic of this part.

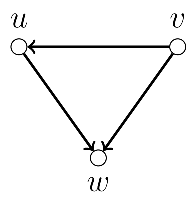
An important approach to learn a Bayesian network structure is the so-called *score-based* approach. Here, given a data set (a table containing observed instantiations of the random variables), one aims to construct the best DAG according to some score function that measures how well the DAG fits the data [89]. In a maximum likelihood-approach, the score function is *decomposable* into local scores for each vertex [40]. That is, for each vertex v and each set of potential parent vertices P of v, there is a value $f_v(P)$ and the score of a particular DAG $D := (N, A)$ is $\sum_{v \in N} f_v(P_v^A)$

P	$f_u(P)$
$\{v, w\}$	2
$\{v\}$	3
$\{w\}$	1
$\emptyset$	0

P	$f_v(P)$
$\{u, w\}$	3
$\{u\}$	1
$\{w\}$	1
$\emptyset$	0

P	$f_w(P)$
$\{u, v\}$	4
$\{u\}$	2
$\{v\}$	2
$\emptyset$	0





$$\underbrace{f_u(P_u^{A_1})}_{=3}+\underbrace{f_v(P_v^{A_1})}_{=1}+\underbrace{f_w(P_w^{A_1})}_{=0}=4 \qquad \underbrace{f_u(P_u^{A_2})}_{=3}+\underbrace{f_v(P_v^{A_2})}_{=0}+\underbrace{f_w(P_w^{A_2})}_{=4}=7$$

Figure III.2: An example of an instance $I := (N, \mathcal{F}, t)$ of Vanilla-BNSL where $N = \{u, v, w\}$ and $t = 7$. The upper part shows the family $\mathcal{F}$ of local scores. The lower part shows two arc sets A_1 and A_2 and the corresponding sums of local scores. Note that A_2 is $(N, \mathcal{F}, t)$-valid and thus, I is a yes-instance.

where P_v^A denotes the parent set of v in D. The goal is to construct a DAG that maximizes the sum of the local scores. There are various approaches of computing such local scores from a given data set [4, 161, 22, 89, 9, 18, 92, 94, 95, 132] that are constructed to be decomposable. Some of these are well-established scores like AIC, BIC, or BDe. Besides these established scores there are multiple works that propose other scores. In the common formulation of the computational problem of BNSL, pre-computed local scores are part of the input and thus, the problem formulation is independent from the concrete scoring function.

Problem Definition. The problem definition of BNSL—to which we will refer as Vanilla-BNSL—has been studied in multiple algorithmic works [141, 69, 113, 112, 131, 65]. Given a vertex set N, we call a family $\mathcal{F} = \{f_v : 2^{N\setminus\{v\}} \to \mathbb{N}_0 \mid v \in N\}$ a family of *local scores* for N. Intuitively, for a vertex $v \in N$ and some $P \in 2^{N\setminus\{v\}}$, the value $f_v(P) \in \mathbb{N}_0$ represents the score we obtain if we choose exactly the vertices of P as parents for v, which corresponds to the idea of decomposable scoring functions. As a shorthand, we define $\text{score}_{\mathcal{F}}(A) := \sum_{v\in N} f_v(P_v^A)$ for the total score of the network structure. The following definition and the corresponding computational problem specify which arc sets we aim to compute.

Definition III.1. Given a vertex set N, local scores $\mathcal{F}$, and some integer $t \in \mathbb{N}_0$, an arc set $A \subseteq N \times N$ is called $(N, \mathcal{F}, t)$-*valid* if (N, A) is a DAG and $\text{score}_{\mathcal{F}}(A) \geq t$.

> Bayesian Network Structure Learning (Vanilla-BNSL)
> **Input**: A set of vertices N, local scores $\mathcal{F} = \{f_v \mid v \in N\}$, and an integer $t \in \mathbb{N}_0$.
> **Question**: Is there an $(N, \mathcal{F}, t)$-valid arc set $A \subseteq N \times N$?

Figure III.2 shows an example of a yes-instance of Vanilla-BNSL. Throughout this work, we let $n := |N|$ denote the number of vertices in a given instance $I = (N, \mathcal{F}, t)$. Furthermore, we assume that for $N = \{v_1, \ldots, v_n\}$, the local scores $\mathcal{F}$ are given as a two-dimensional array $\mathcal{F} := [Q_1, Q_2, \ldots, Q_n]$, where each Q_i is an array containing all triples $(f_{v_i}(P), |P|, P)$ where $f_{v_i}(P) > 0$ or $P = \emptyset$. Note that a triple $(f_{v_i}(P), |P|, P)$ of a non-empty parent set P is only part of the input if its local score $f_{v_i}(P)$ is not 0. This input representation is known as *non-zero representation* [141]. The size $|\mathcal{F}|$ is then defined as the number of bits we need to store this two-dimensional array. As the *size of* I we define $|I| := n + |\mathcal{F}| + \log(t)$.

Overview of Part III. In Chapter 6, we study the parameterized complexity of learning a Bayesian network under additional sparsity constraints. Here, the input consists of an additional integer k and we aim to construct a DAG such that some structural graph parameter is bounded by k. We study parameterization by k for several sparsity constraints. Important in this context are sparsity constraints that are posed on the so-called moralized graph, an undirected graph that is associated to the Bayesian network structure. Furthermore, we provide a kernel lower bound for Vanilla-BNSL by proving that there is presumably no polynomial kernel when parameterized by the number of vertices n.

In Chapter 7, we study a problem called Polytree Learning, which is a version of BNSL where one aims to compute a polytree instead of a DAG. A polytree is a DAG such that the underlying undirected graph (skeleton) is acyclic. For parameterization by the number of vertices n we provide the first FPT-algorithm that is singly-exponential in n. Furthermore, we introduce a new parameter that is potentially smaller than n and show that Polytree Learning is FPT for this parameter on instances that have a bounded parent set size.

In Chapter 8, we study local search algorithms for BNSL. A natural approach for a local search algorithm is a hill climbing strategy, where one replaces a given BNSL solution by a better solution within some pre-defined neighborhood as long as this is possible. Applying this heuristic approach, there is a chance to get stuck in a poor local optimum. One way to decrease this chance is to use parameterized local search, where one has a pre-defined distance function in the solution space and asks for the best solution with distance at most r from the given solution for some search radius r; the parameter r allows to balance between solution quality and running time of the local search algorithm. In Chapter 8 of this work, we study ordering-based local search, where a Bayesian network structure is represented by a topological ordering. We consider four natural distance functions on the space of vertex orderings and analyze the parameterized complexity of the local search problem when parameterization by the radius r. The results range from subexponential FPT-time to W[1]-hardness depending on the distance function. We also outline the limits of ordering-based local search by showing that it cannot be used for some common structural constraints on the network.

Chapter 6

Learning Sparse Network Structures

When one aims to learn a Bayesian network structure from observed data, the goal is to represent the observed data as closely as possible. An intuitive idea is to find a network structure that maximizes the probability of observing the given data set. Following this approach, it may seem appropriate to learn a DAG that is a tournament. A *tournament* is a directed graph where every pair of vertices u and v is connected either by the arc (u, v) or by the arc (v, u). There are, however, several reasons why learning a tournament-like DAG should be avoided: First, such a network gives no information about which variables are conditionally independent. Second, including too many dependencies in the model makes the model vulnerable to overfitting. Finally, avoiding tournament-like network structures can lead to a more efficient inference in the Bayesian network. Recall that the inference problem on Bayesian networks is NP-hard [34] and thus, it is interesting to learn a network structure which might lead to an efficient inference. When the network is tree-like, however, efficient inference algorithms are possible. Intuitively, the *treewidth of a graph* [15] is a graph parameter that measures how tree-like the graph is. If the moralized graph of the network structure has small treewidth, then the inference task can be solved more efficiently [40]; the moralized graph of a network D is the undirected graph on the same vertex set that is obtained by adding an edge between each pair of vertices that is adjacent or has a common child in D. For a detailed discussion about the disadvantages of tournament-like networks we refer to the book by Adnan Darwiche [40].

One idea to decrease the chance of learning tournament-like networks is to use parent scores that reward the choice of smaller parent sets. Many popular scores like AIC [4], BIC [161], or BDe [22, 89] try to achieve this as follows: A vertex v gets a parent set score $a - b$ for a possible set of parents P where a is a measure for how well the arc set $P \times v$ fits to the data and b is a penalty term to favor simpler models [40]. Intuitively, this means: The bigger the size of P, the bigger the value of b. Defining parent scores in such a way may lead to smaller parent set sizes which can be seen as a local sparsity constraint on the network. However, note that global graph parameters like the treewidth might still be large: For example, a directed grid network structure might have a large treewidth while the maximum parent set size is at most 4.

Motivated by the benefits of simpler networks and the efficient inference in sparse networks, it has been proposed to learn optimal networks under global structural constraints which guarantee that the network or its moralized graph have a small treewidth [50, 112, 113, 32, 41, 69]. In this chapter, we continue this line of research, focusing on exact algorithms with worst-case running time guarantees. In other words, we want to find out for which structural constraints there are fast algorithms for learning optimal Bayesian networks under these constraints and for which constraints this is presumably impossible.

Related Work. When the network structure is restricted to be a branching, that is, a directed tree in which every vertex has indegree at most one, then an optimal network structure can be computed in polynomial time [32, 69]. Furthermore, learning a network structure that is a directed path is NP-hard [131]. Thus, since every directed path is a branching, learning a more restricted Bayesian network structure is not necessarily easier. Learning a network structure where the moralized graph is restricted to have treewidth at most ω is NP-hard for every fixed $\omega \geq 2$ and can be solved in $3^n n^{\omega+\mathcal{O}(1)}$ time [112].

Finally, Korhonen and Parviainen [113] considered a structural constraint that restricts the treewidth of the moralized graph by restricting the size of its vertex cover. An optimal network structure where the moralized graph is restricted to have a vertex cover of size at most k can be done in $4^k \cdot n^{2k} \cdot |I|^{\mathcal{O}(1)}$ time [113], where $|I|$ denotes the total input size. Since having a bounded vertex cover number implies that the graph has bounded treewidth, the network structures that are learned by BNSL with bounded-vc moralized graphs allow for fast inference. The XP algorithm provided by Korhonen and Parviainen can presumably not be improved to an FPT algorithm, since learning an optimal network structure with vertex cover number k is W[1]-hard for k [113].

Vanilla-BNSL is NP-hard [29] and can be solved in $2^n n^{\mathcal{O}(1)}$ time by dynamic programming over all subsets of the vertex set [142, 163]. The parameterized complexity of Vanilla-BNSL has been studied for structural parameters of the so-called *undirected superstructure* [141, 65]. The undirected superstructure is an undirected graph on the input vertex set N where two vertices u and v are adjacent if u is an element of a potential parent set of v or vice versa. BNSL is XP and W[1]-hard when parameterized by the treewidth of the undirected superstructure [141], and it is FPT when parameterized by the feedback edge number of the undirected superstructure [65].

Our Results. Extending previous work, we provide an almost complete picture of the parameterized complexity of BNSL with respect to several constraints that guarantee sparse networks or sparse moralized graphs. Since the constraints are formulated in terms of undirected graphs, we will refer to the undirected underlying graph of a network as its *skeleton*. An overview of our results and previous results for the considered problems is given in Table 6.1.

The results for BNSL with bounded-vc moralized graphs [113] form the starting point for our work. We show that BNSL with bounded-vc skeletons can be solved in XP-time and is W[2]-hard when parameterized by the size k of the vertex cover. Since the skeleton is a subgraph of the moralized graph, having bounded-vc skeleton can be seen as a less strict constraint than having a bounded-vc moralized graph. Note that, given a DAG D, the vertex cover number of $\mathcal{M}(D)$ can be arbitrarily larger than the vertex cover number of $\mathcal{S}(D)$: If one vertex v has $n-1$ incoming arcs and there are no further arcs in D, the skeleton of D has vertex cover number 1, while the moralized graph of D has vertex cover number $n-1$. We then consider further related structural constraints that are related to the vertex cover number in the following sense: Recall that a graph has a vertex cover of size k if and only if it can be transformed into a graph with maximum degree 0 by k vertex deletions. Thus, in BNSL with bounded-vc moralized graphs we learn a network whose skeleton or moralized graph is close, in terms of the number of vertex deletions, to the sparse graph class of edgeless graphs. We investigate whether there are further positive examples for such constrained network learning problems.

We consider the constraint that the skeleton or the moralized graph can be transformed into a graph with maximum degree 1 by at most k vertex deletions. This property is also known as having *dissociation number at most* k and we refer to graphs with this property as bounded-diss-number graphs in the following. We show that BNSL with bounded-diss-number moralized graphs can be solved in $n^{3k} \cdot k^{\mathcal{O}(k)} \cdot |I|^{\mathcal{O}(1)}$ time and thus in XP-time for k. Observe that moralized graphs with bounded dissociation number still have bounded treewidth and thus inference on the learned networks will still be solvable efficiently. On the negative side, we show that the problem is

Table 6.1: An overview of the parameterized complexity of constrained BNSL problems where structural parameters of the skeleton or the moralized graph are upper-bounded by a number k. The *distance to degree 2* is the minimum size of a vertex set S such that after removing S, the maximum degree is 2. The *c-component order connectivity (c-COC)* is the minimum size of a vertex set S such that after deleting S, every connected component has order at most c.

Bounded by k	**Skeleton**	**Moralized Graph**
Treewidth	NP-h for $k = 1$ [41]	NP-h for $k = 2$ [112]
Vertex cover number	XP (Thm 6.8) W[2]-h (Thm 6.9)	XP [113] W[2]-h (Cor 6.10)
Dissociation number	W[2]-h (Thm 6.9)	XP (Thm 6.19) W[2]-h (Cor 6.10)
Distance to degree 2	NP-h for $k = 0$ (Thm 6.20)	NP-h for $k = 0$ (Thm 6.20)
c-COC for $c \geq 3$	NP-h for $k = 0$ (Thm 6.21)	NP-h for $k = 0$ (Thm 6.21)
Number of edges	FPT (Cor 6.28) no $k^{\mathcal{O}(1)}$ kernel (Cor 6.34)	XP (Prop 6.29) W[1]-h (Thm 6.30)
Feedback edge set	NP-h for $k = 0$ [41]	W[1]-h (Thm 6.32)

W[2]-hard and thus presumably not FPT for k. The latter hardness result also holds for BNSL with bounded-diss-number skeletons; we did not obtain a positive result for this case, however.

We then consider two further constraints that are related to the dissociation number: We show that learning an optimal network whose skeleton or moralized graph has maximum degree 2 is NP-hard and that learning an optimal network in which every component of the skeleton or the moralized graph has at most c vertices is NP-hard for every $c \geq 3$. The latter constraint is related to the dissociation number since in a graph with maximum degree 1 every connected component has at most two vertices.

Next, we consider constraints that are formulated in terms of edge sets of the skeleton or the moralized graph. We show that optimal networks with at most k arcs can be found in time $2^{\mathcal{O}(k)} \cdot |I|^{\mathcal{O}(1)}$. In contrast, computing an optimal network whose moralized graph has at most k edges is W[1]-hard. Thus, putting structural constraints on the moralized graph may make the problem much harder than putting similar structural constraints on the skeleton. Furthermore, we consider the case where the edge deletion distance to trees is measured, in other words, the case where the skeleton or the moralized graph have a feedback edge set of size at most k. BNSL with tree skeletons is also called POLYTREE LEARNING and it is known to be NP-hard [41]. Thus, the learning problem for skeletons with feedback edge sets of size at most k is NP-hard even for $k = 0$. For BNSL with bounded-fes moralized graph, we obtain a first negative result by proving W[1]-hardness for k; an XP-time algorithm is however still possible.

Finally, we study problem kernelization for VANILLA-BNSL and BNSL under sparsity constraints. We show that VANILLA-BNSL does not admit a polynomial kernel for the number n of vertices unless NP $\subseteq$ coNP/poly. This then implies a kernel lower bound for all constrained BNSL problems studied in this section when parameterized by $n + k$.

Altogether, our results reveal that the difficulty of the learning problem may differ depending on whether we put the constraints on the skeleton or the moralized graph. Moreover, more general networks than those with bounded-vc moralized graphs can be computed efficiently. The room for generalization seems, however, very limited as even learning networks with constant degree or constant component size is NP-hard.

6.1 Constrained BNSL Problems

We provide a problem definition that captures BNSL problems for all sparsity constraints we consider in this chapter. We state the sparsity constraints in terms of vertex- or edge deletion distances to sparse graph classes. Recall that a *graph class* Π is a set of undirected graphs. Given an integer $k \in \mathbb{N}_0$, we let

$$\Pi + kv := \{G = (V, E) \mid \exists V' \subseteq V : (|V'| \leq k \wedge G - V' \in \Pi)\}$$

denote the class of graphs that can be transformed into a graph in Π by performing at most k vertex deletions. Analogously, we let

$$\Pi + ke := \{G = (V, E) \mid \exists E' \subseteq E : (|E'| \leq k \wedge (V, E \setminus E') \in \Pi)\}$$

denote the class of graphs that can be transformed into a graph in Π by performing at most k edge deletions. Furthermore, recall that we call Π *monotone* if Π is closed under edge- and vertex deletions. If Π is monotone, then $\Pi + kv$ and $\Pi + ke$ are monotone for every integer k.

6.1.1 Problem Definitions

The sparsity constraints are posed on the skeleton and the moralized graph [50] of the network. Let $D := (N, A)$ be a DAG. Recall that the *skeleton of* D is the undirected graph $\mathcal{S}(D) := (N, E)$, with $E := \{\{u, v\} \mid (u, v) \in A\}$. The *moralized graph of* D is the undirected graph $\mathcal{M}(D) := (N, E_1 \cup E_2)$ where the edge set is the union of $E_1 := \{\{u, v\} \mid (u, v) \in A\}$ and $E_2 := \{\{u, v\} \mid u$ and v have a common child in $D\}$. The edges in E_2 are called *moral edges*. Given a DAG (N, A), we define $\mathcal{S}(N, A) := \mathcal{S}((N, A))$ and $\mathcal{M}(N, A) := \mathcal{M}((N, A))$ for sake of readability. The problems are defined as follows.

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

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

Furthermore, we define the problems $(\Pi + e)$-Skeleton BNSL and $(\Pi + e)$-Moral BNSL on the same input and the question is if there exists an $(N, \mathcal{F}, t)$-valid arc set A such that $\mathcal{S}(N, A) \in \Pi + ke$ or $\mathcal{M}(N, A) \in \Pi + ke$, respectively. Given a graph class Π, we refer to all problems described above as *constrained BNSL problems for* Π. For a constrained BNSL problem we refer to the constraint on $\mathcal{S}$ or $\mathcal{M}$ as *sparsity constraint*, since in our application we aim to learn sparse network structures. Given an instance I of a constrained BNSL problem for some Π, we call the requested arc set A a *solution* of I.

Consider the input representation for constrained BNSL problems. Recall that the family of local scores $\mathcal{F}$ for a vertex set $N := \{v_1, \ldots, v_n\}$ is given in *non-zero representation*. That is, $\mathcal{F}$ is given as a two-dimensional array $\mathcal{F} := [Q_1, \ldots, Q_n]$ where each Q_i is an array containing all triples $(f_{v_i}(P), |P|, P)$ where $f_{v_i}(P) > 0$ or $P \neq \emptyset$. As the size of an instance $I := (N, \mathcal{F}, t, k)$ of a constrained BNSL problem, we define $|I| := n + |\mathcal{F}| + \log(t) + \log(k)$, where $|\mathcal{F}|$ denotes the number of bits needed to store the two-dimensional array representing $\mathcal{F}$.

In this work we also refer to the so-called superstructures. The *directed superstructure* and the *undirected superstructure* are auxiliary graphs that can be associated with an instance I of a constrained BNSL problem. These graphs can be used as tools for designing algorithms for BNSL problems [141, 65]. They are defined as follows.

Definition 6.1. *Let N be a vertex set with local scores $\mathcal{F}$. The* directed superstructure *of N and $\mathcal{F}$ is the directed graph $S_{\vec{\mathcal{F}}} := (N, A_{\mathcal{F}})$ with*

$$A_{\mathcal{F}} := \{(u, v) \mid \exists P \subseteq N \setminus \{v\} : f_v(P) > 0 \wedge u \in P\}.$$

The undirected superstructure *is the skeleton of the directed superstructure.*

Note that there exists an arc $(u, v) \in A_{\mathcal{F}}$ if and only if the two-dimensional array representing $\mathcal{F}$ contains an entry $(f_v(P), |P|, P)$ with $u \in P$. Given N and $\mathcal{F}$, the directed superstructure and the undirected superstructure can be constructed in linear time.

6.1.2 Basic Observations

Note that if Π is monotone and contains infinitely many graphs and $k = n$, then the sparsity constraints $\mathcal{S}(N, A) \in \Pi + kv$ and $\mathcal{M}(N, A) \in \Pi + kv$ always hold, since the empty graph belongs to Π. Moreover, if Π is monotone and contains infinitely many graphs and $k = n^2$, then the sparsity constraints $\mathcal{S}(N, A) \in \Pi + ke$ and $\mathcal{M}(N, A) \in \Pi + ke$ always hold, since every edgeless graph belongs to Π. Hence, all problems considered in this chapter are generalizations of Vanilla-BNSL and thus, the NP-hardness of Vanilla-BNSL implies the following.

Proposition 6.2. *Let Π be a monotone and infinite graph class. Then, the constrained BNSL problems for Π are NP-hard.*

We next consider the solution structure for constrained BNSL problems. Let I be a yes-instance of a constrained BNSL problem. We call a solution A of I *nice* if $f_v(P_v^A) \leq f_v(\emptyset)$ implies $P_v^A = \emptyset$. In this work, we consider constrained BNSL problems for some monotone graph classes Π. For such classes, every yes-instance has a nice solution A.

Proposition 6.3. *Let Π be a monotone graph property, and let $(N, \mathcal{F}, t, k)$ be a yes-instance of a constrained BNSL problem for Π. Then, there exists a nice solution A for $(N, \mathcal{F}, t, k)$.*

Proof. Let A be a solution for $I := (N, \mathcal{F}, t, k)$ and let $v_1, \ldots, v_\ell \in N$ be all vertices with $P_{v_i}^A \neq \emptyset$ and $f_{v_i}(P_{v_i}^A) \leq f_{v_i}(\emptyset)$. We set $A' := A \setminus \{(u, v_i) \mid u \in N, i \in \{1, \ldots, \ell\}\}$. Observe that $P_{v_i}^{A'} = \emptyset$ for all $i \in \{1, \ldots, \ell\}$. Moreover, $f_v(P_v^{A'}) \geq f_v(P_v^A)$ for every $v \in N$ and (N, A') is a DAG. Therefore, A' is $(N, \mathcal{F}, t)$-valid. Finally, since Π is monotone and $\mathcal{S}(N, A)$ (or $\mathcal{M}(N, A)$, respectively) satisfies the sparsity constraint, $\mathcal{S}(N, A')$ (or $\mathcal{M}(N, A')$, respectively) also satisfies the sparsity constraint. □

Given an instance $I := (N, \mathcal{F}, t, k)$ and some $v \in N$, we denote the *potential parent sets of* v by $\mathcal{P}_{\mathcal{F}}(v) := \{P \subseteq N \setminus \{v\} : f_v(P) > 0\} \cup \{\emptyset\}$, which are exactly the parent sets stored in $\mathcal{F}$. If Π is monotone, then we can assume by Proposition 6.3 that in a solution A of I, every vertex v has a parent set $P_v^A \in \mathcal{P}_{\mathcal{F}}(v)$. An important measurement for the running times of our algorithms is the maximum number of potential parent sets $\delta_{\mathcal{F}}$ which is formally defined by $\delta_{\mathcal{F}} := \max_{v \in N} |\mathcal{P}_{\mathcal{F}}(v)|$ [141]. Given a vertex $v \in N$, we can iterate over all potential parent sets of v and the vertices in these sets in $\mathcal{O}(\delta_{\mathcal{F}} \cdot n)$ time.

With the next proposition we introduce a normalization of the local scores such that $f_v(\emptyset) = 0$ for every vertex v. To this end, let $I := (N, \mathcal{F}, t, k)$ be an instance of a constrained BNSL problem for some monotone Π. If $\sum_{v \in N} f_v(\emptyset) \geq t$, then I is a trivial yes-instance, since the empty arc set is a solution of I. We next show that every non-trivial instance can be preprocessed in $\mathcal{O}(|\mathcal{F}|)$ time into an instance where $f_v(\emptyset) = 0$ for every vertex v.

Proposition 6.4. *Let Π be a graph class, and let $I := (N, \mathcal{F}, t, k)$ be an instance of a constrained BNSL problem for Π where $t \geq \sum_{v \in N} f_v(\emptyset)$. Then, there exist $\mathcal{F}' := \{f'_v \mid v \in N\}$ with $f'_v(\emptyset) = 0$ for every $v \in N$ and $t' \in \mathbb{N}_0$, such that an arc set A is a nice solution for I if and only if A is a nice solution for $I' := (N, \mathcal{F}', t', k)$. Furthermore, I' can be computed in $\mathcal{O}(|\mathcal{F}|)$ time.*

Proof. Let $v \in N$. We define the new local scores f'_v by setting $f'_v(P) := f_v(P) - f_v(\emptyset)$, if $f_v(P) \geq f_v(\emptyset)$, and $f'_v(P) := 0$ otherwise. Note that $f'_v(\emptyset) = 0$ for all $v \in N$. Furthermore, we set $t' := t - \sum_{v \in N} f_v(\emptyset)$. Obviously, $\mathcal{F}'$ and t' can be computed in $\mathcal{O}(|\mathcal{F}|)$ time by iterating over the two-dimensional array representing $\mathcal{F}$. Moreover, $t' \geq 0$ since $t \geq \sum_{v \in N} f_v(\emptyset)$. We next show that $A \subseteq N \times N$ is a nice solution for I if and only if A is a nice solution for I'.

($\Rightarrow$) Let A be a nice solution for I. Obviously, (N, A) is a DAG and the sparsity constraint is satisfied. Furthermore, we have

$$\text{score}_{\mathcal{F}'}(A) = \sum_{v \in N} f'_v(P_v^A) \geq \sum_{v \in N} (f_v(P_v^A) - f_v(\emptyset)) \geq t - \sum_{v \in N} f_v(\emptyset) = t'.$$

It remains to show that A is nice for I'. To this end, let $f'_v(P_v^A) \leq f'_v(\emptyset)$. We conclude $f'_v(P_v^A) = 0$ and therefore $f_v(P_v^A) \leq f_v(\emptyset)$. Since A is nice for I, we conclude $P_v^A = \emptyset$. Hence, A is a nice solution for I'.

($\Leftarrow$) Conversely, let A be nice for I'. We show that A is a nice solution for I. Obviously (N, A) is a DAG and the sparsity constraint is satisfied. Hence, it remains to show that $\text{score}_{\mathcal{F}}(A) \geq t$ and that A is nice for I.

To this end, we first show that $f'_v(P^A_v) = f_v(P^A_v) - f_v(\emptyset)$ for every $v \in N$. Assume towards a contradiction that there exists some $v \in N$ such that $f'_v(P^A_v) \neq f_v(P^A_v) - f_v(\emptyset)$. Then, by the definition of $\mathcal{F}'$ we have $f_v(P^A_v) < f_v(\emptyset)$ and $f'_v(P^A_v) = 0$. Note that $f'_v(P^A_v) = 0$ implies $f'_v(P^A_v) \leq f'_v(\emptyset)$ and therefore $P^A_v = \emptyset$ since A is nice for I'. This contradicts the fact that $f_v(P^A_v) < f_v(\emptyset)$.

Since $f'_v(P^A_v) = f_v(P^A_v) - f_v(\emptyset)$ for every $v \in N$ the sum of the local scores is

$$\text{score}_{\mathcal{F}}(A) = \sum_{v \in N} f_v(P^A_v) = \sum_{n \in N} (f'_v(P^A_v) + f_v(\emptyset)) \geq t' + \sum_{n \in N} f_v(\emptyset) = t.$$

To show that A is nice for I, let $f_v(P^A_v) \leq f_v(\emptyset)$. By the construction of $\mathcal{F}$, this implies that $f'_v(P^A_v) = 0 = f'_v(\emptyset)$. Since A is nice for I' we conclude $P^A_v = \emptyset$. Hence, A is a nice solution for I. □

6.2 BNSL with Bounded Vertex Cover Number

We first study the task of learning Bayesian network structures with a bounded vertex cover number. In the framework of constrained BNSL problems, these are the problems $(\Pi_0 + v)$-Skeleton BNSL and $(\Pi_0 + v)$-Moral BNSL, where Π_0 is the class of edgeless graphs. Note that Π_0 is monotone. Korhonen and Parviainen [113] analyzed the parameterized complexity of $(\Pi_0 + v)$-Moral BNSL parameterized by k. In their work, they provided an XP-time algorithm and proved W[1]-hardness. We adapt their approach to obtain an XP-time algorithm for $(\Pi_0 + v)$-Skeleton BNSL. Furthermore, we prove that (Π_0+v)-Skeleton BNSL and (Π_0+v)-Moral BNSL are both W[2]-hard when parameterized by k.

6.2.1 An XP-time Algorithm for Skeletons with Small Vertex Cover

The XP-time algorithm for $(\Pi_0 + v)$-Skeleton BNSL follows the basic idea of the XP-time algorithm for $(\Pi_0 + v)$-Moral BNSL [113]: First, iterate over every possible choice of the vertex cover S and then split the arc set into two parts which are the arcs between S and the parents of S and the arcs between S and the children of S. These two arc sets can be learned and combined independently.

Our algorithm for learning a network structure with bounded vertex cover number in the skeleton differs from the moralized version in one technical point, however. In the moralized graph, every vertex of a vertex cover S has at most one parent outside S. For (Π_0+v)-Moral BNSL this can be exploited to find the arcs between the vertices of S and their parents. However, this does not hold for the skeleton: Consider a star where all the arcs are directed towards the center. Here, the central vertex forms a minimum vertex cover but the vertex has many parents. In the moralized graph, such star becomes a clique and the vertex cover number is large. Throughout this chapter, we let Q denote the set of parents of S. To overcome the issue described above, we split the resulting network into three disjoint arc sets: The incoming arcs of vertices of S, the incoming arcs of parents Q of vertices of S, and the incoming arcs of the remaining vertices.

In summary, the intuitive idea behind the algorithm is to find the vertex cover S and all parent sets of vertices of S via brute force. For each choice, we compute two further arc sets and combine them all to a solution of (Π_0+v)-SKELETON BNSL. To find the incoming arcs of parents of S, we adapt a dynamic programming algorithm for VANILLA-BNSL [142, 163]. With the next two lemmas, we formalize how our solution is built from disjoint arc sets.

Lemma 6.5. *Let $(N, \mathcal{F}, t, k)$ be an instance of $(\Pi_0 + v)$-SKELETON BNSL, and let S and Q be disjoint subsets of N. Furthermore, let there be arc sets $B_1 \subseteq (Q \cup S) \times S$, $B_2 \subseteq S \times Q$, and $B_3 \subseteq S \times (N \setminus (S \cup Q))$. If $D' := (S \cup Q, B_1 \cup B_2)$ is a DAG where S is a vertex cover of $\mathcal{S}(D')$, then*

a) $D := (N, A)$ with $A := B_1 \cup B_2 \cup B_3$ is a DAG,

b) S is a vertex cover of $\mathcal{S}(D)$, and

c) $\sum_{v\in N} f_v(P_v^A) = \sum_{v\in S} f_v(P_v^{B_1}) + \sum_{v\in Q} f_v(P_v^{B_2}) + \sum_{v\in N\setminus(S\cup Q)} f_v(P_v^{B_3})$.

Proof. Consider Statement *a*). Observe that if $(v, w) \in B_3$, then w is a sink in D. Together with the fact that D' is a DAG, this implies that D is a DAG. Moreover, Statement *b*) holds, since every arc in A has at least one endpoint in S. For Statement *c*), observe that S, Q, and $(N \setminus (S \cup Q))$ form a partition of N, and thus, every $v \in N$ has incoming arcs from either B_1, B_2, or B_3. $\square$

Lemma 6.6. *Let $D := (N, A)$ be a DAG such that $S \subseteq N$ is a vertex cover in $\mathcal{S}(D)$. Then, there exists a set $Q \subseteq N \setminus S$ and arc sets $B_1 \subseteq (Q \cup S) \times S$, $B_2 \subseteq S \times Q$, and $B_3 \subseteq S \times (N \setminus (S \cup Q))$ such that*

a) B_1, B_2, and B_3 form a partition of A, and

b) every vertex in Q has a child in S.

Proof. We set $Q := \{v \in N \setminus S \mid v \text{ has a child in } S\}$. Then, every vertex in Q has a child in S by definition. Furthermore, we set $B_1 := ((Q \cup S) \times S) \cap A$, $B_2 := (S \times Q) \cap A$, and $B_3 := (S \times (N \setminus (S \cup Q))) \cap A$.

Obviously, $B_1 \cup B_2 \cup B_3 \subseteq A$, and the sets are pairwise disjoint, since S, Q, and $N \setminus (S \cup Q)$ are disjoint subsets of N. It remains to show that $B_1 \cup B_2 \cup B_3 \supseteq A$. To this end, let $(v, w) \in A$. If $w \in S$, then v has a child in S. Consequently, $v \in S \cup Q$ and therefore, $(v, w) \in B_1$. Otherwise, if $w \notin S$, then $v \in S$, since S is a vertex cover of $\mathcal{S}(D)$. Therefore, $(v, w) \in B_2 \cup B_3$. □

Intuitively, the algorithm works as follows: We iterate over all possible choices of S, Q, and B_1. Then, for each such choice, we compute B_2 and B_3 that maximize the sum of local scores for $A := B_1 \cup B_2 \cup B_3$. In the following, we describe how to compute B_2 when S, Q, and B_1 are given. This step is the main difference between this algorithm and the XP-time algorithm for $(\Pi_0 + v)$-Moral BNSL [113].

Proposition 6.7. *Let $I := (N, \mathcal{F}, t, k)$ be an instance of (Π_0+v)-*Skeleton BNSL*, and let S and Q be disjoint subsets of N. Furthermore, let $B_1 \subseteq (Q \cup S) \times S$ be an arc set such that $(Q \cup S, B_1)$ is a DAG and every $w \in Q$ has a child in S. Then, we can compute an arc set B_2 that maximizes $\sum_{v \in Q} f_v(P_v^{B_2})$ among all arc sets where $(Q \cup S, B_1 \cup B_2)$ is a DAG and $B_2 \subseteq S \times Q$ in $2^{|S|} \cdot |I|^{\mathcal{O}(1)}$ time.*

Proof. We describe a dynamic programming algorithm.

Intuition. Before we present the algorithm, we provide some intuition. Given a subset $S' \subseteq S$ and the set $Q' \subseteq Q$ containing parents of vertices in S, we want to compute an arc set $B \subseteq S' \times Q'$ such that the sum of local scores for the arc set $B_1 \cup B$ is maximized. This is done by recursively choosing a vertex $v \in S'$ that is a sink in the resulting DAG and letting all $w \in Q'$ whose only child is v choose their best possible parent set in $S' \setminus \{v\}$.

Algorithm. To describe the algorithm, we introduce some notation. Given some $w \in Q$, we let $C_w^{B_1}$ denote the set of children of w in $(S \cup Q, B_1)$. Note that $C_w^{B_1} \subseteq S$ for all $w \in Q$. Given a subset $S' \subseteq S$, we let $Q(S') := Q \cap (\bigcup_{v \in S'} P_v^{B_1})$ denote the set of parents of vertices in S' that belong to Q, and $D(S')$ denote the DAG with vertex set $S' \cup Q(S')$ and arc set $B_1 \cap ((S' \cup Q(S')) \times S')$. Furthermore, given $S' \subseteq S$ and $v \in S'$, we let

$$X(S', v) := \{w \in Q(S') \mid C_w^{B_1} \cap S' = \{v\}\}$$

denote the vertices of $Q(S')$ whose only child in S' is v. Finally, given $S' \subseteq S$ and $w \in Q$, we define $\widehat{f}_w(S') := \max_{S'' \subseteq S'} f_w(S'')$ as the best possible score for a parent set of w containing only vertices from S'. The values $\widehat{f}_w(S')$ for all $S' \subseteq S$ and $w \in Q$ can be computed in overall $2^{|S|} \cdot |I|^{\mathcal{O}(1)}$ time [142].

The dynamic programming table T has entries of the type $T[S']$ where $S' \subseteq S$. Each entry stores the score of the best possible arc set $B \subseteq S' \times Q(S')$ such that $(S \cup Q, B_1 \cup B)$ is a DAG. For one-element sets $\{v\} \subseteq S$, we set $T[\{v\}] := \sum_{w \in P_v^{B_1}} f_w(\emptyset)$. Note that, due to Proposition 6.4 we may assume that $T[\{v\}] = 0$. The recurrence to compute an entry for S' with $|S'| > 1$ is

$$T[S'] := \max_{\substack{v \in S' \\ v \text{ is a sink in } D(S')}} \left(T[S' \setminus \{v\}] + \sum_{w \in X(S',v)} \widehat{f}_w(S' \setminus \{v\}) \right).$$

The score of the best possible arc set $B_2 \subseteq S \times Q$ such that $(S \cup Q, B_1 \cup B_2)$ is a DAG can be computed by evaluating $T[S]$. The corresponding arc set can be found via traceback. The correctness proof is straightforward and thus omitted.

Running Time. Recall that all values $\widehat{f}_w(S')$ with $S' \subseteq S$ and $w \in Q$ can be computed in $2^{|S|} \cdot |I|^{\mathcal{O}(1)}$ time. The dynamic programming table has $2^{|S|}$ entries and each entry can be computed in $|I|^{\mathcal{O}(1)}$ time. Thus, the overall running time is $2^{|S|} \cdot |I|^{\mathcal{O}(1)}$ as claimed. □

We now present the XP-time algorithm for $(\Pi_0 + v)$-Skeleton BNSL. This algorithm uses the algorithm behind Proposition 6.7 as a subroutine.

Theorem 6.8. $(\Pi_0 + v)$-Skeleton BNSL *can be solved in* $(n\delta_{\mathcal{F}})^k \cdot 2^k \cdot |I|^{\mathcal{O}(1)}$ *time.*

Proof. Algorithm. Let $I := (N, \mathcal{F}, t, k)$ be an instance of $(\Pi_0 + v)$-Skeleton BNSL. The following algorithm decides whether I is a yes-instance or a no-instance: First, consider every possible choice of a vertex set S with $|S| \leq k$ forming the vertex cover of the skeleton of the resulting network. For each choice of S consider every choice of potential parent sets for the vertices of S. Let B_1 be the corresponding arc set, and let Q be the set of parents of S in (N, B_1). For each choice of S and B_1, do the following:

- Use the algorithm behind Proposition 6.7 to compute an arc set $B_2 \subseteq S \times Q$ that maximizes $\sum_{v \in Q} f_v(P_v^{B_2})$ among all arc sets where $(Q \cup S, B_1 \cup B_2)$ is a DAG.

- For every $v \in N \setminus (S \cup Q)$, compute a potential parent set that maximizes $f_v(P)$ among all potential parent sets with $P \subseteq S$. Let $B_3 \subseteq S \times (N \setminus (S \cup Q))$ be the resulting arc set.

- If $\sum_{v \in S} f_v(P_v^{B_1}) + \sum_{v \in Q} f_v(P_v^{B_2}) + \sum_{v \in N \setminus (S \cup Q)} f_v(P_v^{B_3}) \geq t$, then return *yes*.

If for none of the choices of S and B_1 the answer *yes* was returned, then return *no*.

Running Time. First, we discuss the running time of the algorithm. Since $|S| \leq k$, there are $\mathcal{O}(n^k)$ choices for S and $\mathcal{O}({\delta_{\mathcal{F}}}^k)$ choices for B_1. For each such choice, the algorithm behind Proposition 6.7 can be applied in $2^k \cdot |I|^{\mathcal{O}(1)}$ time and the choice of the parent sets of vertices in $N \setminus (S \cup Q)$ can be done in $|I|^{\mathcal{O}(1)}$ time. This gives an overall running time of $(n\delta_{\mathcal{F}})^k \cdot 2^k \cdot |I|^{\mathcal{O}(1)}$ as claimed.

Correctness. Second, we show that the algorithm returns *yes* if and only if I is a yes-instance.

($\Rightarrow$) Suppose the algorithm returns *yes*. Then, there exist disjoint subsets S and Q of N, with $|S| \leq k$ and arc sets $B_1 \subseteq (S \cup Q) \times S$, $B_2 \subseteq S \times Q$, and $B_3 \subseteq S \times N \setminus (S \cup Q)$ such that $(Q \cup S, B_1 \cup B_2)$ is a DAG. Due to Lemma 6.5, $D := (N, A)$ with $A := B_1 \cup B_2 \cup B_3$ is a DAG and S is a vertex cover of $\mathcal{S}(D)$. Moreover, $\text{score}_{\mathcal{F}}(A) \geq t$ and therefore, I is a yes-instance.

($\Leftarrow$) Let I be a yes-instance. Then, there exists an $(N, \mathcal{F}, t)$-valid arc set A such that the skeleton of $D := (N, A)$ has a vertex cover S of size at most k. By Lemma 6.6, there exists a set $Q \subseteq N \setminus S$ and arc sets $B_1 \subseteq (S \cup Q) \times S$, $B_2 \subseteq S \times Q$, and $B_3 \subseteq S \times N \setminus (S \cup Q)$ that form a partition of A. Since the algorithm iterates over all choices of S and B_1 with $|S| \leq k$, it considers S and B_1 at some point. For this choice of S and B_1, the algorithm then computes an arc set $B_2' \subseteq S \times Q$ with

$$\sum_{v \in Q} f_v(P_v^{B_2'}) \geq \sum_{v \in Q} f_v(P_v^{B_2})$$

and an arc set $B_3' \subseteq S \times N \setminus (S \cup Q)$ with

$$\sum_{v \in N \setminus (S \cup Q)} f_v(P_v^{B_3'}) \geq \sum_{v \in N \setminus (S \cup Q)} f_v(P_v^{B_3}).$$

Then, since the sum of the local scores under A is at least t, the algorithm returns *yes*. □

6.2.2 W[2]-hardness for Skeletons with Small Vertex Cover

We complement the XP-time algorithm from the previous subsection by proving W[2]-hardness of $(\Pi_0 + v)$-Skeleton BNSL. Thus, $(\Pi_0 + v)$-Skeleton BNSL is not FPT for parameter k unless W[2] = FPT. We show that the hardness also holds for the task of learning a Bayesian network where the skeleton has a a bounded dissociation number. Formally, this is $(\Pi_1 + v)$-Skeleton BNSL, with $\Pi_1 := \{G \mid G \text{ has maximum degree } 1\}$. Observe that Π_1 is monotone.

Theorem 6.9. *Let $\Pi \in \{\Pi_0, \Pi_1\}$. Then, $(\Pi + v)$-*Skeleton BNSL *is W[2]-hard for k even when the directed superstructure is a DAG, the maximum parent set size is 1, and every local score is either 1 or 0.*

Proof. We give a parameterized reduction from Set Cover which is defined as follows.

Set Cover
Input: A finite universe $U \subseteq \mathbb{N}$, a family $\mathcal{X} \subseteq 2^U$, and an integer $\ell \in \mathbb{N}$.
Question: Is there a subfamily $\mathcal{X}' \subseteq \mathcal{X}$ with $|\mathcal{F}'| \leq \ell$ and $\bigcup_{X \in \mathcal{X}'} X = U$?

Set Cover is W[2]-hard when parameterized by ℓ [38]. We first describe a parameterized reduction from Set Cover to (Π_0+v)-Skeleton BNSL and afterwards, we describe how this construction can be modified to obtain W[2]-hardness for (Π_1+v)-Skeleton BNSL.

Construction. Let $(U, \mathcal{X}, \ell)$ be an instance of Set Cover. We describe how to construct an equivalent instance $I := (N, \mathcal{F}, t, k)$ with $k = \ell$. First, we set $N := U \cup \{v_X \mid X \in \mathcal{X}\}$. Next, we define the local scores $\mathcal{F}$. All local scores are either 1 or 0. For every $u \in U$ we set $f_u(P) = 1$ if and only if $P = \{v_X\}$ for some $X \in \mathcal{X}$ that contains u. Furthermore, for every $v \in \{v_X \mid X \in \mathcal{X}\}$, we set $f_v(P) = 0$ for every P. To finish the construction, we set $k := \ell$ and $t := |U|$.

Observe that for every arc (u, v) of the directed superstructure, we have $u \in U$ and $v \in \{v_X \mid X \in \mathcal{X}\}$. Consequently, the directed superstructure is a DAG. Furthermore, by the construction of $\mathcal{F}$, the maximum parent set size is 1.

Intuition. Before we show the correctness, we provide some intuition. To obtain a score of $t = |U|$, every vertex in U has to choose one parent vertex. The chosen parent vertices correspond to the subfamily $\mathcal{X}' \subseteq \mathcal{X}$ that covers U. The vertex cover constraint on the network ensures that $\mathcal{X}'$ has size at most k.

Correctness. We show that $(U, \mathcal{X}, \ell)$ is a yes-instance of Set Cover if and only if I is a yes-instance of $(\Pi_0 + v)$-Skeleton BNSL.

($\Rightarrow$) Let $\mathcal{X}' \subseteq \mathcal{X}$ be a subfamily of size at most k that covers U. Then, for every $u \in U$, there exists some set $X^u \in \mathcal{X}'$ that contains u. We define $A := \{(v_{X^u}, u) \mid u \in U\}$ and show that A is a solution of I.

Consider the skeleton $\mathcal{S}(N, A)$. Each connected component of $\mathcal{S}(N, A)$ is either an isolated vertex or a star consisting of a central vertex from $\{v_X \mid X \in \mathcal{X}'\}$ and leaf vertices from U. Thus, (N, A) is a DAG and $\{v_X \mid X \in \mathcal{X}'\}$ is a vertex cover of the skeleton. Thus, $\mathcal{S}(N, A) \in \Pi_0 + kv$, since $|\mathcal{X}'| \leq k$. Moreover, observe that $f_u(P_u^A) = 1$ for every $u \in U$. Therefore, A is $(N, \mathcal{F}, t)$-valid.

($\Leftarrow$) Conversely, let A be an $(N, \mathcal{F}, t)$-valid arc set such that $\mathcal{S}(N, A)$ has a vertex cover of size at most k. Then, since $t = |U|$, we have $f_u(P_u^A) = 1$ for every $u \in U$.

Thus, for every $u \in U$ we have $P_u^A = \{v_X\}$ for some $X \in \mathcal{X}$ containing u. We define $\mathcal{X}' := \{X \in \mathcal{X} \mid P_u^A = \{v_X\}$ for some $u \in U\}$.

Let u be an element of the universe U. Then, $P_u^A = \{v_X\}$ for some X containing u and therefore $X \in \mathcal{X}'$. Thus, $\mathcal{X}'$ covers U. It remains to show that $|\mathcal{X}'| \leq k$. Assume towards a contradiction that $|\mathcal{X}'| > k$. Then, there exist pairwise distinct vertices $u_1, \dots, u_{k+1}$ in U and $v_1, \dots, v_{k+1}$ in $\{v_X \mid X \in \mathcal{X}'\}$ such that $(v_i, u_i) \in A$ for $i \in \{1, \dots, k+1\}$. Then, the edges $\{v_i, v_i\}$ form a matching of size $k+1$ in $\mathcal{S}(N, A)$. This contradicts the fact that $\mathcal{S}(N, A)$ has a vertex cover of size at most k.

BNSL with bounded Dissociation Number. We now explain how to modify the construction described above, to obtain W[2]-hardness for (Π_1+v)-Skeleton BNSL when parameterized by k.

In the construction, we set $N := U \cup \{v_X \mid X \in X\} \cup \{w_X \mid X \in X\}$. As above, for $u \in U$ we set $f_u(P) := 1$ if and only if $P = \{v_X\}$ for some $X \in \mathcal{X}$ containing u, and for $v \in \{v_X \mid X \in X\}$ we set $f_v(P) := 0$ for every P. Additionally, for every w_X, we set $f_{w_X}(P) := 1$ if and only if $P = \{v_X\}$. Furthermore, we set $k := \ell$ and $t := |U| + |\mathcal{X}|$

($\Rightarrow$) Let $\mathcal{X}' \subseteq \mathcal{X}$ be a subfamily with $|\mathcal{X}'| \leq k$ that covers U. We set $A := \{(v_{X^u}, u) \mid u \in U\} \cup \{(v_X, w_X) \mid X \in \mathcal{X}\}$. Then, (N, A) is a DAG and the sum of the local scores is t. Furthermore, the connected components of $\mathcal{S}(N, A)$ are isolated edges or disjoint stars with central vertex in $\{v_X \mid X \in \mathcal{X}'\}$. Then, $|\mathcal{X}'| \leq k$ implies $\mathcal{S}(N, A) \in \Pi_1 + kv$.

($\Leftarrow$) Let A be a solution of I. Again, we define $\mathcal{X}' := \{X \in \mathcal{X} \mid P_u^A = \{v_X\}$ for some $u \in U\}$, which covers U by the same arguments as above. Note that the skeleton of (N, A) contains an edge $\{v_X, w_X\}$ for every $X \in \mathcal{X}$, since the sum of local scores under A is at least t. Then, assuming $|\mathcal{X}'| > k$ implies that there exist $k+1$ vertex disjoint sets $\{u, v_X, w_X\}$ where v_X is adjacent with u and w_X in $\mathcal{S}(N, A)$. This contradicts the fact that $\mathcal{S}(N, A)$ has a dissociation set of size at most k. $\square$

Observe that for a DAG $D := (N, A)$ where each vertex has at most one parent, the skeleton $\mathcal{S}(D)$ and the moralized graph $\mathcal{M}(D)$ are the same. Thus, Theorem 6.9 also implies W[2]-hardness if the sparsity constraints are posed on the moralized graph. Note that a W[1]-hardness for $(\Pi_0 + v)$-Moral BNSL when parameterized by k has been shown [113]. Theorem 6.9 now implies a slightly stronger hardness result with an additional restriction on the maximum parent set size.

Corollary 6.10. *Let* $\Pi \in \{\Pi_0, \Pi_1\}$. *Then,* $(\Pi + v)$-Moral BNSL *is W[2]-hard for k even when the directed superstructure is a DAG, the maximum parent set size is 1, and every local score is either 1 or 0.*

Consider networks where the maximum parent set size is 1. These networks are also known as *branchings*. Learning a branching without further restrictions can be done in polynomial time [32, 69]. Due to Theorem 6.9 and Corollary 6.10, there is presumably no such polynomial-time algorithm if we add a sparsity constraint on the vertex cover size. Thus, learning a branching becomes harder if under this additional constraint.

6.3 BNSL with Bounded Dissociation Number

We now provide an algorithm for $(\Pi_1 + v)$-Moral BNSL, that is, for Bayesian network learning where the moralized graph has dissociation number at most k. By Corollary 6.10, an FPT algorithm for k is unlikely. We show that $(\Pi_1 + v)$-Moral BNSL can be solved in XP-time when parameterized by k. As detailed in the introduction, this shows that we can find optimal networks for a class of moral graphs that is larger than the ones with bounded vertex cover number, while maintaining the highly desirable property that the treewidth is bounded. In fact, graphs with dissociation number at most k have treewidth at most $k + 1$ and thus Bayesian inference can be performed efficiently if k is small [40].

Before we describe the main idea of the algorithm, we make the following simple observation about Bayesian networks whose moralized graph has a bounded dissociation number.

Proposition 6.11. *Let $D = (N, A)$ be a DAG and let $S \subseteq N$ be a dissociation set of $\mathcal{M}(D)$. Then, at most $2 \cdot |S|$ vertices in $N \setminus S$ have descendants in S.*

Proof. Let $v \in S$. We call a vertex $w \in N \setminus S$ is an *external ancestor of* v if there exists a path $(w, w_1, \dots, w_\ell, v)$ in D such that $w_i \in N \setminus S$ for all $i \in \{1, \dots, \ell\}$. We show that every vertex in S has at most two external ancestors.

First, assume that v has three distinct parents w_1, w_2, and w_3 outside S. Then, there are moral edges $\{w_1, w_2\}$, $\{w_2, w_3\}$, and $\{w_3, w_1\}$ forming a triangle outside S in $\mathcal{M}(D)$. This contradicts the fact that S is a dissociation set of $\mathcal{M}(D)$. Hence, every $v \in S$ has at most two parents outside S. Next, consider the following cases.

Case 1: $|P_v^A \setminus S| = 0$. Then, v has no external ancestors.

Case 2: $|P_v^A \setminus S| = 1$. Then, let $P_v^A \setminus S = \{w\}$. Since S is a dissociation set of $\mathcal{M}(D)$ we have $\deg_{\mathcal{M}(D)-S}(w) \leq 1$. Hence, w has at most one parent w' outside S. Moreover, since $\deg_{\mathcal{M}(D)-S}(w') \leq 1$, the vertex w' has no parent in $N \setminus S$. Therefore, v has at most two external ancestors.

Case 3: $|P_v^A \setminus S| = 2$. Then, let $P_v^A \setminus S = \{w_1, w_2\}$. Note that $\{w_1, w_2\}$ is a moral edge in $\mathcal{M}(D)$. Then, since $\deg_{\mathcal{M}(D)-S}(w_1) \leq 1$ and $\deg_{\mathcal{M}(D)-S}(w_2) \leq 1$, the vertices w_1 and w_2 do not have parents in $N \setminus S$. Therefore, v has exactly two external ancestors. □

The main idea of the algorithm for $(\Pi_1 + v)$-Moral BNSL presented in this section is closely related to XP-algorithms for $(\Pi_0 + v)$-Moral BNSL [113] and $(\Pi_0 + v)$-Skeleton BNSL (Theorem 6.8): If we know which vertices form the dissociation set S and the set Q of vertices that are the ancestors of S, the arcs of the network can be found efficiently if S and Q are small. Roughly speaking, the steps of the algorithm are to iterate over every possible choice of S and Q and then find the arc set of the resulting network respecting this choice. Finding the arc set can then be done in two steps: First, we find all the arcs between the vertices of $S \cup Q$ and afterwards, we find the remaining arcs of the network. Even though the basic idea of the algorithm is similar to algorithms for BNSL with bounded vertex cover number, several obstacles occur when considering Π_1 instead of Π_0.

First, the arcs between $S \cup Q$ and the remaining arcs of the DAG cannot be computed independently, since there might be arcs between vertices of Q and $N \setminus (Q \cup S)$. See Figure 6.1 for an example of a DAG D whose moralized graph has a dissociation set S. We overcome this obstacle by partitioning Q into two sets Q_0 and Q_1 and by considering arc sets $A_Q \subseteq (S \cup Q) \times (S \cup Q)$ that respect a specific constraint regarding this partition.

Second, the vertices in $N \setminus (S \cup Q)$ cannot choose their parent sets greedily from S, since they may also choose one parent from $N \setminus S$. Thus, we need a new technique to find this part of the network. To overcome this obstacle, we show that the parent sets of vertices in $N \setminus (S \cup Q)$ can be found in polynomial time by using an algorithm that computes a maximum matching [133].

This section is organized as follows: In Section 6.3.1, we introduce the terms of *attic arc sets* and *basement arc sets*, the two parts of the arc set that we later combine to a solution. In Section 6.3.2, we describe how to find the attic arc set and in Section 6.3.3, we describe how to find the basement arc set. Finally, in Section 6.3.4, we show how to solve $(\Pi_1 + v)$-Moral BNSL in XP-time using the previous results.

6.3.1 Attic Arc Sets and Basement Arc Sets

In this subsection we formally define *attic arc sets* and *basement arc sets.* As mentioned above, these are the two parts of the resulting network that our algorithm finds when the vertices of the dissociation set and their ancestor vertices are known. The idea behind the names *attic arc set* and *basement arc set* is that the dissociation set S forms the center of the network, the arcs between S and the ancestors of S

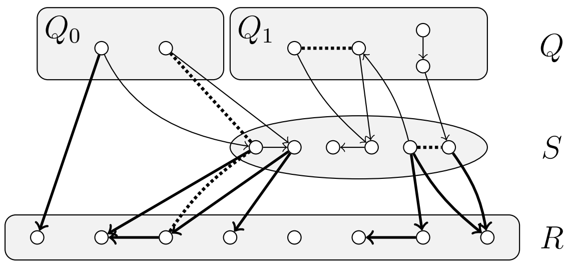


Figure 6.1: A DAG D whose moralized graph has a dissociation set S. The arc set of D is decomposed into an attic arc set A_Q and a basement arc set A_R. The thin arrows correspond to the arcs of A_Q and the thick arrows correspond to the arcs of A_R. The dotted edges are the moral edges.

form the upper part of the network (attic) and the remaining arcs form the lower part (basement) of the network. Figure 6.1 shows a DAG D where the arc set is decomposed into an attic arc set and a basement arc set.

Throughout this section, we let S denote the set of the vertices that form the dissociation set and we let Q denote the set of their ancestors. Furthermore, we assume that Q is partitioned into two sets Q_0 and Q_1. Intuitively, in the moralized graph of the resulting network, the vertices in Q_0 have no neighbors in Q and the vertices in Q_1 may have one neighbor in Q.

Definition 6.12. *Let N be a vertex set and let S, Q_0, and Q_1 be disjoint subsets of N. An arc set A_Q is called* attic arc set of S, Q_0, and Q_1, *if*

a) $D_Q := (S \cup Q_0 \cup Q_1, A_Q)$ is a DAG,

b) in the moralized graph $\mathcal{M}(D_Q)$, no vertex of Q_0 has neighbors outside S, and every vertex of Q_1 has at most one neighbor outside S.

If S, Q_0, and Q_1 are clear from the context we may refer to A_Q as *attic arc set*. Throughout this section we use the following notation as a shorthand for some vertex sets: Given a vertex set N and disjoint subsets S, Q_0, and Q_1 of N, we let $Q := Q_0 \cup Q_1$, and we let $R := N \setminus (S \cup Q)$ denote the remaining vertices of N. We next define basement arc sets.

Definition 6.13. *Let N be a vertex set and let S, Q_0, and Q_1 be disjoint subsets of N. An arc set $A_R \subseteq (S \cup R \cup Q_0) \times R$ is called* basement arc set for S, Q_0, and Q_1 *if A_R contains no self-loops and every $w \in Q_0 \cup R$ has at most one incident arc in $A_R \cap ((R \cup Q_0) \times R)$.*

If S, Q_0, and Q_1 are clear from the context we may refer to A_R as *basement arc set.* The idea is that basement arc sets and attic arc sets can be combined to a solution of $(\Pi_1 + v)$-Moral BNSL and that a solution can be splitted into an attic arc set and a basement arc set. With the next two lemmas, we formalize this idea. First, an attic arc set and a basement arc set can be combined to a DAG where S is a dissociation set of the moralized graph.

Lemma 6.14. *Let N be a vertex set and let S, Q_0 and Q_1 be disjoint subsets of N. Furthermore, let A_Q be an attic arc set, and let A_R be a basement arc set. It then holds that*

a) $D := (N, A_Q \cup A_R)$ is a DAG, and

b) S is a dissociation set of $\mathcal{M}(D)$.

Proof. We first show that D is a DAG. Assume towards a contradiction that there is a directed cycle in D. Since A_Q is an attic arc set we conclude from Definition 6.12 *a)* that there is no directed cycle in (N, A_Q). Thus, the cycle contains an edge $(v, w) \in A_R$. Note that $w \in R$ and there exists an outgoing edge $(w, w') \in A_Q \cup A_R$ that is also part of the cycle. Since no edge in A_Q is incident with vertices of R we conclude $(w, w') \in A_R$ and therefore $w' \in R$. Note that $w' \neq w$, since A_R contains no self-loops. Since (w, w') is part of the directed cycle, there exists an edge $(w', w'') \in A_R$ with $w'' \in R$. Then, w' is incident with two arcs in $A_R \cap ((R \cup Q_0) \times R)$ which is a contradiction to the fact that A_R is a basement arc set. Consequently, there is no directed cycle in D.

It remains to show that S is a dissociation set of $\mathcal{M}(D)$. That is, we show that every vertex v has degree at most 1 in $G := \mathcal{M}(D) - S$.

If $v \in Q_1$, then v has degree at most one in $\mathcal{M}(D_Q) - S$. Then, since no arc in A_R is incident with v, we have $\deg_G(v) = 1$. Otherwise, $v \in Q_0 \cup R$. Then, there is no arc in A_Q connecting v with a vertex in $N \setminus S$. Moreover, by Definition 6.13, there is at most one arc in $A_R \cap ((R \cup Q_0) \times R)$ that is incident with v. To prove $\deg_G(v) \leq 1$, it remains to show that there is no moral edge of $\mathcal{M}(D)$ connecting v with some other vertex in $N \setminus S$.

Assume towards a contradiction that there exists some $v' \in N \setminus S$ such that v and v' have a common child w. If $w \in S \cup Q$, then $v \in Q_0$ and $v' \in Q$. Consequently, $\{v, v'\}$ is a moral edge in $\mathcal{M}(D_Q)$ which contradicts the fact that vertices in Q_0 have degree 0 in $\mathcal{M}(D_Q) - S$. Hence, we conclude $w \in R$ and therefore $(v, w), (v', w) \in A_R$. Then, w has two incident arcs in $A_R \cap ((R \cup Q_0) \times R)$ which contradicts the fact that A_R is a basement arc set. Consequently, we have $\deg_G(v) \leq 1$. $\square$

Next, we show that, conversely, the arc set of every DAG whose moralized graph has a dissociation set S can be partitioned into an attic arc set and a basement arc set for some Q_0 and Q_1.

Lemma 6.15. *Let $D = (N, A)$ be a DAG, let $S \subseteq N$ be a dissociation set of $\mathcal{M}(D)$, and let $Q \subseteq N$ be the set of vertices that have at least one descendant in S. Furthermore, let $A_Q := ((S \cup Q) \times (S \cup Q)) \cap A$ and let Q be partitioned into*

$$\begin{aligned} Q_0 &:= \{v \in Q \mid v \text{ has degree } 0 \text{ in } \mathcal{M}(S \cup Q, A_Q) - S\}, \text{ and} \\ Q_1 &:= Q \setminus Q_0. \end{aligned}$$

Then, A_Q is an attic arc set and $A \setminus A_Q$ is a basement arc set. Moreover, $|Q| \leq 2|S|$.

Proof. Note that Proposition 6.11 implies $|Q| \leq 2|S|$. We first show that Properties $a)$ and $b)$ from Definition 6.12 hold for A_Q. Since $D = (N, A)$ is a DAG, $S \cup Q \subseteq N$, and $A_Q \subseteq A$, it clearly holds that D_Q is a DAG. Consequently, Property $a)$ holds. Consider Property $b)$. By the definition of Q_0, no vertex in Q_0 has neighbors in Q. Furthermore, since S is a dissociation set of $\mathcal{M}(D)$, every vertex in Q_1 has at most one neighbor in Q.

It remains to show that $A \setminus A_Q$ is a basement arc set. To this end, we first show $A \setminus A_Q \subseteq (S \cup R \cup Q_0) \times R$. Assume towards a contradiction that $A \setminus A_Q \not\subseteq (S \cup R \cup Q_0) \times R$. Consider the following cases.

Case 1: There exists an arc $(v, w) \in A \setminus A_Q$ with $w \notin R$**.** Then, $w \in S \cup Q$ and therefore, v is an ancestor of S. Hence, $(v, w) \in A_Q$ which contradicts the choice of (v, w).

Case 2: For every arc $(u, w) \in A \setminus A_Q$ we have $w \in R$**.** Then, by our assumption, there exists an arc $(v, w) \in A \setminus A_Q$ with $v \in Q_1$. Since v has degree 1 in $\mathcal{M}(N, A_Q) - S$ and an incident arc to some vertex in R we have $\deg_{\mathcal{M}(D)-S}(v) \geq 2$ which contradicts the fact that S is a dissociation set of $\mathcal{M}(D)$. Since Cases 1 and 2 are contradictory, we have $A \setminus A_Q \subseteq (S \cup R \cup Q_0) \times R$.

Finally, we show that Definition 6.13 holds for $A \setminus A_Q$. Since D is a DAG we conclude that $A \setminus A_Q$ contains no self-loops. Moreover, since S is a dissociation set of $\mathcal{M}(D)$ every $w \in (Q_0 \cup R)$ has at most one incident edge in $(A \setminus A_Q) \cap ((R \cup Q_0) \times R)$. □

In general, if we consider a union $A_1 \cup A_2$ of two disjoint arc sets, one vertex v may have incoming arcs from A_1 and A_2. Thus, for the local scores we may have $f_v(P_v^{A_1}) \neq f_v(P_v^{A_1 \cup A_2})$. Given an attic arc set A_Q and a basement arc set A_R, all arcs in A_R have endpoints in R and all arcs in A_Q have endpoints in $Q \cup S$. Since $Q \cup S$ and R are disjoint, for every vertex v either all incoming arcs are in A_Q or in A_R. Thus, the local scores under $A_Q \cup A_R$ can be decomposed as follows.

Lemma 6.16. *Let $(N, \mathcal{F}, t, k)$ be an instance of* $(\Pi_1{+}v)$-Moral BNSL *and let S, Q_0, and Q_1 be disjoint subsets of N. Furthermore, let A_Q be an attic arc set and let A_R be a basement arc set. Then, the score of $A := A_Q \cup A_R$ under $\mathcal{F}$ is*

$$\mathrm{score}_{\mathcal{F}}(A) = \sum_{v \in S \cup Q} f_v(P_v^{A_Q}) + \sum_{v \in R} f_v(P_v^{A_R}).$$

6.3.2 Finding the Attic Arc Set

Recall that the idea of the XP-time algorithm is to iterate over all possible vertices that may form the dissociation set and their possible ancestors. Then, for each choice we find an attic arc set and a basement arc set. In this subsection, we present an algorithm to efficiently compute the attic arc set when S, Q_0, and Q_1 are given.

Let $I := (N, \mathcal{F}, t, k)$ be an instance of $(\Pi_1 + v)$-Moral BNSL and let S, Q_0, and Q_1 be disjoint subsets of N. An attic arc set A_Q is called *optimal*, if the sum $\sum_{v \in S \cup Q_0 \cup Q_1} f_v(P_v^{A_Q})$ is maximal among all attic arc sets for S, Q_0, and Q_1.

Let $\lambda := |S \cup Q_0 \cup Q_1|$. Observe that, by iterating over every possible set of arcs between the vertices in $S \cup Q_0 \cup Q_1$, one can enumerate all possible A_Q in $2^{\mathcal{O}(\lambda^2)} \cdot |I|^{\mathcal{O}(1)}$ time. Alternatively, by iterating over all possible parent sets of the vertices of $S \cup Q_0 \cup Q_1$, one can enumerate all possible A_Q in ${\delta_{\mathcal{F}}}^{\lambda} \cdot |I|^{\mathcal{O}(1)}$ time. However, this might be expensive, since $\delta_{\mathcal{F}}$ can be exponentially large in the number of vertices. We show that an optimal attic arc set can be computed in $\lambda^{\mathcal{O}(\lambda)} \cdot |I|^{\mathcal{O}(1)}$ time. The intuitive idea of this algorithm is to find the connected vertex pairs in Q_1 via brute force and use an algorithm for Vanilla-BNSL as a subroutine to find the arcs of A_Q.

Proposition 6.17. *Let $I := (N, \mathcal{F}, t, k)$ be an instance of* $(\Pi_1 + v)$-Moral BNSL, *and let S, Q_0, and Q_1 be disjoint subsets of N. An optimal attic arc set for S, Q_0, and Q_1 can be computed in $\lambda^{\mathcal{O}(\lambda)} \cdot |I|^{\mathcal{O}(1)}$ time, where $\lambda := |S \cup Q_0 \cup Q_1|$.*

Proof. Throughout this proof, let $Q := Q_0 \cup Q_1$ and $N' := S \cup Q$. Consider Q_1. An *auxiliary graph* H is defined as an undirected graph with vertex set Q_1, such that each connected component of H has size at most 2. Note that there are $\binom{\lambda^2}{\lambda} \in \lambda^{\mathcal{O}(\lambda)}$ many auxiliary graphs, since $|Q_1| \leq \lambda$.

Let H be a fixed auxiliary graph. For two vertices $w_1 \in Q_1$ and $w_2 \in Q_1$, we write $w_1 \sim_H w_2$ if they belong to the same connected component of H. In the following, we define a family $\mathcal{F}^H$ of local scores for N'. To this end, we introduce the term of *feasible parent sets regarding H*: First, let $v \in Q_0$. A set $P \subseteq N' \setminus \{v\}$ is called *feasible for v* if $P \subseteq S$. Second, let $v \in Q_1$. A set $P \subseteq N' \setminus \{v\}$ is feasible for v if $P \cap Q \subseteq \{w\}$ where $w \sim_H v$. Finally, let $v \in S$. A set $P \subseteq N' \setminus \{v\}$ is *feasible for* v, if $|P \cap Q| \leq 1$, or $P \cap Q = \{w_1, w_2\}$ for some $w_1, w_2 \in Q_1$ with $w_1 \sim_H w_2$. We then define $\mathcal{F}^H$ by

$$f_v^H(P) := \begin{cases} f_v(P) & \text{if } P \text{ is feasible for } v \text{, or} \\ 0 & \text{otherwise.} \end{cases}$$

Note that for every vertex $v \in N'$, every potential parent set $P \in \mathcal{P}_{\mathcal{F}^H}(v)$ is feasible for v by the definition of $\mathcal{F}^H$.

Algorithm. The algorithm to compute an optimal attic arc set for S, Q_0, and Q_1 can be described as follows: Iterate over all auxiliary graphs H. For every choice of an auxiliary graph H compute an arc set $A_H \subseteq N' \times N'$ that maximizes $\sum_{v \in N'} f_v^H(P_v^{A_H})$. Finally, return an arc set A that maximizes $\sum_{v \in N'} f_v(P_v^A)$ among all arc sets in $\{A_H \mid H \text{ is an auxiliary graph}\}$.

Running time. We first consider the running time of the algorithm. As mentioned above, there are $\lambda^{\mathcal{O}(\lambda)}$ possible auxiliary graphs. The auxiliary graphs can be enumerated in $\lambda^{\mathcal{O}(\lambda)}$ time. For every auxiliary graph, we compute the arc set A_H. This can be done by solving Vanilla-BNSL for the vertex set N' and local scores $\mathcal{F}^H$ in $2^\lambda \cdot |I|^{\mathcal{O}(1)}$ time [142, 163]. Thus, the overall running time of the algorithm is $\lambda^{\mathcal{O}(\lambda)} \cdot |I|^{\mathcal{O}(1)}$ as claimed.

Correctness. It remains to show that the algorithm is correct. That is, the returned arc set A is an optimal attic arc set for S, Q_0, and Q_1. Note that $A = A_H$ for some auxiliary graph H. Therefore, A is a solution of an instance of Vanilla-BNSL with vertex set N' and local scores $\mathcal{F}^H$. By Proposition 6.3, we may assume that A is nice and therefore, for every $v \in N'$ the parent set P_v^A is feasible for v regarding H. Consequently, $f_v^H(P_v^A) = f_v(P_v^A)$ for every $v \in N$.

We first show that A is an attic arc set for S, Q_0, and Q_1. That is, we show that Properties *a*) and *b*) from Definition 6.12 hold. Since A is a solution of a Vanilla-BNSL instance with vertex set N', the graph (N', A) is a DAG. Thus, Property *a*) from Definition 6.12 holds. We next check Property *b*). First, consider $v \in Q_0$ and assume towards a contradiction that v has a neighbor $w \notin S$ in $\mathcal{M}(N', A)$. If $(v, w) \in A$ or $(w, v) \in A$, then either v or w has a non-feasible parent set regarding H. A contradiction. Otherwise, if $\{v, w\}$ is a moral edge, then there exists a vertex $u \in N'$ with $\{v, w\} \in P_u^A$. Then, P_u^A is not feasible for u regarding H, a contradiction. Second, consider $v \in Q_1$. Then, by the definition of feasible parent sets, v can only be adjacent to a vertex $w \in Q \setminus \{v\}$ if $v \sim_H w$. Since the connected components in H have size at most 2, v has at most one neighbor outside S in $\mathcal{M}(N', A)$. Therefore, Property *b*) from Definition 6.12 holds. Thus, A is an attic arc set.

We next show that A is optimal. That is, we show that $\sum_{S \cup Q} f_v(P_v^A)$ is maximal among all attic arc sets for Q_0, Q_1, and S. To this end, let $A' \neq A$ be another attic arc set. Consider $\mathcal{M}(N', A')$. Since every vertex in Q_1 has at most one neighbor outside S in $\mathcal{M}(N', A')$, the graph $H' := \mathcal{M}(N', A')[Q_1]$ has connected components of size at most 2. Consequently, H' is an auxiliary graph. To show that $\sum_{S \cup Q} f_v(P_v^{A'}) \leq \sum_{S \cup Q} f_v(P_v^A)$ we use the following claim.

Claim 1. *For every $n \in N'$, the parent set $P_v^{A'}$ is feasible for v regarding the auxiliary graph H'.*

Proof. We consider the following cases.

Case 1: $v \in Q_0$**.** Then, v has no neighbors outside S in $\mathcal{M}(N', A')$. Thus, v has only incoming arcs from S. Therefore, $P_v^{A'}$ is feasible for v.

Case 2: $v \in Q_1$**.** Then, v has at most one neighbor w outside S in $\mathcal{M}(N', A')$. Observe that $w \sim_{H'} v$ by the definition of H'. Therefore, $P_v^{A'}$ is feasible for v.

Case 3: $v \in S$**.** Then, if $|P_v^{A'} \cap Q| \leq 1$, $P_v^{A'}$ is feasible for v. Furthermore, if $|P_v^{A'} \cap Q| \geq 3$, the vertices in $P_v^{A'} \cap Q$ have degree at least 2 outside S in $\mathcal{M}(N', A')$ contradicting the fact that A' is an attic arc set. Thus, it remains to consider the case where $|P_v^{A'} \cap Q| = 2$. Let $P_v^{A'} \cap Q = \{w_1, w_2\}$. Then, w_1 and w_2 are connected by a moral edge in $\mathcal{M}(N', A')$ implying $w_1 \sim_{H'} w_2$. Thus, $P_v^{A'}$ is feasible for v. $\diamondsuit$

Since every $v \in N'$ has a feasible parent set under A' regarding H', we have $f_v^{H'}(P_v^{A'}) = f_v(P_v^{A'})$. Since the score of A under $\mathcal{F}^H$ is at least as big as the score of the best possible DAG under $\mathcal{F}^{H'}$, we conclude

$$\sum_{S \cup Q} f_v(P_v^{A'}) = \sum_{S \cup Q} f_v^{H'}(P_v^{A'}) \leq \sum_{S \cup Q} f_v^{H}(P_v^{A}) = \sum_{S \cup Q} f_v(P_v^{A}).$$

$\square$

6.3.3 Finding the Basement Arc Set

We now show that we can compute a basement arc set with maximal score in polynomial time if S, Q_0, and Q_1 are given. More precisely, we solve the following problem.

Basement Learning
Input: A set of vertices N, disjoint subsets S, Q_0, Q_1 of N, local scores $\mathcal{F} = \{f_v \mid v \in N\}$, and an integer t.
Question: Is there a basement arc set A_R for S, Q_0, and Q_1 with $\sum_{v \in N \setminus (S \cup Q_0 \cup Q_1)} f_v(P_v^{A_R}) \geq t$?

Proposition 6.18. Basement Learning *can be solved in $\mathcal{O}(n^3 \delta_{\mathcal{F}})$ time.*

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

Construction: Let $I := (N, S, Q_0, Q_1, \mathcal{F}, t)$ be an instance of Basement Learning. Throughout this proof let $Q := Q_0 \cup Q_1$, and let $R := N \setminus (S \cup Q)$. We construct an equivalent instance (G, ω, ℓ) of Maximum Weight Matching. We first define $G := (V, E)$ with $V := Q_0 \cup R \cup R'$, where $R' := \{v' \mid v \in R\}$, and $E := X \cup Y \cup Z$, where

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

Next, we define edge-weights $\omega : E \to \mathbb{N}$: For $e = \{v, v'\} \in Z$, we set

$$\omega(e) := \max_{S' \subseteq S} f_v(S').$$

Furthermore, for $e = \{v, w\} \in Y$ with $v \in R$ and $w \in Q_0$, we set

$$\omega(e) := \max_{S' \subseteq S} f_v(S' \cup \{w\}).$$

Finally, for $e = \{v, w\} \in X$, we set $\omega(e) := \max(\varphi(v, w), \varphi(w, v))$, where

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

To complete the construction of (G, ω, ℓ), we set $\ell := t$.

Intuition: Before we prove the correctness of the reduction we provide some intuition. A maximum-weight matching M in G corresponds to the parent sets of vertices in R and therefore to arcs in a solution A_R of I. More precisely, an edge $\{v, v'\} \in Z$ with $v \in R$ corresponds to a parent set of v that contains only vertices from S. An edge $\{v, w\} \in Y$ with $v \in R$ corresponds to a parent set of v that contains $w \in Q_0$ and vertices from S. Finally, an edge $\{v, w\} \in X$ means that either $v \in P_w^{A_R}$ or $w \in P_v^{A_R}$. An example of the construction is shown in Figure 6.2.

Correctness: We now prove the correctness of the reduction, that is, we show that I is a yes-instance of Basement Learning if and only if (G, ω, ℓ) is a yes-instance of Maximum Weight Matching.

($\Rightarrow$) Let A_R be a basement arc set of S, Q_0, and Q_1 with $\sum_{v \in R} f_v(P_v^{A_R}) \geq t$. We define a matching M with $\sum_{e \in M} \omega(e) \geq t$. To this end, we describe which edges

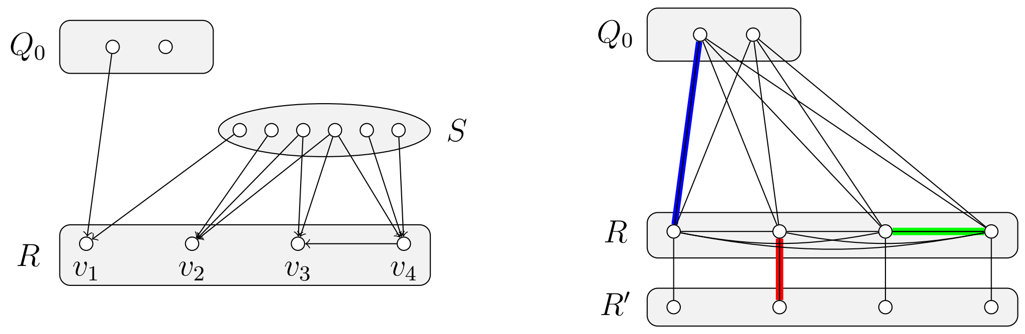


Figure 6.2: An example of the construction from the proof of Proposition 6.18. The left side shows the vertex sets Q_0, S, and R of an instance of BASEMENT LEARNING together with an optimal basement arc set. The right side shows the edges of the corresponding instance of WEIGHTED MATCHING where the edges of a solution are labeled with colors blue, red, and green. The blue edge corresponds to the parent set of v_1, the red edge corresponds to the parent set of v_2, and the green edge corresponds to the parent sets of v_3 and v_4.

of X, Y, and Z we add to M by defining sets M_X, M_Y, and M_Z and set $M := M_X \cup M_Y \cup M_Z$.

First, for every pair $v, w \in R$ with $v \in P_w^{A_R}$ or $w \in P_v^{A_R}$, we add $\{v, w\} \in X$ to M_X. Second, for every pair v, w with $v \in R$, $w \in Q_0$, and $w \in P_v^{A_R}$, we add $\{v, w\}$ to M_Y. Third, for every $v \in R$ that is not incident with one of the edges in $M_X \cup M_Y$, we add $\{v, v'\}$ to M_Z. Obviously, M_X, M_Y, and M_Z are pairwise disjoint.

We first show that M is a matching by proving that there is no pair of distinct edges in M that share an endpoint. Consider the following cases.

Case 1: $e_1, e_2 \in M_Z$**.** Then, if e_1 and e_2 share one endpoint $v \in R$ or $v' \in R'$, it follows by the definition of M_Z that $e_1 = e_2 = \{v, v'\}$. Thus, there are no distinct edges $e_1, e_2 \in M_Z$ that share one endpoint.

Case 2: $e_1, e_2 \in M_X \cup M_Y$**.** Then, assume towards a contradiction that $e_1 = \{u, v\}$ and $e_2 = \{v, w\}$ have a common endpoint v. Now, $\{u, v\} \in M_X \cup M_Y$ implies $(u, v) \in A_R$ or $(v, u) \in A_R$. Moreover $\{v, w\} \in M_X \cup M_Y$ implies $(w, v) \in A_R$ or $(v, w) \in A_R$. Then, $v \in R \cup Q_0$ is incident with two arcs in $A_R \cap ((R \cup Q_0) \times R)$ which contradicts the fact that A_R is a basement arc set.

Case 3: $e_1 \in M_X \cup M_Y, e_2 \in M_Z$**.** Then, e_1 and e_2 can only have a common endpoint in R which is not possible by the definition of M_Z.

By the above, M is a matching. It remains to show that $\sum_{e \in M} \omega(e) \geq t$. Observe that every $v \in R$ is incident with some edge in M. Conversely, every edge in $M_Y \cup M_Z$

has exactly one endpoint in R, and every edge in M_X has both endpoints in R. Given an edge $e \in M_Y \cup M_Z$, we let $\pi(e)$ denote its unique endpoint in R. By the construction of M_X and the fact that A_R is a basement arc set we know that for every $\{v, w\} \in M_X$ it holds that either $(v, w) \in A_R$ or $(w, v) \in A_R$. We let $\pi_1(e)$ and $\pi_2(e)$ denote the endpoints of $e = \{v, w\}$ such that $(\pi_2(e), \pi_1(e)) \in A_R$. Since every $v \in R$ is incident with some edge in M and M is a matching, the following sets form a partition of R.

$$\begin{aligned} R_1 &:= \{\pi_1(e) \mid e \in M_X\}, & R_2 &:= \{\pi_2(e) \mid e \in M_X\}, \\ R_3 &:= \{\pi(e) \mid e \in M_Y\}, & R_4 &:= \{\pi(e) \mid e \in M_Z\}. \end{aligned}$$

Observe that by the definitions of M_X, M_Y, and M_Z all $v \in R_2 \cup R_4$ have a parent set S' under A_R, where $S' \subseteq S$. Moreover, all $\pi(e) \in R_3$ have parent set $P^{A_R}_{\pi(e)} = S' \cup (e \setminus \{\pi(e)\})$ with $S' \subseteq S$, and all $\pi_1(e) \in R_1$ have parent sets $P^{A_R}_{\pi_1(e)} = S' \cup \{\pi_2(e)\}$ with $S' \subseteq S$. For the weight of M it then holds that

$$\begin{aligned} &\sum_{e \in M_X} \omega(e) + \sum_{e \in M_Y} \omega(e) + \sum_{e \in M_Z} \omega(e) \\ = &\sum_{e \in M_X} \max_{S' \subseteq S} f_{\pi_1(e)}(S' \cup \{\pi_2(e)\}) + \sum_{e \in M_X} \max_{S' \subseteq S} f_{\pi_2(e)}(S') \\ &+ \sum_{e \in M_Y} \max_{S' \subseteq S} f_{\pi(e)}(S' \cup (e \setminus \{\pi(e)\})) + \sum_{e \in M_Z} \max_{S' \subseteq S} f_{\pi(e)}(S') \\ \geq &\sum_{v \in R_1 \cup R_2 \cup R_3 \cup R_4} f_v(P^{A_R}_v) \geq t, \end{aligned}$$

and therefore $\sum_{e \in M} \omega(e) \geq t$.

($\Leftarrow$) Conversely, let $M \subseteq E$ be a matching of G with $\sum_{e \in M} \omega(e) \geq t$. Note that in G, every edge $e \in E$ has at least one endpoint in R and consequently every $e \in M$ has at least one endpoint in R. Moreover, without loss of generality we can assume that every vertex of R is incident with an edge of M: If a vertex $v \in R$ is not incident with an edge of M, then we replace M by $M' := M \cup \{\{v, v'\}\}$. Then, $\sum_{e \in M'} \omega(e) \geq t + \omega(\{v, v'\}) \geq t$ and M' is still a matching since $\deg_G(v') = 1$.

We define a set $A_R \subseteq (S \cup Q_0 \cup R) \times R$ and show that $\sum_{v \in R} f_v(P^{A_R}_v) \geq t$ and that A_R is a basement arc set. To this end, we define a parent set with vertices in $S \cup Q_0 \cup R$ for every $v \in R$. First, if v is incident with an edge $\{v, v'\} \in M \cap Z$, we set $P^{A_R}_v := \operatorname{argmax}_{S' \subseteq S} f_v(S')$. Second, if v is incident with an edge $\{v, w\} \in M \cap Y$, then $w \in Q_0$ and we set $P^{A_R}_v := \{w\} \cup \operatorname{argmax}_{S' \subseteq S} f_v(S' \cup \{w\})$. Third, it remains to define the parent sets of vertices in R that are endpoints of some edge in $M \cap X$. Let $\{v, w\} \in M \cap X$, where $\varphi(v, w) \geq \varphi(w, v)$. We then set $P_v^{A_R} := \{w\} \cup \operatorname{argmax}_{S' \subseteq S} f_v(S' \cup \{w\})$ and $P_w^{A_R} := \operatorname{argmax}_{S' \subseteq S} f_w(S')$.

We first show that A_R is a basement arc set. Obviously, A_R does not contain self-loops and no $v \in R$ has a parent in Q_1. It remains to show that every vertex in $Q_0 \cup R$ has at most one incident arc in $A_R \cap ((R \cup Q_0) \times R)$. Let $v \in Q_0 \cup R$. Assume towards a contradiction that v is incident with two distinct arcs in $A_R \cap ((R \cup Q_0) \times R)$. Then, there exists a vertex $w_1 \in Q_0 \cup R$ with $(v, w_1) \in A_R$ or $(w_1, v) \in A_R$. Moreover, there exists a vertex $w_2 \in (Q_0 \cup R) \setminus \{w_1\}$ with $(w_2, v) \in A_R$ or $(v, w_2) \in A_R$. Then, by the definition of A_R we conclude $\{v, w_1\}, \{v, w_2\} \in M$ which contradicts the fact that no two edges in M share one endpoint. We conclude that A_R is a basement arc set.

It remains to show that $\sum_{v \in R} f_v(P_v^{A_R}) \geq t$. To this end consider the following.

Claim 1.

a) If $\{v, w\} \in M \cap (Y \cup Z)$ with $v \in R$, then $\omega(\{v, w\}) = f_v(P_v^{A_R})$.

b) If $\{v, w\} \in M \cap X$, then $\omega(\{v, w\}) = f_v(P_v^{A_R}) + f_w(P_w^{A_R})$.

Proof. *a*) If $\{v, w\} \in M \cap Z$, then $w = v'$ and it follows by the definition of A_R that $f_v(P_v^{A_R}) = \max_{S' \subseteq S} f_v(S') = \omega(\{v, v'\})$. Otherwise, if $\{v, w\} \in M \cap Y$, then $w \in Q_0$ and analogously $f_v(P_v^{A_R}) = \max_{S' \subseteq S} f_v(S' \cup \{w\}) = \omega(\{v, w\})$.

b) If $\{v, w\} \in M \cap X$, then $v, w \in R$. We only consider the case $\varphi(v, w) \geq \varphi(w, v)$, since the other case is analogue. It then follows from the definition of A_R, that

$$\begin{aligned} & f_v(P_v^{A_R}) + f_w(P_w^{A_R}) \\ = & \max_{S' \subseteq S} f_v(S' \cup \{w\}) + \max_{S' \subseteq S} f_w(S') \\ = & \max(\varphi(v, w), \varphi(w, v)) = \omega(\{v, w\}). \end{aligned}$$

$\Diamond$

Now, let $\tilde{R} \subseteq R$ be the set of vertices in R that are incident with an edge in $M \cap X$. We conclude by Claim 1 and the assumption that every $v \in R$ is incident with an edge in M that

$$\begin{aligned} \sum_{v \in R} f_v(P_v^{A_R}) &= \sum_{v \in R \setminus \tilde{R}} f_v(P_v^{A_R}) + \sum_{v \in \tilde{R}} f_v(P_v^{A_R}) \\ &= \sum_{e \in M \cap (Y \cup Z)} \omega(e) + \sum_{e \in M \cap X} \omega(e) \\ &= \sum_{e \in M} \omega(e) \geq t. \end{aligned}$$

Running Time. The constructed instance of Maximum Weight Matching contains $\mathcal{O}(n)$ vertices and $\mathcal{O}(n^2)$ edges. For each edge e, the edge weight $\omega(e)$ can be computed in $\mathcal{O}(n\delta_\mathcal{F})$ time. Hence, we can compute the described instance of Maximum Weight Matching from an instance of Basement Learning in $\mathcal{O}(n^3\delta_\mathcal{F})$ time. Together with the fact that Maximum Weight Matching can be solved in $\mathcal{O}(\sqrt{|V|} \cdot |E|)$ time [133], we conclude that Basement Learning can be solved in $\mathcal{O}(n^3\delta_\mathcal{F})$ time. $\square$

6.3.4 An XP-time Algorithm for $(\boldsymbol{\Pi_1 + v})$-Moral BNSL

We now combine the previous results to obtain an XP-time algorithm for $(\Pi_1 + v)$-Moral BNSL. Recall that the intuitive idea of the algorithm is to find the dissociation set S and the ancestors $Q = Q_0 \cup Q_1$ of S via brute force. Then, for every choice of S, Q_0, and Q_1 we find an attic arc set and a basement arc set and combine these arc sets to a solution of $(\Pi_1 + v)$-Moral BNSL.

Theorem 6.19. $(\Pi_1 + v)$-Moral BNSL *can be solved in* $n^{3k} \cdot k^{\mathcal{O}(k)} \cdot |I|^{\mathcal{O}(1)}$ *time.*

Proof. We first describe the algorithm. Afterwards, we analyze the running time and show the correctness.

Algorithm. Let $I = (N, \mathcal{F}, t, k)$ be an instance of $(\Pi_1 + v)$-Moral BNSL. The following algorithm decides whether I is a yes-instance or a no-instance: First, iterate over all possible choices of S, Q_0, and Q_1 where $|S| \leq k$ and $|Q_0 \cup Q_1| \leq 2k$. For each such choice do the following:

- Compute an optimal attic arc set A_Q using the algorithm behind Proposition 6.17.
- Let $t' := t - \sum_{v \in S \cup Q} f_v(P_v^{A_Q})$ and check if $(N, S, Q_0, Q_1, \mathcal{F}, t')$ is a yes-instance of Basement Learning. If this is the case, return *yes*.

If for none of the choices of S, Q_0, and Q_1 the answer *yes* was returned, then return *no*.

Running Time. Since $|S| \leq k$ and $|Q_0 \cup Q_1| \leq 2k$, there are at most $\binom{n}{k} \cdot \binom{n-k}{2k} \in \mathcal{O}(n^{3k})$ choices for S and $Q := Q_0 \cup Q_1$. For each such choice we can compute all 4^k possible partitions of Q into two sets. Hence, we can iterate over all possible choices of S, Q_0 and Q_1 in $\mathcal{O}(n^{3k} \cdot 4^k)$ time. Afterwards, for each such choice we apply the algorithm behind Proposition 6.17 in $k^{\mathcal{O}(k)} \cdot |I|^{\mathcal{O}(1)}$ time, and the algorithm behind Proposition 6.18 in $\mathcal{O}(n^3\delta_\mathcal{F})$ time. This gives an overall running time of $n^{3k} \cdot k^{\mathcal{O}(k)} \cdot |I|^{\mathcal{O}(1)}$ as claimed.

Correctness. We show the correctness of the algorithm by proving that the algorithm returns yes if and only if I is a yes-instance of $(\Pi_1 + v)$-Moral BNSL.

($\Rightarrow$) Suppose that the algorithm returns yes for I. Then, there exist disjoint sets S, Q_0, and Q_1 with $|S| \leq k$, an attic arc set A_Q, and a basement arc set A_R such that

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

By Lemma 6.14, the graph $(N, A_Q \cup A_R)$ is a DAG whose moralized graph has dissociation set S and by Lemma 6.16 its score is the sum of the local scores for A_R and A_Q. Consequently, I is a yes-instance.

($\Leftarrow$) Conversely, let $(N, \mathcal{F}, t, k)$ be a yes-instance. Then, there exists an $(N, \mathcal{F}, t)$-valid arc set A and the moralized graph of (N, A) has a dissociation set S of size at most k. Then, by Lemma 6.15, there exist disjoint sets Q_0, Q_1, an attic arc set A_Q and a basement arc set A_R such that $A_R \cup A_Q = A$. Furthermore, $|Q_0 \cup Q_1| \leq 2k$. Since the algorithm iterates over all possible choices for S, Q_0, and Q_1 with $|S| \leq k$ and $|Q_0 \cup Q_1| \leq 2k$, it considers S, Q_0, and Q_1 at some point. Since A is $(N, \mathcal{F}, t)$-valid, the arc set A_R satisfies $\sum_{v \in R} f_v(P_v^{A_R}) \geq t - \sum_{v \in S \cup Q} f_v(P_v^{A_Q})$. Consequently, the algorithm returns yes. $\square$

The running time stated in Theorem 6.19 contains a factor of $k^{\mathcal{O}(k)}$. Let us remark that the constant in the exponent hidden by the $\mathcal{O}$-notation is not too high: The constant relies on the running time of the algorithm behind Proposition 6.17 where we iterate over the possible auxiliary graphs. Since $|Q_1| \leq 2k$, the number of iterations is $\mathcal{O}(\binom{(2k)^2}{2k})$. Thus, the hidden constant is 4. While it seems possible that this can be improved, it would be more interesting to determine whether the factor of $k^{\mathcal{O}(k)}$ can be replaced by $2^{\mathcal{O}(k)}$.

6.4 Constrained BNSL for Related Graph Classes

We now outline the limits of learning Bayesian networks under sparsity constraints that are related to bounded vertex cover number and bounded dissociation number. Recall that a bound of k on the vertex cover number or the dissociation number implies that the treewidth is not larger than $k + \mathcal{O}(1)$, and that is desirable for efficient inference [40]. In addition to bounded vertex cover number and bounded dissociation number we now study further graph parameters that provide a similar upper bound on the treewidth, while, for every k, the network structures satisfying this constraint form a superclass of the network structures with dissociation number at most k.

In terms of graph-classes, the bound on the vertex cover number is formalized as the graph class $\Pi_0 + kv$ and the bound on the dissociation number is formalized as the graph class $\Pi_1 + kv$. Recall that Π_0 is the class of edgeless graphs. Equivalently, Π_0 is the class of graphs with maximum degree 0, or the graphs with maximum connected component size 1. Analogously, Π_1 is the class of graphs with maximum degree 1, or the class of graphs with maximum connected component size 2. In this section, we consider two superclasses of Π_1 and show that XP-time algorithms for constrained BNSL problems regarding these superclasses are presumably not possible.

Let Π_2 be the class of graphs that have maximum degree 2, and let Π_c^{COC} be the class of graphs where each connected component has size at most c for a fixed integer $c \geq 3$. These graph classes are superclasses of Π_1, that is $\Pi_1 \subseteq \Pi_2$ and $\Pi_1 \subseteq \Pi_c^{\text{COC}}$. Consequently, if a graph G belongs to the graph class $\Pi_1 + kv$ for some $k \in \mathbb{N}_0$, then there exists some $k' \leq k$ such that $G \in \Pi_2 + k'v$ and $G \in \Pi_c^{\text{COC}} + k'v$. Moreover, observe that the treewidth of G is not bigger than $k' + \mathcal{O}(1)$. Thus, as described above, we have a similar treewidth bound as in the case of bounded dissociation number while considering a larger class of possible network structures.

With the next two theorems, we show that there is little hope that $(\Pi + v)$-Skeleton BNSL or $(\Pi + v)$- Moral BNSL with $\Pi \in \{\Pi_2, \Pi_3^{\text{COC}}\}$ has an XP-time algorithm when parameterized by k. To prove the result for Π_2, we use a reduction from Hamiltonian Path. This construction was already used to show that BNSL is NP-hard if one adds the restriction that the resulting network must be a directed path [131]. In the following we show that it also works for $(\Pi_2 + v)$-Skeleton BNSL and $(\Pi_2 + v)$-Moral BNSL.

Theorem 6.20. *$(\Pi_2 + v)$-Skeleton BNSL and $(\Pi_2 + v)$-Moral BNSL are NP-hard even if $k = 0$ and the maximum parent set size is 1.*

Proof. We give a polynomial-time reduction from the NP-hard Hamiltonian Path problem to $(\Pi_2 + v)$-Skeleton BNSL. Afterwards we show that the reduction is also correct for $(\Pi_2 + v)$-Moral BNSL. In Hamiltonian Path one is given an undirected graph G and the question is whether G has a *Hamiltonian path*, that is, a path which contains every vertex of G exactly once.

Construction. Let $G = (V, E)$ be an instance of Hamiltonian Path with n vertices. We describe how to construct an equivalent instance of $(\Pi_2 + v)$-Skeleton BNSL where $k = 0$. We first set $N := V$. Next, for every $v \in N$ we set $f_v(\{w\}) = 1$ if $w \in N_G(v)$ and $f_v(P) = 0$ for every other $P \subseteq N \setminus \{v\}$. Finally, we set $t := n - 1$ and $k := 0$.

Correctness. We show that G is a yes-instance of Hamiltonian Path if and only if $(N, \mathcal{F}, t, 0)$ is a yes-instance of $(\Pi_2 + v)$-Skeleton BNSL.

($\Rightarrow$) Let $P = (v_1, v_2, \dots, v_n)$ be a Hamiltonian path in G. We set $A := \{(v_i, v_{i+1}) \mid i \in \{1, \dots, n-1\}\}$ and show that A is $(N, \mathcal{F}, t)$-valid and that $\mathcal{S}(N, A)$ has maximum degree 2.

Since P is a Hamiltonian path, no vertex appears twice on P. Hence, (N, A) does not contain directed cycles. Moreover, it holds that $v_i \in N_G(v_{i-1})$ for every $i \in \{1, \dots, n-1\}$ and therefore $\text{score}_{\mathcal{F}}(A) = n - 1 = t$. Hence, A is $(N, \mathcal{F}, t)$-valid. Moreover, observe that $\mathcal{S}(N, A) = (N, \{\{v_i, v_{i+1}\} \mid i \in \{1, \dots, n-1\})$. Consequently, $\mathcal{S}(N, A)$ has maximum degree 2.

($\Leftarrow$) Conversely, let A be an $(N, \mathcal{F}, t)$-valid arc set such that $\mathcal{S}(D)$ has maximum degree at most 2, where $D := (N, A)$. Since $t = n - 1$ and every local score is either 1 or 0 we conclude that $f_v(P_v^A) = 1$ for at least $n - 1$ vertices. Then, there are at least $n - 1$ arcs in A, and thus there are at least $n - 1$ edges in $\mathcal{S}(D)$. Furthermore, by Proposition 6.3 we may assume that in D no vertex v has a non-empty parent set with score $f_v(P_v^A) = 0$. This implies that every vertex has at most one parent in D. Consequently, $\mathcal{S}(D)$ is acyclic.

Since $\mathcal{S}(D)$ is acyclic, the maximum degree is 2, and there are at least $n - 1$ edges, there is a Hamiltonian path $P := (v_1, \dots, v_n)$ in $\mathcal{S}(D)$. We show that P is a Hamiltonian path in G. Let v_i and v_{i+1} be two consecutive vertices on P. Then, either $(v_i, v_{i+1}) \in A$ or $(v_{i+1}, v_i) \in A$. By the construction of $\mathcal{F}$ we have $v_i \in N_G(v_{i+1})$ and thus, $\{v_i, v_{i+1}\} \in E$. Therefore, P is a Hamiltonian path in G.

Moralized Graph. We next argue why the construction described above is also a correct reduction from Hamiltonian Path to $(\Pi_2 + v)$-Moral BNSL.

($\Rightarrow$) For the forward direction, let P be a Hamiltonian path in G and let the arc set A be defined as above. Since every vertex has at most one incoming arc from A, the moralized graph $\mathcal{M}(N, A)$ has no moral edges and therefore $\mathcal{M}(N, A)$ and $\mathcal{S}(N, A)$ have the same set of edges. Thus, $\mathcal{M}(N, A)$ has maximum degree 2.

($\Leftarrow$) For the backwards direction, let A be an $(N, \mathcal{F}, t)$-valid arc set such that the moralized graph $\mathcal{M}(N, A)$ has maximum degree at most 2. Since the edge set of the skeleton of (N, A) is a subset of the edge set of $\mathcal{M}(N, A)$, we conclude that $\mathcal{S}(N, A)$ has maximum degree at most 2. Then, by the above argumentation, there exists a Hamiltonian path in G. □

Theorem 6.21. *Let $c \geq 3$. Then, $(\Pi_c^{\text{COC}} + v)$-*Skeleton BNSL *and $(\Pi_c^{\text{COC}} + v)$-*Moral BNSL *are NP-hard even if $k = 0$.*

Proof. We give a polynomial-time reduction from Size-c Clique Cover, which is NP-hard for every $c \geq 3$ [104], to $(\Pi_c^{\text{COC}} + v)$-Skeleton BNSL. Afterwards, we show that the reduction is also correct for $(\Pi_c^{\text{COC}} + v)$-Moral BNSL. In Size-c Clique Cover one is given an undirected graph $G = (V, E)$ and the question is whether there exists a partition $\mathcal{P} := \{K_c^i \mid i \in \{1, \dots, \frac{|V|}{c}\}\}$ of the vertex set V, where every K_c^i is a clique of size c.

Construction. Let $G = (V, E)$ be an instance of c-Clique Cover with n vertices. We describe how to construct an equivalent instance of $(\Pi_c^{\text{COC}}+v)$-Skeleton BNSL where $k = 0$. We first set $N := V$. Next, for every $v \in N$ we set $f_v(P) = 1$ if $G[P]$ is a K_{c-1} and $P \subseteq N_G(v)$. Otherwise, we set $f_v(P) = 0$. Note that $\mathcal{F}$ can be computed in polynomial time since c is a constant. Finally, we set $t := \frac{n}{c}$ and $k := 0$.

Correctness. We next show that G is a yes-instance of Size-c Clique Cover if and only if $(N, \mathcal{F}, t, 0)$ is a yes-instance of $(\Pi_c^{\text{COC}} + v)$-Skeleton BNSL.

($\Rightarrow$) Let $\mathcal{P} = \{K_c^i \mid i \in \{1, \dots, \frac{n}{c}\}\}$ be a packing of vertex-disjoint cliques of size c that cover G. For each i, let v_i be one vertex of K_c^i. We set $A := \bigcup_{i\in\{1,\dots,\frac{n}{c}\}}\{(u, v_i) \mid u \in K_c^i \setminus \{v_i\}\}$. Then, $D := (N, A)$ is a union of disjoint stars, where the arcs of each star point to the center v_i. Therefore, D is a DAG and every connected component of $\mathcal{S}(D)$ has order c. Thus, $\mathcal{S}(D) \in \Pi_c^{\text{COC}}$. Finally, observe that $f_{v_i}(P_{v_i}^A) = 1$ for each vertex v_i by the definition of $\mathcal{F}$. Consequently, $\text{score}_{\mathcal{F}}(A) \geq \frac{n}{c} = t$.

($\Leftarrow$) Conversely, let A be an $(N, \mathcal{F}, t)$-valid arc set such that every connected component of $\mathcal{S}(N, A)$ has order at most c. Since $t = \frac{n}{c}$ and the local scores are either 0 or 1, there are pairwise distinct vertices $v_1, \dots, v_{\frac{n}{c}}$ with $f_v(P_{v_i}^A) = 1$ for every $i \in \{1, \dots, \frac{n}{c}\}$. We define $\mathcal{P} := \{K^i := \{v_i\} \cup P_{v_i}^A \mid i \in \{1, \dots, \frac{n}{c}\}\}$ and show that $\mathcal{P}$ is a packing of vertex disjoint size c cliques that cover G.

By the definition of $\mathcal{F}$, each $P_{v_i}^A$ is a clique of size $c - 1$ that is completely contained in $N_G(v_i)$. Thus, every $K^i \in \mathcal{P}$ is a size-c clique in G. Next, assume towards a contradiction that there are distinct indices i and j with $K^i \cap K^j \neq \emptyset$. Then, since $v_i \neq v_j$ it follows that $K^i \cup K^j$ is a connected component of size bigger than c in $\mathcal{S}(N, A)$. This contradicts the choice of A.

Moralized Graph. We next argue why the construction described above is also a correct reduction from Size-c Clique Cover to $(\Pi_3^{\text{COC}} + v)$-Moral BNSL.

($\Rightarrow$) Let $\mathcal{P}$ be a packing of vertex disjoint size-c cliques that cover G, and let A be defined as above. Then, (N, A) is a disjoint union of stars that point to the center of v_i. Thus, $\mathcal{M}(N, A)$ is a disjoint union of cliques of size c. Consequently $\mathcal{M}(N, A) \in \Pi_c^{\text{COC}}$.

($\Leftarrow$) Let A be an $(N, \mathcal{F}, t)$-valid arc set such that every connected component of $\mathcal{M}(N, A)$ has order at most c. Since the edge set of the skeleton of (N, A) is a subset of the edge set of the moralized graph, we conclude that $\mathcal{S}(N, A) \in \Pi_c^{\text{COC}}$. Then, by the above argumentation, G is a yes-instance of c-Clique Cover. $\square$

Given a graph class Π, we have $\Pi = \Pi + 0v = \Pi + 0e$. Thus, Theorems 6.20 and 6.21 imply the following.

Corollary 6.22. *Let* $\Pi \in \{\Pi_2\} \cup \{\Pi_c^{COC} \mid c \geq 3\}$*. Then,* $(\Pi + e)$-Skeleton BNSL *and* $(\Pi + e)$-Moral BNSL *are NP-hard even if* $k = 0$*.*

6.5 BNSL with a Bounded Number of Edges

In this section, we study BNSL, where we aim to learn a network such that the skeleton or the moralized graph have a bounded number of edges. Formally, we study $(\Pi_0 + e)$-Skeleton BNSL and $(\Pi_0 + e)$-Moral BNSL, where Π_0 is the class of edgeless graphs. Clearly, Π_0 is monotone.

First, we consider $(\Pi_0 + e)$-Skeleton BNSL in Section 6.5.1. We give a simple dynamic programming algorithm that solves instances with an acyclic directed superstructure in polynomial time. Then, we provide a randomized algorithm which shows that $(\Pi_0 + e)$-Skeleton BNSL is FPT when parameterized by k.

Second, we consider $(\Pi_0 + e)$-Moral BNSL in Section 6.5.2. We observe that it has an XP-time algorithm when parameterized by k and we show that it is W[1]-hard for parameterization by $k + t$. Thus, putting the constraint of a bounded number of edges on the moralized graph makes the learning problem harder than putting a similar constraint on the skeleton.

6.5.1 $(\Pi_0 + e)$-Skeleton BNSL

We first show that $(\Pi_0 + e)$-Skeleton BNSL becomes polynomial-time solvable if the directed superstructure is a DAG. The algorithm uses dynamic programming over a topological ordering of $S_{\vec{\mathcal{F}}}$. Recall that this is an ordering $(v_1, \dots, v_n)$ of the vertices of N such that $i < j$ for every arc (v_i, v_j) of $S_{\vec{\mathcal{F}}}$.

Proposition 6.23. $(\Pi_0 + e)$-Skeleton BNSL *can be solved in* $\mathcal{O}(\delta_{\mathcal{F}} \cdot k \cdot n)$ *time if the directed superstructure is a DAG.*

Proof. Let $N := \{1, \dots, n\}$, and let $(N, \mathcal{F}, t, k)$ be an instance of $(\Pi_0 + e)$-Skeleton BNSL such that $S_{\vec{\mathcal{F}}}$ is a DAG. Without loss of generality, let $(n, n-1, \dots, 2, 1)$ be a topological ordering of $S_{\mathcal{F}}$. Hence, for every arc (a, b) of $S_{\vec{\mathcal{F}}}$ it holds that $a > b$.

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

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

and the result can then be computed by checking if $T[n,k] \geq t$. The corresponding network can be found via traceback. The correctness proof is straightforward and thus omitted. The size of T is $\mathcal{O}(n \cdot k)$ and each entry $T[i,j]$ can be computed in $\mathcal{O}(\delta_{\mathcal{F}})$ time by iterating over the at most $\delta_{\mathcal{F}}$ triples $(f_i(P), |P|, P)$ in $\mathcal{F}$ for the vertex i. Therefore, $(\Pi_0 + e)$-Skeleton BNSL can be solved in $\mathcal{O}(\delta_{\mathcal{F}} \cdot k \cdot n)$ time if $S_{\vec{\mathcal{F}}}$ is a DAG. □

We would like to remark that in Chapter 8 we provide a result that generalizes Proposition 6.23. In particular, we study a version of BNSL where a potential parent set P of a vertex v has a parent score s and a parent cost c and one aims to learn a DAG with maximum score among all DAGs where the sum of parent costs does not exceed some budget. We show that this version of BNSL can be solved in polynomial time if the directed superstructure is a DAG. Note that $(\Pi_0 + e)$-Skeleton BNSL is the special case where the parent costs are $c := |P|$ for each potential parent set P.

The dynamic programming algorithm behind Proposition 6.23 can be extended to obtain an FPT algorithm for $(\Pi_0 + e)$-Skeleton BNSL when parameterized by the number of arcs k. The algorithm is based on color coding [7]: In a Bayesian network with at most k arcs, there are at most $2k$ vertices which are endpoints of such arcs. The idea of color coding is to randomly color the vertices of N with $2k$ colors and find a solution A where all vertices that are incident with arcs of A are colored with pairwise distinct colors. By repeating this algorithm multiple times with multiple independently-chosen colorings, one increases the success probability.

To describe the color coding algorithm, we introduce some notation. Let N be a set of vertices. A function $\chi : N \to \{1, \dots, 2k\}$ is called a *coloring (of N with $2k$ colors)*. Given a color $c \in \{1, \dots, 2k\}$, we call $\chi^{-1}(c) := \{v \in N \mid \chi(v) = c\}$ the *color class of c*. For a subset $N' \subseteq N$, we let $\chi(N') := \{\chi(v) \mid v \in N'\}$, and for a subset $C \subseteq \{1, \dots, 2k\}$ we let $\chi^{-1}(C) := \bigcup_{c \in C} \chi^{-1}(c)$. In the following, we define which arc-sets our algorithm finds for a fixed coloring.

Definition 6.24. *Let N be a set of vertices and let $\chi : N \to \{1, \dots, 2k\}$ be a coloring of N. An arc set $A \subseteq N \times N$ is called* color-ordered for χ *if*

a) every color class $\chi^{-1}(c)$ contains at most one vertex v with $P_v^A \neq \emptyset$.

b) there exists an ordering $(c_1, \dots, c_{2k})$ of the colors $1, \dots, 2k$ such that every $(v, w) \in A$ satisfies $v \in \chi^{-1}(c_i)$ and $w \in \chi^{-1}(c_j)$ for some $i < j$.

Consider the following auxiliary problem.

Colored $(\Pi_0 + e)$-Skeleton BNSL
Input: A set of vertices N, local scores $\mathcal{F} = \{f_v \mid v \in N\}$, two integers $t, k \in \mathbb{N}_0$, and a coloring $\chi : N \to \{1, \dots, 2k\}$.
Question: Is there an $(N, \mathcal{F}, t)$-valid arc set $A \subseteq N \times N$ of size at most k that is color-ordered for χ?

Recall that the intuitive idea behind the color coding algorithm is to randomly color the vertices of N with $2k$ colors and find a solution A satisfying a constraint regarding the random coloring. Colored (Π_0+e)-Skeleton BNSL is the problem that we solve after we randomly choose a coloring. The correspondence between (Π_0+e)-Skeleton BNSL and its colored version is stated in the following proposition.

Proposition 6.25. *Let $I = (N, \mathcal{F}, t, k)$ be an instance of* $(\Pi_0 + e)$-Skeleton BNSL*. If I is a yes-instance of* $(\Pi_0 + e)$-Skeleton BNSL*, then there exist at least $(2k)!(2k)^{(n-2k)}$ colorings $\chi : N \to \{1, 2, \dots, 2k\}$ such that $(N, \mathcal{F}, t, k, \chi)$ is a yes-instance of* Colored $(\Pi_0 + e)$-Skeleton BNSL*.*

Proof. Let I be a yes-instance of $(\Pi_0 + e)$-Skeleton BNSL. Then, there exists an $(N, \mathcal{F}, t)$-valid arc set A with $|A| \leq k$. Observe that $|A| \leq k$ implies that at most $2k$ vertices of N are endpoints of arcs in A.

We define a set $\mathbb{X}$ of colorings of N such that $\chi \in \mathbb{X}$ if χ assigns all vertices incident with arcs of A to pairwise distinct colors. Since at most $2k$ vertices are endpoints of arcs in A, we conclude that $|\mathbb{X}| \geq (2k)!(2k)^{(n-2k)}$. Let $I' := (N, \mathcal{F}, t, k, \chi)$ be an instance of Colored $(\Pi_0 + e)$-Skeleton BNSL for some arbitrary $\chi \in \mathbb{X}$. We show that A is a solution of I'. Recall that A is $(N, \mathcal{F}, t)$-valid, so it remains to show that A is color-ordered for χ.

Since all endpoints of arcs in A have pairwise distinct colors under χ, at most one vertex in each color class $\chi^{-1}(c)$ has a non-empty parent set. Let $D' := (N', A)$ be the subgraph we obtain when removing all isolated vertices from (N, A). Then, D' has at most $2k$ vertices. Consider a topological ordering $\tau := (v_1, \dots, v_{|N'|})$ of the DAG D'. Since the vertices of N' belong to pairwise different color classes, there exists a color sequence $(c_1, \dots, c_{2k})$ where $v_i \in \chi^{-1}(c_i)$ for all $i \in \{1, \dots, |N'|\}$. Let $(v, w) \in A$. Since τ is a topological ordering of D' we have $v \in \chi^{-1}(c_i)$ and $w \in \chi^{-1}(c_j)$ for some $i < j$ Thus, A is color loyal for χ. Consequently, I' is a yes-instance of Colored $(\Pi_0 + e)$-Skeleton BNSL. □

We next show that Colored $(\Pi_0 + e)$-Skeleton BNSL parameterized by k is fixed-parameter tractable.

Proposition 6.26. Colored $(\Pi_0 + e)$-Skeleton BNSL *can be solved in $\mathcal{O}(4^k \cdot k^2 n^2 \delta_{\mathcal{F}})$ time.*

Proof. Intuition. Before we present the algorithm, we provide some intuition. Given a subset C' of colors and an integer k' we want to compute the best color-ordered arc set on $\chi^{-1}(C)$ with at most k arcs. Our algorithm builds the ordering from Definition 6.24 *b*) in a bottom-up manner by computing suffixes of the ordering. It starts with an empty ordering and then adds the next color $c \in C'$ and finds the vertex $v \in \chi^{-1}(c)$ that may choose a non-empty parent set by Definition 6.24 *a*). We then combine every possible choice of a parent set P of v with a solution on $\chi^{-1}(C \setminus \{c\})$ that has at most $k' - |P|$ arcs.

Algorithm. Let $I = (N, \mathcal{F}, t, k, \chi)$ be an instance of Colored (Π_0+e)-Skeleton BNSL and let $C := \{1, 2, \ldots, 2k\}$ denote the set of colors. We fill a dynamic programming table T with entries of type $T[C', k']$ where $C' \subseteq C$ and $k' \in \{0, 1, \ldots, k\}$. Every entry stores the maximum value of the sum $\sum_{v \in \chi^{-1}(C')} f_v(P_v^A)$ over all possible DAGs $D = (N, A)$, where $A \subseteq \chi^{-1}(C') \times \chi^{-1}(C')$ is color-ordered for χ and contains at most k' arcs. We set $T[\{c\}, k'] := \sum_{w \in \chi^{-1}(c)} f_w(\emptyset) = 0$ for every $c \in C$ and $k' \in \{0, 1, \ldots, 2k\}$. The recurrence to compute the entry for $C' \subseteq C$ with $|C'| > 1$ is

$$\begin{aligned} &T[C', k'] \\ &:= \max_{c \in C'} \max_{v \in \chi^{-1}(c)} \max_{\substack{P \in \mathcal{P}_{\mathcal{F}}(v) \\ |P| \leq k' \\ \chi(P) \subseteq C' \setminus \{c\}}} T[C' \setminus \{c\}, k' - |P|] + f_v(P) + \sum_{w \in \chi^{-1}(c) \setminus \{v\}} f_w(\emptyset). \end{aligned}$$

The result can be computed by checking if $T[C, k] \geq t$. Note that the corresponding network can be found via traceback. The correctness proof is straightforward and thus omitted.

Running Time. We next consider the running time. The size of T is $\mathcal{O}(2^{2k} \cdot k)$. Note that we may assume $\sum_{w \in \chi^{-1}(c) \setminus \{v\}} f_w(\emptyset) = 0$ by Proposition 6.4. Therefore, each entry can be computed in $\mathcal{O}(2k \cdot n^2 \cdot \delta_{\mathcal{F}})$ time by iterating over all $2k$ possible colors c, all $\mathcal{O}(n)$ vertices v in the corresponding color class, and all $\mathcal{O}(\delta_{\mathcal{F}} n)$ vertices in possible parent sets of v. Altogether, Colored $(\Pi_0 + e)$-Skeleton BNSL can be solved in $\mathcal{O}(4^k k^2 n^2 \delta_{\mathcal{F}})$ time. □

Propositions 6.25 and 6.26 give the following.

Theorem 6.27. *There exists a randomized algorithm for* (Π_0+e)-Skeleton BNSL *that, in* $\mathcal{O}((2e)^{2k} \cdot k^2 n^2 \delta_{\mathcal{F}})$ *time returns* no, *if given a no-instance and returns* yes *with probability at least* $1 - \frac{1}{e}$, *if given a yes-instance.*

Proof. Algorithm. We describe the randomized algorithm applied on an instance $I = (N, \mathcal{F}, t, k)$. Repeat the following two steps e^{2k} times independently:

1. Color the vertices of N uniformly at random with colors from the set $\{1,\dots,2k\}$. Let $\chi : N \to \{1,\dots,2k\}$ be the resulting coloring.

2. Apply the algorithm behind Proposition 6.26 to decide if $(N,\mathcal{F},t,k,\chi)$ is a yes-instance of Colored (Π_0+e)-Skeleton BNSL. If this is the case, then return *yes*.

If for none of the e^{2k} applications the answer *yes* was returned in Step 2, then return *no*.

Running Time. We first consider the running time of the algorithm. By Proposition 6.26, one application of the algorithm described above can be performed in $\mathcal{O}(2^{2k}\cdot k^2n^2\delta_{\mathcal{F}})$ time. Thus, the overall running time of the algorithm is $\mathcal{O}((2e)^{2k}\cdot k^2n^2\delta_{\mathcal{F}})$ as claimed.

Error Probability. We next analyze the error probability of the algorithm. This is a standard analysis in context of color coding [7]. Given a no-instance, there exists no $(N,\mathcal{F},t)$-valid arc set A with $|A|\le k$ and therefore the answer *no* is always returned in Step 2. Otherwise, given a yes-instance I, we conclude from Proposition 6.25 that there exist at least $(2k)!(2k)^{(n-2k)}$ colorings χ such that $(N,\mathcal{F},t,k,\chi)$ is a yes-instance of Colored (Π_0+e)-Skeleton BNSL. The probability of randomly choosing such coloring χ is at least

$$\frac{(2k)!(2k)^{(n-2k)}}{(2k)^n}\ge e^{-2k}.$$

Since we repeat the algorithm independently e^{2k} times, the probability that *no* is returned is at most

$$(1-e^{-2k})^{e^{2k}}\le (e^{-e^{-2k}})^{e^{2k}}=\frac{1}{e},$$

where the first inequality relies on the inequality $1+x\le e^x$. Consequently, our algorithm returns *yes* with probability at least $1-\frac{1}{e}$. □

By repeating the algorithm from Theorem 6.27 c times (for some constant c) and returning the arc set with maximum score among all resulting arc sets, the error probability is at most $\left(\frac{1}{e}\right)^c$. This way, one can solve (Π_0+e)-Skeleton BNSL with an arbitrarily small error probability by keeping a running time bound of $\mathcal{O}((2e)^{2k}\cdot k^2n^2\delta_{\mathcal{F}})$, since c is a constant. It is also possible to derandomize the algorithm by using standard techniques [136, 38]. This way, one adds a factor of $k^{\mathcal{O}(\log(k))}$ to the running time.

Corollary 6.28. (Π_0+e)-Skeleton BNSL *can be solved in* $(2e)^{2k}\cdot k^{\mathcal{O}(\log(k))}\cdot|I|^{\mathcal{O}(1)}$ *time.*

A bound on the number of arcs appears to be not so relevant for practical use. However, the algorithm might be useful as a heuristic upper bound: If we want to add a restricted number of dependencies to a given Bayesian network, the result of (Π_0+e)-Skeleton BNSL gives an upper bound for the profit we can expect from that modification: Consider a network structure (N, A) and one wants to add up to ℓ arcs to A to obtain an arc set that has maximum score among all arc sets A' of size at most $|A|+\ell$ with $A \subseteq A'$. An upper bound for the resulting score can be found by computing the best possible arc set with $|A|+\ell$ arcs by solving (Π_0+e)-Skeleton BNSL.

Note that the problem described above becomes polynomial-time tractable if we aim to obtain a score larger than score(A) instead of maximizing the score. This holds, since the score of A can be increased if and only if the local score of one parent set P_v^A can be increased by replacing it with a vertex set P with $P_v^A \subseteq P$ and $|P \setminus P_v^A| \leq \ell$ [141].

6.5.2 (Π_0+e)-Moral BNSL

We now study a version of BNSL where we aim to learn a network whose moralized graph has a bounded number of edges. Formally, this is the (Π_0+e)-Moral BNSL problem, where Π_0 is the class of edgeless graphs.

Observe that there is a simple XP-time algorithm that solves (Π_0+e)-Moral BNSL when parameterized by k: Let $I = (N, \mathcal{F}, t, k)$ be an instance of (Π_0+e)-Moral BNSL. If (N, A) is a Bayesian network whose moralized graph has k or less edges, then $|A| \leq k$. We can find A by iterating over all $\mathcal{O}(n^{2k})$ possible arc sets A with $|A| \leq k$. More efficiently, if we consider the directed superstructure $S_{\vec{\mathcal{F}}} = (N, A_{\mathcal{F}})$ we can instead iterate over all possible subsets $A' \subseteq A_{\mathcal{F}}$ with $|A'| \leq k$. Afterwards, we check if A' is $(N, \mathcal{F}, t)$-valid and if $\mathcal{M}(N, A') \in \Pi_0 + ke$. This implies the following.

Proposition 6.29. (Π_0+e)-BNSL *can be solved in* $m^k\cdot|I|^{\mathcal{O}(1)}$ *time where* m *denotes the number of arcs in the directed superstructure.*

To put this simple XP-time algorithm into context, we show that (Π_0+e)-BNSL is W[1]-hard when parameterized by $t+k$. Hence, there is little hope that (Π_0+e)-BNSL is FPT for $t+k$.

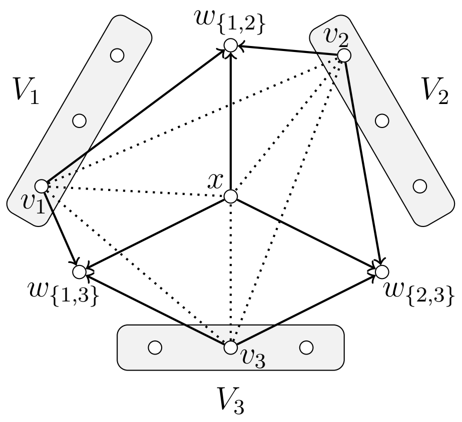


Figure 6.3: An example of the construction given in the proof of Theorem 6.30. The original instance contains a multicolored clique on the vertices $v_1 \in V_1$, $v_2 \in V_2$, and $v_3 \in V_3$. The directed edges represent the arcs of a DAG with score 3 such that the moralized graph contains 15 edges. The dotted edges correspond to the moralized edges.

Theorem 6.30. $(\Pi_0 + e)$-Moral BNSL *is W[1]-hard when parameterized by* $t + k$*, even when* $S_{\vec{\mathcal{F}}}$ *is a DAG, the maximum parent set size is 3, and every local score is either 1 or 0.*

Proof. We prove W[1]-hardness by giving a parameterized reduction from Multicolored Clique. Recall that in Multicolored Clique one is given a graph $G = (V, E)$ together with a partition $(V_1, \ldots, V_\ell)$ of V, where every V_r is an independent set. The question is if there exists a multicolored clique in G, that is, a clique containing one vertex from each set V_r. Multicolored Clique is W[1]-hard when parameterized by ℓ [147, 57].

Construction. Let $G = (V, E)$ with partition $(V_1, \ldots, V_\ell)$ be an instance of Multicolored Clique. We describe how to construct an equivalent instance $I = (N, \mathcal{F}, t, k)$ of $(\Pi_0 + e)$-Moral BNSL from G.

First, we define the vertex set N. Every vertex $v \in V$ becomes a vertex in N and for every pair $\{V_i, V_j\}$ with $i \neq j$, we add a vertex $w_{\{i,j\}}$ to N. Let W be the set of all such vertices $w_{\{i,j\}}$. Moreover, we add a vertex x to N to which we refer as the *central vertex* throughout this proof.

Second, we define the local scores $\mathcal{F}$. For every vertex $u \in V \cup \{x\}$ and every $P \subseteq N \setminus \{u\}$, we set $f_u(P) := 0$. It remains to define the local scores for the vertices in W. Let $i, j \in \{1, \ldots, \ell\}$ with $i \neq j$. We set $f_{w_{\{i,j\}}}(\{u, v, x\}) := 1$ if there is an edge $\{u, v\} \in E$ connecting a vertex $u \in V_i$ and $v \in V_j$. For all other sets P, we set $f_{w_{\{i,j\}}}(P) := 0$. Observe that the value of the local scores is either 0 or 1. We complete the reduction by setting $t := \binom{\ell}{2}$ and $k := 4\binom{\ell}{2} + \ell$. Note that $t+k \in \mathcal{O}(\ell^2)$.

Observe that the maximum parent set size is 3 and the directed superstructure $S_{\vec{\mathcal{F}}}$ is a DAG since every vertex in $V \cup \{x\}$ has in-degree 0 in $S_{\vec{\mathcal{F}}}$ and every vertex in W is a sink in $S_{\vec{\mathcal{F}}}$. Figure 6.3 shows an example of the construction.

Intuition. Before we show the correctness of the reduction, we start with some intuition. To obtain score $t = \binom{\ell}{2}$, every vertex in W must choose a parent set with score 1. Hence, every $w_{\{i,j\}}$ chooses a parent set $\{u, v, x\}$ with $u \in V_i$ and $v \in V_j$. This choice represents the choice of an edge $\{u, v\} \in E$ between the vertices u and v of a multicolored clique in G. Considering the moralized graph of the resulting Bayesian network, the bound on the number of edges gives a bound on the number of moral edges that are incident with the central vertex x. This guarantees that the chosen edges form a multicolored clique in the following sense: If the parent sets of vertices in W do not correspond to the edges of a multicolored clique in G, then the moralized graph has more than $k = 4\binom{\ell}{2} + \ell$ edges.

Correctness. We show that G is a yes-instance of Multicolored Clique if and only if $(N, \mathcal{F}, t, k)$ is a yes-instance of $(\Pi_0 + e)$-Moral BNSL.

($\Rightarrow$) Let $S := \{v_1, \dots, v_\ell\}$ with $v_i \in V_i$ for $i \in \{1, \dots, \ell\}$ be a multicolored clique in G. We define the arc set $A := \{(v_i, w_{\{i,j\}}), (v_j, w_{\{i,j\}}), (x, w_{\{i,j\}}) \mid w_{\{i,j\}} \in W\}$. We show that A is $(N, \mathcal{F}, t)$-valid and that there are at most k edges in $\mathcal{M}(N, A)$.

Since the vertices of S are pairwise adjacent in G, we have $f_{w_{\{i,j\}}}(P^A_{w_{\{i,j\}}}) = 1$ for every $w_{\{i,j\}} \in W$ and therefore $\text{score}_{\mathcal{F}}(A) = \binom{\ell}{2} = t$. Moreover, (N, A) is a DAG since $S_{\vec{\mathcal{F}}}$ is a DAG. Thus, A is $(N, \mathcal{F}, t)$-valid.

It remains to check that there are at most $k = 4 \cdot \binom{\ell}{2} + \ell$ edges in $\mathcal{M}(N, A)$. First, we consider the number of arcs in (N, A). Since every vertex in W has three parents in (N, A), we conclude $|A| = 3 \cdot \binom{\ell}{2}$. Next, we consider the moral edges in $\mathcal{M}(N, A)$. Let $a, b \in N$ be two vertices that have a common child in (N, A). Observe that all vertices in $V \setminus S \cup W$ have out-degree 0 in (N, A). We conclude that $a \in S \cup \{x\}$ and $b \in S \cup \{x\}$. Then, there are at most $|\{\{a, b\} \mid a \in S, b \in S \cup \{x\}\}| = \binom{\ell}{2} + \ell$ moral edges. Hence, there are at most $k = 4 \cdot \binom{\ell}{2} + \ell$ edges in $\mathcal{M}(N, A)$.

($\Leftarrow$) Conversely, let $A \subseteq N \times N$ be an $(N, \mathcal{F}, t)$-valid arc set such that the moralized graph $\mathcal{M}(N, A)$ has at most $k = 4\binom{\ell}{2} + \ell$ edges. We show that there exists a multicolored clique S in G.

Since A is $(N, \mathcal{F}, t)$-valid, we know that $\text{score}_{\mathcal{F}}(A) = \binom{\ell}{2}$ and thus $f_{w_{\{i,j\}}}(P^A_{w_{\{i,j\}}}) = 1$ for every $w_{\{i,j\}} \in W$. By the construction of $\mathcal{F}$ this implies $|P^A_{w_{\{i,j\}}}| = 3$ for every $w_{\{i,j\}} \in W$. We conclude $|A| = 3\binom{\ell}{2}$. Hence, there are at most $\binom{\ell}{2} + \ell$ moral edges in $\mathcal{M}(N, A)$.

Before we define the multicolored clique S, we take a closer look at the moral edges that are incident with vertices of $V_1, \dots, V_\ell$. Let V_i and V_j be distinct color classes. Then, since $P^A_{w_{\{i,j\}}}$ contains one vertex from V_i and one vertex from V_j, there exists a moral edge between the vertices of V_i and V_j. Hence, there are at least $\binom{\ell}{2}$ moral edges between the sets $V_1, \dots, V_\ell$. Now, since the overall number of moral edges in $\mathcal{M}(N, A)$ is at most $\binom{\ell}{2} + \ell$, we may conclude that there are at most ℓ moral edges that are incident with the central vertex x. We use the following claim to define a multicolored clique S in G.

Claim 1. *For every class V_i it holds that $|E_{\mathcal{M}(N,A)}(V_i, \{x\})| = 1$.*

Proof. Consider some class V_i. Note that there is no arc in A connecting x with some vertices in V_i. So, $E_{\mathcal{M}(N,A)}(V_i, \{x\})$ contains only moral edges. For every $j \in \{1, \dots, \ell\}$ with $j \neq i$, the vertex $w_{\{i,j\}}$ has a parent set $P^A_{w_{\{i,j\}}}$ containing some $v \in V_i$, $u \in V_j$ and x. Then, there exist moral edges $\{u, v\}$, $\{v, x\}$, and $\{u, x\}$. Therefore, every class contains a vertex that is adjacent to x by a moral edge. Since there are at most ℓ moral edges incident with x, we conclude $|E_{\mathcal{M}(N,A)}(V_i, \{x\})| = 1$. $\Diamond$

We now define $S := \{v_1, v_2, \dots, v_\ell\}$, where v_i is the unique element contained in $E_{\mathcal{M}(N,A)}(V_i, \{x\})$. Note that this implies $P^A_{w_{\{i,j\}}} = \{v_i, v_j, x\}$ for all $v_i, v_j \in S$ with $i \neq j$. We show that S is a multicolored clique in G. Obviously, the vertices of S are elements of distinct color classes. Thus, it remains to show that the vertices in S are pairwise adjacent in G. Let $v_i, v_j \in S$ with $i \neq j$. Then, $P^A_{w_{\{i,j\}}} = \{v_i, v_j, x\}$ and since $f_{w_{\{i,j\}}}(P^A_{w_{\{i,j\}}}) = 1$ it follows from the construction of $\mathcal{F}$ that there is an edge $\{v_i, v_j\} \in E$. Hence, S is a multicolored clique in G. $\square$

6.6 BNSL with a Bounded Feedback Edge Set

We now provide a first step into the study of the parameterized complexity of learning a Bayesian network whose moralized graph has a feedback edge set of bounded size. Formally, this is the $(\Pi_F + e)$-Moral BNSL problem, where Π_F is the class of forests. Recall that for efficient inference it is desirable to have a small treewidth in the moralized graph [40]. As all other parameters considered in this work, the size of a feedback edge set is an upper bound for the treewidth. Thus, learning a network where the moralized graph has a bounded feedback edge set is motivated from a practical point of view.

Before we consider $(\Pi_F + e)$-Moral BNSL, we briefly discuss $(\Pi_F + e)$-Skeleton BNSL. When $k = 0$, this is the problem of learning a Bayesian network with an acyclic skeleton, also known as polytree. Finding an optimal polytree is NP-hard even on instances with maximum parent set size 2 [41]. Consequently, $(\Pi_F + e)$-Skeleton BNSL is NP-hard even if $k = 0$ is fixed. If we consider the moralized graph instead of the skeleton, the case $k = 0$ can be solved efficiently. This can be seen as follows. Let $D := (N, A)$ be a DAG such that $\mathcal{M}(D)$ is acyclic. Then, each $v \in N$ has at most one parent in D, since otherwise $\mathcal{M}(D)$ contains a triangle. Thus, $D := (N, A)$ is a branching. Consequently, (Π_F+e)-Moral BNSL with $k = 0$ can be solved by computing an optimal branching which can be done in polynomial time [32, 69].

Proposition 6.31. $(\Pi_F + e)$-Moral BNSL *can be solved in polynomial time when limited to instances with* $k = 0$.

This positive result makes it interesting to study the parameterized complexity of $(\Pi_F + e)$-Moral BNSL when parameterized by k. In the following, we provide a first step into this parameterized complexity analysis and show that (Π_F+e)-Moral BNSL is W[1]-hard when parameterized by k. Thus, $(\Pi_F + e)$-Moral BNSL is presumably not FPT for k. However, an XP-time algorithm might still be possible.

Theorem 6.32. $(\Pi_F + e)$-Moral BNSL *is W[1]-hard when parameterized by k, even if $S_{\vec{\mathcal{F}}}$ is a DAG and the maximum parent set size is 4.*

Proof. We give a parameterized reduction from $(\Pi_0 + e)$-Moral BNSL parameterized by the number of edges k which is W[1]-hard even on instances where the directed superstructure is a DAG and the maximum parent set size is 3 due to Theorem 6.30.

Construction. Let $I := (N, \mathcal{F}, t, k)$ be such an instance of $(\Pi_0 + e)$-Moral BNSL. We describe how to construct an equivalent instance $I' := (N', \mathcal{F}', t', k')$ of $(\Pi_F + e)$-Moral BNSL where $k' = k$.

We define the vertex set by setting $N' := N \cup \{x\}$ for some $x \notin N$. To define the local scores $\mathcal{F}'$, we set $\ell^+ := 1 + \sum_{v\in N} \max_{P\subseteq N\setminus\{v\}} f_v(P)$. For every $v \in N$ we set $f'_v(P) = f_v(P \setminus \{x\}) + \ell^+$ if $x \in P$ and $P \setminus \{x\} \in \mathcal{P}_{\mathcal{F}}(v)$. Otherwise, we set $f'_v(P) = 0$. For the vertex x, we set $f'_x(P) = 0$ for every P. Finally, we set $t' := t + n \cdot \ell^+$.

We can obviously compute I' from I in polynomial time. Since I is an instance where $S_{\vec{\mathcal{F}}}$ is a DAG and the maximum parent set size is 3, we conclude that the maximum parent set size of I' is 4 and that $S_{\vec{\mathcal{F}'}}$ is a DAG.

Intuition. Before we prove the correctness of the reduction we provide some intuition. To obtain an $(N', \mathcal{F}', t')$-valid arc set A', the vertex x must be a parent of every vertex of N. Hence, for every $v \in N$, there exists an edge $\{x, v\}$ in $\mathcal{M}(N', A')$.

The idea is that $\mathcal{M}(N', A')$ can be transformed into an acyclic graph by deleting all edges between the vertices of N.

Correctness. We now prove that I is a yes-instance of $(\Pi_0 + e)$-Moral BNSL if and only if I' is a yes-instance of $(\Pi_F + e)$-Moral BNSL.

($\Rightarrow$) Let $A \subseteq N \times N$ be an $(N, \mathcal{F}, t)$-valid arc set such that $\mathcal{M}(D)$ for $D := (N, A)$ contains at most k edges. We then define $A' := A \cup \{(x, v) \mid v \in N\}$ and let $D' := (N', A')$. We show that A' is $(N', \mathcal{F}', t')$-valid and $\mathcal{M}(D')$ has a feedback edge set of size at most k.

We first show that A' is $(N', \mathcal{F}', t')$-valid. Since $S_{\vec{\mathcal{F}}'}$ is a DAG we conclude that D is a DAG. Moreover, $P_v^{A'} = P_v^A \cup \{x\}$ for every $v \in N$ and therefore

$$\text{score}_{\mathcal{F}'}(A') = \sum_{v \in N'} f'_v(P_v^{A'}) = \sum_{v \in N} (f_v(P_v^A) + \ell^+) = t + n \cdot \ell^+ = t'.$$

Consequently, D' is $(N', \mathcal{F}', t')$-valid. It remains to show that $\mathcal{M}(D')$ has a feedback edge set of size at most k. To this end, consider the following claim.

Claim 1. *Let $v, w \in N$. Then, $\{v, w\}$ is a moral edge in $\mathcal{M}(D)$ if and only if $\{v, w\}$ is a moral edge in $\mathcal{M}(D')$.*

Proof. Let $\{v, w\}$ be a moral edge in $\mathcal{M}(D)$. Then, there exists a vertex $u \in N$ such that $(v, u) \in A$ and $(w, u) \in A$. Since $A \subseteq A'$ we conclude that $\{v, w\}$ is a moral edge in $\mathcal{M}(D')$.

Conversely, let $\{v, w\}$ be a moral edge in $\mathcal{M}(D')$. Then, v and w have a common child u in D'. Since x has no incoming arcs, we conclude $u \in N$ and therefore $(v, u) \in A$ and $(w, u) \in A$. Thus, $\{v, w\}$ is a moral edge in $\mathcal{M}(D)$. $\Diamond$

Claim 1 together with the fact that $(N \times N) \cap A' = A$ implies that $v, w \in N$ are adjacent in $\mathcal{M}(D)$ if and only if they are adjacent in $\mathcal{M}(D')$. Hence, if we delete every edge of $\mathcal{M}(D)$ from $\mathcal{M}(D')$ we obtain the graph $G := (N', \{\{x, v\} \mid v \in N\})$ which is acyclic. Since there are at most k edges in $\mathcal{M}(D)$, we conclude that there exists a feedback edge set of size at most k for $\mathcal{M}(D')$.

($\Leftarrow$) Conversely, let A' be an $(N', \mathcal{F}', t')$-valid arc set such that $\mathcal{M}(D')$ for $D' := (N', A')$ has a feedback edge set of size at most k. We define $A := (N \times N) \cap A'$. Note that $P_v^A = P_v^{A'} \setminus \{x\}$ for every $v \in N$.

We first show that $D := (N, A)$ is $(N, \mathcal{F}, t)$-valid. Obviously, D is a DAG since $S_{\vec{\mathcal{F}}}$ is a DAG. Moreover, it holds that

$$\text{score}_{\mathcal{F}}(A) = \sum_{v \in N} f_v(P_v^A) = \sum_{v \in N} f_v(P_v^{A'} \setminus \{x\}) = t' - n \cdot \ell^+ = t.$$

Consequently, D is $(N, \mathcal{F}, t)$-valid. It remains to show that there are at most k edges in $\mathcal{M}(D)$. To this end, observe that $x \in P_v^{A'}$ for every $v \in N$: If there exists a vertex $w \in N$ with $x \notin P_w^{A'}$, then $f'_w(P_w^{A'}) = 0$ and therefore the sum of the local scores is smaller than $n \cdot \ell^+$. This contradicts the fact that A' is $(N', \mathcal{F}', t')$-valid.

Next, assume towards a contradiction that there are at least k edges in $\mathcal{M}(D)$. Then, in $\mathcal{M}(D')$ there are at least $k+1$ edges between the vertices of N. Furthermore, since $x \in P_v^{A'}$ for every $v \in N$ we conclude that every vertex in N is adjacent to x in $\mathcal{M}(D')$. Hence, $\mathcal{M}(D')$ consists of $n+1$ vertices and at least $n+k+1$ edges which contradicts the fact that $\mathcal{M}(D')$ has a feedback edge set of size at most k. □

6.7 On Problem Kernelization

We now study problem kernelization for Vanilla-BNSL and constrained BNSL problems. We show that under the standard assumption NP $\nsubseteq$ coNP/poly, Vanilla-BNSL does not admit a polynomial problem kernel when parameterized by the number of vertices. The kernel lower bound even holds for instances where all local scores are either 0 or 1. Thus, the kernel lower bound is not based on the fact that large local scores might be incompressible.

We then use the kernel lower bound for Vanilla-BNSL to complement the FPT result from Corollary 6.28 and show that there is little hope that (Π_0+e)-Skeleton BNSL admits a polynomial problem kernel.

Theorem 6.33. Vanilla-BNSL *parameterized by* $n+t$ *does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly *even when restricted to instances where all local scores are either 0 or 1.*

Proof. We describe a polynomial parameter transformation from Multicolored Independent Set. Recall that in Multicolored Independent Set one is given a graph $G = (V, E)$ together with a partition $(V_1, \dots, V_\ell)$ of V, where every V_r is an independent set. The question is, if there exists a multicolored independent set in G, that is, an independent set containing one vertex from each set V_r. Multicolored Independent Set does not admit a polynomial kernel when parameterized by $\sum_{r=1}^{\ell-1} |V_r|$ due to Proposition 1.4.

Construction. Let $G = (V, E)$ be an instance of Multicolored Independent Set with the color classes $V_1, \dots, V_\ell$. We describe how to construct an equivalent instance $(N, \mathcal{F}, t)$ of Vanilla-BNSL. First, set $N := V_1 \cup V_2 \cup \dots \cup V_{\ell-1} \cup \{x\}$ for some $x \notin V$. Second, we define the local scores $\mathcal{F}$ as follows: Let $i \in \{1, \dots, \ell-1\}$. For each $v \in V_i$ we set $f_v(P) = 1$ if $P = (V_i \setminus \{v\}) \cup (N_G(v) \setminus V_\ell) \cup \{x\}$. Otherwise, we set $f_v(P) = 0$. For x, we set $f_x(P) = 1$ if there exists some $w \in V_\ell$ with $N_G(w) = P$. Otherwise, we set $f_x(P) = 0$. Finally, we set $t := \ell$.

Observe that the value of the local scores is either 1 or 0, and that there are exactly $|V|$ values where $f_v(P) = 1$. Hence, $|\mathcal{F}| \in \mathcal{O}(|V|)$. We can obviously compute $(N, \mathcal{F}, t)$ in polynomial time from G. Furthermore, recall that $|N| = |V_1 \cup \dots \cup V_{\ell-1}| + 1$ and therefore, $n + t$ is polynomially bounded in $\sum_{r=1}^{\ell-1} |V_r|$.

Intuition: Before we show the correctness of the polynomial parameter transformation, we provide some intuition. To reach the score ℓ, exactly one vertex per color class $V_1, \dots, V_{\ell-1}$ and the vertex x must learn a parent set with score 1. The vertices from $V_1, \dots, V_{\ell-1}$ and the choice of the parent set of x then correspond to a multicolored set in G. The condition that the resulting directed graph must be a DAG guarantees that the chosen vertices form an independent set.

Correctness. We show that G contains a multicolored independent set if and only if $(N, \mathcal{F}, t)$ is a yes-instance of VANILLA-BNSL.

($\Rightarrow$) Let $S = \{v_1, \dots, v_\ell\}$ be a multicolored independent set in G with $v_i \in V_i$ for all $i \in \{1, \dots, \ell\}$. We define the arc set A by defining the parent sets of all vertices in N: For all $v \in N \setminus \{v_1, \dots, v_{\ell-1}\}$ we set $P_v^A := \emptyset$. Next, for each $v_i \in \{v_1, \dots, v_{\ell-1}\}$ we set $P_{v_i}^A := (V_i \setminus \{v_i\}) \cup (N_G(v_i) \setminus V_\ell) \cup \{x\}$. Finally, we set $P_x^A = N_G(v_\ell)$. We now prove that A is $(N, \mathcal{F}, t)$-valid. By the definition of $\mathcal{F}$, we have $f_v(P_v^A) = 1$ for every $v \in \{v_1, \dots, v_{\ell-1}, x\}$. Hence, $\text{score}_{\mathcal{F}}(A) = t$.

It remains to show that $D := (N, A)$ is a DAG. If D contains a directed cycle, all vertices on the directed cycle have incoming and outgoing arcs. Observe that $v_1, \dots, v_{\ell-1}$, and x are the only vertices with incoming arcs.

We first prove that every $v \in \{v_1, \dots, v_{\ell-1}\}$ is a sink in D. Assume towards a contradiction that there is some $v \in \{v_1, \dots, v_{\ell-1}\}$ with an outgoing arc $(v, w) \in A$. Without loss of generality, let $v = v_1$. Since $v_1 \notin P_{v_1}^A$ and only the vertices $v_1, v_2, \dots, v_{\ell-1}, x$ have parents under A, we conclude $w \in \{v_2, \dots, v_{\ell-1}, x\}$. If $w \in \{v_2, \dots, v_{\ell-1}\}$, then $v \in P_w^A$ and therefore $v \in N_G(w)$. Otherwise, if $w = x$, then $v \in P_x^A$ and therefore $v \in N_G(v_\ell)$. Both cases contradict the fact that S is an independent set in G and therefore every $v \in \{v_1, \dots, v_{\ell-1}\}$ is a sink in D.

We conclude that x is the only vertex that may have incoming and outgoing arcs in D. Then, D is a DAG since $x \notin P_x^A$.

($\Leftarrow$) Let A be an $(N, \mathcal{F}, t)$-valid arc set. We show that there exists a multicolored independent set S in G. To this end, consider the following claim.

Claim 1. *For every V_i with $i \in \{1, \dots, \ell - 1\}$ there is exactly one vertex $v_i \in V_i$ with $f_{v_i}(P_{v_i}^A) = 1$.*

Proof. Since all local scores are either 0 or 1 and since $\text{score}(A) = \ell$, there are at least $\ell - 1$ vertices of $V_1 \cup \cdots \cup V_{\ell-1}$ that have a parent set of score 1 under A. Next, assume towards a contradiction that there are distinct vertices $u \in V_i$ and $v \in V_i$ with $f_u(P_u^A) = f_v(P_v^A) = 1$ for some $i \in \{1, \dots, \ell - 1\}$. Then, by the construction of $\mathcal{F}$, we know that $V_i \setminus \{u\} \subseteq P_u^A$ and $V_i \setminus \{v\} \subseteq P_v^A$. Hence $(u, v) \in A$ and $(v, u) \in A$ which contradicts the fact that (N, A) is a DAG. $\Diamond$

By Claim 1 there are unique vertices $v_1, \dots, v_{\ell-1}$ with $v_i \in V_i$ and $f_{v_i}(P_{v_i}^A) = 1$. Furthermore, since $\text{score}(A) = \ell$ we have $f_x(P_x^A) = 1$. Then, $f_x(P_x^A) = 1$ implies that there exists a vertex $v_\ell \in V_\ell$ with $N_G(v_\ell) = P_x^A$. We define $S := \{v_1, \dots, v_{\ell-1}, v_\ell\}$ and show that S is a multicolored independent set in G.

Obviously, the vertices of S are from pairwise distinct color classes. Thus, it remains to show that no two vertices in S are adjacent in G. Assume towards a contradiction that there exist $v \in S$ and $w \in S$ such that $\{v, w\} \in E$. Without loss of generality, let $v = v_1$. Consider the following cases.

Case 1: $w \in \{v_2, \dots, v_{\ell-1}\}$**.** Then, $\{v, w\} \in E$ implies $w \in N_G(v) \setminus V_\ell$ and $v \in N_G(w) \setminus V_\ell$. Together with the fact that $f_v(P_v^A) = f_w(P_w^A) = 1$ we conclude that $(v, w) \in A$ and $(w, v) \in A$ which contradicts the fact that (N, A) is a DAG.

Case 2: $w = v_\ell$**.** Then, $\{v, w\} \in E$ implies $v \in N_G(w)$ and therefore $v \in P_x^A$. Moreover $f_v(P_v^A) = 1$ implies $x \in P_v^A$. Hence, $(v, w) \in A$ and $(w, v) \in A$ contradicting the fact that (N, A) is a DAG.

Consequently, S is a multicolored independent set in G. $\square$

We may also use Theorem 6.33 to complement the FPT result for $(\Pi_0 + e)$-Skeleton BNSL by a kernel lower bound for constrained BNSL problems. Consider an arbitrary constrained BNSL problem for some monotone and infinite graph class Π. Recall that, if $k = n^2$ the sparsity constraint always holds. Thus, the constrained BNSL problem is the same as Vanilla-BNSL on instances with $k = n^2$. Together with the kernel lower bound from Theorem 6.33, this implies the following.

Corollary 6.34. *Let* Π *be a monotone graph class that contains infinitely many graphs. Then, each constrained BNSL problem for* Π *parameterized by* $n + k + t$ *does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly.

6.8 Concluding Remarks

We have outlined the tractability borderline of Bayesian Network Structure Learning with respect to several structural constraints on the learned network or on its moralized graph. In context of this work, this chapter initialized our study of

BNSL with a broad study on constrained BNSL problems. In the remaining chapters of this work, we provide a study of further specific variants of BNSL that also fit into the framework of constrained BNSL problems.

Open Questions. The two most important concrete questions left open by our work are the following. First, can we compute an optimal network where the skeleton has dissociation number at most k in XP-time for k? In case of the moralized graph, our algorithm is based on the observation that every vertex of the dissociation set has at most two ancestors outside the dissociation set. This does not hold for the dissociation set of the skeleton. Second, can we compute an optimal network structure whose moralized graph has a feedback edge set of size at most k in XP-time for k? Observe that if the moralized graph has a feedback edge set of size most k, the network structure has at most k vertices with more than one parent. This fact might be exploited to obtain an XP algorithm. For both open questions we have shown W[1]-hardness of the corresponding problem which presumably implies that FPT algorithms do not exist. However, XP algorithms might still exist.

Another interesting topic is to compare the (parameterized) complexity of constrained BNSL problems for the skeleton and the moralized graph. We have shown that restricting the number of edges on the moralized graph may make the problem harder than putting the same constraints on the skeleton. This is somewhat counterintuitive since the moralized graph is a supergraph of the underlying undirected graph of the network. On the other side, learning a network where the moralized graph is acyclic can be done in polynomial time [32, 69], while putting the same constraint on the skeleton makes the problem NP-hard [41]. It seems interesting to investigate this issue further, that is, to understand which structural constraints make the problem harder when putting them on the moralized graph instead of the skeleton and vice versa.

When it comes to the parameterized complexity of constrained BNSL problems, one may ask for further interesting parameters besides k. A next step could be to consider parameters that have previously been studied for Vanilla-BNSL. An interesting class of possible parameters are structural parameters of the directed or undirected superstructure: recall that Vanilla-BNSL is XP when parameterized by the treewidth of the undirected superstructure [141], W[1]-hard when parameterized by the vertex cover number of the undirected superstructure, and FPT when parameterized by the feedback edge number of the undirected superstructure [65]. An open question is, whether the positive results for Vanilla-BNSL can be adapted for constrained BNSL problems. As an example, consider the problem of learning a Bayesian network structure with a feedback edge set of size at most k in the skeleton.

Recall that this problem is NP-hard even if $k = 0$ [41]. An interesting starting point might be to investigate whether this version of BNSL is FPT for the feedback edge number of the superstructure. Let ℓ be the feedback edge number of the undirected superstructure. If $k \geq \ell$, every potential solution has a feedback edge number of at most k and thus, one may use the FPT algorithm for Vanilla-BNSL parameterized by ℓ to handle this case. Furthermore, the problem is FPT for ℓ in case of $k = 0$ [65]. Thus, it might be interesting to consider cases where $0 < k < \ell$.

The superstructures can be seen as auxiliary graphs that display the structure of instances of constrained BNSL problems. It is interesting to investigate whether one can think of further such auxiliary graphs and study structural parameters of these graphs. Recall that an arc (u, v) in the directed superstructure means that u is in a potential parent set P of v. However, the directed superstructure provides no information on which other vertices are elements of P. Such information may be captured in a bipartite auxiliary graph $G = (N \cup X, E)$, where N is the set of vertices in the BNSL instance and X is the family of all potential parent sets. There is an edge $\{v, P\}$ if and only if P contains v. It might be interesting to consider structural parameters of G as possible parameterizations for constrained BNSL problems. Note that $|X|$ might be exponential in the number of vertices of the input instance, but is bounded in the encoding size of the local scores and therefore not bigger than the total size of an instance.

Finally, it is important to analyze how well the algorithms for $(\Pi_0 + v)$-Skeleton BNSL and $(\Pi_1 + v)$-Moral BNSL perform on benchmark data sets. This way, one could extend the work of Korhonen and Parviainen [113] who experimentally evaluated the performance of their XP algorithm for $(\Pi_0 + v)$-Moral BNSL. With reasonable vertex cover number bounds their algorithm scales up to instances with about 15 vertices. They also experimentally evaluate an ILP formulation for $(\Pi_0 + v)$-Moral BNSL which scales up to about 60 vertices for reasonable vertex cover bounds. Their experiments also showed that network structures with a moderate vertex cover number (for example 7 or 6) in the moralized graph tend to have higher scores than optimal branchings. Do network structures with a dissociation number s in the moralized graph have a significantly higher score than network structures with vertex cover number s in the moralized graph?

Chapter 7

Learning Optimal Polytrees

Learning Bayesian networks under sparsity constraints is a computationally hard task even on very restricted instances as we have shown in the previous chapter. One of the earliest sparsity constraints that has received attention is to require that the network structure is a *branching*. A branching is a DAG where every vertex has at most one parent. In other words, every connected component is a directed out-tree. Learning a branching and performing Bayesian inference on branchings can be done in polynomial time [32, 69, 144].

Branchings are, however, very limited in their modeling power since every variable may depend only on at most one other variable. To overcome this problem, a generalization of branchings called *polytrees* has been proposed [41]. A polytree is a DAG whose skeleton is a forest. Recall that the skeleton of a DAG is its underlying undirected graph. On the positive side, polytrees have a higher modeling power than branchings while the inference task maintains polynomial-time tractable on them [144, 84]. However, on the negative side, Polytree Learning, the problem of learning an optimal polytree structure from parent scores is NP-hard [41]. Polytrees have been used, for example, in image-segmentation for microscopy data [53]. In this chapter, we study exact algorithms for Polytree Learning.

Related Work. Polytree Learning is NP-hard even if every parent set with strictly positive score has size at most 2 [41]. Motivated by the contrast between the NP-hardness of Polytree Learning and the fact that learning a branching has a polynomial-time algorithm, the problem of optimally learning polytrees that are close to branchings has been considered. More precisely, it has been shown that the best polytree among those that can be transformed into a tree by at most k edge deletions can be found in $n^{\mathcal{O}(k)}|I|^{\mathcal{O}(1)}$ time [69, 157] where n is the number of variables and $|I|$

is the overall input size. Thus, the running time of these algorithms is polynomial for every fixed k. As noted by Gaspers et al. [69], a brute-force algorithm for POLYTREE LEARNING would need to consider $n^{n-2} \cdot 2^{n-1}$ directed trees [25].

Our Results. We obtain an algorithm that solves POLYTREE LEARNING in $3^n \cdot |I|^{\mathcal{O}(1)}$ time. This is the first algorithm for POLYTREE LEARNING where the running time is singly-exponential in the number of vertices n. This is a substantial improvement over the brute-force algorithm mentioned above, thus positively answering a question of Gaspers et al. [69] on the existence of such algorithms. We then show that POLYTREE LEARNING has—just like VANILLA-BNSL—no polynomial kernel for n unless NP $\subseteq$ coNP/poly.

We then consider a parameter that is potentially smaller than n and determine whether POLYTREE LEARNING can be solved efficiently when this parameter is small. The parameter d, which we call the number of *dependent vertices*, is the number of vertices v for which at least one entry of f_v is strictly positive. The parameter essentially counts how many variables might receive a nonempty parent set in an optimal solution. We show that POLYTREE LEARNING is XP and W[1]-hard for d. Consequently, in order to obtain FPT results for the parameter d, one needs to consider further restrictions on the structure of the input instance. We make a first step in this direction and consider the case where all parent sets with a strictly positive score have size at most p. Using this parameterization, we show that every input instance can be solved in $2^{\omega dp} \cdot |I|^{\mathcal{O}(1)}$ time where ω is the matrix multiplication constant. With the current-best known value for ω [179] this gives a running time of $5.18^{dp} \cdot |I|^{\mathcal{O}(1)}$. We then consider again data reduction approaches. This time we obtain a positive result: Any instance of POLYTREE LEARNING where p is constant can be reduced in polynomial time to an equivalent one of size $d^{\mathcal{O}(1)}$. Informally, this means that if the instance has only few dependent variables, the parent sets with strictly positive score are small, and there are many non-dependent variables, then we can identify some non-dependent variables that are irrelevant for an optimal polytree representing the input data. We note that this result is tight in the following sense: By the kernel lower bound for parameterization by n, it is presumably impossible to replace each input instance in polynomial time by an equivalent one with $(d+p)^{\mathcal{O}(1)}$ variables. Thus, the assumption that p is a constant is necessary.

Problem Definition. Given a vertex set N, recall that a family $\mathcal{F} := \{f_v : 2^{N \setminus \{v\}} \to \mathbb{N}_0 \mid v \in N\}$ is a *family of local scores for* N. Given a directed graph $D := (N, A)$ we define $\text{score}_{\mathcal{F}}(A) := \sum_{v \in N} f_v(P_v^A)$. We may omit the subscript $\mathcal{F}$ if the local scores are clear from the context. Recall that a polytree is a DAG whose skeleton is acyclic. The problem is formally defined as follows.

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

Let Π_F be the class of acyclic graphs. Then, speaking in terms of Chapter 6, Polytree Learning corresponds to $(\Pi_F + v)$-Skeleton BNSL on instances where $k = 0$. Given an instance $I := (N, \mathcal{F}, t)$ of Polytree Learning, an arc set A is a *solution of* I if (N, A) is a polytree and $\text{score}(A) \geq t$.

Analogously to the constrained BNSL problems studied in Chapter 6, we assume that the local scores $\mathcal{F}$ are given in *non-zero representation*. Thus, $f_v(P)$ is only part of the input if it is different from 0. For $N = \{v_1, \dots, v_n\}$, the local scores $\mathcal{F}$ are represented by a two-dimensional array $[Q_1, Q_2, \dots, Q_n]$, where each Q_i is an array containing all triples $(f_{v_i}(P), |P|, P)$ where $f_{v_i}(P) > 0$. The size $|\mathcal{F}|$ is defined as the number of bits needed to store this two-dimensional array. Given an instance $I := (N, \mathcal{F}, t)$, we define $|I| := n + |\mathcal{F}| + \log(t)$.

Recall that for a vertex v, the set $\mathcal{P}_{\mathcal{F}}(v) := \{P \subseteq N \setminus \{v\} \mid f_v(P) > 0\} \cup \{\emptyset\}$ is the *set of potential parents of* v. By Proposition 6.4, we may assume without loss of generality that $f_v(\emptyset) = 0$ for every $v \in N$. Given a yes-instance $I := (N, \mathcal{F}, t)$ of Polytree Learning, there exists a solution A such that $P_v^A \in \mathcal{P}_{\mathcal{F}}(v)$ for every $v \in N$ by Proposition 6.3. The running times presentend in this chapter will also be measured in the maximum number of potential parent sets $\delta_{\mathcal{F}} := \max_{v \in N} |\mathcal{P}_{\mathcal{F}}(v)|$ [141].

7.1 Parameterization by the Number of Vertices

We study the complexity of Polytree Learning when parameterized by n, the number of vertices. Note that there are up to $n \cdot 2^{n-1}$ entries in $\mathcal{F}$ and thus, the total input size of an instance of Polytree Learning might be exponential in n. On the positive side, we show that the problem can be solved in $3^n \cdot |I|^{\mathcal{O}(1)}$ time. On the negative side, we complement this FPT algorithm by proving that Polytree Learning does not admit a polynomial kernel when parameterized by n.

Theorem 7.1. Polytree Learning *can be solved in* $3^n \cdot |I|^{\mathcal{O}(1)}$ *time.*

Proof. We first describe the algorithm, and afterwards, we analyze the running time.

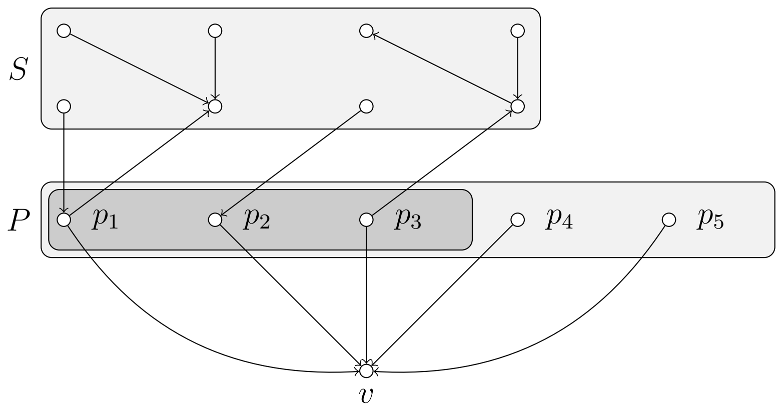


Figure 7.1: Illustration of an entry $T[v, P, S, i]$ where $i = 3$. The vertices p_4 and p_5 are only incident with the arcs (p_4, v) or (p_5, v), respectively.

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

The dynamic programming table T has entries of type $T[v, P, S, i]$ with $v \in N$, $P \in \mathcal{P}_{\mathcal{F}}(v)$, $S \subseteq N \setminus (P \cup \{v\})$, and $i \in \{0, \dots, |P|\}$. Each entry stores the maximal score of an arc set A of a polytree on $\{v\} \cup P \cup S$ where

a) v has no children,

b) v learns exactly the parent set P under A, and

c) for each $j \in \{i + 1, \dots, |P|\}$, only the arc (p_j, v) is incident with p_j under A.

Figure 7.1 shows an illustration of a table entry.

We initialize the table T by setting $T[v, P, \emptyset, i] := f_v(P)$ for all $v \in N$, $P \in \mathcal{P}_{\mathcal{F}}(v)$, and $i \in \{0, \dots, |P|\}$. Intuitively, we compute an entry for $S \neq \emptyset$ and $i \geq 1$ by considering every partition of S into S' and $S \setminus S'$ and combine the best possible polytree on the vertex set $S' \cup \{p_i\}$ with the polytree on the vertex set $P \cup (S \setminus S') \cup \{v\}$ that corresponds to a table entry with $i - 1$. Formally, the recurrence to compute an entry for $S \neq \emptyset$ and $i \geq 1$ is

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

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

In case of $i = 0$, all vertices p_j are only incident with the arc (p_j, v). Thus, the polytree corresponding to an entry $T[v, P, S, 0]$ is the disjoint union of the arc set $P \times v$ and the best polytree on S. The recurrence to compute an entry for $S \neq \emptyset$ and $i = 0$ is

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

Since the two vertex sets $P \cup \{v\}$ and S are disjoint, combining the polytree with edge set $P \times v$ and the best polytree on vertex set S results in a polytree.

The result can be computed by checking if $T[v, P, N \setminus (P \cup \{v\}), |P|] \geq t$ for some $v \in N$ and some $P \in \mathcal{P}_{\mathcal{F}}(v)$. The corresponding polytree can be found via traceback. The correctness proof is straightforward and thus omitted.

Running Time. For every fixed set S, the dynamic programming table T has at most $\delta_{\mathcal{F}} \cdot n \cdot (n+1)$ entries. Each entry can be computed in $\mathcal{O}(2^{|S|} \cdot |I|)$ time. Thus, all entries can be computed in $\sum_{i=0}^{n} \binom{n}{i} 2^i \cdot \mathcal{O}(n^2 \cdot \delta_{\mathcal{F}} \cdot |I|) = 3^n \cdot \mathcal{O}(n^2 \cdot \delta_{\mathcal{F}} \cdot |I|)$ time in total. To evaluate if there is some $v \in N$ and some $P \in \mathcal{P}_{\mathcal{F}}(v)$ such that $T[v, P, N \setminus (P \cup \{v\}), |P|] \geq t$ can afterwards be done in $\mathcal{O}(n \cdot \delta_{\mathcal{F}})$ time. Consequently, the algorithm runs in $\mathcal{O}(3^n \cdot |I|^{\mathcal{O}(1)})$ time. □

We also obtain a kernel lower bound for Polytree Learning parameterized by n. The proof is a slight adaption of the proof the kernel lower bound for Vanilla-BNSL parameterized by n (Theorem 6.33).

Theorem 7.2. Polytree Learning *does not admit a polynomial kernel when parameterized by n unless* NP $\subseteq$ coNP/poly *even when restricted to instances where the directed superstructure is a DAG and all local scores are either 0 or 1.*

Proof. We describe a polynomial parameter transformation from Multicolored Independent Set. Recall that in Multicolored Independent Set one is given a graph $G = (V, E)$ together with a partition $(V_1, \ldots, V_\ell)$ of V. Throughout this proof, we assume without loss of generality that each color class V_i is a clique in G. The question is, if there exists a multicolored independent set in G, that is, an independent set containing one vertex from each set V_i. Multicolored Independent Set does not admit a polynomial kernel when parameterized by $\sum_{r=1}^{\ell-1} |V_r|$ due to Proposition 1.4.

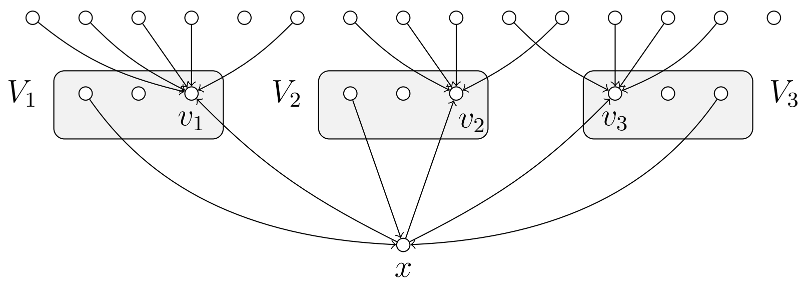


Figure 7.2: An example of the construction given in the proof of Theorem 7.2. The original instance contains a multicolored independent set on the vertices $v_1 \in V_1$, $v_2 \in V_2$, $v_3 \in V_3$, and $v_4 \in V_4$. The directed edges represent the arcs of a polytree with score 4. The choice of vertex $v_4 \in V_4$ is encoded in the parent set of x.

Construction. Let $G = (V, E)$ be an instance of Multicolored Independent Set with a partition $(V_1, \dots, V_\ell)$. We describe how to construct an equivalent instance $I := (N, \mathcal{F}, t)$ of Polytree Learning. Let $V^{<\ell} := V \setminus V_\ell$. We set

$$N := V^{<\ell} \cup \{w_e \mid e \in E_G(V^{<\ell})\} \cup \{x\}.$$

For each $v \in V^{<\ell}$, we set $P_v := \{x\} \cup \{w_{\{v,u\}} \mid u \in N_G(v) \cap V^{<\ell}\}$ and $f_v(P_v) := 1$. Moreover, for each $v \in V_\ell$, we set $f_x(N_G(v) \cap V^{<\ell}) := 1$. All other local scores are set to 0. Finally, we set $t := \ell$. Observe that $|N|$ is polynomially bounded by $\sum_{r=1}^{\ell-1} |V_r|$, that the directed superstructure is acyclic, and that the local scores are either 1 or 0. An example of the construction is shown in Figure 7.2.

Intuition. Before we show the correctness of the polynomial parameter transformation, we provide some intuition. Analogously to the construction for Vanilla-BNSL from the proof of Theorem 6.33, we obtain score ℓ if one vertex per color class $V_1, \dots, V_{\ell-1}$ and the vertex x learn a parent set with score 1. The vertices from $V_1, \dots, V_{\ell-1}$ and the choice of the parent set of x then correspond to a multicolored set in G. The condition that the resulting directed graph must be a polytree guarantees that the chosen vertices form an independent set.

Correctness. We show that G has a multicolored independent set if and only if I is a yes-instance of Polytree Learning.

($\Rightarrow$) Let $S = \{v_1, \dots, v_\ell\}$ be a multicolored independent set in G such that $v_i \in V_i$ for every $i \in \{1, \dots, \ell\}$. We define the arc set A by defining the parent sets of the vertices of N under A. We set

$$P^A_{v_i} := \{x\} \cup \{w_{\{v_i,u\}} \mid u \in N_G(v_i) \cap V^{<\ell}\}$$

for each $i \in \{1, \dots, \ell - 1\}$ and $P_x^A := N_G(v_\ell) \cap V^{<\ell}$. All remaining vertices have an empty parent set under A. By construction, $f_x(P_x^A) = f_{v_i}(P_{v_i}^A) = 1$ for each $i \in \{1, \dots, \ell - 1\}$. Thus, $\text{score}(A) = \ell$.

It remains to show that $D := (N, A)$ is a polytree. To this end, we show that for each distinct pair $u, u' \in \{v_1, \dots, v_{\ell-1}, x\}$ the sets $P_u^A \cup \{u\}$ and $P_{u'}^A \cup \{u'\}$ may only share the vertex x. Note that $v_i \notin N_G(v_\ell)$ for any $i \in \{1, \dots, \ell-1\}$ and, thus, $v_i \notin P_x^A$. Consequently, P_x^A is disjoint from all the sets $\{v_i\} \cup P_{v_i}^A$ with $i \in \{1, \dots, \ell-1\}$. Next, we show that $P_{v_i}^A \cup \{v_i\}$ and $P_{v_j}^A \cup \{v_j\}$ only share the vertex x for distinct $i, j \in \{1, \dots, \ell-1\}$. Since S is an independent set, there is no edge $e = \{v_i, v_j\} \in E_G(V^{<\ell})$. Thus, there is no $w_e \in P_{v_i}^A \cap P_{v_j}^A$. Hence, $P_{v_i}^A \cap P_{v_j}^A = \{x\}$ and, thus, D is a polytree. Consequently, I' is a yes-instance of Polytree Learning.

($\Leftarrow$) Conversely, let $A \subseteq N \times N$ be an arc set such that $D := (N, A)$ is a polytree with score at least ℓ. We show that there is a multicolored independent set in G. To this end, consider the following claim.

Claim 1. *For every V_i with $i \in \{1, \dots, \ell - 1\}$ there is exactly one vertex $v_i \in V_i$ with $f_{v_i}(P_{v_i}^A) = 1$.*

Proof. Since all local scores are either 0 or 1 and since $\text{score}(A) = \ell$, there are at least $\ell - 1$ vertices of $V^{<\ell}$ that have a parent set of score 1 under A. Next, assume towards a contradiction that there are distinct vertices $u \in V_i$ and $v \in V_i$ with $f_u(P_u^A) = f_v(P_v^A) = 1$ for some $i \in \{1, \dots, \ell - 1\}$. Then, since $G[V_i]$ is a clique, the skeleton of D contains the cycle $(x, u, w_{\{u,v\}}, v)$ which contradicts the fact that (N, A) is a polytree. Consequently, the claim holds. $\Diamond$

By Claim 1 there are unique vertices $v_1, \dots, v_{\ell-1}$ such that $v_i \in V_i$ and $f_{v_i}(P_{v_i}^A) = 1$. Furthermore, since $\text{score}(A) = \ell$ we have $f_x(P_x^A) = 1$. Consequently, there exists a vertex $v_\ell \in V_\ell$ with $N_G(v_\ell) = P_x^A$. We define $S := \{v_1, \dots, v_{\ell-1}, v_\ell\}$ and show that S is a multicolored independent set in G.

Obviously, the vertices of S are from pairwise distinct color classes. Thus, it remains to show that no two vertices in S are adjacent in G. Assume towards a contradiction that there exist $u \in S$ and $v \in S$ such that $\{u, v\} \in E$. Without loss of generality, let $v = v_1$. Consider the following cases.

Case 1: $u \in \{v_2, \dots, v_{\ell-1}\}$**.** Then, there is a vertex $w_{\{u,v\}}$ with $w_{\{u,v\}} \in P_u^A \cap P_v^A$. Thus, the skeleton of D contains the cycle $(x, u, w_{\{u,v\}}, v)$ which contradicts the fact that (N, A) is a polytree.

Case 2: $u = v_\ell$**.** Then, $\{u, v\} \in E$ implies $v \in N_G(u)$ and therefore $v \in P_x^A$. Moreover $f_v(P_v^A) = 1$ implies $x \in P_v^A$. Hence, $(v, w) \in A$ and $(w, v) \in A$ contradicting the fact that (N, A) is a polytree.

Consequently, S is a multicolored independent set in G. $\Box$

7.2 Parameterization by the Number of Dependent Vertices

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

Theorem 7.3. Polytree Learning *can be solved in* $(\delta_{\mathcal{F}})^d \cdot |I|^{\mathcal{O}(1)}$ *time.*

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

We next show that there is little hope for a significant running time improvement on this simple brute-force algorithm. More precisely, we show that Polytree Learning is W[1]-hard for d. Our reduction also implies a stronger ETH-based running time bound.

Theorem 7.4. Polytree Learning *is* W[1]*-hard when parameterized by the number of dependent vertices d. If the ETH holds, then it has no* $(\delta_{\mathcal{F}})^{o(d)} \cdot |I|^{\mathcal{O}(1)}$*-time algorithm. Both results even hold for instances where the directed superstructure $S_{\mathcal{F}}$ is a DAG.*

Proof. To prove W[1]-hardness we give a parameterized reduction from Independent Set. Afterwards, we argue why the parameterized reduction also implies the claimed ETH-based lower bound. Recall that, in Independent Set one is given an undirected graph $G = (V, E)$ and an integer k and the question is whether there is a subset $S \subseteq V$ of size at least k such that no two vertices in S are connected by an edge. We may assume that there are no isolated vertices in G. Independent Set is W[1]-hard when parameterized by k [46].

Construction. Given an instance $(G = (V, E), k)$ of Independent Set, we describe how to construct an equivalent instance $I = (N, \mathcal{F}, t)$ of Polytree Learning in polynomial time such that I has at most k dependent vertices. We define

$$N := \{v_1, \ldots, v_k\} \cup \{w_e \mid e \in E\} \cup \{x\}.$$

For every vertex $v \in V$, we define $P_v := \{w_{\{v,u\}} \mid u \in N_G(v)\} \cup \{x\}$ and we set $f_{v_i}(P_v) := 1$ for each $i \in \{1, \ldots, k\}$. All other local scores are set to 0. Finally, we set $t := k$. This completes the construction of I. Note that there are k dependent vertices $v_1, \ldots, v_k$ and that the directed superstructure is a DAG.

Intuition. Before we show the correctness we provide some intuition. To obtain a score of k, each dependent vertex $v_1, \ldots, v_k$ must choose a parent set with local score 1. Each such parent set P_v corresponds to a vertex $v \in V$. Thus, the parent sets then guarantees that these k vertices of V form an independent set in G.

Correctness. Next, we show that there is an independent set of size k in G if and only if I is a yes-instance of Polytree Learning.

($\Rightarrow$) Let $S := \{u_1, \ldots, u_k\}$ be an independent set of size k in G. We set $A := \cup_{i=1}^{k} P_{u_i} \times v_i$ and show that $D := (N, A)$ is a polytree with score k. By the definition of $\mathcal{F}$, we have $f_{v_i}(P_{v_i}^{A}) = 1$ for $i \in \{1, \ldots, k\}$ and therefore $\text{score}(A) = k$. Furthermore, since S is an independent set there is no edge $e \in E$ such that w_e is contained in $P_{u_i} \cap P_{u_j}$ with $i \neq j$. Thus, for every $i \neq j$ the sets $P_{u_i} \cup \{v_i\}$ and $P_{u_j} \cup \{v_j\}$ only share the vertex x. Consequently, D is a polytree.

($\Leftarrow$) Let $A \subseteq N \times N$ be an arc set such that $D = (N, A)$ is a polytree with score at least k. By Proposition 6.3 we may assume that only the vertices $v_1, \ldots, v_k$ have a nonempty parent set under A. Since the local scores are either 0 or 1,, for every $i \in \{1, \ldots, k\}$ there is some $u_i \in V$ such that v_i learns the parent set P_{u_i} under A.

We show that $S := \{u_1, \ldots, u_k\}$ is an independent set of size k in G. By construction of the local scores, each P_{u_i} contains the vertex x. Since every vertex of G has degree at least one, each such parent set has size at least two. Hence, the vertices u_i and u_j are distinct if $i \neq j$ as, otherwise, the skeleton of D would contain the cycle (v_i, x, v_j, w_e) for each $w_e \in P_{u_i} \setminus \{v^*\}$. Moreover, since D is a polytree, distinct vertices u_i and u_j are not adjacent in G as, otherwise, the vertex $w_{\{u_i,u_j\}}$ is contained in both parent sets P_{u_i} and P_{u_j} and, hence, the cycle $(v_i, x, v_j, w_{\{u_i,u_j\}})$ is contained in the skeleton of D. Consequently, no two vertices in S are connected by an edge and therefore S is an independent set in G.

ETH-Based Lower Bound. In the constructed instance I we have $d = k$ and $\delta_{\mathcal{F}} = n + 1$. Unless the ETH fails, Independent Set cannot be solved in $n^{o(k)}$ time [26] and, hence, Polytree Learning cannot be solved in $(\delta_{\mathcal{F}})^{o(d)} \cdot |I|^{\mathcal{O}(1)}$ time. □

Theorem 7.4 points out a difference between Vanilla-BNSL and Polytree Learning. In Vanilla-BNSL, a nondependent vertex v can be easily removed from the input instance $(N, \mathcal{F}, t)$ by setting $N' := N \setminus \{v\}$ and modifying the local scores to $f'_u(P) := \max(f_u(P), f_u(P \cup \{v\}))$. Therefore, Vanilla-BNSL is FPT for d.

7.3 Dependent Vertices and Small Parent Sets

Due to Theorem 7.4, fixed-parameter tractability for Polytree Learning parameterized by d is presumably impossible. However, in instances constructed in the proof of Theorem 7.4 the maximum parent set size p is not bounded by some computable function in d. In practice, there are many instances where p is relatively small or upper-bounded by some small constant [174]. In this section we study parameterization by d in combination with a bounded parent set size. More precisely, we first provide an FPT algorithm for the parameter $d+p$, the sum of the number of dependent vertices and the maximum parent set size. Second, we provide a polynomial kernel for the parameter d if the maximum parent set size p is constant. Both results are based on computing max q-representative sets in a matroid [60, 128].

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

Before we present the results of this section we introduce the basic concepts of the technique of representative sets. We first state the important definitions and theorems and provide some intuition about representative sets in the context of Polytree Learning.

We first give the formal definition of matroids and representative sets. We also recall a theorem by Lokshtanov et al. [128] about the efficient computation of representative sets. Finally, we define the concrete matroid that we use in this section. We start with the definition.

Definition 7.5. *A pair $M = (E, \mathcal{I})$, where E is a set and $\mathcal{I}$ is a family of subsets of E is a* matroid *if*

a) $\emptyset \in \mathcal{I}$,

b) if $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$, and

c) if $A, B \in \mathcal{I}$ and $|A| < |B|$, then there is some $b \in B \setminus A$ with $A \cup \{b\} \in \mathcal{I}$.

Matroids can be seen as an algebraic structure that generalizes the concept of linear independence in vector spaces. The intuition behind matroids in an algorithmic context is, that an input instance of a problem can be assoicated with a finite matroid $(E, \mathcal{I})$. If one—for example—aims to find an edge set in a graph, the set E of the matroid corresponds to the edges of the input graph, and the sets in the family $\mathcal{I}$ can model feasible solutions of the instance or perhaps supersets of feasible solutions. Many algorithmic results are based on matroids [128, 60, 120, 173, 69].

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

Definition 7.6. *Let $M = (E, \mathcal{I})$ be a matroid, let $\mathcal{A}$ be a family of subsets of E, and let $w : \mathcal{A} \to \mathbb{N}_0$ be a weight function. A subfamily $\widehat{\mathcal{A}} \subseteq \mathcal{A}$* max q-represents *$\mathcal{A}$ (with respect to w) if for every set $B \subseteq E$ with $|B| = q$ the following holds: If there is a set $A \in \mathcal{A}$ that fits B, there exists some $\widehat{A} \in \widehat{\mathcal{A}}$ that fits B, and $w(\widehat{A}) \geq w(A)$. If $\widehat{\mathcal{A}}$ max q-represents $\mathcal{A}$, then we write $\widehat{\mathcal{A}} \subseteq^q \mathcal{A}$.*

We refer to a set family $\mathcal{A}$ where every $A \in \mathcal{A}$ has size exactly $x \in \mathbb{N}_0$ as an *x-family*. Our results rely on the fact that max q-representative sets of an x-family can be computed efficiently as stated in a theorem by Lokshtanov et al. [128] that is based on an algorithm by Fomin et al. [60]. In the following, $\omega < 2.373$ is the matrix multiplication constant [179].

Theorem 7.7 ([128]). *Let $M = (E, \mathcal{I})$ be a linear matroid whose representation can be encoded with a $k \times |E|$ matrix over the field $\mathbb{F}_2$ for some $k \in \mathbb{N}$. Let $\mathcal{A}$ be an x-family containing ℓ sets, and let $w : \mathcal{A} \to \mathbb{N}_0$ be a weight function. Then,*

a) there exists some $\widehat{\mathcal{A}} \subseteq^q \mathcal{A}$ of size $\binom{x+q}{x}$ that can be computed with

$$\mathcal{O}\left(\binom{x+q}{x}^2 \cdot \ell x^3 k^2 + \ell \binom{x+q}{q}^\omega kx\right) + (k + |E|)^{\mathcal{O}(1)}$$

operations in $\mathbb{F}_2$, and

b) there exists some $\widehat{\mathcal{A}} \subseteq^q \mathcal{A}$ of size $\binom{x+q}{x} \cdot k \cdot x$ that can be computed with

$$\mathcal{O}\left(\binom{x+q}{x} \cdot \ell x^3 k^2 + \ell \binom{x+q}{q}^{\omega-1} (kx)^{\omega-1}\right) + (k + |E|)^{\mathcal{O}(1)}$$

operations in $\mathbb{F}_2$.

We next define the matroid that we use for Polytree Learning. Recall that, given an instance $(N, \mathcal{F}, t)$ of Polytree Learning, the directed superstructure $S_\mathcal{F}$ is defined as $S_\mathcal{F} := (N, A_\mathcal{F})$ where $A_\mathcal{F}$ is the set of arcs that are potentially present in a solution. We set $m := |A_\mathcal{F}|$. In this work we consider the *super matroid* $M_\mathcal{F}$ which we define as the graphic matroid [172] of the super structure. Formally, $M_\mathcal{F} := (A_\mathcal{F}, \mathcal{I})$ where $A \subseteq A_\mathcal{F}$ is independent if and only (N, A) is a polytree

We provide some intuition for the super matroid and for representative sets in context of Polytree Learning. We consider families $\mathcal{A}$ of arc sets of directed graphs where some of the dependent vertices receive a parent set of size exactly p. Assume that one of the arc sets $A \in \mathcal{A}$ can be extended to a solution of Polytree Learning. In other words, assume there exists an arc set Q such that $(N, A \cup Q)$ is a polytree with maximum score. If we now consider a $|Q|$-representation $\widehat{\mathcal{A}}$ of $\mathcal{A}$, there also exists an arc set $\widehat{A} \in \widehat{\mathcal{A}}$ that can be extended to a solution using the same arc set Q. In other words, if $(N, A \cup Q)$ is a polytree of maximum score, then there is some $\widehat{A} \in \widehat{\mathcal{A}}$ such that $(N, \widehat{A} \cup Q)$ is also a polytree of maximum score. Speaking in terms of Definition 7.6, the property that $\widehat{A}$ fits Q together with the definition of independent sets in the super matroid guarantees that $(N, \widehat{A} \cup Q)$ is a polytree. Furthermore, the restriction on the weight function guarantees that the score of $\widehat{A} \cup Q$ is maximum. Figure 7.3 shows a sketch of this situation. Summarizing, if we have a family $\mathcal{A}$ that potentially contains partial solutions, it suffices to consider a representing family $\widehat{\mathcal{A}}$ of $\mathcal{A}$. Recall that we consider parameterization by the number of dependent vertices d in combination with the maximum parent set size p. Since we assume that every dependent vertex chooses a parent set of size p, a solution consists of $d \cdot p$ arcs. Thus, if there is a family $\mathcal{A}$ that contains a partial solution where only $i < d$ dependent vertices have a non-empty parent set, every arc set in $\mathcal{A}$ consists of $i \cdot p$ arcs and it suffices to consider a max $(d - i) \cdot p$-representing family $\widehat{\mathcal{A}}$ of $\mathcal{A}$.

We next show that the super matroid is a linear matroid. The super matroid is closely related to the *acyclicity matroid* that has been used for a constrained version of Polytree Learning [69]. The proof of the following proposition is along the lines of the proof that the graphic matroid is a linear matroid. We provide it here for sake of completeness.

Proposition 7.8. *Let* $(N, \mathcal{F}, t)$ *be an instance of* Polytree Learning. *Then, the super matroid* $M_\mathcal{F}$ *is a linear matroid and its representation can be encoded by an* $n \times m$ *matrix over the field* $\mathbb{F}_2$.

Proof. We first show that $M_\mathcal{F}$ is a matroid. Note that $\emptyset \in \mathcal{I}$ since $(N, \emptyset)$ contains no cycles. Next, if (N, A) is a polytree for some $A \subseteq A_\mathcal{F}$, then (N, B) is a polytree

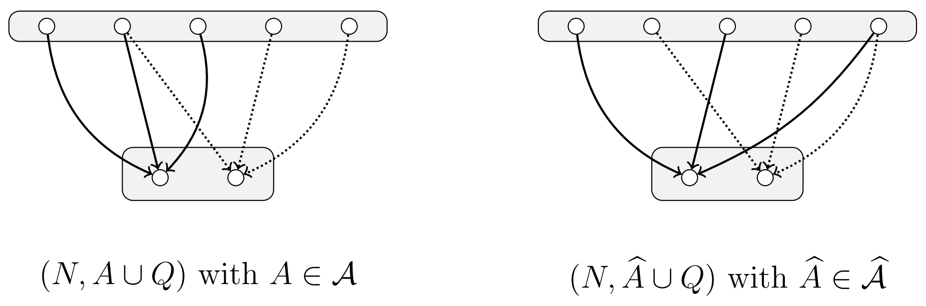

$(N, A \cup Q)$ with $A \in \mathcal{A}$
$(N, \widehat{A} \cup Q)$ with $\widehat{A} \in \widehat{\mathcal{A}}$

Figure 7.3: Two optimal polytrees on a vertex set N. The lower part shows the dependent vertices and the upper part shows the non-dependent vertices. The dotted arcs correspond to an arc set Q of size 3. The left solution is a union of Q and an arc set A that belongs to a family $\mathcal{A}$. The right solution is the union of Q and an arc set $\widehat{A}$ that belongs to a max 3-representation $\widehat{\mathcal{A}}$ of $\mathcal{A}$. The fact that $\widehat{\mathcal{A}}$ max 3-represents $\mathcal{A}$ guarantees the existence of $\widehat{A}$.

for every $B \subseteq A$. Thus, Conditions a) and b) from Definition 7.5 hold.

We next show that Condition 3 holds. Consider $A, B \in \mathcal{I}$ with $|A| < |B|$. Let $N' \subseteq N$ be the vertices of a connected component of (N, A). Since (N, B) is a polytree, the number of arcs in B between vertices of N' is at most the number of arcs in A between the vertices of N'. Then, since $|B| > |A|$, there exists some $(u, v) \in B \setminus A$ that has endpoints in two distinct connected components of (N, A). Thus, $(N, A \cup \{(u, v)\})$ is a polytree. Consequently, $M_{\mathcal{F}}$ is a matroid.

We next consider the representation of $M_{\mathcal{F}}$. Let $v_1, \ldots, v_n$ be the elements of N. We define the mapping $\varphi : A_{\mathcal{F}} \to {\mathbb{F}_2}^n$ by letting $\varphi((v_i, v_j))$ be the vector where the ith and the jth entry equal 1 and all other entries equal 0. Note that $\varphi(a_1) = \varphi(a_2)$ for two arcs a_1 and a_2 if and only if a_1 and a_2 share both endpoints. Clearly, φ can be encoded by an $n \times m$ matrix over the field $\mathbb{F}_2$.

It remains to show that an arc set A is independent in $M_{\mathcal{F}}$ if and only if $\varphi|_A$ is injective and $\{\varphi(a) \mid a \in A\}$ is linearly independent in ${\mathbb{F}_2}^n$.

($\Rightarrow$) Let A be an independent arc set. Then, (N, A) is a polytree and therefore A does not contain two arcs (u, v) and (v, u) that share both endpoints. Consequently, $\varphi|_A$ is injective. We now prove by induction over the size of subsets A' of A that $\{\varphi(a) \mid a \in A\}$ is linearly independent in ${\mathbb{F}_2}^n$.

Base Case: Let $|A'| = 1$. Then $\{\varphi(a) \mid a \in A'\}$ contains a single vector $\varphi(a) \neq \vec{0}$. Thus, the set is linearly independent.

Inductive Step: Let $|A'| > 1$. Then, since (N, A') is a polytree, the skeleton of (N, A') contains a leaf v_i. Let $x \in A'$ be the unique arc incident with that leaf. Then, $\varphi(x)$ is the unique vector in $\{\varphi(a) \mid a \in A'\}$ where the ith entry is not 0. Consequently, $\sum_{a\in A} \lambda_a \cdot \varphi(a) = \vec{0}$ with coefficients $\lambda_a \in \mathbb{F}_2$ implies $\lambda_x = 0$. Furthermore, the inductive hypothesis states that $\{\varphi(a) \mid a \in A' \setminus \{x\}\}$ is linearly independent. Therefore, $\sum_{a\in A} \lambda_a \cdot \varphi(a) = \vec{0}$ implies that all coefficients λ_a are 0. Consequently, $\{\varphi(a) \mid a \in A'\}$ is linearly independent in ${\mathbb{F}_2}^n$.

($\Leftarrow$) Conversely, let A be an arc set such that $\varphi|_A$ is injective and $\{\varphi(a) \mid a \in A\}$ is linearly independent in ${\mathbb{F}_2}^n$. Assume towards a contradiction that A is an arc set that is not independent in $M_{\mathcal{F}}$. Then, (N, A) is not a polytree. Furthermore, since $\varphi|_A$ is injective, there are no two arcs (u, v) and (v, u) in A that share both endpoints. Consequently, the skeleton of (N, A) contains a cycle. Let $a_1, \ldots, a_\ell \in A$ such that the corresponding edges of the skeleton form this cycle. Then $\sum_{i=1}^{\ell} \varphi(a_i) = \vec{0}$, which contradicts the fact that $\{\varphi(a) \mid a \in A\}$ is linearly independent in ${\mathbb{F}_2}^n$. □

7.3.1 An FPT algorithm for $\mathbf{d + p}$

We now use the super matroid $M_{\mathcal{F}}$ to show that Polytree Learning can be solved in $2^{\omega dp} \cdot |I|^{\mathcal{O}(1)}$ time where ω is the matrix multiplication constant. The idea of the algorithm is simple: Let $H := \{v_1, \ldots, v_d\}$ be the set of dependent vertices, and for $i \in \{0, 1, \ldots, d\}$ let $H_i := \{v_1, \ldots, v_i\}$ be the set containing the first i dependent vertices. The idea is that, for every H_i, we compute a family $\mathcal{A}_i$ of possible polytrees where only the vertices from $\{v_1, \ldots, v_i\}$ learn a nonempty potential parent set. We use the algorithm behind Theorem 7.7 as a subroutine to delete arc sets from $\mathcal{A}_i$ that are not necessary to find a solution. We next define the operation $\oplus$. Intuitively, $\mathcal{A} \oplus_v P$ means that we extend each possible solution in the family $\mathcal{A}$ by the arc set that defines P as the parent set of a vertex v.

Definition 7.9. *Let $v \in N$, and let $\mathcal{A}$ be an x-family of subsets of $A_{\mathcal{F}}$ such that $P_v^A = \emptyset$ for every $A \in \mathcal{A}$. For a vertex set $P \subseteq N$ we define*

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

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

Lemma 7.10. *Let $w : 2^{A_{\mathcal{F}}} \to \mathbb{N}_0$ be a weight function with $w(A) := \mathrm{score}(A)$. Let $\mathcal{A}$ be an x-family of subsets of $A_{\mathcal{F}}$.*

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

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

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

Proof. Statements *a)* and *b)* are well-known facts [38, 60]. We prove Statement *c)*. Let B be a set of size q. Let there be a set $A \cup (P \times v) \in \mathcal{A} \oplus_v P$ that fits B. That is,

$$B \cap (A \cup (P \times v)) = \emptyset, \text{ and} \tag{7.1}$$
$$B \cup (A \cup (P \times v)) \in \mathcal{I}. \tag{7.2}$$

We show that there is some $\widehat{A} \cup (P \times v) \in \widehat{\mathcal{A}} \oplus_v P$ that fits B and $w(\widehat{A} \cup (P \times v)) \geq w(A \cup (P \times v))$. To this end, observe that Property (7.1) implies

$$B \cap A = \emptyset, \text{ and} \tag{7.3}$$
$$B \cap (P \times v) = \emptyset. \tag{7.4}$$

We define $\overline{B} := B \cup (P \times v)$. Observe that Property (7.3) together with $A \cap (P \times v) = \emptyset$ implies $\overline{B} \cap A = \emptyset$, and that Property (7.2) implies $A \cup \overline{B} \in \mathcal{I}$. Consequently, A fits $\overline{B}$. Then, since $|\overline{B}| = q + |P|$ due to Property (7.4) and $\widehat{\mathcal{A}} \subseteq^{q+|P|} \mathcal{A}$, there exists some $\widehat{A} \in \widehat{\mathcal{A}}$ that fits $\overline{B}$ and $w(\widehat{A}) \geq w(A)$.

Consider $\widehat{A} \cup (P \times v)$. Since $\widehat{A}$ fits $\overline{B}$ it holds that $(\widehat{A} \cup (P \times v)) \cup B \in \mathcal{I}$. Moreover, observe that $w(\widehat{A} \cup (P \times v)) = w(\widehat{A}) + f_v(P) \geq w(A) + f_v(P) = w(A \cup (P \times v))$. It remains to show that $(\widehat{A} \cup (P \times v)) \cap B = \emptyset$. Since $\widehat{A}$ fits $\overline{B}$ and $B \subseteq \overline{B}$ it holds that $\widehat{A} \cap B = \emptyset$. Then, Property (7.4) implies $(\widehat{A} \cup (P \times v)) \cap B = (\widehat{A} \cap B) \cup ((P \times v) \cap B)) = \emptyset$. Thus, $\widehat{A} \cup (P \times v)$ fits B and therefore $\widehat{\mathcal{A}} \oplus_v P \subseteq^q \mathcal{A} \oplus_v P$. □

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

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

The algorithm is based on the following recurrence.

Lemma 7.11. *If $i = 0$, then $\mathcal{A}_i = \{\emptyset\}$. If $i > 0$, $\mathcal{A}_i$ can be expressed as*

$$\mathcal{A}_i = \bigcup_{P \in \mathcal{P}_{\mathcal{F}}(v_i) \setminus \{\emptyset\}} \mathcal{A}_{i-1} \oplus_{v_i} P.$$

Algorithm 2 FPT-algorithm for the parameter $d + p$

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

Intuitively, Lemma 7.11 states that $\mathcal{A}_i$ can be expressed by considering $\mathcal{A}_{i-1}$ and combining every $A \in \mathcal{A}_{i-1}$ with every arc set that defines a nonempty potential parent set of v_i. The correctness proof is straightforward and thus omitted.

We next present the FPT algorithm.

Theorem 7.12. Polytree Learning *can be solved in* $2^{\omega dp} \cdot |I|^{\mathcal{O}(1)}$ *time, where* ω *is the matrix multiplication constant.*

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

The subroutine `ComputeRepresentation`$(\widetilde{\mathcal{A}}_i, (d-i) \cdot p)$ in Algorithm 2 is an application of the algorithm behind Theorem 7.7 *b*). It computes a max $((d-i) \cdot p)$-representing family for $\widetilde{\mathcal{A}}_i$. As a technical remark we mention that the algorithm as described by Lokshtanov et al. [128] evaluates the weight $w(A)$ for $|\widetilde{\mathcal{A}}_i|$ many arc sets A. We assume that each such evaluation $w(A)$ is replaced by the computation of score$(A) = \sum_{v \in H} f_v(P_v^A)$ which can be done in $|I|^{\mathcal{O}(1)}$ time. We first prove the correctness of the algorithm and afterwards, we analyze the running time.

Correctness. We prove the following loop invariant.

Claim 1. *The family* $\widehat{\mathcal{A}}_i$ *max* $((d-i) \cdot p)$*-represents* $\mathcal{A}_i$ *and*

$$|\widehat{\mathcal{A}}_i| \leq \max(1, \binom{dp}{ip} \cdot n \cdot i \cdot p).$$

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

First, consider Line 4. Since we assume that every nonempty potential parent set contains exactly p vertices, Lemma 7.10 and Lemma 7.11 imply that $\widetilde{\mathcal{A}}_i$ max $((d-i)\cdot p)$-represents $\mathcal{A}_i$. Note that at this point $\widetilde{A}_i$ contains up to $\max(1, \binom{dp}{(i-1)p}\cdot n \cdot (i-1)\cdot p)\cdot \delta_{\mathcal{F}}$ sets.

Next, consider Line 5. Since we assume that every nonempty potential parent set contains exactly p vertices, the family $\widetilde{A}_i$ is an $(i\cdot p)$-family. Then, the algorithm behind Theorem 7.7 computes a $((d-i)\cdot p)$-representing family $\widehat{\mathcal{A}}_i$. Then, by Theorem 7.7 *b*) and Lemma 7.10 *a*), $\widehat{\mathcal{A}}_i$ max $((d-i)\cdot p)$-represents $\mathcal{A}_i$ and $|\widehat{\mathcal{A}}_i| \leq \binom{dp}{ip}\cdot n\cdot i\cdot p$ after the execution of Line 5. $\Diamond$

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

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

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

We next analyze the running time of one execution of Line 5. Recall that $\widetilde{\mathcal{A}}_i$ is an $(i\cdot p)$-family of size at most $\binom{dp}{(i-1)p}\cdot n\cdot i\cdot p\cdot\delta_{\mathcal{F}}$. Furthermore, recall that there are $|\widetilde{\mathcal{A}}_i|$ many evaluations of the weight function. Considering the parameters x, ℓ, k, and q from Theorem 7.7, we have

$$x := i\cdot p, \qquad \ell := \binom{dp}{(i-1)p}\cdot n\cdot i\cdot p\cdot\delta_{\mathcal{F}},$$
$$k := n, \text{ and} \qquad q := (d-i)\cdot p.$$

Combining the running time from Theorem 7.7 *b*) with the time for evaluating w, the subroutine takes time

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

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

7.3.2 A Problem Kernel for d with Constant p

We now study problem kernelization for Polytree Learning parameterized by d when the maximum parent set size p is constant. We provide a problem kernel consisting of at most $(dp)^{p+1}+d$ vertices where each vertex has at most $(dp)^p$ potential parent sets. The kernel can be computed in $(dp)^{\omega p} \cdot |I|^{\mathcal{O}(1)}$ time. Observe that both, the running time and the kernel size, are polynomial for every constant p. Note also that, since $d + p \in \mathcal{O}(n)$, Theorem 7.2 implies that there is presumably no kernel of size $(d + p)^{\mathcal{O}(1)}$ that can be computed in $(d + p)^{\mathcal{O}(1)}$ time.

Let v be a dependent vertex. The basic idea of the kernelization is to use the technique of representing sets to identify those potential parent sets P such that the arc set $P \times v$ can be extended to a solution. Doing this for every dependent vertex we identify nondependent vertices that are not necessary to find a solution. This idea is closely related to a problem kernel for d-Set Packing [38].

Theorem 7.13. *There is an algorithm that, given an instance $(N, \mathcal{F}, t)$ of* Polytree Learning *computes in time $(dp)^{\omega p} \cdot |I|^{\mathcal{O}(1)}$ an equivalent instance $(N', \mathcal{F}', t)$ such that $|N'| \leq (dp)^{p+1} + d$ and $\delta_{\mathcal{F}'} \leq (dp)^p$.*

Proof. Let H be the set of dependent vertices of $(N, \mathcal{F}, t)$.

Computation of the reduced instance. We describe how to compute a reduced instance $(N', \mathcal{F}', t)$. We define the family $\mathcal{A}_v := \{P \times v \mid P \in \mathcal{P}_{\mathcal{F}}(v)\}$ for every $v \in H$ and the weight function $w : \mathcal{A}_v \to \mathbb{N}_0$ by $w(P \times v) := f_v(P)$. We then apply the algorithm behind Theorem 7.7 *a*) and compute a max $((d-1) \cdot p)$-representing family $\widehat{\mathcal{A}}_v$ for every $\mathcal{A}_v$.

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

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

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

Observe that we use the symmetry of the binomial coefficient to obtain this running time from the Running time stated in Theorem 7.7 *a*). After computing all $\widehat{\mathcal{A}}_v$, we compute N' and $\mathcal{F}'$ in polynomial time in $|I|$. The overall running time is $(dp)^{\omega p} \cdot |I|^{\mathcal{O}(1)}$.

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

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

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

Bound on the size of $|N'|$ and $\delta_{\mathcal{F}'}$. By Theorem 7.7, each family $\widehat{\mathcal{A}}_i$ has size at most $\binom{(d-1)p+p}{p} = \binom{dp}{p}$. Consequently, $\delta_{\mathcal{F}'} \leq (dp)^p$ and $N' \leq d \cdot \binom{dp}{p} \cdot p + d \leq (dp)^{p+1} + d$. □

Observe that the instance $I := (N', \mathcal{F}', t)$ from Theorem 7.13 is technically not a kernel since the encoding of the integer t and the values of $f_v(P)$ might not be bounded in d and thus the size of the instance $|I|$ is not bounded in d. We use the following lemma [51, 62] to show that Theorem 7.13 implies an actual polynomial kernel for the parameter d when p is constant.

Lemma 7.14 ([51])**.** *There is an algorithm that, given a vector $w \in \mathbb{Q}^r$ and some $W \in \mathbb{Q}$ computes in polynomial time a vector $\overline{w} = (w_1, \ldots, w_r) \in \mathbb{Z}^r$ with* $\max_{i \in \{1,\ldots,r\}} |w_i| \in$

$2^{\mathcal{O}(r^3)}$ and an integer $\overline{W} \in \mathbb{Z}$ with total encoding length $\mathcal{O}(r^4)$ such that $w \cdot x \geq W$ if and only if $\overline{w} \cdot x \geq \overline{W}$ for every $x \in \{0,1\}^r$.

Corollary 7.15. Polytree Learning *with constant parent set size p admits a polynomial kernel when parameterized by d.*

Proof. Let $(N', \mathcal{F}', t)$ be the reduced instance from Theorem 7.13. Let r be the number of triples $(P, |P|, f_v(P))$ in the two-dimensional array representing $\mathcal{F}'$. Clearly, $r \leq |N'| \cdot \delta_{\mathcal{F}'}$. Let w be the r-dimensional vector containing all values $f_v(P)$ for all v and P. Applying the algorithm behind Lemma 7.14 on w and t computes a vector $\overline{w}$ and an integer $\bar{t}$ that has encoding length $\mathcal{O}((|N'| \cdot \delta_{\mathcal{F}'})^4)$ with the property stated in Lemma 7.14.

Substituting all local scores stored in $\mathcal{F}'$ with the corresponding values in $\overline{w}$ and substituting t by $\bar{t}$ converts the instance $(N', \mathcal{F}', t)$ into an equivalent instance whose size is polynomially bounded in d if p is constant. □

7.4 Concluding Remarks

With Polytree Learning we have studied the problem of learning Bayesian network structures with an acyclic skeleton. This fits into the line of research on BNSL under sparsity constraints that were discussed in Chapter 6. In fact, learning a polytree is a sparsity constraint that has been studied from an algorithmical point of view [29, 69, 157] since it allows for efficient inference [144, 84] and practical applications specifically ask for a polytree structure [53]. We presented the first algorithm with singly-exponential running time in the number of vertices, answering a question of Gaspers et al. [69]. We also introduced the number of dependent vertices d as a parameter and showed that Polytree Learning is W[1]-hard for d which is a contrast to Vanilla-BNSL.

Open Questions. Next, one might aim for improved parameterizations. For example, Theorem 7.3 gives an FPT algorithm for the parameter $\delta_{\mathcal{F}} + d$. Note that $\delta_{\mathcal{F}} \leq 2^{\Delta_{\text{in}}}$, where Δ_{in} is the maximum in-degree of the directed superstructure. Can we replace $\delta_{\mathcal{F}}$ or d by smaller parameters? Instead of $\delta_{\mathcal{F}}$, one could consider parameters that are small when only few dependent vertices have many potential parent sets. Ganian and Korchemna [65] showed that Polytree Learning is FPT for the local feedback edge number of the undirected superstructure. One open question is if Polytree Learning is FPT when parameterized by the sum of p and the vertex cover number s of the undirected superstructure; note that the set of dependent vertices forms a vertex cover of the undirected superstructure and thus, s never exceeds d.

To obtain the positive results for the number of dependent vertices when p is small, we used the technique of representative sets. There are problems, where using representative sets leads to an algorithm that is faster than an algorithm based on color coding [38]. Recall that the FPT algorithm for learning a DAG with a bounded number of arcs presented in Chapter 6 is based on color coding. Since using representative sets worked for Polytree Learning, it is a natural question if we can also use it for other constrained BNSL problems. However, using representative sets to learn a DAG might differ quite substantially from the approach for polytrees presented in this chapter: We used the super matroid, where an arc set is independent if it corresponds to a polytree. Thus, we were able to use a very simple recurrence to generate all parent sets (Lemma 7.11) since the computation of representative sets guarantees that the remaining arc sets correspond to polytrees. If we instead consider a structure $M := (A_{\mathcal{F}}, \mathcal{I})$, where an arc set $A \subseteq A_{\mathcal{F}}$ belongs to $\mathcal{I}$ if it corresponds to a DAG, Property c) from Definition 7.5 is violated. Thus, one might need another idea of a matroid where the independence definition includes more arc sets than the ones corresponding to DAGs. In a *uniform matroid of rank* r, every subset of the universe of size at most r is independent [38]. However, using such a matroid to learn DAGs might mean that the steps of an algorithm using representative sets may be more complicated than the ones in the algorithm from Section 7.3.1.

We believe that there is potential for practically relevant exact Polytree Learning algorithms and that this work could constitute a first step. We think that the algorithm with running time $3^n \cdot |I|^{\mathcal{O}(1)}$ might be practical for n up to 20 based on experience with dynamic programs with a similar running time [108]. A next step should be to combine this algorithm with heuristic data reduction and pruning rules to further increase the range of tractable values of n.

Chapter 8

Learning by Ordering-Based Local Search

The task of learning the structure of a Bayesian network is NP-hard [29]. In Chapters 6 and 7, we studied multiple variants of BNSL under additional sparsity constraints, provided exact algorithms, and outlined the limits of exactly solving constrained BNSL problems. Analyzing the complexity borderlines of constrained BNSL problems helps us to increase our theoretical understanding of the problem and we believe that there is potential for practically relevant exact algorithms.

However, exact algorithms might become impractical for instances with a large number of variables. In practice, BNSL is thus often solved using heuristics. One of the most successful heuristic approaches relies on local search [171]. A natural approach for a local search algorithm is a hill climbing strategy, where one replaces a given BNSL solution by a better solution within some pre-defined neighborhood as long as this is possible. More precisely, in this approach one starts with an arcless DAG D and adds, removes, or reverses an arc while this results in an increased network score. More recent local search approaches do not use DAGs as solution representations but rather orderings of the variables [8, 125, 159]. This approach is motivated by the fact that given an ordering τ of the variables, one may greedily find the optimal DAG D among all DAGs for which τ is a topological ordering [141].

In this chapter, we study different versions of local search on variable orderings. In contrast to previous work that defined the local neighborhood of an ordering as all orderings that can be reached via *one* operation, we consider *parameterized* local search [59, 129, 68, 56, 141]. Here, one sets a parameter r and aims to find a better network that can be reached via at most r modifications of the ordering. Intuitively, r can be seen as the search radius. Then, using a parameterized local search algorithm as a hill climbing strategy, one ends up with a solution that is the best possible solution within a circle of radius r in the search space. The hope is that, by considering such a larger neighborhood, one may avoid being stuck in a bad local optimum.

Related Work. The highly competitive MMHC algorithm [171] for learning large network structures is a local search approach that uses a hill-climbing strategy to find solutions in a previously defined skeleton. Ordyniak and Szeider [141] studied a parameterized local search approach, where one is given a network structure D and an integer r and the question is whether one can perform at most r operations on D (deleting arcs, inserting arcs, or reversing arcs) such that the resulting network structure has a higher sum of local scores. They showed that the local search problem is XP and W[1]-hard when parameterized by r. Furthermore, if one restricts the possible operations to only deleting arcs or only adding arcs, then the problem can be solved in polynomial time since it suffices to find one vertex v for which the local score can be increased by modifying its parent set using the allowed operations [141]. Other approaches use possible topological orderings of the resulting network structure as the search space for a local search approach [8, 125, 159]. Lee and van Beek [125] study a local search approach that is based on an operation, where one interchanges the positions of two consecutive vertices on the ordering as long as this results in an improvement of the network score. We refer to this operation as an *inversion*. Alonso-Barba et al. [8] an *insert* operation, where one removes a vertex from the ordering and inserts this vertex at a new position. Scanagatta et al. [159] considered inversions, insertions, and a further generalized operation where one removes a larger block of consecutive vertices from the ordering and inserts the block of consecutive vertices at a new position.

Our Results. We consider three different kinds of operations on orderings in this work. We first study *insertions*, where one may move one variable to an arbitrary position in the ordering and *swaps* where two arbitrary variables may exchange their positions. Recall that local search strategies based on one insertion have been studied previously [8, 159]. We introduce the corresponding local search problem, where, for a given ordering τ and a search radius r, one aims to find the best possible network when performing at most r insertions (or swaps, respectively) on τ. We observe that the corresponding parameterized local search problems are XP for r and prove W[1]-hardness for r on instances where the maximum parent set size is 2. Afterwards, we study inversions, which are swaps of adjacent vertices. Our main result is a randomized FPT algorithm with running time $2^{\mathcal{O}(\sqrt{r}\cdot\log r)} \cdot |I|^{\mathcal{O}(1)}$

for deciding for a given variable ordering, whether there is a better ordering that can be reached via at most r inversions. The distance that measures the minimum number of inversions needed to transform one ordering in another is also known as the *Kendall tau distance* as it is an adaption of Kedall tau rank correlation [101]. We then introduce a further distance that we call *inversion-window distance*. Intuitively, given an ordering τ we find an ordering τ' that has inversion-window distance at most r by partitioning τ into multiple windows of consecutive vertices and performing up to r inversions inside each window. This new distance extends the number of inversions in the following sense: Given a search radius r, the search space for orderings with inversion-window distance r is potentially larger than the number of orderings one can obtain with at mots r inversions. We provide a randomized algorithm with running time $2^{\mathcal{O}(\sqrt{r} \cdot \log r)} \cdot |I|^{\mathcal{O}(1)}$ that decides for a given ordering, whether there is a better ordering that has inversion-window distance at most r. An overview of the distances studied in this work is shown in Figure 8.1.

Our algorithms work not only for the Vanilla-BNSL but also for some structural constraints that we may wish to impose on the final DAG. To formulate the algorithms compactly, we introduce a generalization of BNSL, where each possible parent set is associated with a score and a weight and the aim is to find the highest scoring network that does not exceed a specified weight bound. We show that this captures natural types of structural constraints like a bounded number of edges in the skeleton which we studied in Chapter 6. As a side result, we show that a previous polynomial-time algorithm for acyclic directed superstructures [141] can be generalized to this new more general problem. Note that this generalizes the polynomial-time algorithm for $(\Pi_0 + e)$-Skeleton BNSL on instances with an acyclic directed superstructure that we provided in Proposition 6.23.

Finally, we show that for using an ordering-based local search approach in the presence of structural constraints it is essentially necessary, that the corresponding variant of BNSL is polynomial-time solvable on instances where the directed super structure is acyclic. This implies for several important structural constraints that ordering-based local search is unlikely to be useful.

Orderings. In this paragraph, we give formal definitions of orderings and distances between orderings. Given a vertex set N, an *ordering of* N is an n-tuple $\tau = (v_1, \dots, v_n)$ containing every vertex of N. For $i \leq n$, we let $\tau(i)$ denote the ith vertex appearing on τ. We write $u <_\tau v$ if the vertex u appears before v on τ. A *partial ordering of* τ is an ordering σ of a subset $S \subseteq N$ such that $u <_\tau v$ if and only if $u <_\sigma v$ for all $u, v \in S$. Given a vertex set S, we let $\tau[S]$ denote the partial ordering containing exactly the vertices from S, and we let $\tau - S := \tau[N \setminus S]$ denote

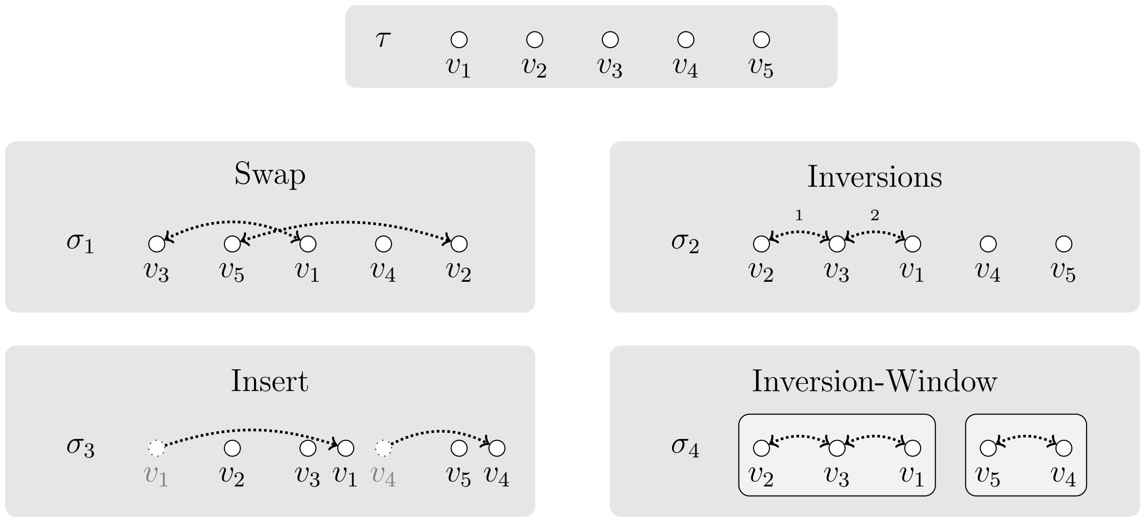

τ
v_1 v_2 v_3 v_4 v_5
Swap
σ_1
v_3 v_5 v_1 v_4 v_2
Inversions
1
2
σ_2
v_2 v_3 v_1 v_4 v_5
Insert
σ_3
v_1 v_2 v_3 v_1 v_4 v_5 v_4
Inversion-Window
σ_4
v_2 v_3 v_1 v_5 v_4

Figure 8.1: Examples of the distances studied in this work. The upper part shows an ordering τ of the vertices $v_1, \dots, v_5$. The lower part shows orderings $\sigma_1, \dots, \sigma_4$, that have distance 2 from τ with respect to the swap-, inversions-, insert-, or inversion-window distance. The dotted arrows correspond to the operations performed to obtain the ordering σ_i from τ. In case of the swap distance, σ_1 is obtained from τ by swapping the vertex pairs marked in the figure. To obtain the ordering σ_2, a first inversion swapped the positions of v_1 and v_2, and afterwards, a second inversion swapped the positions of v_1 and v_3. In case of the insert distance, σ_3 is obtained by removing the vertices v_1 and v_4 and inserting them at new positions. For the inversion-window distance, there are two windows such that at most two inversions are performed inside each window to obtain the ordering σ_4 from τ.

the partial ordering we obtain when removing all vertices of S from τ. For $i \leq j \leq n$ we define $\tau(i,j)$ as the partial ordering $(\tau(i), \tau(i+1), \dots, \tau(j))$. Given a partial ordering σ of τ, we let $N(\sigma)$ be the set of all elements appearing on σ. If $\sigma = \tau(i,j)$, we may write $N_\tau(i,j) := N(\tau(i,j))$.

A *distance* d of the orderings of N is a mapping that assigns an integer to every pair of orderings τ and τ' of N such that $d(\tau,\tau') = d(\tau',\tau)$ and $d(\tau,\tau) = 0$. For an integer r, we say that an ordering τ' is *r-close to τ with respect to d* if $d(\tau,\tau') \leq r$. If τ and d are clear from the context we may only write *τ' is r-close.*

Let $D := (N, A)$ be a directed graph. An ordering τ of N is a *topological ordering of D* if $u <_\tau w$ for every arc $(u,w) \in A$. A directed graph has a topological ordering if and only if it is a DAG.

8.1 BNSL with Multiweights

We provide parameterized local search algorithms that also work for some generalizations of Vanilla-BNSL like $(\Pi_0 + e)$-Skeleton BNSL where one aims for example to find a DAG with a restricted number of edges in the skeleton. To capture these generalizations we introduce the problem Weighted Bayesian Network Structure Learning (W-BNSL).

Let N be a set of vertices. A mapping $\mathcal{F}$ is a collection of *local multiscores for* N if $\mathcal{F}(v) \subseteq 2^{N\setminus\{v\}} \times \mathbb{N}_0 \times \mathbb{N}_0$ for each $v \in N$. Intuitively, if $(P, s, \omega) \in \mathcal{F}(v)$ for some vertex $v \in N$, then choosing P as the parent set of v may simultaneously give a local score of s and a local weight of ω. For the same parent set P, there might be another local multiscore $(P, s', \omega') \in \mathcal{F}(v)$ with a different local score and a different local weight. Throughout this work, we assume that for every vertex v there exists some $s \in \mathbb{N}_0$ such that $(\emptyset, s, 0) \in \mathcal{F}(v)$, that is, every vertex has a local multiscore for the empty parent set with weight zero. Given $v \in N$ and $\mathcal{F}$, the *possible parent sets* are defined by $\mathcal{P}_{\mathcal{F}}(v) := \{P \mid (P, s, \omega) \in \mathcal{F}(v) \text{ for some integers } s \text{ and } w\}$. Given N and $\mathcal{F}$, the directed graph $S_{\mathcal{F}} := (N, A_{\mathcal{F}})$ with $A_{\mathcal{F}} := \{(u, v) \mid u \in P \text{ for some } P \in \mathcal{P}_{\mathcal{F}}(v)\}$ is called the *directed superstructure* of N and $\mathcal{F}$ [141].

We say that an $\mathcal{F}$-valid arc set has weight at most k if for every $v \in N$ there is some $(P_v^A, s_v, \omega_v) \in \mathcal{F}(v)$ such that $\sum_{v\in N} \omega_v \leq k$. For a given integer k and an $\mathcal{F}$-valid arc set A we define $\text{score}_{\mathcal{F}}(A, k) := \sum_{v\in N} s_v$ as the maximal score one can obtain from any choice of triples $(P_v^A, s_v, \omega_v) \in \mathcal{F}_v, v \in N$, with $\sum_{v\in N} \omega_v \leq k$. If $\mathcal{F}$ is clear from the context we may write $\text{score}(A, k) := \text{score}_{\mathcal{F}}(A, k)$. We now formally define the problem.

Definition 8.1. *Let N be a vertex set, let $\mathcal{F}$ be multiscores for N. An arc set $A \subseteq N \times N$ is called $\mathcal{F}$-*valid *if (N, A) is a DAG and $P_v^A \in \mathcal{P}_{\mathcal{F}}(v)$ for all $v \in N$.*

Weighted Bayesian Network Structure Learning (W-BNSL)
Input: A set of vertices N, local multiscores $\mathcal{F}$, and two integers $t, k \in \mathbb{N}_0$.
Question: Is there an $\mathcal{F}$-valid arc set $A \subseteq N \times N$ such that $\text{score}(A, k) \geq t$?

For $N = \{v_1, \dots, v_n\}$, the local multiscores $\mathcal{F}$ are given as a two-dimensional array $\mathcal{F} := [Q_1, \dots, Q_n]$, where each Q_i is an array containing a quadruple $(s, \omega, |P|, P)$ for each $(P, s, \omega) \in \mathcal{F}(v)$. The size of $\mathcal{F}$ is the number of bits needed to store this two-dimensional array. The size of an instance I is defined as $|I| := n + |\mathcal{F}| + \log(t) + \log(k)$. Throughout this work we assume that $k \in |I|^{\mathcal{O}(1)}$ for every instance $I := (N, \mathcal{F}, t, k)$ of W-BNSL. Note that this also implies $\omega \in |I|^{\mathcal{O}(1)}$ for each multiscore (P, s, ω).

W-BNSL generalizes Vanilla-BNSL. Recall that in Vanilla-BNSL one is given a set N of vertices and one local score s for each pair consisting of a vertex $v \in N$

and a possible parent set $P \subseteq N \setminus \{v\}$ and the goal is to learn a DAG such that the sum of the local scores is maximal. Thus, Vanilla-BNSL can be modeled with local multiscores $\mathcal{F}(v)$ containing triples $(P, s, 0)$. Since Vanilla-BNSL is NP-hard [29] and the weights ω are not used for this construction, W-BNSL is NP-hard even if $k = 0$.

W-BNSL also allows to model the task of Bayesian network structure learning under additional sparsity constraints: Recall that one example for such a constrained version is $(\Pi_0 + e)$-Skeleton BNSL, where one aims to learn a DAG that consists of at most k arcs for some given integer k. This can be modeled with multiscores containing triples $(P, s, |P|)$.

A further arguably natural example is Bounded Indegree c-BNSL (BI-c-BNSL) which we define for every constant c. In BI-c-BNSL one aims to learn a network that contains at most k vertices that have more than c parents for a given integer k. This scenario can be modeled with triples (P, s, ω) where $\omega = 1$ if $|P| > c$ and $\omega = 0$, otherwise. To the best of our knowledge, this is a sparsity constraint that has not been analyzed so far. Next, we observe that W-BNSL is solvable in polynomial time if the directed superstructure is a DAG. This generalizes algorithms for BNSL [141] and $(\Pi_0 + e)$-Skeleton BNSL (Proposition 6.23).

Theorem 8.2. W-BNSL *is solvable in $\mathcal{O}(k \cdot |I|)$ time if $S_{\mathcal{F}}$ is a DAG.*

Proof. Let $I = (N, \mathcal{F}, t, k)$ be an instance of W-BNSL where the directed superstructure $S_{\mathcal{F}}$ is a DAG and let τ be a topological ordering of $S_{\mathcal{F}}$. We describe a dynamic programming algorithm to solve I. The dynamic programming table T has entries of type $T[i, k']$ with $i \in \{1, \dots, n+1\}$ and $k' \in \{0, \dots, k\}$. Each entry stores the maximal score of an arc set A of weight at most k' where only the vertices of $N_\tau(i, n)$ are allowed to learn a non-empty parent set and only the local multiscores of the vertices of $N_\tau(i, n)$ count towards the score and weight of A. We start to fill the table T by setting $T[n+1, k'] := 0$ for all $k' \in \{0, \dots, k\}$. The recurrence to compute an entry for $i \in \{1, \dots, n\}$ and $k' \in \{0, \dots, k\}$ is

$$T[i, k'] := \max_{\substack{(P,s,\omega) \in \mathcal{F}(v) \\ \omega \leq k'}} s + T[i+1, k' - \omega].$$

where $v := \tau(i)$. Thus, to determine if I is a yes-instance of W-BNSL, it remains to check if $T[1, k] \geq t$. The corresponding network can be found via traceback. The formal correctness proof is straightforward and thus omitted.

The dynamic programming table T consists of $k + 1$ entries for each $v \in V$ and each such entry can be computed in $\mathcal{O}(|\mathcal{F}(v)|)$ time plus $k + 1$ entries which can be computed in $\mathcal{O}(1)$ time. Hence, the total running time is $\mathcal{O}(k \cdot |I|)$. Consequently, the algorithm runs in polynomial time since $k \in |I|^{\mathcal{O}(1)}$. $\square$

Assume we are given an instance $I := (N, \mathcal{F}, t, k)$ of W-BNSL and consider an arbitrary but fixed optimal solution A for I. Suppose, we are given some topological ordering τ of (N, A) but not the arc set A itself. Then, Theorem 8.2 implies that we can solve I in polynomial time by restricting the possible parent sets to parent sets respecting the ordering τ. This gives rise to the ordering-based local search approach which we study in this work. More precisely, we consider a version of W-BNSL where one is additionally given an ordering of the vertex set and an integer r and aims to learn a DAG that has a topological ordering that is r-close to the given ordering. For any fixed distance d, this problem is formally defined as follows.

d-Local W-BNSL
Input: A set of vertices N, local multiscores $\mathcal{F}$, an ordering τ of N, and three integers $t, k, r \in \mathbb{N}_0$.
Question: Is there an $\mathcal{F}$-valid arc set A such that $\mathrm{score}(A, k) \geq t$ and (N, A) has a topological ordering τ' that is r-close to τ with respect to d?

Let I be an instance of d-Local W-BNSL. We call an arc set A *feasible for I* if A is $\mathcal{F}$-valid with weight at most k and (N, A) has a topological ordering that is r-close to τ. For theoretical reasons d-Local W-BNSL is stated as a decision problem since this allows us to prove the presented conditional lower bounds for some cases of d. However, the algorithms presented in this chapter solve the corresponding optimization problem, where the input is an instance $I := (N, \mathcal{F}, \tau, k, r)$ and the task is to compute a feasible arc set that maximizes $\mathrm{score}(A, k)$. Such an arc set is called a *solution of I*. Note that not every feasible arc set is necessarily a solution.

8.2 Preliminary Experiments

To assess whether parameterized local search is in principle a viable approach to ordering-based Bayesian network structure learning, we perform some preliminary experiments for the standard BNSL problem with a particularly simple neighborhood. Consider the *window-distance* Win where $\mathrm{Win}(\tau, \tau')$ is the distance r between the first and last position in which τ and τ' differ. Formally, for two orderings of length n, we define $\mathrm{Win}(\tau, \tau') := 0$ if $\tau = \tau'$ and otherwise

$$\mathrm{Win}(\tau, \tau') := \max_{\substack{j \in \{1,\dots,n\} \\ \tau(j) \neq \tau'(j)}} j - \min_{\substack{i \in \{1,\dots,n\} \\ \tau(i) \neq \tau'(i)}} i.$$

Algorithm 3 A local search algorithm combining two simple neighborhoods, herein $\text{score}_{\mathcal{F}}(\tau)$ for an ordering τ denotes maximum score of any DAG D such that τ is a topological ordering of D.

```
1: **Input:** A set of vertices $N$, local multiscores $\mathcal{F}$, an ordering $\tau$ of $N$ and an
   integer $r$.
2: **Output** An $r$-optimal ordering $\tau$.
3: currentScore := $\text{score}_{\mathcal{F}}(\tau)$
4: improvable := true
5: **while** improvable = true **do**
6:     improvable $\leftarrow$ false
7:     **for each** $\tau'$ such that $\tau$ can be obtained from $\tau'$ via one insertion **do**
8:         **if** currentScore $<$ $\text{score}_{\mathcal{F}}(\tau')$ **then**
9:             currentScore $\leftarrow$ $\text{score}_{\mathcal{F}}(\tau')$; $\tau \leftarrow \tau'$
10:            improvable $\leftarrow$ true
11:            **break**
12:    **if** improvable = false **then**
13:        **for** $i = 1$ **to** $n - r$ **do**
14:            $\tau_i$ := permutation of $\tau(i, i+r)$ such that
                   $\text{score}_{\mathcal{F}}(\tau(1, i-1) \circ \tau_i \circ \tau(i+r+1, n))$ is maximum.
15:            $\tau'$ := $\tau(1, i-1) \circ \tau_i \circ \tau(i+r+1, n)$
16:            **if** currentScore $<$ $\text{score}_{\mathcal{F}}(\tau')$ **then**
17:                currentScore $\leftarrow$ $\text{score}_{\mathcal{F}}(\tau')$; $\tau \leftarrow \tau'$
18:                improvable $\leftarrow$ true
19:                **break**
   **return** $\tau$
```

We now use a hill-climbing algorithm that combines the *window-distance* with insertion operations. That is, we say that two orderings τ and τ' are r-*close* if $\text{Win}(\tau, \tau') \leq r$ or τ can be obtained from τ' via one insertion operation. We say that an ordering τ is r-*optimal* if there is no ordering τ' that is r-close to τ such that the best DAG with topological ordering τ' achieves a better score than the best DAG with topological ordering τ.

The algorithm to compute an r-*optimal* ordering works as follows; see Algorithm 3 for the pseudocode. Given a start ordering τ, we repeat the following steps until no further improvement was found: First, check if some single insert operation on τ gives an improvement and apply it if this is the case. Otherwise, slide a window of size r over the ordering τ and find a permutation for this window that optimizes the score of the total ordering. Apply the permutation to the window if it leads to an improved

Table 8.1: Results of the experiments for local search with inserts and with inversions inside a window of size r. The boldface entries mark the maximal average score obtained for each instance.

instance	$r = 3$, avg	$r = 5$, avg	$r = 7$, avg	$r = 9$, avg	$r = 11$, avg
alarm-10000	-105308.81	**-105300.16**	**-105300.16**	**-105300.16**	**-105300.16**
alarm-1000	-11265.42	-11265.16	-11264.91	-11264.96	**-11264.60**
alarm-100	-1357.26	-1357.26	-1357.26	-1356.33	**-1355.77**
asia-10000	-22467.52	**-22466.40**	**-22466.40**	-22467.52	-22467.52
asia-1000	**-2317.49**	**-2317.49**	**-2317.49**	**-2317.49**	**-2317.49**
asia-100	-246.96	-246.30	**-245.81**	-246.96	-246.96
carpo-10000	-174451.06	-174450.96	-174450.96	-174404.82	**-174397.24**
carpo-1000	-17747.31	-17745.21	-17746.77	-17746.31	**-17739.14**
carpo-100	-1844.54	-1844.54	-1844.54	-1843.20	**-1842.42**
hailfinder-10000	-498133.54	-498133.54	-498133.54	**-498099.56**	**-498099.56**
hailfinder-1000	-52508.02	-52508.02	-52508.02	-52508.02	**-52507.98**
hailfinder-100	**-6021.58**	**-6021.58**	**-6021.58**	**-6021.58**	**-6021.58**
insurance-10000	-133108.70	-133086.47	-133086.47	**-133055.32**	**-133055.32**
insurance-1000	-13931.21	-13929.06	-13924.03	-13915.04	**-13914.44**
insurance-100	-1695.65	-1695.46	-1694.27	**-1693.40**	-1693.91
kredit-family	-16702.23	-16702.23	-16700.58	**-16697.58**	-16698.53
water-1000	-13274.46	-13274.46	-13274.46	-13274.16	**-13270.91**
water-100	-1502.62	-1502.62	-1502.40	-1502.14	**-1501.85**

score. Repeat these two steps until no further improvement has been found. To find the optimal permutation of the window one may try all permutations of the window entries. We use a more efficient dynamic programming algorithm; since we were not too interested in a detailed running time evaluations in this preliminary experiment, we omit further details on this dynamic programming algorithm.

We performed an experimental evaluation of this algorithm on data sets provided at the GOBNILP [37] homepage.[1] Given an instance, we compute 20 random topological orderings. We ran experiments for each $r \in \{3, 5, 7, 9, 11\}$ using the same 20 random orderings for a fair comparison.

Table 8.1 shows for each instance and each r the average score of the computed r-optimal orderings; Table 8.2 shows the maximum score of the 20 computed r-optimal orderings. For most of the instances the best results are obtained for $r \in \{9, 11\}$ in terms of both average score and maximum score. Thus, the experiments show that it can be worthwhile to consider larger local search neighborhoods, demonstrated here for the combination of window distance with parameter r and insertion operation. It is notable that we see the positive effect of a larger search radius for the window dis-

[1] `https://www.cs.york.ac.uk/aig/sw/gobnilp/`

Table 8.2: Results of the experiments for local search with inserts and with inversions inside a window of size r. The boldface entries mark the maximal maximum score obtained for each instance.

instance	max	$r = 5$, max	$r = 7$,max	$r = 9$, max	$r = 11$, max
alarm-10000	**-105226.51**	**-105226.51**	**-105226.51**	**-105226.51**	**-105226.51**
alarm-1000	**-11247.28**	**-11247.28**	**-11247.28**	**-11247.28**	**-11247.28**
alarm-100	-1351.92	-1351.92	-1351.92	-1351.01	**-1350.55**
asia-10000	**-22466.40**	**-22466.40**	**-22466.40**	**-22466.40**	**-22466.40**
asia-1000	**-2317.41**	**-2317.41**	**-2317.41**	**-2317.41**	**-2317.41**
asia-100	**-245.64**	**-245.64**	**-245.64**	**-245.64**	**-245.64**
carpo-10000	-174269.97	-174269.97	-174269.97	**-174137.03**	-174139.69
carpo-1000	-17728.98	-17728.98	-17725.29	-17724.56	**-17724.05**
carpo-100	**-1839.16**	**-1839.16**	**-1839.16**	**-1839.16**	**-1839.16**
hailfinder-10000	**-497730.35**	**-497730.35**	**-497730.35**	**-497730.35**	**-497730.35**
hailfinder-1000	**-52486.74**	**-52486.74**	**-52486.74**	**-52486.74**	**-52486.74**
hailfinder-100	**-6019.47**	**-6019.47**	**-6019.47**	**-6019.47**	**-6019.47**
insurance-10000	**-132968.58**	**-132968.58**	**-132968.58**	**-132968.58**	**-132968.58**
insurance-1000	-13909.50	-13888.03	-13888.03	**-13887.90**	-13888.58
insurance-100	-1689.90	-1689.90	-1689.90	-1689.24	**-1689.17**
kredit-family	**-16695.67**	**-16695.67**	**-16695.67**	**-16695.67**	**-16695.67**
water-1000	**-13263.38**	**-13263.38**	**-13263.38**	**-13263.38**	**-13263.38**
water-100	-1501.26	-1501.26	-1501.26	-1501.19	**-1501.03**

tance even when the search neighborhood allows for another non-window operation. This shows that the positive effect of the larger search radius is not due to choosing a too restrictive search neighborhood in the first place. Finally, we remark that the running time bottleneck in our preliminary experiments was not the combinatorial explosion in r but rather the slow implementation of the insert operation.

8.3 Parameterized Local Search for Insert Distance and Swap Distance

A *swap operation* on two vertices v and w on an ordering τ interchanges the positions of v and w. The distance $\text{Swap}(\tau, \tau')$ is the minimum number of swap operations needed to transform τ into τ'. An *insert operation* on an ordering τ removes one arbitrary vertex from τ and inserts it at a new position. We define $\text{Insert}(\tau, \tau')$ as the minimum number of insert operations needed to transform τ into τ'. This number can be computed as $\text{Insert}(\tau, \tau') = |N| - \text{LCS}(\tau, \tau')$, where $\text{LCS}(\tau, \tau')$ is the length of the longest common subsequence of τ and τ'. That is, if $\text{Insert}(\tau, \tau') = r$, then there is a subset $S \subseteq V$ of size r such that $\tau - S = \tau' - S$ and vice versa.

For both distances, local search approaches for BNSL have been studied previously [8, 125, 159]. We now focus on the parameterized complexity regarding the parameter r which is the radius of the local search neighborhood. We first prove that Insert-LOCAL W-BNSL and Swap-LOCAL W-BNSL are XP when parameterized by r. The algorithm is straight forward and simply computes for each possible orderings τ' which is r-close to τ a feasible arc set A such that τ' is a topological ordering of (N, A) and $\mathrm{score}(A, k)$ is maximal. The latter is done by applying Theorem 8.2 after restricting the local multiscores to those that respect the ordering τ'.

Theorem 8.3. Insert-LOCAL W-BNSL *and* Swap-LOCAL W-BNSL *are solvable in* $n^{\mathcal{O}(r)} \cdot |I|^{\mathcal{O}(1)}$ *time.*

Proof. Given an instance $I = (N, \mathcal{F}, \tau, t, k, r)$ of Insert-LOCAL W-BNSL, we compute all subsets $S \subseteq V$ of size r and all orderings τ' with $\tau' - S = \tau - S$ in $n^{\mathcal{O}(r)}$ time. For each such τ', we compute the instance $I' = (N, \mathcal{F}', t, k)$ of W-BNSL, where $\mathcal{F}'(v) := \{(P, s, \omega) \in \mathcal{F} \mid \forall u \in P : u <_{\tau'} v\}$ for each $v \in N$. Intuitively, $\mathcal{F}'$ is the collection of local multiscores restricted to parent sets that respect the topological ordering τ'. Due to Theorem 8.2, each of these instances can be solved in polynomial time since $S_{\mathcal{F}'}$ is a DAG. If one of these instances is a yes-instance, then there is a set of arcs $A \subseteq N \times N$ such that $\mathrm{score}_{\mathcal{F}}(A, k) \geq \mathrm{score}_{\mathcal{F}'}(A, k) \geq t$. Consequently, I is a yes-instance of Insert-LOCAL W-BNSL. The converse clearly holds as well since we iterate over all possible choices of S and τ'.

The algorithm for Swap-LOCAL W-BNSL works analogously: We iterate over all $n^{\mathcal{O}(r)}$ collections of r vertex pairs that swap their positions. For each such choice we restrict the local multiscores with respect to the corresponding ordering and apply the algorithm behind Theorem 8.2. □

Next we show that there is little hope that this algorithm can be improved to a fixed-parameter algorithm by showing that both problems are W[1]-hard when parameterized by r.

Theorem 8.4. Insert-LOCAL W-BNSL *and* Swap-LOCAL W-BNSL *are* W[1]*-hard when parameterized by r even if $k = 0$, $|\mathcal{F}(v)| \leq 2$ for all $v \in N$, $S_{\mathcal{F}}$ is a DAG, and every potential parent set has size at most 2.*

Proof. We describe a parameterized reduction from CLIQUE. In CLIQUE one is given an undirected graph G together with an integer k and the question is if G contains a clique of size k, that is, a set of pairwise adjacent vertices. CLIQUE is W[1]-hard when parameterized by k [46]. We first describe a parameterized reduction from CLIQUE to Insert-Local W-BNSL and afterwards, we describe how this construction can be modified to obtain the desired hardness result for Swap-Local W-BNSL.

Construction. Given an instance $I := (G = (V, E), k)$ of Clique where G has n vertices and m edges, we compute an equivalent instance $I' := (N, \mathcal{F}, \tau, t, k', r)$ of Insert-Local W-BNSL in polynomial time. Let $V = \{v_1, \dots, v_n\}$, and let $E = \{e_1, \dots, e_m\}$. The vertex set N consists of the vertices in V together with vertices w_i and $\{w_i^1, \dots, w_i^k\}$ for every edge $e_i \in E$ and k additional vertices $x_1, \dots, x_k$.

For every $e_i \in E$, we set $\mathcal{F}(w_i) := \{(e_i, k, 0)\}$ and $\mathcal{F}(w_i^j) := \{(\{w_i\}, n^9, 0)\}$ with $j \in \{1, \dots, k\}$. Further, we set $\mathcal{F}(x_i) := \{(\{x_{i-1}\}, n^9, 0)\}$ for all $i \in \{2, \dots, k\}$ and $\mathcal{F}(v) := \{(\{x_k\}, \binom{k}{2} - 1, 0)\}$ for every vertex $v \in V$. Moreover, for each $v \in N$, we also add $(\emptyset, 0, 0)$ to $\mathcal{F}(v)$. Note that $S_{\mathcal{F}}$ is a DAG. Next, we describe the ordering τ of the instance I'. For each $j \in \{1, \dots, m\}$, we set $\tau_j := (w_j, w_j^1, w_j^2, \dots, , w_j^k)$ and $\tau := \tau_1 \cdot \tau_2 \cdot \ldots \cdot \tau_m \cdot (x_1, \dots, x_k, v_1, \dots, v_n)$. Finally, we set $r := k$, $k' := 0$, and $t := (mk + k - 1)n^9 + n(\binom{k}{2} - 1) + k$ which completes the construction of I'.

Throughout this proof, we let

$$A^* := \{(w_i, w_i^j) \mid e_i \in E, j \in \{1, \dots, k\}\} \cup \{(x_{i-1}, x_i) \mid i \in \{2, \dots, k\}\}$$

denote the set of arcs for parent sets of score n^9 and we let $\widehat{A} := A^* \cup \{(x_k, v_i) \mid v_i \in V\}$ denote the set of all arcs of parent sets with positive score that do not violate the topological ordering τ. By construction, $\text{score}(\widehat{A}, 0) = t - k$. An example of the construction is shown in Figure 8.2

Intuition. The idea is that every arc set A of score at least t has to contain all the arcs of A^*. Moreover, if $D = (N, A)$ has a topological ordering τ' which is r-close to τ, then at most r of the vertices of N change their position. Intuitively, vertices $S \subseteq V$ should be inserted a the beginning of the new ordering τ' so that for each edges $e_i \in E(S)$, the vertex w_i can learn the parent set e_i. To obtain the maximal score, the number of edges in $E(S)$ should be maximal, which is the case if and only if S is a clique of size $k = r$ in G.

Correctness. Next, we show that I is a yes-instance of Clique if and only if I' is a yes-instance of Insert-Local W-BNSL. Since $k' = 0$ and every parent set has weight 0, we set $\text{score}(A) := \text{score}(A, 0)$ for ease of notation.

($\Rightarrow$) Let I be a yes-instance of Clique. Then, there is a clique $S \subseteq V$ of size k in G. Let τ' be the permutation of N obtained by moving all the vertices of S in an arbitrary ordering to the beginning of τ. Thus, $\text{Insert}(\tau, \tau') = k = r$. Since S is a clique in G, we have $\{u, v\} \in E$ for all distinct vertices $u, v \in S$. We set $A := (\widehat{A} \setminus \{(x_k, v) \mid v \in S\}) \cup \{(u, w_i), (v, w_i) \mid e_i = \{u, v\} \in E(S)\}$. By construction, τ' is a topological ordering of $D = (N, A)$. Moreover, we have $\text{score}(A) - \text{score}(\widehat{A}) =$

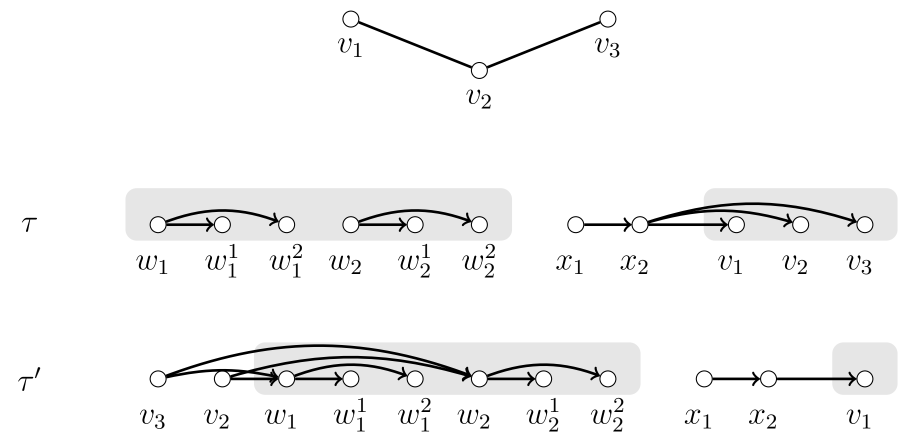

v_1
v_3
v_2
τ
w_1 w_1^1 w_1^2 w_2 w_2^1 w_2^2 x_1 x_2 v_1 v_2 v_3
τ'
v_3 v_2 w_1 w_1^1 w_1^2 w_2 w_2^1 w_2^2 x_1 x_2 v_1

Figure 8.2: An example of the construction from the proof of Theorem 8.4. The upper part shows the graph G of a clique instance $(G, 2)$. Below, there are the vertices of the Insert-Local W-BNSLinstance I, the ordering τ, and the arc set $\widehat{A}$. The first six vertices on τ correspond to the edge set of G and the last three vertices on τ correspond to the vertex set of G. The lower part shows a 2-close ordering τ' together with an optimal arc set. The parent sets of w_1 and w_2 correspond to the clique on the vertices v_2 and v_3 in G.

$|E(S)|\cdot k - k(\binom{k}{2}-1) = \binom{k}{2}k - k(\binom{k}{2}-1) = k$ and, thus, $\text{score}(A) = t$. Consequently, I' is a yes-instance of Insert-Local W-BNSL.

($\Leftarrow$) Let I' be a yes-instance of Insert-Local W-BNSL. Consequently, there is an arc set $A \subseteq N \times N$ and a topological ordering τ' for $D = (N, A)$ such that $\text{score}(A) \geq t = \text{score}(\widehat{A}) + k$ and $\text{Insert}(\tau, \tau') \leq r = k$. By construction, $n \cdot (\binom{k}{2}-1) + |A^*| \cdot n^9 + mk$ is the sum of all positive scores of potential parent sets and therefore it is an upper bound of the score one can obtain from any arc set. Observe that $A^* \subseteq A$ as, otherwise

$$\begin{aligned}\text{score}(A) &\leq n \cdot (\binom{k}{2} - 1) + (|A^*| - 1) \cdot n^9 + mk \\ &< n \cdot (\binom{k}{2} - 1) + |A^*| \cdot n^9 = \text{score}(\widehat{A}),\end{aligned}$$

which contradicts the fact that $\text{score}(A) \geq t$.

Since $\text{Insert}(\tau, \tau') \leq k$, there is a set of vertices $S \subseteq N$ of size at most k such that $\tau - S = \tau' - S$. Since $\text{score}(A) \geq \text{score}(\widehat{A}) + k$, there is a nonempty set of edges $E' \subseteq E$ of G such that $(u, w_i) \in A$ and $(v, w_i) \in A$ for all $e_i = \{u, v\} \in E'$. Hence, $w_i \in S$ or $\{u, v\} \subseteq S$ for each $e_i = \{u, v\} \in E'$. We set $S' := \{u, v \mid \{u, v\} \in$

$E'\}$ and show that S' forms a clique of size k in G. To this end, we prove the following properties of S.

Claim 1. *It holds that*

a) $S' \subseteq S$, *and*

b) $(x_k, v) \notin A$ *for every* $v \in S'$.

Proof. *a*) Assume towards a contradiction that there is an edge $e_i = \{u, v\} \in E'$ such that $\{u, v\} \nsubseteq S$. Then, $w_i \in S$. Without loss of generality, let $u \notin S$. Since $w_i \in S$ and $|S| \leq k$, there is one vertex w_i^j with $j \in \{1, \ldots, k\}$ that is not an element of S. Then, since $u \notin S$, $w_i^j \notin S$ and $w_i^j <_\tau u$, we have $w_i^j <_{\tau'} u$. Then, we have $w_i^j <_{\tau'} w_i$ since $u <_{\tau'} w_i$ due to the fact that $e_i \in E'$. Consequently, $(w_i, w_i^j) \notin A$ contradicting the fact that $A^* \subseteq A$.

b) Assume towards a contradiction that $(x_k, v) \in A$ for some $v \in S'$. Then, $x_k <_{\tau'} v$. Note that $v <_{\tau'} w_i$ where w_i is the vertex corresponding to some edge $e_i \in E'$. Thus, we have $x_k <_{\tau'} w_i$. Since $v \in S' \subseteq S$ is an element of V and $|S| \leq k$, there is one vertex w_i^j with $j \in \{1, \ldots, k\}$ and one vertex x_ℓ with $\ell \in \{1, \ldots, k\}$ which both do not belong to S. We thus have $w_i^j <_{\tau'} x_\ell$, since $w_i^j <_\tau x_\ell$. If $x_\ell = x_k$ or $x_\ell <_{\tau'} x_k$ we have $w_i^j <_{\tau'} w_i$ and thus $(w_i, w_i^j) \notin A$. This contradicts the fact that $A^* \subseteq A$. Otherwise, if $x_k <_{\tau'} x_\ell$, one of the arcs (x_i, x_{i+1}) is not an element of A which also contradicts the fact that $A^* \subseteq A$. Since both cases are contradictory, Statement *b*) follows. $\Diamond$

Due to Claim 1 *b*), the vertices in S' do not have a parent set with score $\binom{k}{2} - 1$ under A. Moreover, note that there are $|E'|$ vertices that have a parent set with score k. Formally, we have

$$\begin{aligned}\text{score}(A) - \text{score}(\widehat{A}) &= |E'| \cdot k - |S'|(\binom{k}{2} - 1) \\ &\leq |E_G(S')| \cdot k - |S'|(\binom{k}{2} - 1)\end{aligned}$$

Since $\text{score}(A) \geq \text{score}(\widehat{A}) + k$ and $|S'| \leq |S| \leq k$ by Claim 1 *a*) we have $k \leq |E_G(S')| \cdot k - |S'|(\binom{k}{2} - 1)$ which is only possible if $|S'| = k$ and $|E_G(S')| = \binom{k}{2}$. Consequently, S' forms a clique of size k in G and, thus, I is a yes-instance of CLIQUE.

Hardness for Swap Distance. Next, we describe how we can modify this construction to obtain the hardness result for Swap-LOCAL W-BNSL. The idea is that we add k additional vertices at the beginning of the topological ordering that need to be swapped with the k vertices of a clique S to obtain the required score. We add k

additional vertices $y_1, \dots, y_k$ with $\mathcal{F}_2(y_j) := \{(\emptyset, 0, 0), (\{x_k\}, n^8, 0)\}$ for each $j \in \{1, \dots, k\}$ and $\mathcal{F}_2(v) := \mathcal{F}(v)$ for each $v \in N$. Moreover, we set $\sigma = (y_1, \dots, y_k) \cdot \tau$ and $t_2 := t + k \cdot n^8$. It remains to show that I' is a yes-instance of Insert-Local W-BNSL if and only if $I_2 := (N_2, \mathcal{F}_2, \sigma, t_2, k', r)$ is a yes-instance of Swap-Local W-BNSL. By construction and the above argumentation, an arc set A has score at least $t_2 - k = \text{score}(\widehat{A}) + k \cdot n^8$ if and only if $A^* \cup \{(x_k, y_j) \mid 1 \leq j \leq k\} \subseteq A$. Consequently, each swap operation has to swap a different vertex from $\{y_1, \dots, y_k\}$ with a vertex which appears after x_k in the current topological ordering, that is, with a vertex of V. Let S be the vertices of V that are swapped with the vertices of $\{y_1, \dots, y_k\}$. Then, $\text{score}(A) \geq t_2$ if and only if S forms a clique in G. □

8.4 Parameterized Local Search for Inversions Distance

We study parameterized local search for the *inversions* distance. An inversion on an ordering is an operation that swaps the positions of two consecutive vertices of the ordering.

We describe a randomized algorithm to solve Inv-Local W-BNSL. The algorithm has constant error probability and runs in subexponential FPT time for r, the radius of the local search neighborhood. The algorithm is closely related to a parameterized local search algorithm for finding minimum weight feedback arc sets in tournaments by Fomin et al. [59]. A *tournament* is a directed graph $T = (N, A)$ where either $(u, v) \in A$ or $(v, u) \in A$ for every pair of distinct vertices u and v. A *feedback arc set* is a subset $A' \subseteq A$ such that $(N, A \setminus A')$ is a DAG. The problem is defined as follows.

Feedback Arc Set in Tournaments (FAST)
Input: A tournament $T := (N, A)$, a weight function $\omega : A \to \mathbb{N}$, and an integer k.
Question: Is there a feedback arc set $A' \subseteq A$ such that $\sum_{a \in A'} \omega(a) \leq k$?

In a local search version of FAST one is given a feedback arc set A' and aims to find a feedback arc set A'' where the sum of the weights is strictly smaller, $|A' \setminus A''| \leq r$, and $|A'' \setminus A'| \leq r$ for a given search radius $r \in \mathbb{N}_0$. The approach of Fomin et al. [59] is ordering-based in the following sense: A feedback arc set A' is identified with the unique topological ordering τ of (N, A_{rev}), where A_{rev} results from A by reversing all directions of arcs in A'. Interchanging the positions of two consecutive vertices on τ

then corresponds to removing exactly one arc from the current feedback arc set (or adding it to the feedback arc set, respectively).

FAST and W-BNSL are related, since in both problems we aim to find a DAG. Furthermore, removing one arc (u, v) in a tournament can be seen as changing the parent set of v. However, note that the seemingly more general problem W-BNSL is not an actual generalization of FAST: A parent set of a vertex contains up to $n-1$ vertices and therefore, to model all possible deletions of its incoming arcs with local multiscores, there are up to 2^{n-1} potential parent sets. Thus, modeling an instance of FAST as an instance of W-BNSL in this natural way takes exponential time and space. For our algorithm, we adapt the techniques used for FAST and show that we can use these to obtain a local search algorithm for W-BNSL.

We first introduce some formalism of inversions. Let $\tau = (v_1, \dots, v_n)$ be an ordering of a set N. An *inversion on position* $i \in \{1, \dots, n-1\}$ transforms τ into the ordering $h_i(\tau) := (v_1, \dots, v_{i-1}, v_{i+1}, v_i, \dots, v_n)$. A *sequence of inversions* $S = (s_1, \dots, s_\ell)$ is a finite sequence of elements in $\{1, \dots, n-1\}$. Applying S on τ transforms τ into the ordering $S(\tau) := h_{s_\ell} \circ h_{s_{\ell-1}} \circ \dots \circ h_{s_1}(\tau)$. The distance $\text{Inv}(\tau, \tau')$ is the length of the shortest sequence of inversions S such that $S(\tau) = \tau'$. Recall that the inversions-distance is also known as the *Kendall tau distance* [101].

The key idea behind the algorithm is as follows: If we know a partition of the vertex set N into relatively few sets which do not change their relative positions in the topological ordering, then we have a limited space of possible orderings which can be obtained by performing at most r inversions. To specify which vertices keep their relative positions we employ a technique called color coding [7]. Intuitively, we randomly color all vertices with $\mathcal{O}(\sqrt{r})$ colors in a way that vertices of the same color keep their relative positions. As the local search algorithm for FAST [59], our color coding algorithm is closely related to a subexponential time algorithm for FAST [6].

To describe the color coding technique, we need some definitions: Let N be a set of vertices. A function $\chi : N \to \{1, \dots, \ell\}$ is called a *coloring (of N with ℓ colors)*. For each color $i \in \{1, \dots, \ell\}$, we call $Z^i := \{v \in N \mid \chi(v) = i\}$ the *color class of i*. We next define color-restricted arc sets and color-restricted solutions which are important for the color coding algorithm.

Definition 8.5. *Let $I := (N, \mathcal{F}, \tau, k, r)$ be an instance of* Inv-Local W-BNSL*, let $\chi : N \to \{1, \dots, \ell\}$ be a coloring, and let A be an $\mathcal{F}$-valid arc set. We say that A is a* color-restricted arc set *if there is a topological ordering τ' of (N, A) with $\text{Inv}(\tau, \tau') \le r$ and $\tau[Z^i] = \tau'[Z^i]$ for every color class Z^i. A color-restricted arc set A that maximizes* $\text{score}(A, k)$ *is called a* color-restricted solution *of I and χ.*

We next describe a deterministic algorithm that efficiently finds a color-restricted solution.

Proposition 8.6. *Given an instance $I := (N, \mathcal{F}, \tau, k, r)$ of* Inv-Local W-BNSL *and a coloring $\chi : N \to \{1, \dots, \ell\}$ for I, a color-restricted solution A can be computed in $(r+2)^\ell \cdot |I|^{\mathcal{O}(1)}$ time.*

Proof. Before we present the algorithm, we provide some intuition.

Intuition. Our algorithm is based on dynamic programming. Every color class can be seen as a chain of vertices that keep their relative position in the ordering. An integer vector of length ℓ then describes which prefixes of these chains we consider. Our dynamic programming algorithm starts with empty chains and then adds the next vertex of one of the chains and finds a solution of the instance with the extended prefix vectors.

Notation. To formally describe the algorithm, we introduce some notation. For every color class Z^i consider the sub-ordering $\tau[Z^i]$ that contains only the vertices of Z^i. Given some integer $x \leq |Z^i|$ we define $Z^i_x := \{z_i(1), z_i(2), \dots, z_i(x)\}$ as the set of the first x vertices appearing on $\tau[Z^i]$, where $z_i(j)$ denotes the jth vertex on $\tau[Z^i]$.

Note that $Z^i_0 = \emptyset$. Given an integer vector $\vec{p} = (p_1, \dots, p_\ell)$ with $p_i \in [0, |Z^i|]$, we define $\tau(\vec{p}) := \tau[Z^1_{p_1} \cup Z^2_{p_2} \cup \dots \cup Z^\ell_{p_\ell}]$. As a shorthand, for the set of vertices appearing on $\tau(\vec{p})$ we define $N(\vec{p}) := N(\tau(\vec{p}))$ and we define $\mathcal{F}_{\vec{p}}$ by $\mathcal{F}_{\vec{p}}(v) := \{(P, s, \omega) \in \mathcal{F}(v) \mid P \subseteq N(\vec{p})\}$ for every $v \in N(\vec{p})$ as the *restriction of $\mathcal{F}$ to $N(\vec{p})$*. Note that, given a partial ordering $\tau(\vec{p})$, the vertex $z_i(p_i)$ is the last vertex of color class Z^i appearing on $\tau(\vec{p})$. Throughout this proof, we let $\vec{0}$ denote the integer vector of length ℓ where all entries equal 0, and $\vec{e_i}$ the integer vector of length ℓ where the ith entry equals 1 and all other entries equal 0.

Algorithm. The dynamic programming table T has entries of the type $T[\vec{p}, k', r']$ with $k' \in \{0, \dots, k\}$ and $r' \in \{0, \dots, r\}$. Each entry stores the score of a color-restricted solution of the Inv-Local W-BNSL instance

$$I^{\vec{p}}_{k',r'} := (N(\vec{p}), \mathcal{F}_{\vec{p}}, \tau(\vec{p}), k', r').$$

The instance $I^{\vec{p}}_{k',r'}$ is an instance obtained by only considering the prefixes of the chains of color classes which are specified by the vector $\vec{p}$. The idea behind this algorithm is to recursively find the best sink of the current network and combine this with a color-restricted solution of the remaining network. To specify the contribution of a sink to the score, we introduce the following definition: For given $i \in \{0, \dots, \ell\}$, $k' \in \{0, \dots, k\}$, and $\vec{p}$, we define the value $f_{\vec{p}}(i, k')$ as the maximal local score of a parent set $P \subseteq N(\vec{p})$ of $z_i(p_i)$ that simultaneously has weight at most k'. More formally,

$$f_{\vec{p}}(i,k') := \max_{\substack{(P,s,\omega)\in\mathcal{F}(z_i(p_i))\\ P\subseteq N(\vec{p})\\ \omega\le k'}} s.$$

The value of $f_{\vec{p}}(i,k')$ can be computed in $|I|^{\mathcal{O}(1)}$ time by iterating over the array representing $\mathcal{F}(v)$.

We next describe how to fill the dynamic programming table. As base case we set $T[\vec{0},k',r'] := 0$ for all $k' \in \{0,\dots,k\}$ and $r' \in \{0,\dots,r\}$. The recurrence to compute entries for $\vec{p} \neq \vec{0}$ is

$$T[\vec{p},k',r'] := \max_{k''\le k'} \max_{\substack{i,\ p_i>0\\ R(\vec{p},i)\le r'}} \Big(f_{\vec{p}}(i,k'') + T[\vec{p}-\vec{e}_i, k'-k'', r'-R(\vec{p},i)]\Big),$$

where $R(\vec{p},i) := |\{v\in N(\vec{p}) \mid z_i(p_i) <_{\tau(\vec{p})} v\}|$ is the number of elements that appear after $z_i(p_i)$ in $\tau(\vec{p})$. Intuitively, $R(\vec{p},i)$ is the number of inversions that need to be performed to move $z_i(p_i)$ to the end of the ordering $\tau(\vec{p})$. The score of a color-restricted solution of I and χ can be computed by evaluating $T[(|Z^1|,|Z^2|,\dots,|Z^\ell|),k,r]$. The corresponding network can be found via traceback.

Correctness. We next show that the dynamic programming recurrence is correct. That is, we prove the following claim.

Claim 2. *The table entry $T[\vec{p},k',r']$ contains the score of a color-restricted solution of $I^{\vec{p}}_{k',r'}$ and $\chi|_{N(\vec{p})}$ for each combination $\vec{p}$, k', and r'.*

Proof. We prove the claim by induction over $|\vec{p}| := \sum_{i=1}^{\ell} p_i$.

Base Case: $|\vec{p}|=0$. Then $\vec{p}=\vec{0}$ and $I^{\vec{0}}_{k',r'}$ is an instance with an empty vertex set. Thus, the score of a color-restricted solution is $0 = T[\vec{0},k',r']$.

Inductive Step: Let the claim hold for all $\vec{q}$ with $|\vec{q}| < |\vec{p}|$. Let A be a color-restricted solution of $I^{\vec{p}}_{k',r'}$ and $\chi|_{N(\vec{p})}$. We prove

$$\mathrm{score}(A,k') = T[\vec{p},k',r'].$$

($\geq$) We first show $\mathrm{score}(A,k') \geq T[\vec{p},r',k']$. Let i and k'' be the integers that maximize the right hand side of the recurrence. Taking the vertex $z_i(p_i)$ and moving it to the rightmost position of $\tau(\vec{p})$ can be done with exactly $R(\vec{p},i) \le r'$ inversions. Let P be the parent set of the tuple (P,s,ω), that maximizes $f_{\vec{p}}(i,k'')$. Defining P as the parent set with weight at most k'' of $z_i(p_i)$ and combining this with a color-restricted solution A'' of $I^{\vec{p}-\vec{e}_i}_{k'-k'',r'-R(\vec{p},i)}$ defines a color-restricted arc set A' for $I^{\vec{p}}_{k',r'}$ with $\text{score}(A',k') = \text{score}(A'',k'-k'') + f_{\vec{p}}(i,k'')$. The inductive hypothesis implies $\text{score}(A'',k'-k'') = T[\vec{p}-\vec{e}_i, k'-k'', r'-R(\vec{p},i)]$. We thus have $\text{score}(A',k') = T[\vec{p},k',r']$, and therefore $\text{score}(A,k') \geq T[\vec{p},k',r']$.

($\leq$) We next show $\text{score}(A,k') \leq T[\vec{p},r',k']$. Let $Y^1,\dots,Y^\ell$ be the color classes of $\chi|_{N(\vec{p})}$. Since A is a color-restricted solution of $I^{\vec{p}}_{k',r'}$ and $\chi|_{N(\vec{p})}$, there exists a topological ordering τ' of $(N(\vec{p}),A)$ with $\text{Inv}(\tau(\vec{p}),\tau') \leq r'$ and $\tau(\vec{p})[Z^i] = \tau'[Y^i]$ for every color i. It follows that the last vertex appearing on τ' is $z_i(p_i)$ for some color i. By the inductive hypothesis the entry $T[\tau(\vec{p}-\vec{e}_i), k'-k'', r'-R(\vec{p},i)]$ stores the score of a color-restricted solution of $I^{\vec{p}-\vec{e}_i}_{k'-k'',r'-R(\vec{p},i)}$ for each i and k'. Since we iterate over every choice of i and k'', we consider every possible choice of $z_i(p_i)$ and combine its best possible parent set with a color-restricted solution of the remaining instance. Consequently, $\text{score}(A,k') \leq T[\vec{p},k',r']$. $\diamondsuit$

Running Time. We now analyze the running time of the algorithm. The table T has $\mathcal{O}(n^\ell \cdot (k+1) \cdot (r+1))$ entries. Each entry can be computed in time polynomial in $|I|$. This would lead to a running time of $n^\ell \cdot |I|^{\mathcal{O}(1)}$. However, we can obtain a running time of $(r+1)^\ell \cdot |I|^{\mathcal{O}(1)}$ by using a more elaborate analysis similar to the one used by Fomin et al. [59]. The key idea is to compute the entry $T[(|Z^1|,|Z^2|,\dots,|Z^\ell|),k,r]$ in a top-down manner using a memoization table to store the results of computed table entries. To obtain the running time bound, we give a bound on the number of vectors $\vec{p}$ such that an entry $T[\vec{p},k',r']$ is evaluated by the algorithm in order to compute the entry $T[(|Z^1|,|Z^2|,\dots,|Z^\ell|),k,r]$. Throughout the rest of this proof we call such $\vec{p}$ an *important vector*.

Let $\vec{p}$ be an important vector. An index $j \in \{1,\dots,n\}$ is called a *hole in* $\tau(\vec{p})$ if $\tau(j) \notin N(\vec{p})$ and there exists some $j' > j$ with $\tau(j') \in N(\vec{p})$. Given a hole j, we let $\Phi(j)$ denote the set of vertices in $N(\vec{p})$ that appear after $\tau(j)$ on $\tau(\vec{p})$.

The intuition behind holes is the following: Throughout the top-down algorithm we have a 'budget' of at most r inversions. A hole at some position j is caused by a recursive call of the algorithm with some vector $\vec{p}-\vec{e}_i$ where i corresponds to the color of the vertex $\tau(j)$. Intuitively, this means that the vertex $\tau(j)$ is chosen to be a sink in a possible solution of an instance $I^{\vec{p}}_{k',r'}$ and therefore it is moved to the last position of the ordering $\tau(\vec{p})$. For this movement, the vertex $\tau(j)$ needs to pass all the vertices appearing on $\tau(\vec{p})$ after $\tau(j)$. Consequently, this decreases our budget of inversions by at least $|\Phi(j)|$. We then use the fact that the total budget of inversions is at most r which implies that the number and the position of holes in $\tau(\vec{p})$ is restricted when $\vec{p}$ is an important vector. Afterwards, we use this restriction to give an upper bound on the total number of important vectors. To formally show that the number of important vectors is bounded, we need the inequality stated in the following claim.

Claim 3. *If $T[\vec{p}, k', r']$ is an entry that is evaluated by the algorithm, we have*

$$\sum_{j \text{ is a hole in } \tau(\vec{p})} |\Phi(j)| \leq r.$$

Proof. Consider a sequence of table entries $(T[\vec{q}^{\,1}, k_1, r_1], \ldots, T[\vec{q}^{\,s}, k_s, r_s])$ such that

$$T[\vec{q}^{\,1}, k_1, r_1] = T[(|Z^1|, |Z^2|, \ldots, |Z^\ell|), k, r],$$
$$T[\vec{q}^{\,s}, k_s, r_s] = T[\vec{p}, k', r'],$$

and for each $x \in \{1, \ldots, s-1\}$ the entry $T[\vec{q}^{\,x+1}, k_{x+1}, r_{x+1}]$ has been evaluated by the algorithm in order to compute the entry $T[\vec{q}^{\,x}, k_x, r_x]$. Such sequence exists, since $T[\vec{p}, k', r']$ is evaluated by the algorithm. Note that $\sum_{x=1}^{s-1}(r_x - r_{x+1}) \leq r$.

Let j be a hole of $\tau(\vec{p})$. Then, there is a unique index $x \in \{1, \ldots, s-1\}$ such that the computation of $T[\vec{q}^{\,x}, k_x, r_x]$ creates the hole j by recursively calling $T[\vec{q}^{\,x+1}, k_{x+1}, r_{x+1}]$ with $\vec{q}^{\,x+1} = \vec{q}^{\,x} - e_i$ and $z_i({q^x}_i) = \tau(j)$ for some i. By the recurrence we then have $r_x = R(\vec{q}^{\,x}, i) + r_{x+1}$. Therefore, $r_x - r_{x+1} = R(\vec{q}^{\,x}, i) \geq |\Phi(j)|$ since $\tau(\vec{p})$ is an induced subordering of $\tau(\vec{q}^{\,x})$. Thus, we have

$$\sum_{j \text{ is a hole in } \tau(\vec{p})} |\Phi(j)| \leq \sum_{x=1}^{s-1}(r_x - r_{x+1}) \leq r.$$

◊

We next use Claim 3 to give a bound on the number of important vectors. Let $\vec{p}$ be important and let $x \in \{1, \ldots, n\}$ such that $\tau(x)$ is the last vertex appearing on $\tau(\vec{p})$. We show that $\tau(j) \in N(\vec{p})$ for every $j \in \{1, \ldots, x-(r+1)-1\}$. Intuitively, this means that all holes in $\tau(\vec{p})$ are close to x. Assume towards a contradiction that there is some $j \in \{1, \ldots, x-(r+1)-1\}$ with $\tau(j) \notin N(\vec{p})$. Note that j is a hole in $\tau(\vec{p})$ with $\tau(x) \in \Phi(j)$. Consider an index $j' \in \{x-(r+1), \ldots, x-1\}$. If $\tau(j') \notin N(\vec{p})$, then j' is a hole in $\tau(j')$ with $\tau(x) \in \Phi(j')$. Otherwise, if $\tau(j') \in N(\vec{p})$, we have $\tau(j') \in \Phi(j)$. Since there are $r+1$ possible values for j', this is a contradiction to the inequality from Claim 3.

We can specify an important vector $\vec{p}$ by specifying the set $N(\vec{p})$. By the above, it suffices to specify which elements of $X := \{\tau(x-(r+1)), \ldots, \tau(x-1)\}$ belong to $N(\vec{p})$. Recall that there are ℓ color classes, and the vertices of each color class Z^i which belong to $N(\vec{p})$ are defined by the entry p_i. Since X contains $r+1$ elements, for each color class, there are $r+2$ possible choices of $Z^i \cap X$. Thus, there are $(r+2)^\ell$

possible important vectors. Consequently, the algorithm computes at most $(r+2)^\ell \cdot (k+1) \cdot (r+1)$ table entries. Since each entry can be computed in polynomial time in $|I|$, the algorithm runs in $(r+2)^\ell \cdot |I|^{\mathcal{O}(1)}$ time. $\square$

We now describe how to use the algorithm behind Proposition 8.6 to obtain a randomized algorithm for Inv-Local W-BNSL. The idea is to randomly color the vertices with $\sqrt{8r}$ colors and use the algorithm from Proposition 8.6 to find a color-restricted solution. To asses which coloring leads to a solution of Inv-Local W-BNSL, we define *good colorings.*

Definition 8.7. *Given an instance I of* Inv-Local W-BNSL *and a coloring χ, a coloring χ is a* good coloring of I *if every color-restricted solution of χ and I is a solution of I.*

Note that not every coloring is a good coloring. A trivial example is a random coloring, where every vertex receives the same color. In this case, a colored solution has the topological ordering τ since all vertices keep their relative positions. However, if $r \geq 1$, the uncolored instance might have a strictly better solution with a topological ordering that is r-close to τ.

We next analyze the likelihood of randomly choosing a good coloring when using only $\sqrt{8r}$ colors. For ease of notation, we assume that $\sqrt{8r}$ is an integer.

Lemma 8.8. *Let I be an instance of* Inv-Local W-BNSL *and let $\chi : N \to \{1, \dots, \sqrt{8r}\}$ be a coloring that results from assigning a color to each vertex uniformly at random. Then, χ is a good coloring of I with probability at least $(2e)^{-\sqrt{r/8}}$.*

Proof. Let $I := (N, \mathcal{F}, \tau, k, r)$, let A be a solution of I, and let τ' be an r-close topological ordering of (N, A). Consider the graph $G := (N, E)$, where $E := \{(u, v) \mid u <_\tau v \text{ and } v <_{\tau'} u\}$. Intuitively, an arc $(u, v) \in E$ indicates that in τ' the vertices u and v have another relative position than in τ. Note that, when applying a sequence of inversions S on an ordering, at most $|S|$ pairs of vertices change their relative positions. Thus, we have $|E| \leq r$ since τ' is r-close to τ.

A proper vertex coloring of G is a coloring that assigns distinct colors to every pair of vertices that are connected by an edge. It is easy to see that χ is a good coloring of I if χ is a proper vertex coloring of G. The probability of randomly choosing a proper vertex coloring with $\sqrt{8r}$ colors for a graph with r edges is at least $(2e)^{-\sqrt{r/8}}$ [6]. $\square$

We next describe the randomized algorithm. Recall that for every vertex v we have $(\emptyset, s, 0) \in \mathcal{F}(v)$ for some $s \in \mathbb{N}_0$. Thus, given an instance $(N, \mathcal{F}, \tau, k, r)$ of Inv-Local W-BNSL, there always exists the $\mathcal{F}$-valid arc set $A := \emptyset$ with weight at most k such that (N, A) has a topological ordering that is r-close to τ.

Theorem 8.9. *There exists a randomized algorithm for* Inv-Local W-BNSL *that, in time* $2^{\mathcal{O}(\sqrt{r}\cdot\log(r))} \cdot |I|^{\mathcal{O}(1)}$ *returns an* $\mathcal{F}$*-valid arc set* A *with weight at most* k *such that* (N, A) *has a topological ordering that is* r*-close to* τ*. With probability at least* $1 - \frac{1}{e}$*, the arc set* A *is a solution.*

Proof. Let I be an instance of the Inv-Local W-BNSL. The algorithm can be described as follows: Repeat the following two steps $(2e)^{\sqrt{r/8}}$ times and return the color-restricted solution with the maximal score.

1. Color the vertices of N uniformly at random with colors from $\{1, \dots, \sqrt{8r}\}$. Let $\chi : N \to \{1, \dots, \sqrt{8r}\}$ be the resulting coloring.

2. Apply the algorithm behind Proposition 8.6 and compute a color-restricted solution of I and χ.

By Lemma 8.8, the probability of choosing a good coloring in Step 1 and therefore computing a solution of I in Step 2 is at least $(2e)^{-\sqrt{r/8}}$. Thus, by repeating these steps $(2e)^{\sqrt{r/8}}$ times, we obtain a running time of $(2e)^{\sqrt{r/8}} \cdot (r+2)^{\sqrt{8r}} \cdot |I|^{\mathcal{O}(1)}$. As a shorthand, let $x := \sqrt{r/8}$. The probability that the output is not a solution is at most

$$(1 - (2e)^{-x})^{(2e)^x} \leq (e^{-(2e)^{-x}})^{(2e)^x} = \frac{1}{e}.$$

The first inequality relies on the inequality $(1+y) \leq e^y$ for all y. Then, the probability that the output is a solution is at least $1 - \frac{1}{e}$. $\square$

8.5 Parameterized Local Search for Inversion-Window Distance

We now study the *inversion-window distance* $\text{InvWin}(\tau, \tau')$ which is closely related to the number of inversions. The intuition behind this new distance is the following: Given an ordering τ, we obtain an r-close ordering by partitioning τ into multiple windows of consecutive vertices and performing up to r inversions inside each such window. This new distance is an extension of the inversions distance in the sense that $\text{InvWin}(\tau, \tau') \leq \text{Inv}(\tau, \tau')$. In other words, given a search radius r, the search space of r-close orderings is potentially larger when using $\text{InvWin}(\tau, \tau')$ instead of $\text{Inv}(\tau, \tau')$. For this extended distance we also provide a randomized algorithm with running time $2^{\mathcal{O}(\sqrt{r}\log(r))} \cdot |I|^{\mathcal{O}(1)}$.

We now formally define the inversion-window distance. Let τ and τ' be orderings of a vertex set N and let S be a sequence of inversions that transforms τ into τ'. A *window partition* W for S is a partition of the set $\{1, \dots, n\}$ into windows $I_1 = [a_1, b_1], \dots, I_\ell = [a_\ell, b_\ell]$ such that no endpoint b_i appears on S. Here, a *window* $[a, b]$ with $a \leq b$ denotes the set $\{i \in \mathbb{N}_0 \mid a \leq i \leq b\}$. Note that for each window $I_j = [a_j, b_j]$ we have $N_\tau(a_j, b_j) = N_{\tau'}(a_j, b_j)$. In other words, given a window partition W, applying S on τ does not move any vertex from one window of W to another window of W. A window partition always exists since n does not appear on any sequence of inversions and thus $W := \{[1, n]\}$ is a window partition. The width of a window partition W for S is defined as

$$\text{width}(W) := \max_{I \in W} \sum_{j \in I} \#(S, j),$$

where $\#(S, j)$ denotes the number of appearances of index j on the sequence S. The *number of window inversions* $\text{WI}(S)$ is then defined as the minimum width(W) among all window partitions W of S. In other words, $\text{WI}(S)$ is the smallest possible maximum number of inversions inside a window among all possible window partitions. The *inversion-window distance* $\text{InvWin}(\tau, \tau')$ is the minimum number $\text{WI}(S)$ among all sequences S that transform τ into τ'. Observe that $\text{InvWin}(\tau, \tau') \leq \text{Inv}(\tau, \tau')$ since $\{[1, n]\}$ is a window partition of every sequence S.

Our algorithm is based on the following intuition: If an ordering τ' is r-close to τ, the ordering can be decomposed into windows in which at most r inversions are performed and the vertices from distinct windows keep their relative positions. In a bottom-up manner, we compute the best possible ordering of suffixes $\tau(a, n)$ of τ until we find the best possible ordering of $\tau(1, n) = \tau$. This is done by finding the first window on the suffix and combining this with an optimal ordering of the remaining vertices of this suffix. To find a solution of the first window we use the algorithm behind Theorem 8.9 for Inv-Local W-BNSL as a subroutine.

The sub-instances on which we apply the algorithm behind Theorem 8.9 contain the vertices of a *window* $\tau(a, b)$. Changing the ordering in $\tau(a, b)$, these vertices may learn a parent set containing vertices from $\tau(1, b)$. For our purpose, only the new parents in $\tau(a, b)$ are important, since the new parents in $\tau(1, a-1)$ are not important to find an optimal ordering of the suffix $\tau(a, n)$. Intuitively, we hide the parents in $\tau(1, a-1)$. Formally, we define the restricted local multiscores $\mathcal{F}|_a^b$ by

$$\mathcal{F}|_a^b := \{(P \cap N_\tau(a, b), s, \omega) \mid (P, s, \omega) \in \mathcal{F}(v) \text{ and } P \subseteq N_\tau(1, b)\}.$$

Theorem 8.10. *There exists a randomized algorithm for* InvWin-Local W-BNSL *that, in time $2^{\mathcal{O}(\log(r)\sqrt{r})} \cdot |I|^{\mathcal{O}(1)}$, returns an $\mathcal{F}$-valid arc set with weight at most k. With probability at least $1 - \frac{1}{e}$, the returned arc set is a solution.*

Proof. Let $I := (N, \mathcal{F}, \tau, k, r)$ be an instance of InvWin-Local W-BNSL. We describe the algorithm in two steps. We first describe a deterministic algorithm that finds a solution of I in polynomial time when using an oracle that gives solutions of Inv-Local W-BNSL instances where the search radius is at most r. Afterwards, we describe how to replace the oracle evaluations with the randomized algorithm behind Theorem 8.9 to obtain a randomized algorithm for InvWin-Local W-BNSLwith the claimed running time and error probability.

The oracle algorithm. We fill a dynamic programming table T that has entries of the type $T[j, k']$ with $j \in \{1, \ldots, n+1\}$ and $k' \in \{0, k\}$. The idea is that each entry $T[j, k']$ with $j \leq n$ stores the score of a solution of the InvWin-Local W-BNSL-instance

$$I(j, k') := (N_\tau(j, n), \mathcal{F}|_j^n, \tau(j, n), k', r).$$

Intuitively, a solution of $I(j, k')$ gives the best possible ordering of the suffix $\tau(j, n)$ corresponding to an arc set of maximum weight k'. We next describe how to fill T. As base case we set $T[n+1, k'] = 0$ for all k'. The recurrence to compute an entry for $j \leq n$ is

$$T[j, k'] := \max_{p \in \{j, \ldots, n\}} \max_{k'' \leq k'} \Big(W(j, p, k'') + T[p+1, k' - k''] \Big),$$

where $W(j, p, k'')$ denotes the score of a solution of the Inv-Local W-BNSL-instance

$$(N_\tau(j, p), \mathcal{F}|_j^p, \tau(j, p), k'', r)$$

which we obtain by an oracle evaluation. Intuitively, this instance corresponds to the first window of the suffix $\tau(j, n)$. Observe that we evaluate the oracle on Inv-Local W-BNSL-instances with search radius r. The score of a solution of I can be computed by evaluating $T[1, k]$. The ordering of the corresponding network can be found via traceback. The correctness follows from the fact we consider every possible position of the endpoint of the first window on the suffix $\tau(j, n)$.

Replacing the Oracle Evaluations. We replace the oracle by the randomized algorithm from Proposition 8.6. The dynamic programming table has $(n+1)(k+1)$ entries. Each entry can be computed by using at most $n \cdot k$ oracle evaluations. Thus, there are at most $x := (n+1)(k+1) \cdot n \cdot k \in |I|^{\mathcal{O}(1)}$ oracle evaluations. We replace every oracle evaluation by applying the algorithm behind Theorem 8.9 exactly x times and keeping a result with maximum score.

Observe that the algorithm always computes a feasible arc set for I. The probability that all x repetitions fail to compute a solution of the corresponding Inv-Local

W-BNSL-instance is at most $(\frac{1}{e})^x$. Consequently, the probability that the correct result of one oracle evaluation is returned is at least $1 - \frac{1}{e^x}$. Therefore, the success probability of the algorithm is at least $(1 - \frac{1}{e^x})^x \geq (1 - \frac{1}{e})$. The inequality holds since we have equality in the case of $x = 1$ and the left hand side of the inequality strictly increases when $x \geq 1$ increases.

We next analyze the running time of the algorithm. As mentioned above, the dynamic programming table has $(n+1)(k+1) \in |I|^{\mathcal{O}(1)}$ entries. For each entry we apply the algorithm from Theorem 8.9 on x instances of Inv-Local W-BNSL with search radius r. Since $x \in |I|^{\mathcal{O}(1)}$ we have total running time $2^{\mathcal{O}(\log(r)\sqrt{r})} \cdot |I|^{\mathcal{O}(1)}$. □

8.6 Limits of Ordering-Based Local Search

In this section we outline the limits of ordering-based local search. In particular, we observe that ordering-based local search might not be a promising approach to handle variants of BNSL that are NP-hard on instances with an acyclic directed superstructure. As mentioned in Section 8.1, W-BNSL can be used to model Bayesian network learning under additional sparsity constraints like a bounded number of arcs. However, some important sparsity constraints are unlikely to be modeled with our framework, for example sparsity constraints that are posed on the skeleton or on the moralized graph [50] of the resulting network. Recall that the *skeleton* of a DAG is the underlying undirected graph, and that the *moralized graph* of a DAG D is an undirected graph $(N, E_1 \cup E_2)$ with $E_1 := \{\{u,v\} \mid (u,v) \in A\}$, and $E_2 := \{\{u,v\} \mid u$ and v have a common child in $D\}$. Furthermore, recall that the NP-hard task of Bayesian inference can be solved more efficiently if the moralized graph of the network is treelike, that is, it has small treewidth [40]. Thus, it is well motivated to learn a Bayesian network structure that satisfies a sparsity constraint that provides an upper bound on the treewidth of the moralized graph. In Chapter 6, we studied this task for several sparsity constraints.

For the following constraints, the task of learning a restricted network structure with score at least t is NP-hard even on instances where the directed superstructure is a DAG and t is polynomial in the number of vertices:

- Skeleton with bounded treewidth (Theorem 7.4),
  Moralized graph with bounded treewidth [112]

- Skeleton with bounded vertex cover number (Theorem 6.9),
  Moralized graph with bounded vertex cover number [113]

- Skeleton with bounded dissociation number (Theorem 6.9),
  Moralized graph with bounded dissociation number (Corollary 6.10)
- Moralized graph with a bounded number of edges (Theorem 6.30)

Recall that all parameterized reductions contained in this work run in polynomial time and thus, we may derive NP-hardness from these W[i]-hardness results.

We now argue that for these variants of BNSL, there is little hope that one can efficiently find improvements of a given DAG by applying changes on its topological ordering. Observe that an acyclic directed superstructure has a topological ordering τ which is then automatically a topological ordering of every DAG where the parent sets are potential parent sets. Furthermore, observe that all the constraints listed above are true if the network structure does not contain arcs. Assume one can find improvements of a given DAG in polynomial time if the radius r of the local search neighborhood is constant. Then, we can solve instances with acyclic directed superstructure by setting $r = 0$, starting with an empty DAG and a topological ordering of $S_{\mathcal{F}}$, and improve the score t times. This would be a polynomial-time algorithm for instances where t is polynomial in n and $S_{\mathcal{F}}$ is a DAG which would imply P = NP.

8.7 Concluding Remarks

We initiated the study on parameterized ordering-based local search for the task of learning a Bayesian network structure. We studied four distances d and classified for which of these distances d-Local W-BNSL is FPT and for which distances it is W[1]-hard. Note that there is one big difference between this chapter and the other chapters of Part III. While in Chapters 6 and 7 we studied the task of exactly learning a Bayesian network structure, we now proposed local search heuristics that can be used to find locally optimal network structures. Even though we think that there is potential for practically relevant exact algorithms, the local search algorithms described in this chapter might be more relevant in a practical context.

Open Questions. There are several ways of extending our results that seem interesting topics for future research. First, besides the experimental motivation mentioned in Section 8.2, this work is purely theoretical. Recall that the running time bottleneck in the preliminary experiments from Section 8.2 was not the combinatorial explosion in the search radius r but rather the slow implementation of the insert operation. Krambrock [119] provided another implementation of the hill-climbing strategy behind the experiments in Section 8.2 and obtained an improved running time. Thus, one important topic for future work is to investigate how well the theoretical algorithms proposed in this work perform in practice when combined with algorithm engineering tricks like data reduction rules, upper and lower bounds for the solution size, and restarts.

Besides experimental evaluations and practical improvements of the algorithms proposed in this chapter, it is also interesting to further study theoretical running time improvements. Recall that our subexponential-time algorithms for the inversions distance and the inversion-window distance are closely related to an algorithm for FAST [6]. Feige [54] and Karpinski and Schudy [100] improved the running time lower bound for FAST by proposing algorithms with running time $2^{\mathcal{O}(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$ instead of $2^{\mathcal{O}(\sqrt{k}\log(k))} \cdot n^{\mathcal{O}(1)}$. Therefore, a natural open question is whether d-LOCAL W-BNSL for $d \in \{\text{Inv}, \text{InvWin}\}$ can be solved in $2^{\mathcal{O}(\sqrt{r})} \cdot n^{\mathcal{O}(1)}$ time.

The results in this work may be extended by considering further distances. One example for another distance could be a q-swap distance, where a q-swap for some fixed integer q is a swap of two vertices on the ordering which positions differ by at most q. The q-swap distance between two orderings is then the minimum number of q-swaps needed to transform one ordering into the other. Observe that in a 1-swap is an inversion. Observe that a q-swap can be simulated by performing at most q inversions. Consequently, using the algorithm behind Theorem 8.9 we can compute a solution in $2^{\mathcal{O}(\sqrt{q \cdot r}\log(q \cdot r))} \cdot |I|^{\mathcal{O}(1)}$ time that is at least as good as the best solution that can be found by performing at most r q-swaps. One first question might be if there is a more efficient algorithm for q-swap LOCAL W-BNSL. Analogously to q-swaps, one may define q-inserts, where one can remove a vertex at position j and insert it at a new position i with $j - q \leq i \leq j + q$. A hill-climbing strategy that is based on performing single q-insert operations has previously been studied by Alonso-Barba et al. [8].

In Section 8.6, we outlined the limits of ordering-based local search. In a nutshell, we discussed variants of BNSL that are NP-hard even on instances with an acyclic directed superstructure—in other words—instances, where the topological ordering of the solution is known. An idea to tackle such variants of BNSL might be to find parameters ℓ for which these variants are FPT when the directed superstructure is acyclic. Then, it would be interesting to study ordering-based local search parameterized by $r + \ell$. Intuitively, there may be efficient algorithms that iterate over all orderings that are r-close to a given ordering in $n^{f(r)}$ time and then solve the instances respecting each such ordering in $g(\ell) \cdot |I|^{\mathcal{O}(1)}$ time. Maybe it is also possible that this can be improved to an FPT-algorithm for parameterization by $r + \ell$.

Part IV

Conclusion

In this work we studied Strong Triadic Closure (STC) and related edge coloring problems and multiple variants of Bayesian Network Structure Learning (BNSL). A detailed summary of our specific results can be found in the corresponding chapters. Therefore, we will not restate all these results here. Instead, we describe on a high level in which way our contributions extend existing results. Furthermore,some concrete open questions were posed in the concluding remarks of the corresponding chapters. In this conclusion, we provide a collection of possible future research projects that are related to the problems but fall beyond the scope of the chapters of this work.

Strong Triadic Closure

Summary. We studied the problem of partitioning the edge set of a graph into (multiple) strong and weak classes of edges such that the number of weak edges is minimized and the strong triadic closure property is satisfied.

Most previous algorithmic work on strong triadic closure has focused on STC, the problem variant with one strong color. The STC problem was introduced in 2014 [164] and there were results that STC is NP-hard even on restricted graph classes [164, 110, 111]. Konstantinidis et al. [110] studied the relationship between STC and Cluster Deletion (CD) to provide a polynomial-time algorithm for both problems on P_4-free graphs. Furthermore, STC is FPT when parameterized by the number k of weak edges [164] and it admits a linear problem kernel for k [77, 23].

With our work, we extended the knowledge on the relationship between STC and CD. Furthermore, we initiated the study of the number of strong edges ℓ as natural parameter for STC and showed that STC is FPT when parameterized by ℓ. Recall that, independent from our work, the parameter ℓ has been studied by Golovach et al. [71]. We observed the NP-hardness of STC with multiple strong colors and introduced the list variants as further generalizations of the problem. These generalizations allow to model further restrictions on the labeling that are arguably well-motivated in the context of social network analysis. For these list variants, we provided strong ETH-based lower bounds that are superexponential in the number of vertices. We also analyzed how the positive results regarding the parameterized complexity of STC can be lifted to the more general problem variants. Finally, we outlined the relationship between Multi-STC and classic edge coloring. Based on this relationship, we observed that Multi-STC can be solved in polynomial time when the input graphs has low degree. Using this observation, we introduced a distance-from-triviality parameter that measures the distance of the input graph to a low-degree graph. We proved thatMulti-STC admits a polynomial problem kernel when parameterization by the sum of this new parameter and the number of colors.

Outlook. The strong triadic closure property requires that two agents which have the same type of strong relationship with a third agent must be connected in the social network. This is a very simple combinatorial property. Based on the strong triadic closure property, several works from the data mining community identify strong relationships in real-world data. Adriaens et al. [1] claimed that when only relying on the strong triadic closure property, one might fail to capture some strong ties. To capture more strong relationships, it might thus be motivated to study variants of STC that are based on a less strict version of the strong triadic closure property.

In the following, we introduce a possible relaxation of STC: An important graph parameter in context of triadic closure is the so-called *closure number* $\mathrm{cl}(G)$ of a graph G (also known as *c-closure of* G). Fox et al. [61] introduced this graph parameter which is defined as the smallest integer $\mathrm{cl}(G)$ such that every pair of two vertices is connected by an edge if they have at least $\mathrm{cl}(G)$ common neighbors. In recent works on parameterized complexity, $\mathrm{cl}(G)$ has been studied as a possible parameter for classic problems like Independent Set, Dominating Set, or variants of Vertex Cover [106, 107]. The idea that only *multiple* common neighbors indicate an adjacency of two vertices gives rise to a problem which we call Strong α-Closure for some integer α. In Strong α-Closure one is given a graph G and an integer k and one aims to label the edges of G as *strong* or *weak* such that at most k edges are weak and the *strong α-closure property* is satisfied. Here, the *strong α-closure property* requires that no pair of nonadjacent vertices has α or more common strong neighbors. Note that STC is the special case where $\alpha = 1$. In this sense, for $\alpha > 1$, Strong α-Closure can be seen as a less strict version of STC. One interesting observation is, that if an edge-labeling produces a conflict to the strong α-closure property, one can identify a set of 2α edges from which one edge must be labeled as weak. Using this observation one can think of a simple $(2\alpha)^k \cdot n^{\mathcal{O}(1)}$ branching algorithm, and thus, Strong α-Closure is FPT when parameterized by $k + \alpha$. Is Strong α-Closure FPT when parameterized by k alone? Besides direct FPT algorithms, one could also aim for kernelizations. Known kernelizations for STC parameterized by k [77, 23] can not be directly applied for Strong α-Closure when $\alpha > 1$. Is it possible to adapt the known kernelizations or do we need a fundamentally new idea for this new variant of STC?

Another more general direction of research is to further study the correspondence between graphs and their Gallai graphs. Recall that EL-Multi-STC can be solved in $3^m \cdot n^{\mathcal{O}(1)}$ time by solving List Colorable Subgraph on the Gallai graph of the input graph (Proposition II.7). Thus, if List Colorable Subgraph can be solved in polynomial time on some graph class Π, we can identify a graph class on which EL-Multi-STC is solvable in polynomial time: the class of all graphs whose Gallai graph belongs to Π. Since considering the Gallai graph can provide a new view on the input graph, it seems to be interesting to study which other graph problems (besides EL-Multi-STC) become tractable when the Gallai graph of the input graph has a certain structure. To this end, one could further study relations between the structure of graphs and the structure of their Gallai graphs. For example, when given a graph class Π, is it possible to characterize the graphs in Π via forbidden induced subgraphs of their Gallai graphs?

Bayesian Networks

Summary. We studied the problem of finding an optimal DAG structure of a Bayesian network by using local scores. Many previous algorithmic works focused on Vanilla-BNSL, where one aims to learn an optimal network structure without additional constraints [29, 141, 142, 163, 65]. Other works studied the task of learning a branching, a path, or learning a polytree [41, 65, 32, 69, 131]. Korhonen and Parviainen [112, 113] introduced the task where the input has an additional integer k and one asks for a network structure with treewidth or vertex cover at most k. Extending the work of Korhonen and Parviainen [112, 113], we introduced the framework of constrained BNSL problems where one aims to learn a network structure that is close to a sparse graph class Π. We provided an almost complete picture of the parameterized complexity of constrained BNSL problems with respect to several graph classes. This way, we filled complexity gaps left open by previous work. With the class of graphs with maximum degree one, we identified a larger class such that the corresponding learning problem is tractable and the inference task can be performed efficiently on the resulting networks.

Polytree Learning is known to be NP-hard [41]. Gaspers et al. [69] analyzed the parameterized complexity of Polytree Learning under additional constraints and posed an open question whether Polytree Learning can be solved singly-exponential time in n. We positively answered this question and introduced a new parameter called *number of dependent vertices* d that is potentially smaller than n. We showed that Polytree Learning is W[1]-hard for parameterization by d. This is somewhat surprising since Vanilla-BNSL is obviously FPT for this parameter. Thus, we discovered a striking difference in the parameterized complexity of Polytree and Vanilla-BNSL.

For Vanilla-BNSL, there are local search approaches that use possible topological orderings of the resulting network structure as the search space [8, 125, 159]. We extended this line of research by considering parameterized local search approaches on the space of topological orderings. We analyzed the parameterized complexity of this approach with regard to four natural neighborhood definitions. Some of these neighborhood definitions have previously been experimentally evaluated in a nonparameterized local search framework [8, 125, 159]. We believe that our parameterized local search algorithms have the potential to be practically relevant. We also outlined the limits of ordering-based local search by showing that this approach is unlikely to be efficient for some BNSL problems with, for example, sparsity constraints that are posed on the moralized graph.

Outlook. In the context of Bayesian networks in machine learning, there are further interesting research questions regarding the learning task and other well-motivated problems that should be studied from an algorithmic point of view.

The problem formulation of BNSL studied in this work and in multiple other algorithmic studies [141, 69, 112, 113] makes no assumptions on the structure of the local scores. Is it possible that some scores satisfy properties that can be exploited to find efficient algorithms? This question is motivated by the wide range of possible local scores that are used to learn Bayesian networks: recall that besides well-established local scores like AIC, BIC, or BDe [4, 161, 22, 89], proposing new scoring functions is an important research direction [9, 18, 92, 94, 95, 132]. An example of a useful property is a symmetry in case of single-element parents sets: for every pair of vertices v and w, a vertex v choosing a one-element parent set $\{w\}$ receives the same local score as w when choosing the one-element parent set $\{v\}$. Chow and Liu [32] exploited this property and suggested to learn an optimal branching by using Kruskal's algorithm [121] to compute a maximum weight spanning tree. Another example of a useful property is the *additivity* of local scores. Here, one is given the local scores $f_v(\{w\})$ of one-element parent sets and an integer q and the local score of a parent set P is defined by $f_v(P) := \sum_{w \in P} f_v(\{w\})$ if $|P| \leq q$ or $f_v(P) = 0$ if $|P| > q$. Ganian and Korchemna [65] showed that VANILLA-BNSL is FPT for the treewidth of the undirected superstructure when the local scores are *additive*. In contrast, VANILLA-BNSL is W[1]-hard for the vertex cover number of the undirected superstructure when we have *general* local scores [141]. Thus, VANILLA-BNSL parameterized by the treewidth of the undirected super structure is one example where the problem becomes easier if we can assume that the local scores are additive. It would be a well-motivated future project to derive similar properties for concrete local scores and to revisit the (parameterized) complexity of BNSL problems for these more restricted cases. A good starting point might be to revisit hardness reductions like the one given in the proof of Theorem 6.9 and to study whether they also work if the local score satisfies specific properties.

A further approach for the learning problem that does not rely on local scores is *constraint-based Bayesian network structure learning* [146]. In this approach, one uses observed data and performs conditional independence tests between pairs of variables to determine whether there is an arc between the corresponding vertices in the network structure. These independence tests do not always deliver the direction of the arc. The standard algorithm in this context is the *PC algorithm* that was introduced by Spirtes et al. [165]. In this algorithm, one starts with a skeleton that is a clique and removes edges based on conditional independence tests. Most applications adopt chi-square tests or mutual information tests [160]. The algorithm, however, is independent from the concrete conditional independence test. It would be a well-motivated project to study the constraint-based approach from an algorithmic point of view. To the best of our knowledge, there are not many works that put the constraint-based approach into an algorithmic framework in the sense that a formal definition of a corresponding optimization or decision problem is given. Chickering et al. [30] introduced a problem called LEARN in which one is given a probability distribution and a parameter bound d, and the question is whether there is a Bayesian network representing the probability distribution such that the total number of entries in the conditional probability tables is at most d. They proved that LEARN is NP-hard even when given an oracle for independence tests. A starting point for a future project might be to study the parameterized complexity of LEARN. Since the input of LEARN does not consist of a graph, it is particularly interesting and challenging to identify structural parameters for a parameterized complexity analysis.

One of the most important tasks in the context of Bayesian networks is probabilistic inference. This is the problem of using a Bayesian network to compute posterior probability distributions of some variables when given evidence of some other variables. Recall that a Bayesian network is a tuple (D, T) consisting of the DAG D and a set of conditional probability tables T which encode the distribution of each variable given the value of its parents in D. In the POSITIVE INFERENCE problem one is given a Bayesian network (D, T), a variable X and a value x, and a set of evidence variables with joint value assignment e. The question is whether $\Pr(X = x \mid e) > 0$. Kwisthout et al. [123] provided a strong ETH-based lower bound for this problem. More precisely, they showed that POSITIVE INFERENCE cannot be solved in $f(\mathcal{M}(D)) \cdot |(D, T)|^{o\left(\frac{\mathrm{tw}(\mathcal{M}(D))}{\log \mathrm{tw}(\mathcal{M}(D))}\right)}$ time, where f is a computable function, $|(D, T)|$ is the encoding length of the Bayesian network and $\mathrm{tw}(\mathcal{M}(D))$ denotes the treewidth of the moralized graph of D. Note that this lower bound excludes all structural graph parameters as possible candidates for FPT algorithms. In a future project it might be interesting to analyze the parameterized complexity of Bayesian inference for parameters that describe the structure of the conditional probability tables.

Bibliography

[1] Florian Adriaens, Tijl De Bie, Aristides Gionis, Jefrey Lijffijt, Antonis Matakos, and Polina Rozenshtein. Relaxing the strong triadic closure problem for edge strength inference. *Data Mining and Knowledge Discovery*, 34(3):611–651, 2020. (Cited on pp. 33, 34, 252)

[2] Akanksha Agrawal. Fine-grained complexity of rainbow coloring and its variants. In *Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS '17)*, volume 83 of *LIPIcs*, pages 60:1–60:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. (Cited on p. 89)

[3] Akanksha Agrawal, Madhumita Kundu, Abhishek Sahu, Saket Saurabh, and Prafullkumar Tale. Parameterized complexity of maximum edge colorable subgraph. In *Proceedings of the 26th International Conference on Computing and Combinatorics (COCOON '20)*, volume 12273 of *Lecture Notes in Computer Science*, pages 615–626. Springer, 2020. (Cited on p. 113)

[4] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. (Cited on pp. 147, 152, 254)

[5] Alessandro Aloisio and Vahan Mkrtchyan. Algorithmic aspects of the maximum 2-edge-colorable subgraph problem. In *Proceedings of the 35th International Conference on Advanced Information Networking and Applications (AINA '21)*, volume 3 of *Lecture Notes in Networks and Systems*, pages 232–241. Springer, 2021. (Cited on p. 113)

[6] Noga Alon, Daniel Lokshtanov, and Saket Saurabh. Fast FAST. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP '09)*, volume 5555 of *Lecture Notes in Computer Science*, pages 49–58. Springer, 2009. (Cited on pp. 236, 241, 247)

[7] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. (Cited on pp. 184, 187, 236)

[8] Juan Ignacio Alonso-Barba, Luis de la Ossa, and José Miguel Puerta. Structural learning of Bayesian networks using local algorithms based on the space of orderings. *Soft Computing*, 15(10):1881–1895, 2011. (Cited on pp. 221, 222, 231, 247, 254)

[9] Francis R. Bach and Michael I. Jordan. Learning graphical models with mercer kernels. In *Proceedings of the 15th Annual Conference on Neural Information Processing Systems (NIPS '02)*, pages 1009–1016. MIT Press, 2002. (Cited on pp. 147, 254)

[10] Robert D Baller and Kelly K Richardson. The "dark side" of the strength of weak ties: The diffusion of suicidal thoughts. *Journal of Health and Social Behavior*, 50(3):261–276, 2009. (Cited on p. 33)

[11] Mehdi Behzad and Gary Chartrand. *Introduction to the Theory of Graphs*. Allyn and Bacon, 1972. (Cited on p. 129)

[12] Sebastian Böcker and Jan Baumbach. Cluster editing. In *Proceedings of the 9th Conference on Computability in Europe (CiE '13)*, volume 7921 of *Lecture Notes in Computer Science*, pages 33–44. Springer, 2013. (Cited on p. 64)

[13] Sebastian Böcker, Sebastian Briesemeister, and Gunnar W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60(2):316–334, 2011. (Cited on p. 64)

[14] Sebastian Böcker and Peter Damaschke. Even faster parameterized cluster deletion and cluster editing. *Information Processing Letters*, 111(14):717–721, 2011. (Cited on pp. 45, 56, 64)

[15] Hans L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1–21, 1993. (Cited on pp. 140, 151)

[16] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014. (Cited on p. 28)

[17] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011. (Cited on p. 27)

[18] Peter Bühlmann, Jonas Peters, and Jan Ernest. Cam: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 42(6):2526–2556, 2014. (Cited on pp. 147, 254)

[19] Laurent Bulteau, Niels Grüttemeier, Christian Komusiewicz, and Manuel Sorge. Your rugby mates don't need to know your colleagues: Triadic closure with edge colors. In *Proceedings of the 11th International Conference on Algorithms and Complexity (CIAC '19)*, volume 11485 of *Lecture Notes in Computer Science*, pages 99–111. Springer, 2019. (Cited on p. 8)

[20] Laurent Bulteau, Niels Grüttemeier, Christian Komusiewicz, and Manuel Sorge. Your rugby mates don't need to know your colleagues: Triadic closure with edge colors. *Journal of Computer and System Sciences*, 120:75–96, 2021. (Cited on p. 8)

[21] Andrea Bunt and Cristina Conati. Probabilistic student modelling to improve exploratory behaviour. *User Modeling and User-Adapted Interaction*, 13(3):269–309, 2003. (Cited on p. 145)

[22] Wray L. Buntine. Theory refinement on Bayesian networks. In *Proceedings of the 7th Conference in Uncertainty in Artificial Intelligence (UAI '91)*, pages 52–60. Morgan Kaufmann, 1991. (Cited on pp. 147, 152, 254)

[23] Yixin Cao and Yuping Ke. Improved kernels for edge modification problems. In *Proceedings of the 16th International Symposium on Parameterized and Exact Computation (IPEC '21), to appear*, 2021. (Cited on pp. 44, 45, 56, 93, 104, 110, 251, 252)

[24] Vicky Cattell. Poor people, poor places, and poor health: the mediating role of social networks and social capital. *Social Science & Medicine*, 52(10):1501–1516, 2001. (Cited on p. 33)

[25] Arthur Cayley. A theorem on trees. *The Quarterly Journal of Mathematics*, 23:376–378, 1889. (Cited on p. 200)

[26] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006. (Cited on p. 207)

[27] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010. (Cited on pp. 37, 44, 95)

[28] Jianer Chen and Jie Meng. A $2k$ kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 78(1):211–220, 2012. (Cited on p. 97)

[29] David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Proceedings of the 5th International Conference on Artificial Intelligence and Statistics (AISTATS '95)*, pages 121–130. Springer, 1995. (Cited on pp. 152, 218, 221, 226, 253)

[30] David Maxwell Chickering, David Heckerman, and Christopher Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004. (Cited on p. 255)

[31] Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *Proceedings of the 30th Intoernational Workshop on Graph-Theoretic Concepts in Computer Science (WG '04)*, volume 3353 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2004. (Cited on p. 119)

[32] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968. (Cited on pp. 152, 166, 192, 197, 199, 253, 254)

[33] Richard Cole and John E. Hopcroft. On edge coloring bipartite graphs. *SIAM Journal on Computing*, 11(3):540–546, 1982. (Cited on p. 111)

[34] Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990. (Cited on p. 151)

[35] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. (Cited on p. 24)

[36] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. (Cited on p. 140)

[37] James Cussens and Mark Bartlett. Advances in Bayesian network learning using integer programming. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, (UAI '13)*. AUAI Press, 2013. (Cited on p. 229)

[38] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. (Cited on pp. 19, 25, 27, 119, 164, 187, 213, 216, 219)

[39] Espen Dahl and Ira Malmberg-Heimonen. Social inequality and health: the role of social capital. *Sociology of Health & Illness*, 32(7):1102–1119, 2010. (Cited on p. 33)

[40] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009. (Cited on pp. 146, 151, 152, 166, 179, 191, 245)

[41] Sanjoy Dasgupta. Learning polytrees. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pages 134–141. Morgan Kaufmann, 1999. (Cited on pp. 152, 154, 155, 192, 197, 198, 199, 253)

[42] Christopher P. Diehl, Galileo Namata, and Lise Getoor. Relationship identification for social network discovery. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI '07)*, pages 546–552. AAAI Press, 2007. (Cited on p. 33)

[43] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and IDs. *ACM Transactions on Algorithms*, 11(2):13:1–13:20, 2014. (Cited on p. 108)

[44] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995. (Cited on p. 26)

[45] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theorertical Computer Science*, 141(1&2):109–131, 1995. (Cited on p. 26)

[46] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. (Cited on pp. 19, 25, 26, 105, 109, 206, 231)

[47] Arthur Conan Doyle. *A study in scarlet*. Ward, Lock & Co., 1888. (Cited on p. 145)

[48] Nils Jakob Eckstein, Niels Grüttemeier, Christian Komusiewicz, and Frank Sommer. Destroying multicolored paths and cycles in edge-colored graphs. *CoRR*, abs/2104.03138, 2021. (Cited on p. 7)

[49] Ward Edwards. Influence diagrams, Bayesian imperialism, and the collins case: an appeal to reason. *Cardozo Law Review*, 13:1025, 1991. (Cited on p. 145)

[50] Gal Elidan and Stephen Gould. Learning bounded treewidth Bayesian networks. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NIPS '08)*, pages 417–424. Curran Associates, Inc., 2008. (Cited on pp. 152, 156, 245)

[51] Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *Journal of Computer and System Sciences*, 84:1–10, 2017. (Cited on p. 217)

[52] Zhanpeng Fang and Jie Tang. Uncovering the formation of triadic closure in social networks. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI '15)*, pages 2062–2068. AAAI Press, 2015. (Cited on p. 34)

[53] Hamid Fehri, Ali Gooya, Yuanjun Lu, Erik Meijering, Simon A. Johnston, and Alejandro F. Frangi. Bayesian polytrees with learned deep features for multi-class cell segmentation. *IEEE Transactions on Image Processing*, 28(7):3246–3260, 2019. (Cited on pp. 199, 218)

[54] Uriel Feige. Faster FAST (feedback arc set in tournaments). *CoRR*, abs/0911.5094, 2009. (Cited on p. 247)

[55] Uriel Feige, Eran Ofek, and Udi Wieder. Approximating maximum edge coloring in multigraphs. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX '02)*, volume 2462 of *Lecture Notes in Computer Science*, pages 108–121. Springer, 2002. (Cited on pp. 112, 113, 116)

[56] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, and Yngve Villanger. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3):707–719, 2012. (Cited on p. 221)

[57] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. (Cited on p. 189)

[58] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. (Cited on pp. 19, 25)

[59] Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Fast local search algorithm for weighted feedback arc set in tournaments. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI '10)*. AAAI Press, 2010. (Cited on pp. 221, 235, 236, 239)

[60] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*, pages 142–151. SIAM, 2014. (Cited on pp. 208, 209, 213)

[61] Jacob Fox, Tim Roughgarden, C. Seshadhri, Fan Wei, and Nicole Wein. Finding cliques in social networks: A new distribution-free model. *SIAM Journal on Computing*, 49(2):448–464, 2020. (Cited on p. 252)

[62] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. (Cited on p. 217)

[63] Harold N. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC '83)*, pages 448–456. ACM, 1983. (Cited on p. 114)

[64] Tibor Gallai. Transitiv orientierbare Graphen. *Acta Mathematica Hungarica*, 18(1–2):25–66, 1967. (Cited on pp. 35, 37)

[65] Robert Ganian and Viktoriia Korchemna. The complexity of bayesian network learning: Revisiting the superstructure. In *Proceedings of the 34th Annual Conference on Advances in Neural Information Processing Systems (NeurIPS '21), to appear*, 2021. (Cited on pp. 147, 153, 156, 197, 198, 218, 253, 254)

[66] Yong Gao, Donovan R. Hare, and James Nastos. The cluster deletion problem for cographs. *Discrete Mathematics*, 313(23):2763–2771, 2013. (Cited on pp. 45, 51)

[67] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. (Cited on pp. 19, 25)

[68] Serge Gaspers, Eun Jung Kim, Sebastian Ordyniak, Saket Saurabh, and Stefan Szeider. Don't be strict in local search! In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI '12)*. AAAI Press, 2012. (Cited on p. 221)

[69] Serge Gaspers, Mikko Koivisto, Mathieu Liedloff, Sebastian Ordyniak, and Stefan Szeider. On finding optimal polytrees. *Theoretical Computer Science*, 592:49–58, 2015. (Cited on pp. 147, 152, 166, 192, 197, 199, 200, 209, 210, 218, 253, 254)

[70] Jozef Gemela. Financial analysis using Bayesian networks. *Applied Stochastic Models in Business and Industry*, 17(1):57–67, 2001. (Cited on p. 145)

[71] Petr A. Golovach, Pinar Heggernes, Athanasios L. Konstantinidis, Paloma T. Lima, and Charis Papadopoulos. Parameterized aspects of strong subgraph closure. *Algorithmica*, 82(7):2006–2038, 2020. (Cited on pp. 45, 56, 66, 110, 251)

[72] Mark Granovetter. The strength of weak ties. *The American Journal of Sociology*, 78:1360–1380, 1973. (Cited on pp. 19, 33)

[73] Mark Granovetter. The strength of weak ties: A network theory revisited. *Sociological Theory*, pages 201–233, 1983. (Cited on pp. 19, 33)

[74] Mark Granovetter. *Getting a job: A study of contacts and careers*. University of Chicago press, 2018. (Cited on p. 33)

[75] Niels Grüttemeier and Christian Komusiewicz. On the relation of strong triadic closure and cluster deletion. In *Proceedings of the 44th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '18)*, volume 11159 of *Lecture Notes in Computer Science*, pages 239–251. Springer, 2018. (Cited on p. 7)

[76] Niels Grüttemeier and Christian Komusiewicz. Learning Bayesian networks under sparsity constraints: A parameterized complexity analysis. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*, pages 4245–4251. ijcai.org, 2020. (Cited on p. 8)

[77] Niels Grüttemeier and Christian Komusiewicz. On the relation of strong triadic closure and cluster deletion. *Algorithmica*, 82(4):853–880, 2020. (Cited on pp. 7, 44, 56, 64, 94, 97, 251, 252)

[78] Niels Grüttemeier, Christian Komusiewicz, and Nils Morawietz. Maximum edge-colorable subgraph and strong triadic closure parameterized by distance to low-degree graphs. In *Proceedings of the 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT '20)*, volume 162 of *LIPIcs*, pages 26:1–26:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. (Cited on pp. 8, 139)

[79] Niels Grüttemeier, Christian Komusiewicz, and Nils Morawietz. Efficient Bayesian network structure learning via parameterized local search on topological orderings. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '21)*, pages 12328–12335. AAAI Press, 2021. (Cited on p. 9)

[80] Niels Grüttemeier, Christian Komusiewicz, Nils Morawietz, and Frank Sommer. String factorizations under various collision constraints. In *Proceedings of the 31st Annual Symposium on Combinatorial Pattern Matching (CPM '20)*, volume 161 of *LIPIcs*, pages 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. (Cited on p. 7)

[81] Niels Grüttemeier, Christian Komusiewicz, Nils Morawietz, and Frank Sommer. Preventing small $(\mathbf{s},\mathbf{t})$-cuts by protecting edges. In *Proceedings of the 47th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '21)*, volume 12911 of *Lecture Notes in Computer Science*, pages 143–155. Springer, 2021. (Cited on p. 7)

[82] Niels Grüttemeier, Christian Komusiewicz, Jannik Schestag, and Frank Sommer. Destroying bicolored P_3s by deleting few edges. *Discrete Mathematics & Theoretical Computer Science*, 23(1), 2021. (Cited on p. 7)

[83] 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)*, pages 4221–4227. International Joint Conferences on Artificial Intelligence Organization, 8 2021. (Cited on p. 9)

[84] Haipeng Guo and William Hsu. A survey of algorithms for real-time Bayesian network inference. In *Working Notes of the Joint AAAI/UAI/KDD Workshop*

on Real-Time Decision Support and Diagnosis Systems, 2002. (Cited on pp. 199, 218)

[85] Jiong Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8-10):718–726, 2009. (Cited on pp. 45, 56, 97)

[86] Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proceedings of the 1st Workshop on Parameterized and Exact Computation (IWPEC '04)*, volume 3162 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2004. (Cited on p. 112)

[87] András Hajnal and Endre Szemerédi. Proof of a conjecture of P. Erdős. *Combinatorial Theory and its Applications*, 2:601–623, 1970. (Cited on pp. 73, 81)

[88] S. Louis Hakimi and Oded Kariv. A generalization of edge-coloring in graphs. *Journal of Graph Theory*, 10(2):139–154, 1986. (Cited on p. 111)

[89] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995. (Cited on pp. 146, 147, 152, 254)

[90] Ian Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981. (Cited on pp. 52, 71, 111, 112)

[91] Wen-Lian Hsu and Tze-Heng Ma. Substitution decomposition on chordal graphs and applications. In *Proceedings of the 2nd International Symposium on Algorithms (ISA '91)*, volume 557 of *Lecture Notes in Computer Science*, pages 52–60. Springer, 1991. (Cited on p. 97)

[92] Biwei Huang, Kun Zhang, Yizhu Lin, Bernhard Schölkopf, and Clark Glymour. Generalized score functions for causal discovery. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18)*, pages 1551–1560. ACM, 2018. (Cited on pp. 147, 254)

[93] Hong Huang, Jie Tang, Sen Wu, Lu Liu, and Xiaoming Fu. Mining triadic closure patterns in social networks. In *Companion Volume of the 23rd International World Wide Web Conference (WWW '14)*, pages 499–504. ACM, 2014. (Cited on pp. 33, 45)

[94] Aapo Hyvärinen and Stephen M. Smith. Pairwise likelihood ratios for estimation of non-Gaussian structural equation models. *Journal of Machine Learning Research*, 14(1):111–152, 2013. (Cited on pp. 147, 254)

[95] Seiya Imoto, Takao Goto, and Satoru Miyano. Estimation of genetic networks and functional structures between genes by using Bayesian networks and non-parametric regression. In *Biocomputing 2002*, pages 175–186. World Scientific, 2001. (Cited on pp. 147, 254)

[96] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. (Cited on pp. 24, 29, 30, 73)

[97] Tommy R Jensen and Bjarne Toft. *Graph coloring problems*, volume 39. John Wiley & Sons, 2011. (Cited on p. 111)

[98] Marcin Jakub Kaminski and Lukasz Kowalik. Beyond the Vizing's bound for at most seven colors. *SIAM Journal on Discrete Mathematics*, 28(3):1334–1362, 2014. (Cited on p. 113)

[99] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972. (Cited on p. 105)

[100] Marek Karpinski and Warren Schudy. Faster algorithms for feedback arc set tournament, kemeny rank aggregation and betweenness tournament. In *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC '10)*, volume 6506 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2010. (Cited on p. 247)

[101] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. (Cited on pp. 223, 236)

[102] Henry A. Kierstead, Alexandr V. Kostochka, Marcelo Mydlarz, and Endre Szemerédi. A fast algorithm for equitable coloring. *Combinatorica*, 30(2):217–224, 2010. (Cited on pp. 73, 81)

[103] Jin H. Kim and Judea Pearl. Convince: A conversational inference consolidation engine. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(2):120–132, 1987. (Cited on p. 145)

[104] David G. Kirkpatrick and Pavol Hell. On the complexity of general graph factor problems. *SIAM Journal on Computing*, 12(3):601–609, 1983. (Cited on p. 181)

[105] Andreas Klärner, Markus Gamper, Sylvia Keim-Klärner, Irene Moor, Holger von der Lippe, and Nico Vonneilich. *Soziale Netzwerke und gesundheitliche Ungleichheiten: Eine neue Perspektive für die Forschung*. Springer Nature, 2020. (Cited on p. 33)

[106] Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Exploiting c-closure in kernelization algorithms for graph problems. In *Proceedings of the 28th Annual European Symposium on Algorithms (ESA '20)*, volume 173 of *LIPIcs*, pages 65:1–65:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. (Cited on p. 252)

[107] Tomohiro Koana, Christian Komusiewicz, and Frank Sommer. Essentially tight kernels for (weakly) closed graphs. In *Proceedings of the 32nd International Symposium on Algorithms and Computation (ISAAC '21)*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. to appear. (Cited on p. 252)

[108] Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann. Deconstructing intractability - A multivariate complexity analysis of interval constrained coloring. *Journal of Discrete Algorithms*, 9(1):137–151, 2011. (Cited on p. 219)

[109] Christian Komusiewicz and Johannes Uhlmann. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics*, 160(15):2259–2270, 2012. (Cited on pp. 45, 49, 50, 52, 55, 64)

[110] Athanasios L. Konstantinidis, Stavros D. Nikolopoulos, and Charis Papadopoulos. Strong triadic closure in cographs and graphs of low maximum degree. *Theoretical Computer Science*, 740:76–84, 2018. (Cited on pp. 40, 44, 45, 47, 49, 71, 90, 251)

[111] Athanasios L. Konstantinidis and Charis Papadopoulos. Maximizing the strong triadic closure in split graphs and proper interval graphs. *Discrete Applied Mathematics*, 285:79–95, 2020. (Cited on pp. 44, 46, 51, 251)

[112] Janne H. Korhonen and Pekka Parviainen. Exact learning of bounded treewidth Bayesian networks. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS '13)*, pages 370–378. JMLR.org, 2013. (Cited on pp. 147, 152, 154, 245, 253, 254)

[113] Janne H. Korhonen and Pekka Parviainen. Tractable Bayesian network structure learning with bounded vertex cover number. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS '15)*, pages 622–630. MIT Press, 2015. (Cited on pp. 147, 152, 153, 154, 159, 161, 165, 167, 198, 245, 253, 254)

[114] Timo JT Koski and John Noble. A review of Bayesian networks and structure learning. *Mathematica Applicanda*, 40(1), 2012. (Cited on p. 20)

[115] Adrian Kosowski. Approximating the maximum 2- and 3-edge-colorable subgraph problems. *Discrete Applied Mathematics*, 157(17):3593–3600, 2009. (Cited on p. 113)

[116] Lukasz Kowalik. Improved edge-coloring with three colors. *Theoretical Computer Science*, 410(38-40):3733–3742, 2009. (Cited on p. 113)

[117] Lukasz Kowalik, Juho Lauri, and Arkadiusz Socala. On the fine-grained complexity of rainbow coloring. *SIAM Journal on Discrete Mathematics*, 32:1672–1705, 2018. (Cited on pp. 73, 89)

[118] Lukasz Kowalik and Arkadiusz Socala. Tight lower bounds for list edge coloring. In *Proceedings of the 16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT '18)*, volume 101 of *LIPIcs*, pages 28:1–28:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. (Cited on pp. 72, 90)

[119] Daniel Krambrock. Algorithm Engineering für lokale Suchalgorithmen zum Lernen von Bayesnetzwerken. Bachelorarbeit, Philipps-Universität Marburg, 2021. (Cited on p. 246)

[120] Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms*, 10(4):20:1–20:15, 2014. (Cited on pp. 91, 209)

[121] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956. (Cited on p. 254)

[122] Mithilesh Kumar and Daniel Lokshtanov. A $2\ell k$ kernel for ℓ-component order connectivity. In *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC '16)*, pages 20:1–20:14, 2016. (Cited on pp. 115, 124)

[123] Johan Kwisthout, Hans L. Bodlaender, and Linda C. van der Gaag. The necessity of bounded treewidth for efficient inference in Bayesian networks. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI '10)*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 237–242. IOS Press, 2010. (Cited on p. 255)

[124] Van Bang Le. Gallai graphs and anti-Gallai graphs. *Discrete Mathematics*, 159(1-3):179–189, 1996. (Cited on pp. 35, 37, 110)

[125] Colin Lee and Peter van Beek. Metaheuristics for score-and-search Bayesian network structure learning. In *Proceedings of the 30th Canadian Conference on Artificial Intelligence (CCAI '17)*, volume 10233 of *Lecture Notes in Computer Science*, pages 129–141. Springer, 2017. (Cited on pp. 221, 222, 231, 254)

[126] Daniel Leven and Zvi Galil. NP completeness of finding the chromatic index of regular graphs. *Journal of Algorithms*, 4(1):35–44, 1983. (Cited on pp. 71, 90, 111)

[127] Nan Lin, Xiaolan Ye, and Walter M Ensel. Social support and depressed mood: A structural analysis. *Journal of Health and Social behavior*, pages 344–359, 1999. (Cited on p. 33)

[128] Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. *ACM Transactions on Algorithms*, 14(2):14:1–14:20, 2018. (Cited on pp. 208, 209, 214)

[129] Dániel Marx and Ildikó Schlotter. Parameterized complexity and local search approaches for the stable marriage problem with ties. *Algorithmica*, 58(1):170–187, 2010. (Cited on p. 221)

[130] Oliver Mason and Mark Verwoerd. Graph theory and networks in biology. *IET systems biology*, 1(2):89–119, 2007. (Cited on p. 19)

[131] Christopher Meek. Finding a path is harder than finding a tree. *Journal of Artificial Intelligence Research*, 15:383–389, 2001. (Cited on pp. 147, 152, 180, 253)

[132] Osman Ali Mian, Alexander Marx, and Jilles Vreeken. Discovering fully oriented causal networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI '21)*, pages 8975–8982. AAAI Press, 2021. (Cited on pp. 147, 254)

[133] Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science (FOCS '80)*, pages 17–27. IEEE Computer Society, 1980. (Cited on pp. 49, 167, 178)

[134] Nils Morawietz, Niels Grüttemeier, Christian Komusiewicz, and Frank Sommer. Colored cut games. In *Proceedings of the 40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS '20)*, volume 182 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. (Cited on p. 7)

[135] Nils Morawietz, Niels Grüttemeier, Christian Komusiewicz, and Frank Sommer. Refined parameterizations for computing colored cuts in edge-colored graphs. In *Proceedings of the 46th International Conference on Current Trends in Theory and Practice of Informatics (SOFSEM '20)*, volume 12011 of *Lecture Notes in Computer Science*, pages 248–259. Springer, 2020. (Cited on p. 7)

[136] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS '95)*, pages 182–191. IEEE Computer Society, 1995. (Cited on p. 187)

[137] Assaf Natanzon. Complexity and approximation of some graph modification problems. Master thesis, University of Tel-Aviv, 1999. (Cited on p. 50)

[138] Béla Neuendorf. On strong triadic closure with edge insertion. Bachelorarbeit, Philipps-Universität Marburg, 2020. (Cited on pp. 45, 65)

[139] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006. (Cited on pp. 19, 25)

[140] Stephan Olariu. Paw-free graphs. *Information Processing Letters*, 28(1):53–54, 1988. (Cited on pp. 47, 49)

[141] Sebastian Ordyniak and Stefan Szeider. Parameterized complexity results for exact Bayesian network structure learning. *Journal of Artificial Intelligence Research*, 46:263–302, 2013. (Cited on pp. 147, 148, 153, 156, 158, 188, 197, 201, 221, 222, 223, 225, 226, 253, 254)

[142] Sascha Ott and Satoru Miyano. Finding optimal gene networks using biological constraints. *Genome Informatics*, 14:124–133, 2003. (Cited on pp. 152, 160, 162, 172, 253)

[143] Judea Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th conference of the Cognitive Science Society (CSS '85)*, pages 15–17. UCLA, Computer Science Department, Irvine, CA, 1985. (Cited on pp. 19, 145)

[144] Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989. (Cited on pp. 145, 199, 218)

[145] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018. (Cited on pp. 19, 145)

[146] Judea Pearl and Thomas Verma. A theory of inferred causation. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR '91)*, pages 441–452. Morgan Kaufmann, 1991. (Cited on p. 255)

[147] Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences*, 67(4):757–771, 2003. (Cited on p. 189)

[148] Svatopluk Poljak. A note on stable sets and colorings of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 15(2):307–309, 1974. (Cited on p. 52)

[149] Olivier Pourret, Patrick Na, Bruce Marcot, et al. *Bayesian networks: a practical guide to applications*. John Wiley & Sons, 2008. (Cited on p. 145)

[150] Erich Prisner. Hereditary clique-helly graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 14:216–220, 1993. (Cited on p. 48)

[151] Fábio Protti, Maise Dantas da Silva, and Jayme Luiz Szwarcfiter. Applying modular decomposition to parameterized cluster editing problems. *Theory of Computing Systems*, 44(1):91–104, 2009. (Cited on p. 97)

[152] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004. (Cited on p. 72)

[153] Romeo Rizzi. Approximating the maximum 3-edge-colorable subgraph problem. *Discrete Mathematics*, 309(12):4166–4170, 2009. (Cited on p. 113)

[154] Elena Prieto Rodríguez. *Systematic kernelization in FPT algorithm design.* Phd thesis, The University of Newcastle, 2005. (Cited on p. 119)

[155] Rahmtin Rotabi, Krishna Kamath, Jon M. Kleinberg, and Aneesh Sharma. Detecting strong ties using network motifs. In *Proceedings of the 26th International Conference on World Wide Web Companion (WWW '17)*, pages 983–992. ACM, 2017. (Cited on p. 34)

[156] Polina Rozenshtein, Nikolaj Tatti, and Aristides Gionis. Inferring the strength of social ties: A community-driven approach. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*, pages 1017–1025. ACM, 2017. (Cited on pp. 33, 34, 45, 66)

[157] Javad Safaei, Ján Manuch, and Ladislav Stacho. Learning polytrees with constant number of roots from data. In *Proceedings of the 26th Australasian Joint Conference on Advances in Artificial Intelligence (AI '13)*, volume 8272 of *Lecture Notes in Computer Science*, pages 447–452. Springer, 2013. (Cited on pp. 199, 218)

[158] Najiba Sbihi. Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. *Discrete Mathematics*, 29(1):53–76, 1980. (Cited on p. 52)

[159] Mauro Scanagatta, Giorgio Corani, and Marco Zaffalon. Improved local search in Bayesian networks structure learning. In *Proceedings of the 3rd Workshop on Advanced Methodologies for Bayesian Networks (AMBN '17)*, volume 73 of *Proceedings of Machine Learning Research*, pages 45–56. PMLR, 2017. (Cited on pp. 221, 222, 231, 254)

[160] Mauro Scanagatta, Antonio Salmerón, and Fabio Stella. A survey on Bayesian network structure learning from data. *Progress in Artificial Intelligence*, 8(4):425–439, 2019. (Cited on pp. 20, 145, 255)

[161] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978. (Cited on pp. 147, 152, 254)

[162] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004. (Cited on p. 45)

[163] Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence (UAI '06)*. AUAI Press, 2006. (Cited on pp. 152, 160, 172, 253)

[164] Stavros Sintos and Panayiotis Tsaparas. Using strong triadic closure to characterize ties in social networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*, pages 1466–1475. ACM, 2014. (Cited on pp. 19, 33, 34, 35, 36, 37, 44, 45, 47, 56, 66, 69, 70, 90, 93, 94, 109, 251)

[165] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search, Second Edition*. Adaptive computation and machine learning. MIT Press, 2000. (Cited on p. 255)

[166] Michael Stiebitz, Diego Scheide, Bjarne Toft, and Lene M Favrholdt. *Graph edge coloring: Vizing's theorem and Goldberg's conjecture*, volume 75. John Wiley & Sons, 2012. (Cited on p. 116)

[167] Liping Sun. Two classes of perfect graphs. *Journal of Combinatorial Theory, Series B*, 53(2):273–292, 1991. (Cited on pp. 35, 37)

[168] Jie Tang, Tiancheng Lou, and Jon M. Kleinberg. Inferring social ties across heterogenous networks. In *Proceedings of the 5th International Conference on Web Search and Web Data Mining (WSDM '12)*, pages 743–752. ACM, 2012. (Cited on p. 33)

[169] Benjamin Taskar, Ming Fai Wong, Pieter Abbeel, and Daphne Koller. Link prediction in relational data. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems (NIPS '03)*, pages 659–666. MIT Press, 2003. (Cited on p. 19)

[170] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. (Cited on p. 74)

[171] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006. (Cited on pp. 221, 222)

[172] William Thomas Tutte. Lectures on matroids. *Journal of Research of the National Bureau of Standards*, 69:1–47, 1965. (Cited on p. 210)

[173] Shuichi Ueno, Yoji Kajitani, and Shin'ya Gotoh. On the nonseparating independent set problem and feedback set problem for graphs with no vertex degree exceeding three. *Discrete Mathematics*, 72(1-3):355–360, 1988. (Cited on p. 209)

[174] Peter van Beek and Hella-Franziska Hoffmann. Machine learning of Bayesian networks using constraint programming. In *Proceedings of the 21st Conference of Principles and Practice of Constraint Programming (CP '15)*, volume 9255 of *Lecture Notes in Computer Science*, pages 429–445. Springer, 2015. (Cited on p. 208)

[175] René van Bevern, Oxana Yu. Tsidulko, and Philipp Zschoche. Fixed-parameter algorithms for maximum-profit facility location under matroid constraints. In *Proceedings of the 11th International Conference on Algorithms and Complexity (CIAC '19)*, volume 11485 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 2019. (Cited on p. 45)

[176] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001. (Cited on p. 25)

[177] Nate Veldt. Faster deterministic approximation algorithms for correlation clustering and cluster deletion. *CoRR*, abs/2111.10699, 2021. (Cited on p. 46)

[178] Vadim G Vizing. On an estimate of the chromatic class of a p-graph. *Discret Analiz*, 3:25–30, 1964. (Cited on pp. 41, 71, 72, 111, 116)

[179] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC '12)*, pages 887–898. ACM, 2012. (Cited on pp. 200, 209)

[180] Rongjing Xiang, Jennifer Neville, and Monica Rogati. Modeling relationship strength in online social networks. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*, pages 981–990. ACM, 2010. (Cited on p. 33)

[181] Jaewon Yang and Jure Leskovec. Community-affiliation graph model for overlapping network community detection. In *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM '12)*, pages 1170–1175. IEEE Computer Society, 2012. (Cited on p. 33)

[182] CT Zahn, Jr. Approximating symmetric relations by equivalence relations. *Journal of the Society for Industrial and Applied Mathematics*, 12(4):840–847, 1964. (Cited on p. 64)