Elements of Efficient Data Reduction: Fractals, Diminishers, Weights and Neighborhoods

vorgelegt von
M. Sc.
Till Fluschnik

an der Fakultät IV — Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation

Promotionsausschuss:
Vorsitzender: Prof. Dr. Henning Sprekeler
Gutachter: Prof. Dr. Rolf Niedermeier
Gutachter: Prof. Dr. Bart M. P. Jansen
Gutachter: Prof. Dr. Stefan Szeider

Tag der wissenschaftlichen Aussprache: 29. November 2019

Berlin 2020

Zusammenfassung

Algorithmische Lösungsverfahren für Entscheidungs- oder Optimierungsprobleme nutzen oft eine Datenvorverarbeitung um eine Eingabeinstanz auf ihren wesentlichen Kern zu bringen. In der parametrisierten Komplexität ist diese Vorverarbeitung definiert als (Problem-) Kernelisierung. Sie gehört zu den meistgenutzten algorithmischen Werkzeugen. Die vorliegenen Dissertation befasst sich mit oberen und unteren Schranken für polynomielle Kernelisierung (d.h. für Kernelisierung die eine äquivalente Instanz liefert deren Größe polynomiell im Parameter beschränkt ist) sowie mit Varianten von Kernelisierung, z.B. für polynomzeitlösbare Probleme.

Wir entwickeln eine Familie von Graphen, die sogenannten „T-fractals“, die wie Fraktale eine selbstähnliche Struktur haben. Mittels dieser T-fractals beweisen wir untere Schranken für polynomielle Kernelisierung für einige Probleme bei denen es das Ziel ist, durch eine kleine Anzahl von Kantenlöschungen gewisse Knotendistanzen zu maximieren. Eines dieser Probleme ist das Length-Bounded Edge-Cut, dessen polynomielle Kernelisierbarkeit länger offen war.

Zusätzlich erweitern wir das sogenannte „Diminisher“-Konzept zum Ausschließen polynomieller Kernelisierungsvarianten unter der Annahme dass P $\neq$ NP ist. Bei den Varianten handelt es sich um Kernelisierungen, bei denen der Parameter in der resultierenden Instanz nicht vergrößert werden darf, sprich restriktive Kernelisierungen. Zunächst zeigen wir, dass das Diminisher-Konzept auf mehr als die zuvor bekannten Probleme anwendbar ist. Darüber hinaus erweitern wir das Konzept zur Anwendung auf weniger restriktive Kernelisierungen. Wir stellen dabei fest, dass das erweiterte Konzept zwar oft unter gewissen Annahmen nicht anwendbar ist, sich jedoch für den Bereich der polynomzeitlösbaren Probleme als nutzbar erweist. In diesem Bereich zeigen wir für ein Problem eine Kernelisierungsdichotomie bezüglich Laufzeit und Größe der Kernelisierung.

Zuletzt studieren wir klassische Graphprobleme unter Einschränkungen auf die Nachbarschaft des gesuchten Teilgraphen. Eines unserer Probleme ist das klassische Problem des Findens eines kürzesten Weges, der zwei ausgewählte Knoten verbindet. Für dieses Problem beweisen wir eine Hierarchie von strukturellen Graphparametern bezüglich polynomieller Kernelisierung. Einige der dabei erzielten polynomiellen Kernelisierungen nutzen eine Technik zur Reduzierung von Gewichten in Graphen. Wir präsentieren diese Technik und zeigen neue Berechnungsprobleme auf, für die die Technik anwendbar ist. Zuletzt zeigen wir noch für zwei weitere klasssiche Graphprobleme auf, wie Einschränkungen

auf die Nachbarschaft der Lösungsstruktur sich unter anderem auf polynomielle Kernelisierung auswirken können.

Abstract

Preprocessing and data reduction are basic algorithmic tools. In parameterized algorithmics, such preprocessing is defined by (problem) kernelization, where an equivalent instance (the kernel) is computed in polynomial time and its size can be upper-bounded only in a function of the parameter value of the input instance. In this thesis, we study lower and upper bounds on kernelization regarding polynomial-sized kernels as well as variants of kernelization like kernelization for polynomial-time solvable problems.

We introduce a new family of fractal-like graphs that we call T-fractals. Using these T-fractals in a common machinery for proving kernelization lower bounds, we refute (under some complexity-theoretic assumptions) the existence of polynomial kernels for some distance-related cut problems. One of these problems is the Length-Bounded Edge-Cut problem, for which the status of polynomial kernelization remained unknown for some time.

We underline and extend the usage of the so-called diminisher framework for excluding restrictive kernels of polynomial size under the assumption of $\mathrm{P} \neq \mathrm{NP}$. We prove that the original framework applies to more parameterized problems than previously known. In order to exclude less restrictive kernels of polynomial size, we extend the framework. We prove, however, that this extended framework does not apply to several parameterized problems. Yet, we prove that the framework applies to polynomial-time solvable problems, yielding first direct kernelization lower bounds in this class of problems. In addition, we prove a kernelization dichotomy regarding the running time and the size of the kernel for the polynomial-time solvable problem of computing the hyperbolicity of a graph.

Finally, we study classic graph problems under the so-called secluded concept, where constraints on the neighborhood of the subgraph in question are present. One of our problems is a secluded variant of the classic problem of finding a short path connecting two terminal vertices. For this problem, we prove a hierarchy of polynomial kernelizations regarding several structural graph parameters. Herein, we obtain polynomial kernels via the so-called "losing weight" technique. We outline this technique and prove that it applies to more problems than previously known. Eventually, we study two more problems in the secluded setup and prove how different constraints on the neighborhood lead to different complexity-theoretic classifications not only regarding polynomial kernelizability.

Preface

This thesis consists of parts of my research during my time at Technical University of Berlin, Berlin, Germany, in the group "Algorithmics and Computational Complexity", Faculty IV, (subsequently referred to as "our research group") from June 2015 to June 2019. I am grateful to the DFG, projects DAMM (NI 369/13) and TORE (NI 369/18), for their financial support throughout this period.

This thesis is based on and includes contents of the following scientific works I contributed to.

(1) *Fractals for kernelization lower bounds*, with Danny Hermelin, André Nichterlein, and Rolf Niedermeier. Journal: SIAM Journal on Discrete Mathematics [Flu+18a]. Conference: ICALP 2016 [Flu+16].

(2) *When Can Graph Hyperbolicity Be Computed in Linear Time?*, with Christian Komusiewicz, George B. Mertzios, André Nichterlein, Rolf Niedermeier, Nimrod Talmon. Journal: Algorithmica [Flu+19a]. Conference: WADS 2017 [Flu+17b].

(3) *The parameterized complexity of finding secluded solutions to some classical optimization problems on graphs*, with René van Bevern, George B. Mertzios, Hendrik Molter, Manuel Sorge, Ondřej Suchý. Journal: Discrete Optimization [Bev+18]. Conference: IPEC 2016 [Bev+17].

(4) *On the Computational Complexity of Length- and Neighborhood-Constrained Path Problems*, with Max-Jonathan Luckow. Journal: Information Processing Letters [LF20].

(5) *Kernelization Lower Bounds for Finding Constant-Size Subgraphs*, with George B. Mertzios and André Nichterlein. Conference: Computability in Europe (CiE'18) [FMN18].

(6) *Diminishable Parameterized Problems and Strict Polynomial Kernelization*, with Henning Fernau, Danny Hermelin, Andreas Krebs, Hendrik Molter, Rolf Niedermeier. Journal: Computability [Fer+20]. Conference: Computability in Europe (CiE'18) [Fer+18].

(7) *Parameterized algorithms and data reduction for the short secluded s-t-path problem*, with René van Bevern, Oxana Yu. Tsidulko. Journal: Networks [BFT20]. Conference: ATMOS 2018 [BFT18].

In the following, I outline chapter by chapter my contributions relating to the scientific work listed above.

Chapters 2 and 3: These chapters are based on contents from (1) I contributed to. Danny Hermelin, André Nichterlein, Rolf Niedermeier, and me came up with the idea to challenge the open problem regarding the polynomial kernelizability of LENGTH-BOUNDED EDGE-CUT. After several discussions on the problem, I eventually invented the T-fractal, and together we set up the framework, worked out the proofs, and wrote all paper versions. I had the idea for the NP-hardness proof of the planar variants of LENGTH-BOUNDED EDGE-CUT, which we finally worked out together, also with some support by Manuel Sorge. I presented (1) at ICALP'16.

Chapter 4: This chapter is based on contents from (6) I contributed to. This work was initiated during the research retreat of the Theoretical Computer Science group of the University of Tübingen in Sulz (Neckar), September 2016, where Henning Fernau, Danny Hermelin, Andreas Krebs, Hendrik Molter, Rolf Niedermeier, and me were participating. There, Henning Fernau proposed the project idea behind (6). All authors from (6) contributed equally to the results from (6) included in this chapter. My major contributions were to the TERMINAL STEINER TREE problem and to the MULTI-COMPONENT ANNOTATED Π problem. I presented (6) at CiE'18.

Chapter 5: This chapter is based on contents from (6) and from (5) I contributed to. During the project regarding (6), we wondered about the limits of diminishers. Together, we formalized strong diminishers, semi-strict kernels, observed the connection to the ETH, and wrote down the proofs. My major contributions to (6) are in this chapter. Later, Rolf Niedermeier wondered whether diminishers could apply also for polynomial-time solvable problems. Upon this question, George B. Mertzios, André Nichterlein, and me started the project (5). Together, we set up the framework, proved the results, and wrote down the papers. I presented (5) at CiE'18.

Chapter 6: This chapter is based on contents from (2) I contributed to. The project started at the research retreat 2016 of our research group, where the idea was proposed by André Nichterlein and Rolf Niedermeier. The kernelization upper bound was developed together by all authors. Christian Komusiewicz and André Nichterlein firstly had the idea for the lower bound reduction, I was contributing to the proof and to the adaption and proof for the reduction regarding the parameter maximum degree. I presented (2) at WADS'17.

Chapter 7: This chapter is based on unpublished content. At our research group's retreat in 2018, Matthias Bentert, René van Bevern, and André Nichterlein approached Oxana Yu. Tsidulko and me with their problem regarding the polynomial kernelizability of the MIN-POWER SYMMETRIC CONNECTIVITY problem, and explained their idea of making use of the technique due to Frank and Tardos [FT87]. Then, I proved an application of their technique to the problem yielding the polynomial kernel. We then decided to generalize this observation to some framework, which can be found in this chapter.

Chapter 8: This chapter is based on contents from (4) and (7) I contributed to. I proposed to study the problem of finding a two-terminal path in the "small secluded" setup and further three related setups to Max-Jonathan Luckow, who finally did his bachelor thesis on this topic. Upon this, I proposed my idea to focus on the small secluded setup for efficient kernelization (trade-offs) to René van Bevern and Oxana Yu. Tsidulko, eventually resulting in (7). Together we obtained the results from (7) contained in this thesis. My major contribution is the lower bound regarding the parameter $\text{fvs} + \ell$. Oxana Yu. Tsidulko presented (7) at ATMOS'18.

Chapter 9: This chapter is based on contents from (3) I contributed to. I came up with the idea to study (small) secluded classic graph optimization problems, which I proposed as a research topic at our research group's retreat 2016. All authors from (3) contributed equally to the project. Results on the secluded variants of the SEPARATOR problem were obtained together during the time of the retreat (except for the fixed-parameter tractability result). Ondřej Suchý firstly had the idea for the polynomial kernel for SECLUDED FEEDBACK VERTEX SET, I was contributing to the proof. I presented (3) at IPEC'16.

During the time of the thesis, I also contributed to the following.

- *The Parameterized Complexity of the Minimum Shared Edges Problem*, with Stefan Kratsch, Rolf Niedermeier, Manuel Sorge. Conference: FSTTCS 2015 [Flu+15]. Journal: Journal of Computer and System Sciences [Flu+19b].
- *The Minimum Shared Edges Problem on Grid-Like Graphs*, with Meike Hatzel, Steffen Härtlein, Hendrik Molter, Henning Seidler. Conference: WG 2016 [Flu+17a].
- *The complexity of routing with collision avoidance*, with Marco Morik, Manuel Sorge. Conference: FCT 2017 [FMS17]. Journal: Journal of Computer and System Sciences [FMS19].
- *Parameterized aspects of triangle enumeration*, with Matthias Bentert, André Nichterlein, Rolf Niedermeier. Conference: FCT 2017 [Ben+17b]. Journal: Journal of Computer and System Sciences [Ben+19].
- *The Minimum Shared Edges Problem on Planar Graphs*, with Manuel Sorge. Unpublished [FS16].
- *Fair Knapsack*, with Piotr Skowron, Mervin Triphaus, Kai Wilker. Conference: AAAI 2019 [Flu+19d].
- *A more fine-grained complexity analysis of finding the most vital edges for undirected shortest paths*, with Cristina Bazgan, André Nichterlein, Rolf Niedermeier, Maximilian Stahlberg. Journal: Networks [Baz+19].
- *Exact mean computation in dynamic time warping spaces*, with Markus Brill, Vincent Froese, Brijnesh J. Jain, Rolf Niedermeier, David Schultz. Conference: SDM 2018 [Bri+18]. Journal: Data Mining and Knowledge Discovery [Bri+19].
- *Temporal Graph Classes: A View Through Temporal Separators*, with Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, Philipp Zschoche. Conference: WG 2018 [Flu+18b]. Journal: Theoretical Computer Science [Flu+20].
- *The Complexity of Finding Small Separators in Temporal Graphs*, with Philipp Zschoche, Hendrik Molter, Rolf Niedermeier. Conference: MFCS 2018 [Zsc+18]. Journal: Journal of Computer and System Sciences [Zsc+20].
- *On $(1+\varepsilon)$-approximate problem kernels for the Rural Postman Problem*, with René van Bevern, Oxana Yu. Tsidulko. Conference: MOTOR'19 [BFT19].
- *Multistage Vertex Cover*, with Rolf Niedermeier, Valentin Rohm, Philipp Zschoche. Conference: IPEC 2019 [Flu+19c].

Acknowledgments. I want to thank all my coauthors (listed above) for the pleasant and effective collaboration. I further want to thank all my current and former colleagues for the pleasant time and fruitful discussions (in lexicographic order): Matthias Bentert, René van Bevern, Robert Bredereck, Markus Brill, Jiehua Chen, Piotr Faliszewski, Vincent Froese, Klaus Heeger, Anne-Sophie Himmel, Andrzej Kaczmarczyk, Dušan Knop, Leon Kellerhals, Christian Komusiewicz, Junjie Luo, Hendrik Molter, André Nichterlein, Malte Renken, Piotr Skowron, Manuel Sorge, Nimrod Talmon, Oxana Yu. Tsidulko, Philipp Zschoche.

Special thanks also go to Matthias Bentert, Anne-Sophie Himmel, Leon Kellerhals, Hendrik Molter, André Nichterlein, Rolf Niedermeier, and Anne Röhrborn, for their support during the time of writing this thesis.

Very special thanks go to my supervisor, Rolf Niedermeier, for all his support during the whole time.

Finally, I am grateful to the referees, Bart M. P. Jansen, Rolf Niedermeier, and Stefan Szeider, for their time and constructive feedback.

I dedicate this thesis to my beloved parents, Monika and Georg.

Table of Contents

Chapter 1.

Introduction

When we are asked for a given a set of numbers to determine the median, probably we would first sort the numbers in some, say ascending order. When we are asked for a given social network to compute a large subset of persons that are pairwise unknown to each other, probably we would first collect all persons knowing no one. When we are asked for a given street network with two designated terminals to compute a short path connecting the terminals, probably we would first delete all parts from the network that are obviously too far away from any of our terminals. In many problem-solving tasks, preprocessing the given input is a natural, fundamental algorithmic tool. As we have seen, such preprocessing ranges from sorting parts of the input to any form of data reduction, e.g., taking parts of the input already into our solution or deleting parts of the input that are irrelevant for solving the task. In this work, our central concern is efficient preprocessing for parameterized problems:

Definition 1.1. A *parameterized problem* $L \subseteq \{(x,k) \in \Sigma^* \times \mathbb{N}\}$ is a set of *instances* $(x,k) \in \Sigma^* \times \mathbb{N}$, where $x \in \Sigma^*$ for a finite alphabet Σ and $k \in \mathbb{N}$ is referred to as the *parameter*.

Basically, the task is to decide whether a given input instance $(x,k) \in \Sigma^* \times \mathbb{N}$ (e.g., an undirected graph and an integer k) is a yes-instance for L (e.g., deciding whether there are k vertices covering all edges of the graph), that is, whether $(x,k) \in L$.

In the field of parameterized complexity analysis and algorithmics, *kernelization*, coined by Downey and Fellows [DF95b], is the main mathematical concept for provably efficient preprocessing of computationally hard problems to their "computationally hard core" (the *kernel*).

Definition 1.2. A *problem kernelization* for a parameterized problem L is an algorithm that, given an instance (x, k) of L, computes in polynomial time an instance (x', k') of L (the *problem kernel*) such that

(i) $(x, k) \in L$ if and only if $(x', k') \in L$, and

(ii) $|x'| + k' \leq f(k)$ for some computable function f only depending on k.

We say that f measures the *size* of the problem kernel, and if $f \in k^{O(1)}$, we say that L admits a *polynomial* problem kernel.

Throughout this thesis, we will use kernelization and kernel for short. In the first formal definition of kernelization [DFS97, Definition 4.7], condition (ii) of Definition 1.2 was different:

(ii) $k' \leq k$ and $|x'| \leq f(k)$ for some computable function f only depending on k.

In accordance with Abu-Khzam and Fernau [AF06] we refer to this first variant of kernelization as *proper* kernelization. As we will see in Part II of this work, the modification of (ii) in the definition of kernelization to proper kernelization makes some crucial difference in the study of kernelization lower bounds.

Kernelization has been extensively studied (see, e.g., [FS14, GN07, Kra14, LMS12]) and it has great potential for delivering practically relevant algorithms (see, e.g., [Iwa17]). Very recently, Fomin et al. [Fom+19] published a book on kernelization.

Often, kernelization consists of *data reduction rules* followed by an analysis of the size of the obtained instance. A reduction rule is, in a nutshell, an algorithm that turns the input instance into an equivalent instance while guaranteeing for some properties to hold. A classic example for this is given by the NP-complete graph problem Vertex Cover (the “drosophila” of parameterized algorithmics; see [Fel+18] for a recent survey), the problem of deciding for a given undirected graph and an integer k whether k vertices suffice to cover all edges of the graph. A polynomial kernelization for Vertex Cover yielding a kernel with $O(k^2)$ vertices, going back to the work of Buss and Goldsmith [BG93], consists of the following two data reduction rules:

(1) Delete every vertex with no edges incident to it from the graph (since such a vertex can cover no edges).

(2) As long as there is a vertex v adjacent to at least $k + 1$ other vertices, delete v and decrease k by one (we have to take v into every k-sized set covering all edges).

When none of the reduction rules are applicable, the graph still contains at least one edge, and k is still positive, then either the number of vertices is in $O(k^2)$ or we are facing a no-instance. Note that Vertex Cover admits even a kernel with at most $2k$ vertices [CKJ01] employing more involved data reduction rules. Generally speaking, many polynomial kernels require more sophisticated reduction rules, hence considered as “the art of preprocessing”.

Kernelization is not only an important algorithmic tool, but also provides an alternative definition of *fixed-parameter tractability*[1]:

Lemma 1.1 ([Cai+97, DFS97]). *Let L be a decidable parameterized problem. Then there is a computable function f such that each instance (x, k) of L can be decided in $f(k) \cdot |x|^{O(1)}$ time (that is, L is fixed-parameter tractable) if and only if there is a computable function g such that L admits a kernel of size $g(k)$.*

An algorithm with running time $f(k) \cdot |x|^{O(1)}$ for a parameterized problem L implies that L has a kernel of size $f(k)+k$, but in the reverse direction one cannot always take the same function f. For example, Vertex Cover parameterized by the solution size k admits a kernel of $O(k^2)$ size but no algorithm running in $O(k^2) \cdot |x|^{O(1)}$ time assuming $\text{P} \neq \text{NP}$. Typically one wishes to minimize the *size* $f(k)$ of the kernel. This goal leads to the question of what is the smallest possible kernel size for a given parameterized problem. In particular, do all fixed-parameter tractable problems have small kernels, say, of polynomial size?

The latter question was answered negatively by Bodlaender et al. [Bod+09] using a result of Fortnow and Santhanam [FS11] to show that various fixed-parameter tractable problems, for instance Longest Path parameterized by the solution size, admit no polynomial kernel unless $\text{coNP} \subseteq \text{NP}_{/\text{poly}}$ (which implies a collapse of the polynomial hierarchy to its third level). This led to the exclusion of polynomial kernels for various further parameterized problems, and to several extending works building on the framework of Bodlaender et al. (see Figure 1.1 for an overview of the brief history of kernelization lower bounds). In Section 1.1.1, we provide details for kernelization-lower-bound frameworks.

Knowing about kernelization and its lower-bound frameworks, Downey et al.’s [DFS97, Example 1.2] “An Encounter with a Computational Biologist” could have been as follows:

> *“About ten years ago some computer scientists came by and said they had heard that we have some really cool problems. They showed that the problems admit no polynomial kernels unless coNP $\subseteq$ NP$_{/poly}$ and went away!”*

[1] Downey et al. [DFS97, Lemma 4.8] proved the equivalence of being kernelizable to being fixed-parameter tractable, where they use one direction from Cai et al. [Cai+97].

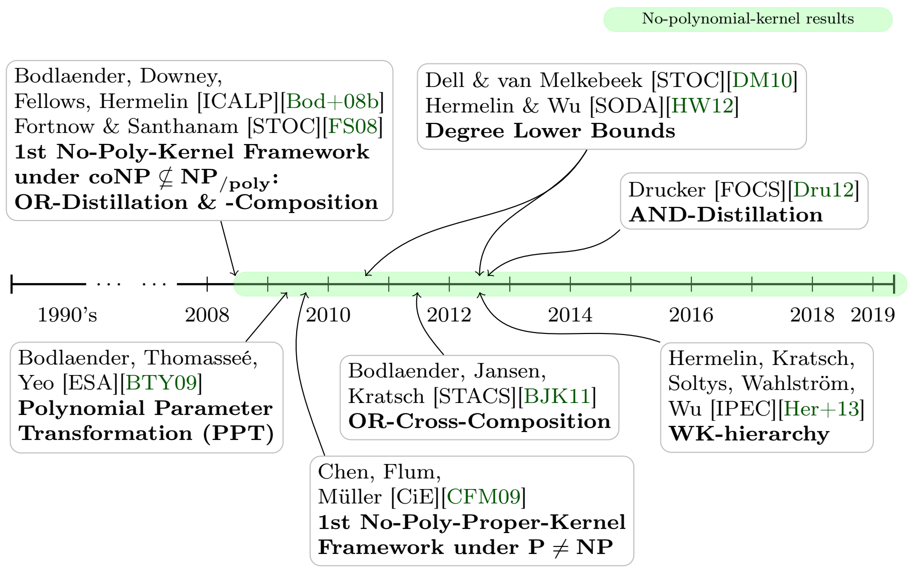

No-polynomial-kernel results
Bodlaender, Downey, Fellows, Hermelin [ICALP][Bod+08b] Fortnow & Santhanam [STOC][FS08] 1st No-Poly-Kernel Framework under coNP ⊈ NP/poly: OR-Distillation & -Composition
Dell & van Melkebeek [STOC][DM10] Hermelin & Wu [SODA][HW12] Degree Lower Bounds
Drucker [FOCS][Dru12] AND-Distillation
1990's
2008
2010
2012
2014
2016
2018
2019
Bodlaender, Thomasseé, Yeo [ESA][BTY09] Polynomial Parameter Transformation (PPT)
Bodlaender, Jansen, Kratsch [STACS][BJK11] OR-Cross-Composition
Hermelin, Kratsch, Soltys, Wahlström, Wu [IPEC][Her+13] WK-hierarchy
Chen, Flum, Müller [CiE][CFM09] 1st No-Poly-Proper-Kernel Framework under P ≠ NP

Figure 1.1.: Overview of the brief history of kernelization lower bounds.

If this interaction had been even more recent, then the computer scientist possibly would have proved the problem to be WK[1]-hard (and hence conjectured to admit no so-called Turing kernelization of polynomial size), or to presumably admit no polynomial-size approximate kernelization scheme.

Facing kernelization lower bounds, researchers have studied variants, mostly relaxations of kernelization such as *partial kernelization* [Bet+11b], *bikernelization* [Alo+11], and *Turing kernelization* [Bin+12, Sch+12]. See Figure 1.2 for an overview of the brief history of some variants of kernelization. Note that while kernelization is a special case of, e.g., partial kernelization or bikernelization, Turing kernelization is an algorithm that *decides* the input instance for the given parameterized problem and hence is conceptually different to kernelization. Indeed, Turing kernelization is more powerful in the following sense: if a parameterized problem admits a polynomial kernelization, then it admits a polynomial Turing kernelization, but the reverse is presumably not true. In Section 1.1.2, we elaborate on these and several other variants of kernelization in more detail.

In this thesis, we aim for lower and upper bounds on efficient kernelization and its variants for NP-hard and polynomial-time solvable graph problems. Thereby,

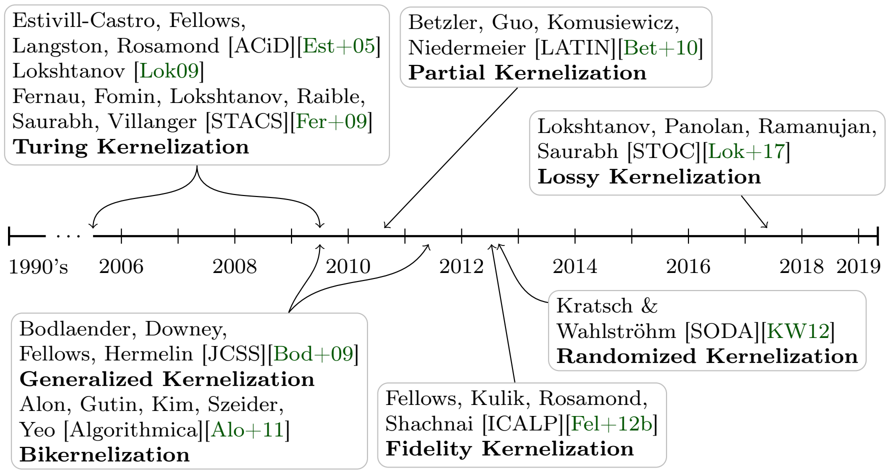

Estivill-Castro, Fellows, Langston, Rosamond [ACiD][Est+05]
Lokshtanov [Lok09]
Fernau, Fomin, Lokshtanov, Raible, Saurabh, Villanger [STACS][Fer+09]
Turing Kernelization
Betzler, Guo, Komusiewicz, Niedermeier [LATIN][Bet+10]
Partial Kernelization
Lokshtanov, Panolan, Ramanujan, Saurabh [STOC][Lok+17]
Lossy Kernelization
1990's
2006
2008
2010
2012
2014
2016
2018
2019
Bodlaender, Downey, Fellows, Hermelin [JCSS][Bod+09]
Generalized Kernelization
Alon, Gutin, Kim, Szeider, Yeo [Algorithmica][Alo+11]
Bikernelization
Kratsch & Wahlströhm [SODA][KW12]
Randomized Kernelization
Fellows, Kulik, Rosamond, Shachnai [ICALP][Fel+12b]
Fidelity Kernelization

Figure 1.2.: Overview of the brief history of some variants of kernelization.

we will discover that fractals, diminishers, and weights and neighborhoods form elements in the field of efficient data reduction.

Contributions of this Thesis. In each chapter, in the beginning the contributions therein are outlined. From a high-level perspective, this work contributes the following.

Part I: Specific properties of graphs admitting a self-similar, fractal-like structure can be exploited for developing OR-(cross-)composition to exclude the existence of polynomial kernels assuming coNP $\not\subseteq$ NP$_{/\text{poly}}$, even for restricted inputs like planar graphs. Using fractal-like graphs, like the T-fractal, in compositions allows for proving polynomial kernel lower bounds for three distance-related edge cut problems like the LENGTH-BOUNDED EDGE-CUT problem, hence answering an open question for the latter problem [GT11].

Part II: A framework by Chen et al. [CFM11] for excluding proper kernels of polynomial size assuming P $\neq$ NP applies to more problems than previously known. However, strengthening the framework for excluding a more relaxed variant of proper kernelization of polynomial size fails for several problems assuming the Exponential Time Hypothesis to hold (Hypothesis 1.10). Yet, the stronger framework enables to exclude several combinations of fast and small proper kernelizations for parameterized problems that are solvable in polynomial time, assuming popular complexity-theoretic conjectures to hold. Still, this framework does not allow for exclusions of fast kernelization of any polynomial size, while such an exclusion can be obtained through running-time lower bounds based on the Strong Exponential Time Hypothesis (Hypothesis 1.11).

Part III: For graph problems related to finding subgraphs fulfilling some property (e.g., being a path connecting two designated terminals), additionally restricting the number of vertices neighboring the subgraph is not only a natural problem modification, but it also allows for interesting kernelization results. While asking for subgraphs with small closed neighborhood allows to transfer for positive kernelization results, polynomial kernelization of finding small-sized subgraphs with small-sized open neighborhoods is more often excluded unless $\text{coNP} \subseteq \text{NP}_{/\text{poly}}$. In the case of finding a short path connecting two designated terminals with small-sized open neighborhood, polynomial kernelization can be obtained by first reducing to a weighted version of the problem and then employing the losing-weight technique due to Frank and Tardos [FT87].

The outline of this work is presented in Figure 1.3. In the present chapter, in Section 1.1 we provide notation and results on parameterized data reduction, and in Section 1.2 we describe our notation and assumptions, and give basic definitions and facts. Moreover, we provide a list of decision and optimization problems relevant to this thesis in Appendix A, and a list of open problems appearing in this thesis in Appendix B.

1.1. Invitation to Parameterized Data Reduction

In this section, we provide basic definitions, notations, and results from parameterized data reduction, in particular kernelization, that are important for this work. In Section 1.1.1, we give an overview of the theory of kernelization lower bounds. In Section 1.1.2, we explain variants of kernelization.

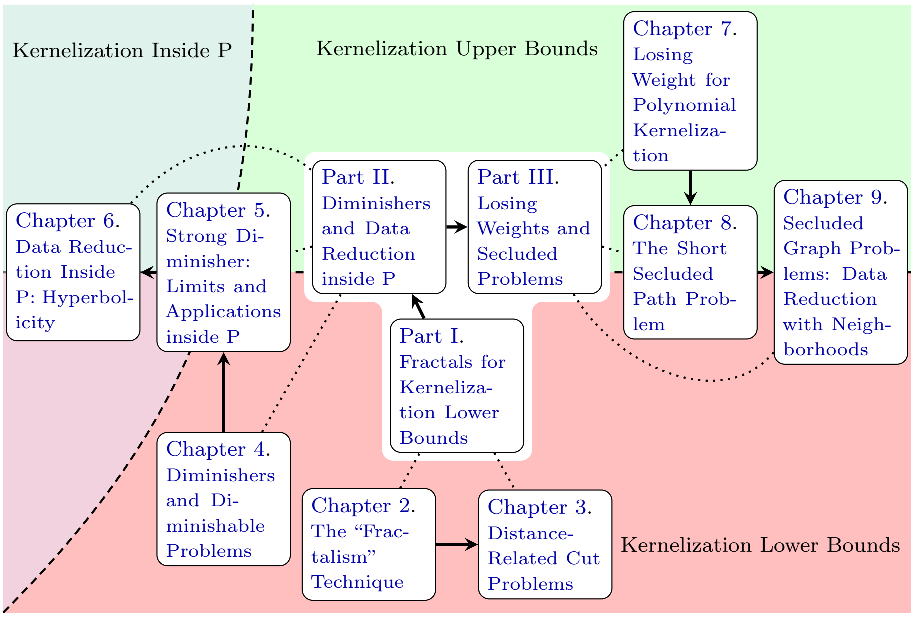

Kernelization Inside P
Kernelization Upper Bounds
Kernelization Lower Bounds
Chapter 7. Losing Weight for Polynomial Kernelization
Part II. Diminishers and Data Reduction inside P
Part III. Losing Weights and Secluded Problems
Chapter 6. Data Reduction Inside P: Hyperbolicity
Chapter 5. Strong Diminisher: Limits and Applications inside P
Chapter 8. The Short Secluded Path Problem
Chapter 9. Secluded Graph Problems: Data Reduction with Neighborhoods
Part I. Fractals for Kernelization Lower Bounds
Chapter 4. Diminishers and Diminishable Problems
Chapter 2. The "Fractalism" Technique
Chapter 3. Distance-Related Cut Problems

Figure 1.3.: Illustrative outline of this thesis. While arrows indicate the order of appearance in this thesis, dotted lines indicate membership of chapters in parts. Chapters are arranged according to their correspondence to the fields kernelization lower and upper bounds, and kernelization of polynomial-time solvable problems (inside P).

1.1.1. Kernelization Lower Bounds

The first framework for excluding polynomial kernels assuming that coNP $\not\subseteq$ NP$_{/\text{poly}}$ by Bodlaender et al. [Bod+09] builds on OR-compositions:

Definition 1.3 ([Bod+09])**.** An *OR-composition* for a parameterized problem L is an algorithm that takes p instances $(x_1, k), \ldots, (x_p, k)$, where $(x_i, k) \in \Sigma^* \times \mathbb{N}$ for all $i \in \{1, \ldots, p\}$, and constructs in time polynomial in $\sum_{i=1}^{p}(|x_i| + k)$ an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that (i) $(x', k') \in L \iff (x_i, k) \in L$ for some $i \in \{1, \ldots, p\}$ and (ii) $k' \in k^{O(1)}$.

Bodlaender et al. [Bod+09] together with Fortnow and Santhanam [FS11] proved the following.

Proposition 1.2 ([Bod+09, FS11])**.** *If a parameterized problem L whose unparameterized problem is NP-complete admits an OR-composition and a polynomial kernelization, then $NP \subseteq coNP_{/poly}$.*

Yap [Yap83] proved that $\mathrm{NP} \subseteq \mathrm{coNP}_{/\mathrm{poly}}$ is equivalent to $\mathrm{coNP} \subseteq \mathrm{NP}_{/\mathrm{poly}}$ (see Lemma 1.9), and that if either of them is true, then the polynomial hierarchy collapses to its third level (see Lemma 1.8).

That replacing "OR" by "AND" in Proposition 1.2 (and hence in Definition 1.3 by restating (i) by "$(x', k') \in L \iff (x_i, k) \in L$ for *every* $i \in \{1, \ldots, p\}$") also holds true was proven by Drucker [Dru15].

Bodlaender et al. [BJK14] extended the notion of OR/AND-composition to OR/AND-cross-composition. The definition of OR/AND-cross-composition uses the following.

Definition 1.4 ([BJK14])**.** An equivalence relation $\mathcal{R}$ on the instances of an NP-hard problem $L \subseteq \Sigma^*$ is a *polynomial equivalence relation* if

(i) one can decide for any two instances x, x' in time polynomial in $|x| + |x'|$ whether they belong to the same equivalence class, and

(ii) for any finite set S of instances, $\mathcal{R}$ partitions S into $(\max_{x \in S} |x|)^{O(1)}$ equivalence classes.

Definition 1.5 ([BJK14])**.** Given an NP-hard problem $L \subseteq \Sigma^*$, a parameterized problem $L' \subseteq \Sigma^* \times \mathbb{N}$, and a polynomial equivalence relation $\mathcal{R}$ on the instances of L, an *OR-cross-composition* of L into L' (with respect to $\mathcal{R}$) is an algorithm that takes p $\mathcal{R}$-equivalent instances $x_1, \ldots, x_p$ of L and constructs in time polynomial in $\sum_{i=1}^{p} |x_i|$ an instance (x, k) of L' such that

(i) k is polynomially upper-bounded in $\max_{1 \leq i \leq p} |x_i| + \log(p)$ and

(ii) $(x, k) \in L' \iff x_i \in L$ for at least one $i' \in \{1, \ldots, p\}$. ("OR")

An *AND-cross-composition* is an OR-cross-composition where (ii) is replaced by

(ii) $(x, k) \in L' \iff x_i \in L$ for every $i \in \{1, \ldots, p\}$. ("AND")

We remark that we can assume that $p = 2^q$ for some $q \in \mathbb{N}$ since we can pad the list of instances by copies of any instance to reach a power of two. For cross-compositions, the connection to $\mathrm{coNP} \subseteq \mathrm{NP}_{/\mathrm{poly}}$ is as before.

Proposition 1.3 ([BJK14])**.** *If an NP-hard problem L OR-cross-composes into a parameterized problem L', then L' does not admit a polynomial kernel with respect to its parameterization, unless $coNP \subseteq NP_{/poly}$.*

In this work, we mostly employ OR-cross-compositions and Proposition 1.3 to prove (conditional) kernelization lower bounds.

For "kernelization hardness", the following works like polynomial-time many-one reductions work for NP-hard problems.

Definition 1.6 ([BTY11])**.** Given two parameterized problems $L, L' \subseteq \Sigma^* \times \mathbb{N}$, a *polynomial parameter transformation* from L to L' is an algorithm that, given an instance (x, k) of L, computes in polynomial time an instance (x', k') of L' such that (i) $(x, k) \in L \iff (x', k') \in L'$ and (ii) $k' \leq k^{O(1)}$.

Polynomial parameter transformations are hence another tool to conditionally exclude polynomial kernels.

Proposition 1.4 ([BTY11])**.** *Let $L, L' \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems such that there is a polynomial parameter transformation from L to L', and the unparameterized version of L and of L' is NP-complete and contained in NP, respectively. If L' admits a polynomial kernel, then also L admits a polynomial kernel.*

A direct corollary of Proposition 1.4 is that (under the same conditions as in Proposition 1.4) if L admits no polynomial kernel unless coNP $\subseteq$ NP$_{/\text{poly}}$, then L' admits no polynomial kernel unless coNP $\subseteq$ NP$_{/\text{poly}}$. We refer to Dom et al. [DLS14] for an extensive demonstration of polynomial parameter transformations. Polynomial parameter transformations also transfer so-called WK[1]-hardness (see next Section 1.1.2).

1.1.2. Variants of Kernelization

In this section, we present some variants of kernelization, namely proper kernelization, bikernelization, partial (bi-)kernelization, and Turing kernelization.

Proper Kernels. As said, the first definition of kernelization [DFS97] was the one of proper kernelization [AF06].

Definition 1.7. A *proper kernelization* is an algorithm that, given an instance (x, k) of a parameterized problem L, computes in polynomial time an instance (x', k') of L (the *proper kernel*) such that (i) $(x, k) \in L \iff (x', k') \in L$, (ii) $|x'| \leq f(k)$ for some computable function f only depending on k, and (iii) $k' \leq k$.

We say that f measures the *size* of the proper kernel, and if $f \in k^{O(1)}$, we say that L admits a *polynomial* proper kernel.

Indeed, proper kernelization is found in several works (see, e.g., [Fel+12b, GN07]) including recent ones (see, e.g., [Lin+17, XK17]) as the definition of kernelization. We will study proper kernelization only in Part II, Chapter 5, in the context of polynomial-time solvable, parameterized problems.

In Chapters 4 and 5, Part II, we will also study relaxations of proper kernelization where we replace (iii) in Definition 1.7 by "$k' \leq k+c$" (we will call this *strict* kernelization) and by "$k' \leq c \cdot k$" (we will call this *semi-strict* kernelization), where c is some constant.

Bikernels and Partial Kernels. Bikernelization, also known as general kernelization [Bod+09], is a natural generalization of kernelization:

Definition 1.8 ([Alo+11])**.** A *bikernelization* is an algorithm that, given an instance (x, k) of a parameterized problem L, computes in polynomial time an instance (x', k') of some parameterized problem L' (the *bikernel*) such that (i) $(x, k) \in L \iff (x', k') \in L'$, and (ii) $|x'| + k' \leq f(k)$ for some computable function f only depending on k.

That is, a bikernelization with $L = L'$ is a kernelization, and also notions like polynomial bikernelization are defined as expected. Note that if we drop the requirements on the unparameterized versions from Proposition 1.4, then L' admitting a polynomial kernel implies L to admit a polynomial bikernel.

A partial kernelization is yet another generalization of kernelization, where we want to bound some *dimension* of each input instance of our problem. We give a definition of partial bikernelization, generalizing the definition of Betzler et al. [Bet+11b].

Definition 1.9 ([Bet+11b])**.** Let L, L' be two parameterized problems. Let $\dim\colon \Sigma^* \to \mathbb{N}$ be a computable function. A *partial bikernelization* (regarding dim) is an algorithm that, given an instance (x, k) of L, computes in polynomial time an instance (x', k') of L' (the *partial bikernel*) such that (i) $(x, k) \in L \iff (x', k') \in L'$, and (ii) $\dim(x') + k' \leq f(k)$ for some computable function f only depending on k.

If $L = L'$, then it is called partial kernelization. Note that partial bikernelization with $\dim(x) = |x|$ is a bikernelization. In Part III, Chapter 8, we will present partial bikernels of polynomial size (again, f is referred to as the size), where dim will count the number of vertices and edges of a graph.

Turing Kernelization. Intuitively, a Turing kernelization is a polynomial-time algorithm that has access to an oracle that can decide in constant time instances of small size, that is, of size bounded by some computable function in the parameter. Formally, it is defined as follows (our definition is due to Binkele-Raible et al. [Bin+12], see also [Kra14]).

Definition 1.10 ([Bin+12, Kra14])**.** A *Turing kernelization* for a parameterized problem L is an algorithm that decides whether any input instance $(x, k) \in L$ in time polynomial in $|x| + k$ having access to an $f(k)$-oracle, where an $f(k)$-oracle for L is an oracle that decides whether any input instance (x, k) with $|x| + k \leq f(k)$ is contained in L in constant time.

A Turing kernelization is a *polynomial* Turing kernelization if $f(k) \in k^{O(1)}$.

Estivill-Castro et al. [Est+05] firstly came up with the idea of Turing kernelization, which was then restated by Guo and Niedermeier [GN07] and Fellows and Guo [Bod+08a], where Guo and Niedermeier named it Turing kernelization, and Fellows and Guo named it cheat kernelization. The first definition of Turing kernelization was given by Binkele-Raible et al. [Bin+12], see also [Lok09].

A Turing kernelization can, during its computation, make decisions based on the answers of the oracle calls, which is referred to as *adaptive* behavior. While the first polynomial Turing kernel due to Binkele-Raible et al. [Bin+12] is non-adaptive, Jansen [Jan17] made use of this adaptive behavior to develop polynomial Turing kernelization for the Longest Path problem on restricted input graphs parameterized by the size k of the path.

Hermelin et al. [Her+15] conjecture that Longest Path on general graphs parameterized by k admits no polynomial Turing kernelization. In fact, they proved the multicolored version of the problem to be WK[1]-complete, where WK[1] is the first complexity class in the WK-hierarchy of problems conjectured to admit no polynomial Turing kernelization [Her+15]. A problem L is WK[1]-hard if for all parameterized problems L' in WK[1] there is a polynomial parameter transformation from L' to L. If there is a polynomial parameter transformation from a WK[1]-hard parameterized problem L' to some parameterized problem L, where both unparameterized versions of the problems are NP-complete, and there is a polynomial Turing kernelization for L, then also L' admits a polynomial Turing kernelization.

Finally, Witteveen et al. [WBT19] recently studied *unconditional* separation between polynomial kernelization and Turing kernelization, as well as between different types of Turing kernelizations. They proved that there are parameterized problems where (i) one admits a polynomial (adaptive) Turing kernelization making only one oracle call, but admits no polynomial kernel, (ii) one admits a polynomial non-adaptive Turing kernelization, but admits no polynomial (adaptive) Turing kernelization making a constant number of oracle calls, and (iii) one admits a polynomial adaptive Turing kernelization, but admits no polynomial non-adaptive Turing kernelization. This hierarchy, in particular (iii), indicates that the adaptive behavior might be powerful.

1.2. Preliminaries and Notations

We use basic notation from "classic" computational complexity [AB09, GJ79, Pap94], parameterized complexity [Cyg+15, DF13, FG06, Nie06], and graph theory [Die10, Wes00]. We denote by $\mathbb{N} = \{1, 2, \ldots\}$ the natural numbers excluding zero, and by $\mathbb{N}_0 \coloneqq \mathbb{N} \cup \{0\}$. We denote by $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ the sets of integers, rational numbers, and real numbers, respectively. By $\mathbb{Q}_+$, we denote the set of positive numbers in $\mathbb{Q}$.

1.2.1. Functions and Vectors

For an algorithm $\mathcal{A}$ on input $x \in \Sigma^*$ and output in Σ^*, we denote by $\mathcal{A}^c(x) = \mathcal{A}(\cdots \mathcal{A}(\mathcal{A}(x)) \cdots)$, where $\mathcal{A}$ appears $c \geq 1$ times on the right-hand side of the equation.

A function f is *computable* if there exists a Turing machine that on input x, outputs either $f(x)$ in some time, if x is in the domain of f, or that x is not in the domain of f, otherwise.

For a vector $x \in \mathbb{R}^n$, the ℓ_p-norm for $p \in \mathbb{N}$ of x is defined as $\|x\|_p = \sqrt[p]{\sum_{i=1}^n |x_i|^p}$. The ℓ_∞-norm (also known as max-norm) of x is defined as $\|x\|_\infty = \max_{i \in \{1,\ldots,n\}} |x_i|$.

For a number $x \in \mathbb{R}$, we define by $\lceil x \rceil \in \mathbb{Z}$ the smallest number in $\mathbb{Z}$ that is at least x, and by $\lfloor x \rfloor \in \mathbb{Z}$ the largest number in $\mathbb{Z}$ that is at most x.

Definition 1.11 (signum)**.** For all $x \in \mathbb{R}$, the *signum* of x is

$$\operatorname{sign}(x) = \begin{cases} x/|x| & \text{if } x \neq 0, \\ 0 & \text{if } x = 0. \end{cases}$$

We denote by $A \uplus B$ the *disjoint union* of sets A and B.

1.2.2. Graph Theory

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set $E \subseteq \{\{v, w\} \mid v, w \in V, v \neq w\}$. We say that edge $\{v, w\} \in E$ has endpoints v and w. We also denote by $V(G)$ and $E(G)$ the vertex and edge set of graph G, respectively. For a vertex set $W \subseteq V$, we denote by $G[W] \coloneqq (W, \{e \in E \mid e \subseteq W\})$ the induced subgraph on W. We denote by $G - W \coloneqq G[V \setminus W]$ the graph obtained from the deletion of the vertex set W from G. For an edge set $F \subseteq E$, we denote by $V(F) \coloneqq \{v \in V \mid \exists e \in F\colon v \in e\}$ and by $G[F] \coloneqq (V(F), F)$. We also denote by $G - F \coloneqq (V, E \setminus F)$. If $W = \{v\}$ ($F = \{e\}$), then we also write $G - v$ ($G - e$) instead of $G - \{v\}$ ($G - \{e\}$). A vertex v is called isolated if there is no edge $e \in E$ with $v \in e$.

For a vertex $v \in V$, we denote by $N_G(v) \coloneqq \{w \in V \mid \{v, w\} \in E\}$ the open neighborhood of v in G. For a vertex $v \in V$, by $N_G[v] \coloneqq N_G(v) \cup \{v\}$ we denote the closed neighborhood of v in G. Two vertices u and v are called *twins* or *false twins* if $N_G(u) = N_G(v)$, and *true twins* if $N_G[u] = N_G[v]$. For a vertex set $W \subseteq V$, we define by $N_G[W] \coloneqq \bigcup_{v \in W} N_G[v]$ the closed neighborhood of W and by $N_G(W) \coloneqq N_G[W] \setminus W$ the open neighborhood of W.

A (simple) path P of length ℓ is a graph with vertex set $V(P) = \{v_1, \ldots, v_{\ell+1}\}$ and edge set $E(P) = \{\{v_i, v_{i+1}\} \mid i \in \{1, \ldots, \ell\}\}$. We refer to $v_1, v_{\ell+1}$ as the endpoints of P (we also call P a v_1-$v_{\ell+1}$-path), and to the vertices in $V(P) \setminus \{v_1, v_{\ell+1}\}$ as the inner vertices of P. Note that if $\ell \geq 1$, then each of the endpoints has exactly one neighbor, and if $\ell \geq 2$, then each of the inner vertices has exactly two neighbors. For two vertices $s, t \in V$, path P is an s-t path in G if P is a subgraph of G, $v_1 = s$, and $v_{\ell+1} = t$. An s-t path in G is called *shortest* if there is no s-t path in G of smaller length. We define by $\operatorname{dist}_G(s, t)$ the length of the shortest s-t path in G. We will also represent a path P with $V(P) = \{v_1, \ldots, v_{\ell+1}\}$ and $E(P) = \{\{v_i, v_{i+1}\} \mid i \in \{1, \ldots, \ell\}\}$ by $P = (v_1, \ldots, v_{\ell+1})$, that is, by a sequence derived from the adjacencies of its vertices. A graph G is *connected* if either it only consists of an isolated vertex or for every two distinct vertices $v, w \in V(G)$, there is a v-w path in G. A *component* (or *connected component*) of G is an inclusion-wise maximal connected induced subgraph of G. The closed q-neighborhood of a vertex $v \in V(G)$ in graph G for $q \in \mathbb{N}$ is the set $N_G^q[v] = \{w \in V(G) \mid \operatorname{dist}_G(v, w) \leq q\}$. Let $s, t \in V$. A vertex set $S \subseteq V \setminus \{s, t\}$ is an s-t vertex cut, or also called s-t separator, if there is no s-t path in $G - S$. An edge set $F \subseteq E$ is an s-t edge cut if there is no s-t path in $G - F$. An s-t vertex/edge cut X is minimal if there is no s-t vertex/edge cut X' with $X' \subset X$ in G. An s-t vertex/edge cut X is minimum if there is no s-t vertex/edge cut X' with $|X'| < |X|$ in G.

A cycle C of length ℓ is a graph with vertex set $V(P) = \{v_1, \ldots, v_\ell\}$ and edge set $E(P) = \{\{v_i, v_{i+1}\} \mid i \in \{1, \ldots, \ell-1\}\} \cup \{\{v_\ell, v_1\}\}$. A forest is a cycle-free graph. A tree T is a connected forest. A vertex $v \in V(T)$ is a leaf of T if it has only one neighbor in T.

Definition 1.12. A rooted balanced binary tree of depth 0 with root r is the graph $(\{r\}, \emptyset)$ with designated root r. A rooted balanced binary tree of depth $\ell \in \mathbb{N}$ with root r is the graph obtained from taking two rooted balanced binary trees with roots r' and r'' each of depth $\ell - 1$, making r adjacent to r' and to r'', and designating r as the root.

In a tree T with root r, the lowest common ancestor of two vertices $x, y \in V(T)$ is the first vertex that appears on both paths from x to r and y to r.

A *directed* graph $G = (V, E)$ consists of a vertex set V and a set $E \subseteq \{(v, w) \in V \times V \mid v \neq w\}$ of *arc*s. (As many notions translate directly from undirected to directed graphs, we only mention few in the following.) A *directed path* P of length ℓ is a directed graph with vertex set $V(P) = \{v_1, \ldots, v_{\ell+1}\}$ and arc set $E(G) = \{(v_i, v_{i+1}) \mid i \in \{1, \ldots, \ell\}\}$. A directed cycle, or also called cycle when considered as a subgraph of a directed graph, of length $\ell + 1$ is a directed path of length ℓ with additional arc $(v_{\ell+1}, v_1)$.

Graph parameters. In Figure 1.4, we give an overview of (some) graph parameters appearing in this thesis, and the relations between them. The order ξ of G is $\xi(G) = |V|$. The diameter of graph $G = (V, E)$ is $\operatorname{diam}(G) = \max_{v,w \in V} \operatorname{dist}_G(v, w)$. The degree of a vertex v in G is $\deg_G(v) = |N_G(v)|$. The maximum degree of G is $\Delta(G) = \max_{v \in V} \deg_G(v)$, and the minimum degree of G is $\delta(G) = \min_{v \in V} \deg_G(v)$. The degeneracy of G is $\operatorname{dgn}(G) = \max_{V' \subseteq V} \delta(G[V'])$. A set $W \subseteq V$ is a

vertex cover if $G - W$ contains no edge;
feedback vertex set if $G - W$ contains no cycle;
dominating set if $N_G[W] = V$.

Consequently, the vertex cover number, feedback vertex number, and domination number are the size of the smallest vertex cover, smallest feedback vertex set, and smallest dominating set in the graph, respectively. An edge set $F \subseteq E$ is a *feedback edge set* if $G - F$ contains no cycle. The feedback edge number is the size of the smallest feedback edge set in the graph.

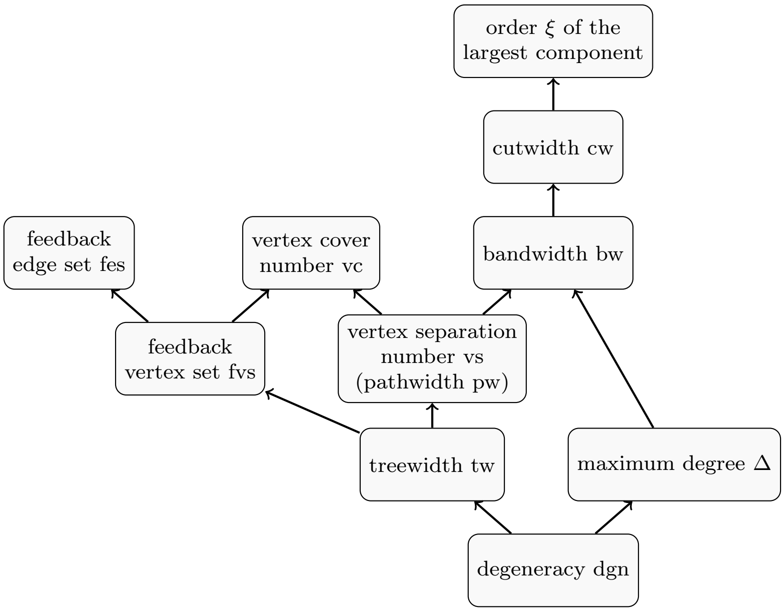


Figure 1.4.: Overview of several structural graph parameters studied in this thesis, with their relations. An arrow from a parameter p to another parameter p' means that there is a function f such that $p \leq f(p')$. Refer to Sorge and Weller [SW18] for details. Note that the vertex separation number equals the pathwidth [Kin92].

Definition 1.13. Given a graph $G = (V, E)$, a tuple $\mathbb{T} = (T, (B_\alpha)_{\alpha \in V(T)})$ consisting of a tree T and subsets $B_\alpha \subseteq V$ for all $\alpha \in V(T)$ is a *tree decomposition* of G if the following holds: (i) $V = \bigcup_{\alpha \in V(T)} B_\alpha$, (ii) for all $e \in E$ there exists an $\alpha \in V(T)$ such that $e \subseteq B_\alpha$, and (iii) for all $v \in V$, the induced graph $T[\{\alpha \in V(T) \mid v \in B_\alpha\}]$ is a tree. The width of $\mathbb{T}$ is $\omega(\mathbb{T}) := \max_{\alpha \in V(T)}\{|B_\alpha| - 1\}$. The *treewidth* of G is the minimum width over all tree decompositions of G.

1.2.3. Parameterized Complexity

Let Σ be some finite alphabet (e.g. $\Sigma = \{0, 1\}$), and let Σ^* denote the set of all finite sequences of elements of Σ (or, strings over Σ). The parameter value of any instance $(x, k) \in \Sigma^* \times \mathbb{N}$ of a parameterized problem we assume to be encoded in unary. Hence, the size of the instance (x, k) is in $O(|x| + k)$. We say that an instance (x, k) is a `yes`-instance of L if and only if $(x, k) \in L$. Otherwise, we say that (x, k) is a `no`-instance of L. We say that two instances (x, k) and (x', k') of parameterized problems L and L' are *equivalent* if $(x, k) \in L$ (i.e., (x, k) is a `yes`-instance of L) if and only if $(x', k') \in L'$ (i.e., (x', k') is a `yes`-instance of L').

An algorithm running in $f(k) \cdot |x|^{O(1)}$ time, where f is a computable function only depending on k, is called a *fixed-parameter algorithm*. A parameterized problem L is *fixed-parameter tractable* if there is a fixed-parameter algorithm solving the problem. The complexity class FPT contains all fixed-parameter tractable parameterized problems.

A *parameterized reduction* from a parameterized problem L to a parameterized problem L' is a fixed-parameter algorithm that on input $(x, k) \in \Sigma^* \times \mathbb{N}$, computes an instance (x', k') such that (i) $(x, k) \in L \iff (x', k') \in L'$, and (ii) $k' \leq g(k)$, for some computable function g only depending on k. Clearly, parameterized reductions provide the following:

Lemma 1.5 ([DF95a]). *If there is a parameterized reduction from a parameterized problem L to a fixed-parameter tractable problem L', then L is fixed-parameter tractable.*

The complexity class XP contains all parameterized problems L such that every instance (x, k) of L can be decided in $f(k) \cdot |x|^{g(k)}$ time, where f and g are some computable function only depending on k. It holds true that FPT $\subseteq$ XP. Inclusionwise, between FPT and XP lies the *W-hierarchy*, which consists of the complexity classes W$[t]$, $t \in \mathbb{N}$:

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \ldots \subseteq \text{XP}.$$

Whether FPT = W[1] is unknown, but all inclusions from above are believed to be strict. A parameterized problem L is W$[t]$-hard, $t \in \mathbb{N}$, if for all parameterized problems L' in W$[t]$ there is parameterized reduction from L' to L.

A parameterized problem is *para-NP-hard* if the problem is NP-hard for some constant value of the parameter. For instance, Coloring parameterized by the number k of colors is para-NP-hard since 3-Coloring is NP-hard. Clearly, if some para-NP-hard parameterized problem is fixed-parameter tractable, then P = NP. For complexity classes even beyond the class of para-NP-hard problems, see e.g. the work of Haan and Szeider [HS17].

1.2.4. Complexity-theoretic Coherence and Conjectures

The complexity class $\mathrm{P}_{/\mathrm{poly}}$ contains all languages that can be decided by a polynomial-time Turing machine with polynomial advice[2]. Similarly, the complexity class $\mathrm{NP}_{/\mathrm{poly}}$ contains all languages that can be decided by a polynomial-time nondeterministic Turing machine with polynomial advice, and the complexity class $\mathrm{coNP}_{/\mathrm{poly}}$ contains the complement languages of $\mathrm{NP}_{/\mathrm{poly}}$.

For a complexity class $\mathcal{C}$, let $\mathrm{NP}^{\mathcal{C}}$ be the class of all languages that can be decided by a polynomial-time nondeterministic Turing machine with an oracle for any problem contained in $\mathcal{C}$. The polynomial hierarchy consists of the complexity classes $\Sigma_i^{\mathrm{PH}} = \mathrm{NP}^{\Sigma_{i-1}^{\mathrm{PH}}}$ and $\Pi_i^{\mathrm{PH}} = \text{co-}\Sigma_{\mathrm{i}}^{\mathrm{PH}}$ (i.e., the complement of Σ_i^{PH}) for every $i \in \mathbb{N}$, where $\Sigma_0^{\mathrm{PH}} = \Pi_0^{\mathrm{PH}} = \mathrm{P}$ (cf., e.g., [HS19, Sto76]). It holds true that $\Sigma_{i-1}^{\mathrm{PH}} \subseteq \Sigma_i^{\mathrm{PH}}$ and $\Pi_{i-1}^{\mathrm{PH}} \subseteq \Pi_i^{\mathrm{PH}}$ for every $i \in \mathbb{N}$, and it is believed that all inclusions are strict. We say that the polynomial hierarchy collapses to its ith level if $\Sigma_i^{\mathrm{PH}} = \Sigma_j^{\mathrm{PH}}$ for all $j > i$. The following collapse is immediate.

Lemma 1.6 (folklore). *If $P = NP$, then the polynomial hierarchy collapses completely, that is, to level zero.*

We exhibit the following two collapsing scenarios, firstly proven by Karp and Lipton [KL82] and Yap [Yap83].

Lemma 1.7 ([KL82]). *If $NP \subseteq P_{/poly}$, then the polynomial hierarchy collapses to its second level.*

Lemma 1.8 ([Yap83]). *If $coNP \subseteq NP_{/poly}$, then the polynomial hierarchy collapses to its third level.*

Lemma 1.8 holds also true if we swap the roles of NP and coNP, due to the following.

Lemma 1.9 ([Yap83]). *$coNP \subseteq NP_{/poly} \iff NP \subseteq coNP_{/poly}$.*

For a discussion on the plausibility of $\mathrm{coNP} \subseteq \mathrm{NP}_{/\mathrm{poly}}$, we refer to the work of Weller [Wel13, Appendix A].

[2]Roughly speaking, for every $n \in \mathbb{N}$, the Turing machine on input x of length n is given additional access to a string (the advice) of size polynomial in n.

Complexity-theoretic Conjectures. We next recall some well-known complexity-theoretic hypotheses and conjectures. The two hypotheses are the Exponential Time Hypothesis and the Strong Exponential Time Hypothesis.

Hypothesis 1.10 (Exponential Time Hypothesis (ETH) [IP01, IPZ01])**.** *There exists some fixed $\varepsilon > 0$ such that* 3-CNF-Sat *cannot be solved in $2^{\varepsilon n} \cdot (n+m)^{O(1)}$ time, where n and m denote the numbers of variables and clauses, respectively.*

Note that the ETH implies that there is no algorithm solving 3-CNF-Sat running in $2^{o(n)} \cdot (n+m)^{O(1)}$ time [IPZ01]. Moreover, the ETH implies FPT $\neq$ W[1] [Che+06] and clearly implies P $\neq$ NP.

Hypothesis 1.11 (Strong Exponential Time Hypothesis (SETH) [IP01, IPZ01])**.** *For every fixed $\varepsilon < 1$ there is an integer $k \in \mathbb{N}$ such that k-*CNF-Sat *cannot be solved in $O(2^{\varepsilon n}) \cdot (n+m)^{O(1)}$ time, where n and m denote the numbers of variables and clauses, respectively.*

Note that the SETH implies that for every fixed $\varepsilon < 1$ there is no $O(2^{\varepsilon n}) \cdot (n+m)^{O(1)}$-time algorithm solving CNF-Sat, and that there is no $(2-\varepsilon)^n \cdot (n+m)^{O(1)}$-time algorithm solving CNF-Sat, where $\varepsilon > 0$. Moreover, the SETH implies the ETH [IPZ01].

The following two conjectures concern polynomial-time solvable problems. We will refer to them only in Chapter 5 (see Appendix A for problem definitions).

Conjecture 1.12 (APSP-conjecture (see, e.g., [AVY18, VW18]))**.** All Pairs Shortest Paths *is not solvable in truly subcubic time, that is, in $O(n^{3-\varepsilon})$ time for any $\varepsilon > 0$, where n denotes the number of vertices of the input graph.*

Conjecture 1.13 (3SUM-conjecture [GO95])**.** 3SUM *is not solvable in truly subquadratic time, that is, in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$, where n denotes the number of numbers.*

For a short discussion of the conjectures, we refer to Abboud et al. [AVY18, Appendix A].

Part I.

Fractals for Kernelization Lower Bounds

OR and AND-(cross-)compositions [BJK14, Bod+09] (Definition 1.5) are tools for excluding polynomial kernelizations under the assumption that coNP $\nsubseteq$ NP$_{/\text{poly}}$. Some parameterized problems such as[3] LONGEST PATH parameterized by the solution size or CLIQUE parameterized by the maximum degree admit a simple OR-composition by taking the disjoint union of the input graphs. Other problems seem to require more involved constructions: For instance, SET COVER parameterized by the universe size [DLS14] or CLIQUE parameterized by the vertex cover number [BJK14]. Devising OR- or AND-(cross-)compositions can be challenging, and the task can be even more challenging when considering combined parameters.

A problem that resisted several attempts for cross-compositions is the NP-hard problem LENGTH-BOUNDED EDGE-CUT (LBEC): Given an undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, and two integers $k, \ell \in \mathbb{N}$, the question is whether it is possible to delete at most k edges such that the shortest s-t path is of length at least ℓ. Golovach and Thilikos [GT11] proved LBEC parameterized by $k + \ell$ to be fixed-parameter tractable, but its status about polynomial kernelization remained open there. In this part, we will resolve this open question.

We would like to apply the OR-(cross-)composition framework to LBEC. Suppose we have a sequence of p input instances of LBEC. Following a first "standard" approach, we concatenate the input instances by identifying the sink vertex of each instance with the source vertex of its succeeding instance, set the first source as global source, and set the last sink as global sink. One might refer to this as "serial composition" (cf. [Flu+19b]). Even if we could manage to guarantee that edges are only deleted in a subgraph corresponding to a graph of exactly one input instance, the length of any shortest path would still depend on the number of input instances.

Following a second "standard" approach, we could introduce a global sink and source vertex, and make each source and sink vertex from the input instances adjacent to the global source and sink vertex, respectively. One might refer to this as a "parallel composition" (cf. [Flu+17a]). This approach would keep the shortest path between the global source and sink, and hence ℓ, small enough. However, we need to ensure that every shortest path, going through each graph of the input instances, is of length at least ℓ. It follows that the number k of edge deletions would depend on the number of instances.

[3]See Appendix A for problem definitions.

Summarizing, the parameter k seems to ask for a serial composition and the parameter ℓ seems to ask for a parallel composition. What does a composition look like that is *between* serial and parallel?

One answer to this question is presented in Chapter 2. We introduce a graph family that we call T-fractals, and explain its application in OR-(cross-)compositions which we baptize "fractalism technique". Indeed, in the fractalism technique the graphs of the input instances are combined in a serial fashion, while a global source and sink vertex in the T-fractal are of short distance to the input graphs' source and sink vertices. Hence, the composition can be seen to be between serial and parallel, yet avoiding each of the issues addressed before: A cut in the T-fractal is small, and works as an instance selector. Once an instance is selected, a shortest path only needs to pass the selected instance. Hence, the edge deletions and the length of a shortest path are both small enough, as required in the (cross-)-compositions framework.

We introduce in Chapter 2 the graph family of T-fractals and prove its application in OR-cross-composition. In Chapter 3, employing the fractalism technique, we show that several parameterized graph modification problems (including LBEC) and several of their variants (planar, directed, vertex deletion) admit no polynomial kernels (unless $\text{coNP} \subseteq \text{NP}_{/\text{poly}}$).

Chapter 2.

The "Fractalism" Technique

We introduce a graph family that we call T-fractals. We prove several properties and provide a manual for devising cross-compositions using T-fractals to rule out polynomial kernels under the assumption coNP $\not\subseteq$ NP$_{/\text{poly}}$.

2.1. Introduction

OR-(cross-)compositions, or in general any kind of hardness reductions, often require *gadgetery* in their constructions. In developing OR-cross-compositions, in particular, gadgets selecting one instance from the set of instances to compose (so-called instance selection gadgets) often form the key for the construction. In this chapter, we introduce an instance selection gadget for cross-compositions in form of a graph family: the *triangle fractals* (*T-fractals* for short). In the next chapter, we prove the application of T-fractals in cross-compositions for problems that we refer to as distance-related cut problems.

Roughly speaking, a T-fractal can be constructed by iteratively putting triangles on top of each other (see Figure 2.1 for four examples). Formally, T-fractals are (iteratively) defined as follows.

Definition 2.1 (iterative)**.** For $q \in \mathbb{N}$, the *q-T-fractal* $\triangle_q$ is the graph constructed as follows:

(1) Set $\triangle_0 \coloneqq (\{\sigma,\tau\},\{\{\sigma,\tau\}\})$ with $\{\sigma,\tau\}$ being a "marked edge" with endpoints σ and τ, subsequently referred to as special vertices.

(2) Let F be the set of marked edges.

This chapter is based on (parts of) *Fractals for kernelization lower bounds* by Till Fluschnik, Danny Hermelin, André Nichterlein, and Rolf Niedermeier (SIAM Journal on Discrete Mathematics [Flu+18a]).

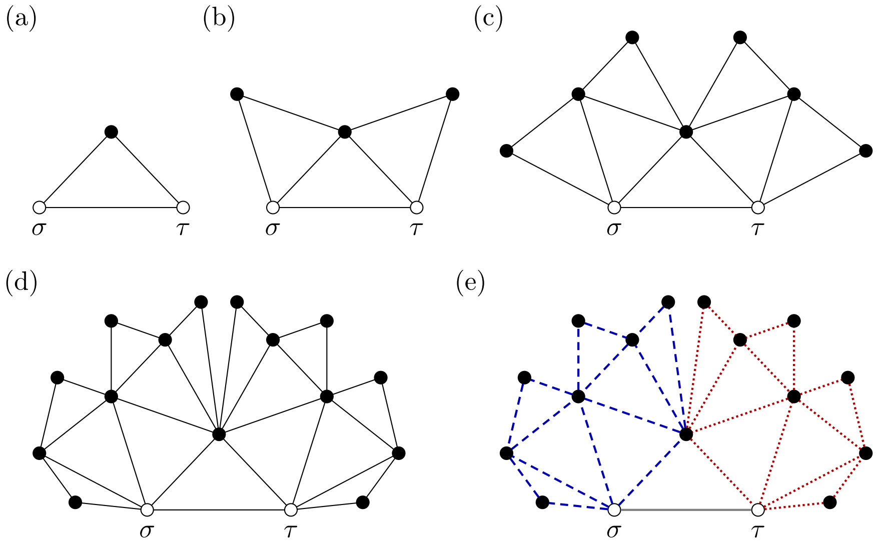


Figure 2.1.: T-fractals (a) $\triangle_1$, (b) $\triangle_2$, (c) $\triangle_3$, (d) $\triangle_4$. The two special vertices σ and τ are highlighted by empty circles. For the T-fractal $\triangle_4$, in (e) the two 3-T-subfractals $\triangle_3'$ (dashed) and $\triangle_3''$ (dotted) are illustrated.

(3) For each edge $e \in F$, add a new vertex and connect it by two new edges with the endpoints of e, and mark the two added edges.
(4) Unmark all edges in F.
(5) Repeat (2)–(4) $q-1$ times.

The T-fractal is self-similar, hence fractal-like: zooming in, one can recognize T-fractals of smaller order (so-called T-subfractals; see Figure 2.1(e)). Indeed, several graph families like cliques or balanced binary trees are self-similar. Fractal-likeness leads to an equivalent, recursive definition of T-fractals (see Definition 2.2 in Section 2.2) which allows for easily proving several of its graph-theoretic properties. For instance, the T-fractal is outerplanar, of small diameter, and admits small σ-τ edge cuts. In Table 2.1 we give an overview of several properties.

We describe how T-fractals help for excluding polynomial kernels by providing a general construction scheme for cross-compositions using T-fractals. To this end, we first define T-fractals and then discuss several of their properties

Table 2.1.: Overview of some properties of T-fractals ($p = 2^q$). Herein, n and m denote the numbers of vertices and edges, respectively, Δ denotes the maximum vertex degree, ϕ denotes the size of a minimum σ-τ edge cut, tw denotes the treewidth, and diam denotes the diameter. BBT_p denotes a balanced binary tree with p leaves.

	n	m	Δ	ϕ	tw	diam
q-T-fractal	$p+1$	$2p-1$	$2q$	$q+1$	≤ 2	$\leq 2q$
BBT_p	$2p-1$	$2p-2$	3	1	1	$2q$

in Section 2.2. Furthermore, we present in Section 2.3 a directed variant and in Section 2.4 an edge-weighted variant. We provide two “manuals” for an application of T-fractals in cross-compositions in Section 2.5. Finally, we discuss a modification of the T-fractal for vertex-deletion problems in Section 2.6.

2.2. Properties of the T-Fractal

The fractal structure of the q-T-fractal $\triangle_q$ might be easier to see when considering the following equivalent recursive definition of $\triangle_q$:

Definition 2.2 (recursive)**.** For the base case we define $\triangle_0 := (\{\sigma, \tau\}, \{\{\sigma, \tau\}\})$. Then, the q-T-fractal $\triangle_q$ is constructed as follows. Take two $(q-1)$-T-fractals $\triangle'_{q-1}$ and $\triangle''_{q-1}$, where σ', τ' and σ'', τ'' are the special vertices of $\triangle'_{q-1}$ and $\triangle''_{q-1}$, respectively. Then $\triangle_q$ is obtained by identifying the vertices τ' and σ'', adding the edge $\{\sigma', \tau''\}$, and setting $\sigma = \sigma'$ and $\tau = \tau''$ as the special vertices of $\triangle_q$.

For $\triangle_q$, we also refer to $\triangle'_{q-1}$ and $\triangle''_{q-1}$ from Definition 2.2 as the $(q-1)$-T-*subfractals* of $\triangle_q$. We remark that we make use of the recursive structure in later proofs. However, by construction, we immediately obtain the following (for the latter, see, e.g., [Bod98]).

Observation 2.1. *The T-fractal is outerplanar and hence the treewidth of* $\triangle_q$ *is* $\operatorname{tw}(\triangle_q) \leq 2$ *for every* $q \in \mathbb{N}$.

In the ith execution of (2)–(4) in Definition 2.1, we obtain 2^{i-1} new triangles. We say that these triangles have depth i. The *i-th boundary* $B_i \subseteq E(\triangle_q)$, $i \in \{1, \ldots, q\}$, are those edges of the triangles of depth i which are not edges of

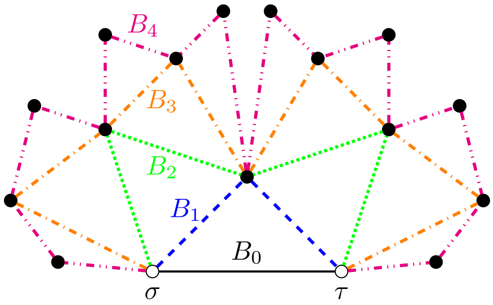


Figure 2.2.: Highlighting the different boundaries $B_0, B_1, \ldots, B_4$ of $\triangle_4$ (see Definition 2.3) by line-types (solid: B_0; dashed: B_1; dotted: B_2; dash-dotted: B_3; dash-dot-dotted: B_4).

the triangles of depth $i-1$. As a convention, the edge $\{\sigma, \tau\}$ connecting the two special vertices σ and τ forms the boundary B_0. We refer to Figure 2.2 for an illustration of the boundaries in the T-fractal $\triangle_4$.

Definition 2.3. For each $\triangle_q$, let $B_0 = \{\{\sigma, \tau\}\}$. The i-th boundary $B_i \subseteq E(\triangle_q)$, $i \in \{1, \ldots, q\}$, of $\triangle_q$ is defined as $B_i = B'_{i-1} \uplus B''_{i-1}$, where B'_{i-1} and B''_{i-1} are the $(i-1)$-th boundaries of the T-subfractals $\triangle'_{q-1}$ and $\triangle''_{q-1}$ of $\triangle_q$, respectively.

By construction, with an easy inductive argument we obtain the following:

Observation 2.2. *In every T-fractal, each boundary forms a σ-τ path, and all boundaries are pairwise edge-disjoint.*

Note that the boundary B_q contains $p = 2^q$ edges. Thus, the number of edges in $\triangle_q$ is $\sum_{i=0}^{q} 2^i = 2^{q+1} - 1 = 2 \cdot p - 1$. Further observe that all vertices of $\triangle_q$ are incident with the edges in B_q, and B_q forms a σ-τ path. Hence, $\triangle_q$ contains $p+1$ vertices.

2.2.1. Minimum σ-τ Edge Cuts

The goal of this subsection is to prove several properties of T-fractals that are used in later constructions. Some key properties of T-fractals appear in the context of σ-τ edge cuts in $\triangle_q$. To prove other properties, we later introduce the notion of the dual structure behind the T-fractals.

The minimum edge cuts in $\triangle_q$ will play a central role when using T-fractals in cross-compositions since the minimum edge cuts serve as instance selectors (see Section 2.5). First, we discuss the size and structure of the minimum edge cuts in $\triangle_q$.

Lemma 2.3. *Every minimum σ-τ edge cut in $\triangle_q$ contains exactly one edge of each boundary and hence is of size $q + 1$.*

Proof. Let C be a minimum σ-τ edge cut in $\triangle_q$. Note that the degrees of σ and τ are exactly $q + 1$, and thus $|C| \leq q + 1$. Moreover, the boundaries in $\triangle_q$ are pairwise edge-disjoint and each boundary forms a σ-τ path (Observation 2.2). Since $\triangle_q$ contains $q+1$ boundaries, it follows that there are at least $q+1$ disjoint σ-τ paths in $\triangle_q$. Menger's theorem [Men27] states that in a graph with distinct source and sink, the maximum number of disjoint source-sink paths equals the minimum size of any source-sink edge cut. Thus, by Menger's theorem, it follows that $|C| \geq q + 1$. Hence $|C| = q + 1$. □

Lemma 2.4. *Let $\triangle'_{q-1}$ and $\triangle''_{q-1}$ be the two $(q-1)$-T-subfractals of $\triangle_q$ and let u be the common neighbor of σ and τ. Then, for every minimum σ-τ edge cut C in $\triangle_q$ it holds true that $C \setminus \{\{\sigma, \tau\}\}$ is either a minimum σ-u edge cut in $\triangle'_{q-1}$, or a minimum u-τ edge cut in $\triangle''_{q-1}$.*

Proof. Note that since C is a minimum σ-τ edge cut in $\triangle_q$ and every σ-τ edge cut in $\triangle_q$ contains the edge $\{\sigma, \tau\}$, the edge set $C' := C \setminus \{\{\sigma, \tau\}\}$ is a minimum σ-τ edge cut in $\triangle_q - \{\{\sigma, \tau\}\}$. Hence, C' is a σ-u edge cut or a u-τ edge cut in $\triangle_q - \{\sigma, \tau\}$, thus $C' \cap (E(\triangle'_{q-1}) \cup E(\triangle''_{q-1})) \neq \emptyset$. Suppose towards a contradiction that

$$C'_1 := C' \cap E(\triangle'_{q-1}) \neq \emptyset \text{ and}$$
$$C'_2 := C' \cap E(\triangle''_{q-1}) \neq \emptyset.$$

If C' is a σ-u edge cut in $\triangle'_{q-1}$, then $C'_1 \cup \{\{\sigma, \tau\}\}$ is a σ-τ edge cut in $\triangle_q$ with $|C'_1 \cup \{\{\sigma, \tau\}\}| < |C|$. If C' is a u-τ edge cut in $\triangle'_{q-1}$, then $C'_2 \cup \{\{\sigma, \tau\}\}$ is a σ-τ edge cut in $\triangle_q$ with $|C'_2 \cup \{\{\sigma, \tau\}\}| < |C|$. Either case yields a contradiction to the minimality of C. It follows that $C' \subseteq E(\triangle'_{q-1})$ or $C' \subseteq E(\triangle''_{q-1})$. □

Lemma 2.5. *There are exactly $p = 2^q$ pairwise different minimum σ-τ edge cuts in $\triangle_q$.*

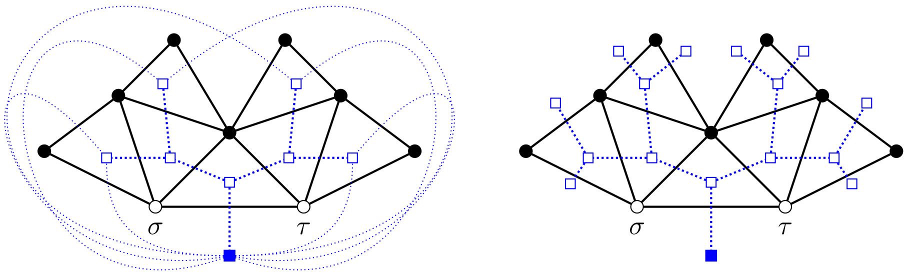


Figure 2.3.: The T-fractal $\triangle_3$ (circles and solid lines) with its dual graph (left-hand side) and its dual structure T_3 (right-hand side). Both, the dual graph and the dual structure are illustrated by squares and dotted lines, where the filled square corresponds to the vertex dual to the outer face (in the dual graph) or the root (of the dual structure).

Proof. We prove the lemma by induction on q. For the base case $q = 0$, observe that $C = \{\{\sigma, \tau\}\}$ is the only σ-τ edge cut in $\triangle_0$.

For the induction step, assume that the statement of the lemma is true for $\triangle_{q-1}$. In $\triangle_q$, denote by u the (unique) vertex that is adjacent to the two special vertices σ and τ. Let $\triangle'_{q-1}$ and $\triangle''_{q-1}$ be the two $(q-1)$-T-subfractals of $\triangle_q$, so that $\triangle'_{q-1}$ ($\triangle''_{q-1}$) has the special vertices σ and u (u and τ). By assumption, $\triangle'_{q-1}$ ($\triangle''_{q-1}$) has exactly $p/2 = 2^{q-1}$ pairwise different minimum σ-u edge cuts (u-τ edge cuts). Moreover, each edge cut in $\triangle'_{q-1}$ is different to each edge cut in $\triangle''_{q-1}$. Observe that for any minimum σ-u (u-τ) edge-cut C in $\triangle'_{q-1}$ ($\triangle''_{q-1}$), we have that $C \cup \{\{\sigma, \tau\}\}$ is a minimum edge-cut in $\triangle_q$. Hence, there are at least $p = 2^q$ pairwise different minimum σ-τ edge cuts $C_1, \dots, C_p$ in $\triangle_q$. Due to Lemma 2.4, we know that there are at most $p = 2^q$ pairwise different minimum σ-τ edge cuts in $\triangle_q$. It follows that there are exactly $p = 2^q$ pairwise different minimum σ-τ edge cuts in $\triangle_q$. $\square$

The Dual Structure

In the following we describe a (hidden) dual structure in $\triangle_q$, that is, a balanced binary tree with p leaves. We refer to Figure 2.3 for an example of the dual structure in $\triangle_3$. To talk about the dual structure by means of duality of plane graphs, we need a plane embedding of $\triangle_q$. Hence we assume that $\triangle_q$ is embedded as in Figure 2.1 (iteratively extended). By T_q we denote the dual structure in $\triangle_q$, where the vertex dual to the outer face is replaced by $p+1$ vertices (*split vertices*) such that each edge incident with the dual vertex is incident with exactly one split vertex. We consider the split vertex incident with the vertex dual to the triangle containing the edge $\{\sigma, \tau\}$ as the root vertex of the dual structure T_q. Thus, the other split vertices correspond to the leaves of T_q. Note that the depth of a triangle one-to-one corresponds to the depth of the dual vertex in T_q.

Observe that there is a one-to-one correspondence between the edges in T_q and the edges in $\triangle_q$. The following lemma states duality of root-leaf paths in T_q and minimum σ-τ edge cuts in $\triangle_q$, demonstrating the utility of the dual structure T_q.

Lemma 2.6. *There is a one-to-one correspondence between root-leaf paths in the dual structure T_q of $\triangle_q$ and minimum σ-τ edge cuts in $\triangle_q$. Moreover, there are exactly $p = 2^q$ pairwise different minimum σ-τ edge cuts in $\triangle_q$.*

Proof. Observe that each path from the root to a leaf in the dual structure T_q corresponds to a cycle in the dual graph. Note that there is a one-to-one correspondence between minimal edge cuts in a plane graph and cycles in its dual graph [Die10, Proposition 4.6.1]. Herein, every cycle in the dual graph that "cuts" the edge $\{\sigma, \tau\}$ in $\triangle_q$ is a root-leaf path in T_q. Thus, the only minimal σ-τ edge cuts are those corresponding to the root-leaf paths. By the one-to-one correspondence of the depth of the triangles in $\triangle_q$ and the depth of the vertices in T_q, these edge cuts are of cardinality $q+1$. Hence, by Lemma 2.3, these edge cuts are minimum edge cuts.

Since $|B_q| = p$, there are exactly p leaves in T_q, and thus there are exactly p different root-leaf paths in T_q. It follows that the number of pairwise different minimum σ-τ edge cuts in $\triangle_q$ is exactly $p = 2^q$. □

2.2.2. Distances between σ and τ

In this subsection, we discuss how edge deletions in T-fractals change the distances of the vertices in the T-fractal to σ and to τ.

Lemma 2.7. *Let C be a minimum σ-τ edge cut in $\triangle_q$. Let $\{\{x, y\}\} = C \cap B_q$, where x and σ are in the same connected component in $\triangle_q - C$. Then* $\operatorname{dist}_{\triangle_q - C}(\sigma, x) + \operatorname{dist}_{\triangle_q - C}(y, \tau) = q$.

Proof. We prove the lemma by induction on q. For the base case $q = 0$, observe that $C = \{\{\sigma, \tau\}\}$ and $\operatorname{dist}_{\triangle_0 - C}(\sigma, x) + \operatorname{dist}_{\triangle_0 - C}(y, \tau) = 0$.

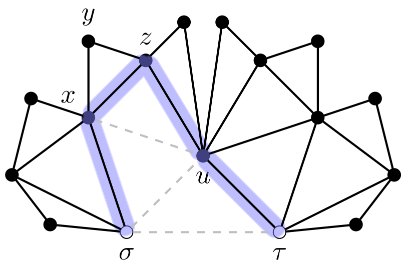


Figure 2.4.: The T-fractal $\triangle_4$ without the edge set $D = \{\{\sigma,\tau\},\{\sigma,u\},\{u,x\}\}$ (left-hand side) and without the edge set $D' = D \cup \{\{x,z\}\}$ (right-hand side). Deleted edges are sketched by dashed lines. For each, the shortest σ-τ path is highlighted. Note that $D' \cup \{\{x,y\}\}$ or $D' \cup \{\{y,z\}\}$ forms a minimum σ-τ edge cut (see right-hand side).

For the induction step, assume that the statement of the lemma is true for $\triangle_{q-1}$. Now, let C be a minimum σ-τ edge cut in $\triangle_q$. Hence, $\{\sigma,\tau\} \in C$. Denote by u the (unique) vertex that is adjacent to the two special vertices σ and τ. Let $\triangle'_{q-1}$ and $\triangle''_{q-1}$ be the two $(q-1)$-T-subfractals of $\triangle_q$, so that $\triangle'_{q-1}$ ($\triangle''_{q-1}$) has the special vertices σ and u (u and τ). By Lemma 2.4, $C' := C \setminus \{\{\sigma,\tau\}\}$ is either a subset of $E(\triangle'_{q-1})$ or of $E(\triangle''_{q-1})$. Assume w.l.o.g. that $C' \subseteq E(\triangle'_{q-1})$. It follows from the induction hypothesis that $\mathrm{dist}_{\triangle'_{q-1}-C'}(\sigma,x) + \mathrm{dist}_{\triangle'_{q-1}-C'}(y,u) = q-1$. Since $\mathrm{dist}_{\triangle_q-C}(y,\tau) = \mathrm{dist}_{\triangle'_{q-1}-C'}(y,u)+1$, it follows that $\mathrm{dist}_{\triangle_q-C}(\sigma,x) + \mathrm{dist}_{\triangle_q-C}(y,\tau) = q$. □

Remark 2.1. By an inductive proof like the one for Lemma 2.5 or for Lemma 2.7, one can easily show that the maximum degree Δ of $\triangle_q$ is exactly $2q$ for $q > 0$. Moreover, due to Lemma 2.7, the diameter of $\triangle_q$ is at most $2q$.

Another observation on $\triangle_q$ is that any deletion of d edges increases the length of any shortest σ-τ path to at most $d+1$, unless the edge deletion forms a σ-τ edge cut.

Lemma 2.8. *Let $D \subseteq E(\triangle_q)$ be a subset of edges of $\triangle_q$. If D is not a σ-τ edge cut, then there is a σ-τ path of length at most $|D|+1$ in $\triangle_q - D$.*

Proof. We prove the statement of the lemma by induction on q. For the induction base with $q = 0$, observe that since D is not a σ-τ edge cut, it follows that $D = \emptyset$ and, hence, σ and τ have distance one.

For the induction step, assume that the statement of the lemma is true for $\triangle_{q-1}$. Now, let $D \subseteq E(\triangle_q)$ be a subset of edges of $\triangle_q$ such that D is not a σ-τ edge cut. If $\{\sigma,\tau\} \notin D$, then there is a σ-τ path of length one and the statement of the lemma holds. Now consider the case $\{\sigma,\tau\} \in D$. Denote by u the (unique) vertex that is adjacent to the two special vertices σ and τ. If $\{\sigma,\tau\} \in D$, then every σ-τ path in $\triangle_q - D$ contains u and hence $\mathrm{dist}_{\triangle_q - D}(\sigma,\tau) = \mathrm{dist}_{\triangle_q - D}(\sigma,u) + \mathrm{dist}_{\triangle_q - D}(u,\tau)$. (If there is no σ-u-path or no u-τ-path in $\triangle_q - D$, then D is a σ-τ edge cut; a contradiction to the assumption of the lemma.) Now let $\triangle'_{q-1}$ and $\triangle''_{q-1}$ be the two $(q-1)$-T-subfractals of $\triangle_q$, so that $\triangle'_{q-1}$ ($\triangle''_{q-1}$) has the special vertices σ and u (u and τ). It follows that D can be partitioned into $D = D' \cup D'' \cup \{\{\sigma,\tau\}\}$ with $D' \subseteq E(\triangle'_{q-1})$ and $D'' \subseteq E(\triangle''_{q-1})$. By induction hypothesis, it follows that there is a σ-u path of length at most $|D'|+1$ in $\triangle'_{q-1} - D'$ and a u-τ path of length at most $|D''|+1$ in $\triangle''_{q-1} - D''$. Hence, there is a σ-τ path of length at most $|D'|+|D''|+2 = |D|+1$ in $\triangle_q - D$. □

By Lemma 2.8, the distance of the two special vertices σ and τ is upper-bounded by the number of edge deletions plus one, where the deleted edges do not form a σ-τ edge cut. Hence, if only few edges are deleted in $\triangle_q$, then σ and τ are not far away from each other. The next lemma generalizes this by stating that the distance of *any* vertex in $\triangle_q$ to σ or to τ is quite small, even if a few edges are deleted. Here "quite small" means that if $O(q)$ edges are deleted, then the distance is still in $O(q)$, which is logarithmic in the size of $\triangle_q$.

Lemma 2.9. *Let $D \subseteq E(\triangle_q)$ be a subset of edges of $\triangle_q$.*
(A) If $\triangle_q - D$ is connected, then $\mathrm{dist}_{\triangle_q - D}(\sigma,x) \le q+|D|+1$ for all $x \in V(\triangle_q)$.
(B) If $\triangle_q - D$ has exactly two connected components, with σ and τ being in different components, then $\min_{z\in\{\sigma,\tau\}}\{\mathrm{dist}_{\triangle_q - D}(z,x)\} \le q+|D|-1$ for all $x \in V(\triangle_q)$.

Proof. We prove the two statements (A) and (B) simultaneously with an induction on depth q of the T-fractal.

The base case is $q = 0$. For statement (A), observe that $D = \emptyset$. Thus, since τ has distance one to σ, statement (A) follows. For statement (B), observe that $D = \{\{\sigma,\tau\}\}$. Thus, statement (B) holds.

As our induction hypothesis, we assume that (A) and (B) hold for $1,\ldots,q-1$. We write IH.(A) and IH.(B) for the induction hypothesis of (A) and (B), respectively. We introduce some notation used for the induction step for both statements. Let $\triangle_q$, $q > 0$, be the T-fractal with special vertices σ and τ and let u be the (unique) vertex in $\triangle_q$ that is adjacent to σ and τ, that is, u is on the boundary B_1 of $\triangle_q$. Denote with $\triangle'_{q-1}$ ($\triangle''_{q-1}$) the left (right) subfractal of $\triangle_q$ with special vertices σ and u (u and τ). Furthermore, let D' (D'') be the subset of edges of D deleted in $\triangle'_{q-1}$ ($\triangle''_{q-1}$).

For the inductive step, we consider the two cases of $\{\sigma,\tau\} \notin D$ and $\{\sigma,\tau\} \in D$.

Case 1: $\{\sigma,\tau\} \notin D$. Obviously, this case excludes (B), since σ and τ are in the same connected component. Thus, we consider the induction step for (A). Let x be in the left subfractal $\triangle'_{q-1}$. If D' does not form an edge cut in $\triangle'_{q-1}$, then by IH.(A) it follows that $\mathrm{dist}_{\triangle'_{q-1}-D'}(\sigma,x) \leq q-1+|D'|+1 \leq q+|D|$. Thus, we consider the case where D' forms an edge cut in $\triangle'_{q-1}$. Observe that such an edge cut fulfills the requirements of statement (B) for $\triangle'_{q-1}$. By IH.(B), it follows that $\min_{z\in\{\sigma,u\}}\{\mathrm{dist}_{\triangle'_{q-1}-D'}(z,x)\} \leq q-1+|D'|-1 < q+|D|$. If $z=\sigma$, then we are done. Thus let $z=u$, where u is the vertex incident to both σ and τ in $\triangle_q$. We know that $\triangle_q - D$ is connected, and thus there exists an u-τ path in the right subfractal $\triangle''_{q-1} - D''$. By Lemma 2.8, it follows that $\mathrm{dist}_{\triangle''_{q-1}-D''}(u,\tau) \leq |D''|+1$. Recall that $\{\sigma,\tau\} \notin D$. In total, we get

$$\begin{aligned}\mathrm{dist}_{\triangle_q-D}(x,\sigma) &\leq \mathrm{dist}_{\triangle'_{q-1}-D'}(u,x) + \mathrm{dist}_{\triangle''_{q-1}-D''}(u,\tau)+1\\ &\leq q-1+|D'|-1+|D''|+1+1 = q+|D|.\end{aligned}$$

We obtain that $\mathrm{dist}_{\triangle_q-D}(x,\sigma) \leq q+|D|$ and hence, $\mathrm{dist}_{\triangle_q-D}(x,\tau) \leq q+|D|+1$. In case that x is in the right subfractal $\triangle''_{q-1}$, it follows by symmetry that $\mathrm{dist}_{\triangle_q-D}(x,\tau) \leq q+|D|$ and, hence, $\mathrm{dist}_{\triangle_q-D}(x,\sigma) \leq q+|D|+1$.

Case 2: $\{\sigma,\tau\} \in D$. First, we consider the step for statement (A). Let x be in the left subfractal $\triangle'_{q-1}$. Observe that D' forms no edge cut in $\triangle'_{q-1}$, since otherwise the graph is not connected. Thus, $\triangle'_{q-1} - D'$ is connected, and by IH.(A) it follows that $\mathrm{dist}_{\triangle'_{q-1}-D'}(\sigma,x) \leq q-1+|D'|+1 < q+|D|$.

Now, let x be in the right subfractal $\triangle''_{q-1}$. Again, D'' forms no edge cut in $\triangle''_{q-1}$. By IH.(A), $\mathrm{dist}_{\triangle''_{q-1}-D''}(u,x) \leq q-1+|D''|+1$. Since u and σ are connected in $\triangle'_{q-1} - D'$, we can apply Lemma 2.8 on u and σ. In total, with $D = D' \cup D'' \cup \{\{\sigma,\tau\}\}$ we get:

$$\begin{aligned}\mathrm{dist}_{\triangle_q-D}(x,\sigma) &\leq \mathrm{dist}_{\triangle'_{q-1}-D'}(u,\sigma) + \mathrm{dist}_{\triangle''_{q-1}-D''}(u,x)\\ &\leq q-1+|D'|+1+|D''|+1\\ &\leq q+|D|.\end{aligned}$$

Next, we consider the step for statement (B). Observe that the edge cut formed by edges in D cannot form edge cuts in $\triangle'_{q-1}$ and in $\triangle''_{q-1}$ at the same time since

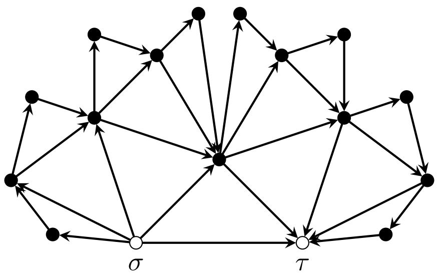


Figure 2.5.: The directed T-fractal $\vec{\triangle}_4$ (see Definition 2.4).

otherwise there are more than two connected components. Let x be in the left subfractal and let the edges in D' form no edge cut in $\triangle'_{q-1}$. Then, by IH.(A), it follows that $\text{dist}_{\triangle'_{q-1}-D'}(\sigma, x) \leq q-1+|D'|+1 \leq q+|D|-1$. Thus, let the edges in D' form an edge cut in $\triangle'_{q-1}$. By IH.(B), either $\text{dist}_{\triangle'_{q-1}-D'}(\sigma, x) \leq q-1+|D'|-1 < q+|D|-1$, or $\text{dist}_{\triangle'_{q-1}-D'}(u, x) \leq q-1+|D'|-1$. For the latter case, recall that the edges in D'' form no edge cut in $\triangle''_{q-1}$, that is, $\triangle''_{q-1} - D''$ is connected. By Lemma 2.8, it follows that $\text{dist}_{\triangle''_{q-1}-D''}(u, \tau) \leq |D''|+1$. In total, we get:

$$\begin{aligned}
\text{dist}_{\triangle_q - D}(x, \tau) &\leq \text{dist}_{\triangle'_{q-1}-D'}(x, u) + \text{dist}_{\triangle''_{q-1}-D''}(u, \tau) \\
&\leq q - 1 + |D'| - 1 + |D''| + 1 \\
&< q + |D| - 1.
\end{aligned}$$

The case where x is in the right subfractal follows by symmetry. □

2.3. Directed Variants of T-Fractals

By definition, a T-fractal $\triangle_q$ is an undirected graph. The directed variant of $\triangle_q$, denoted by $\vec{\triangle}_q$, is (recursively) defined as follows (see Figure 2.5 for an illustration).

Definition 2.4. For the base case we define $\vec{\triangle}_0 := (\{\sigma, \tau\}, \{(\sigma, \tau)\})$. Then, the directed q-T-fractal $\vec{\triangle}_q$ is constructed as follows. Take two directed $(q-1)$-T-fractals $\vec{\triangle}'_{q-1}$ and $\vec{\triangle}''_{q-1}$, where σ', τ' and σ'', τ'' are the special vertices of $\vec{\triangle}'_{q-1}$

and $\vec{\triangle}''_{q-1}$, respectively. Then $\vec{\triangle}_q$ is obtained by identifying the vertices τ' and σ'', adding the arc (σ', τ''), and setting $\sigma = \sigma'$ and $\tau = \tau''$ as the special vertices of $\vec{\triangle}_q$.

We can also obtain $\vec{\triangle}_q$ from $\triangle_q$ as follows: Recall that each boundary forms a σ-τ path (Observation 2.2). For each boundary, we direct the edges in the boundary from σ to τ. By this, the obtained boundary forms a directed σ-τ path. Hereby, we transfer the notion of a boundary (Definition 2.3) to the directed variant.

Observation 2.10. *In $\vec{\triangle}_q$, vertex σ has no incoming arcs and out-degree $q+1$, and vertex τ has no outgoing arcs and in-degree $q+1$. Moreover, $\vec{\triangle}_q$ is acyclic.*

Except for Lemma 2.9, all results from Section 2.2.1 can be transferred to $\vec{\triangle}_q$. Lemmas 2.3 and 2.5 hold since we still have the same degree on σ and on τ and the boundaries still form disjoint (directed) σ-τ paths. We define the dual structure of $\vec{\triangle}_q$ as the dual structure of the underlying undirected variant $\triangle_q$. By this, it is not hard to adapt Lemmas 2.6 and 2.7. For the latter result, and additionally for Lemma 2.8, we make use of the fact that in the undirected case, we traverse the edges of the undirected $\triangle_q$ in the same direction as they are directed in $\vec{\triangle}_q$.

Regarding an equivalent of Lemma 2.9 for the directed variant, with small effort one can modify the proof of Lemma 2.9 to show the following.

Lemma 2.11. *Let $D \subseteq E(\vec{\triangle}_q)$ be a subset of arcs of $\vec{\triangle}_q$ and let x be an arbitrary vertex in $V(\vec{\triangle}_q)$. If $x \in V(\vec{\triangle}_q)$ is reachable from σ in $\vec{\triangle}_q - D$, then* $\operatorname{dist}_{\vec{\triangle}_q - D}(\sigma, x) \leq q + |D| + 1$.

Proof. We prove the statement with an induction on depth q of the T-fractal.

The base case is $q = 0$. If $x = \sigma$, the statement immediately holds. If $x = \tau$, observe that $D = \emptyset$, since x is reachable from σ. Thus, since τ has distance one to σ, the statement follows.

As our induction hypothesis, we assume that the statement holds for $1, \ldots, q-1$. We introduce some notation used for the induction step. Let $\vec{\triangle}_q$, $q > 0$, be the directed T-fractal with special vertices σ and τ and let u be the (unique) vertex in $\vec{\triangle}_q$ that is adjacent to σ and τ, that is, u is on the boundary B_1 of $\vec{\triangle}_q$. Denote with $\vec{\triangle}'_{q-1}$ ($\vec{\triangle}''_{q-1}$) the left (right) T-subfractal of $\vec{\triangle}_q$ with special vertices σ and u (u and τ). Furthermore, let D' (D'') be the subset of arcs of D deleted in $\vec{\triangle}'_{q-1}$ ($\vec{\triangle}''_{q-1}$).

Let $x \in V(\vec{\triangle})$ be an arbitrary vertex reachable from σ in $\vec{\triangle}_q - D$. For the inductive step, we consider the two cases of the position of x in $\vec{\triangle}_q$.

Case 1: $x \in V(\vec{\triangle}'_{q-1})$. By induction hypothesis, it follows that

$$\mathrm{dist}_{\vec{\triangle}_q - D}(\sigma, x) = \mathrm{dist}_{\vec{\triangle}'_{q-1} - D'}(\sigma, x) \leq q - 1 + |D'| + 1 \leq q + |D|.$$

Case 2: $x \in V(\vec{\triangle}''_{q-1})$. Since x is reachable from σ and τ has no outgoing arcs, it follows that u is reachable from σ as well. By the version of Lemma 2.8 for directed T-fractals, it follows that $\mathrm{dist}_{\vec{\triangle}'_{q-1} - D'}(\sigma, u) \leq |D'| + 1$. Together with the induction hypothesis, it follows that

$$\begin{aligned}\mathrm{dist}_{\vec{\triangle}_q - D}(\sigma, x) &\leq \mathrm{dist}_{\vec{\triangle}'_{q-1} - D'}(\sigma, u) + \mathrm{dist}_{\vec{\triangle}'_{q-1} - D'}(u, \tau)\\ &\leq |D'| + 1 + q - 1 + |D''| + 1\\ &\leq q + |D| + 1. \qquad \square\end{aligned}$$

Observe that if x reaches τ, then Lemma 2.11 is symmetric.

2.4. Edge-Weighted T-Fractals

The *weighted* T-fractal is the T-fractal equipped with *edge costs*, that is, the cost for deleting an edge in the T-fractal. If all edges in $\triangle_q$ are of the same edge cost $c \in \mathbb{N}$, then we write $\triangle_q^c$ (we drop the superscript if $c = 1$). In the remainder, we use the weighted case yet obtain results on the unweighted case of T-fractals without multiple edges or loops. This is possible due to the following many-one reduction of the weighted to the unweighted case. Consider the weighted T-fractal $\triangle_q^c$ with $c \geq 2$. To reduce to the case with an unweighted, simple graph, we replace each edge by c copies of it. Thus, to make two adjacent vertices non-adjacent, it requires c edge-deletions. To make the graph simple, we subdivide each of the added edges. We remark that in this way we double the distances of the vertices in the original T-fractal. Hence, whenever we consider distances in the fractal with edge cost and the graph obtained by the reduction above, we have to take into account a factor of two.

Remark 2.2. The treewidth of the graph G obtained by the modification of the T-fractal described above remains at most two, though it is not necessarily outerplanar anymore. To see this, observe that outerplanar graphs are series-parallel. Moreover, a graph obtained by replacing an edge in a series-parallel graph by a number of paths with the same endpoints remains series-parallel [Duf65]. It follows that G has treewidth at most two.

2.5. Application Manuals for T-Fractals

The aim of this section is to provide two general manuals on how to use T-fractals in cross-compositions. To this end, we introduce two general constructions—one for undirected graphs and one for directed graphs. We start with the undirected case.

Construction 2.12. Given $p = 2^q$ instances $\mathcal{I}_1, \dots, \mathcal{I}_p$ of an NP-hard graph problem, where each instance $\mathcal{I}_i$ has a unique source vertex s_i and a unique sink vertex t_i.

(i) Equip $\triangle_q^c$ with some “appropriate” edge cost $c \in \mathbb{N}$.

(ii) Let $v_0, \dots, v_p$ be the vertices of the boundary B_q, labeled by their distances to σ in the σ-τ path corresponding to B_q (observe that $v_0 = \sigma$ and $v_p = \tau$).

(iii) Incorporate each of the p graphs of the input instances into $\triangle_q^c$ as follows: for each $i \in \{1, \dots, p\}$, identify s_i with vertex v_{i-1} in $\triangle_q^c$ and identify t_i with v_i in $\triangle_q^c$.

Refer to Figure 2.6 for an illustrative example of Construction 2.12. In Construction 2.12, the T-fractal works as an instance selector by deleting edges corresponding to a minimum edge cut, which, by Lemma 2.3, is of size $q + 1$. Hence, each minimum edge cut costs $c \cdot (q + 1)$. The idea is that if we choose an appropriate value for c (larger than the budget in the instances $\mathcal{I}_1, \dots, \mathcal{I}_p$) and an appropriate budget in the composed instance (e. g. $c \cdot (q + 1)$ plus the budget in the instances $\mathcal{I}_1, \dots, \mathcal{I}_p$), then we can only afford to delete at most $q + 1$ edges in $\triangle_q^c$. Furthermore, if the at most $q + 1$ edges chosen to be deleted do not form a minimum σ-τ edge cut in $\triangle_q^c$, then, by Lemma 2.8, the shortest σ-τ path has length at most $q + 2$. Thus, by requiring in the composed instance that σ and τ have distance more than $q + 2$, we enforce that any solution for the composed instance contains a minimum σ-τ edge cut in $\triangle_q^c$. By Lemma 2.6, each such minimum edge cut corresponds to one root-leaf path in the dual structure T_q of $\triangle_q^c$. Observe that each leaf in the dual structure of $\triangle_q^c$ one-to-one corresponds to an attached source instance. Hence, with an appropriate choice of c, the budget in the composed instance, and the required distance between σ and τ, the T-fractal ensures that one instance is “selected”. We say that a minimum

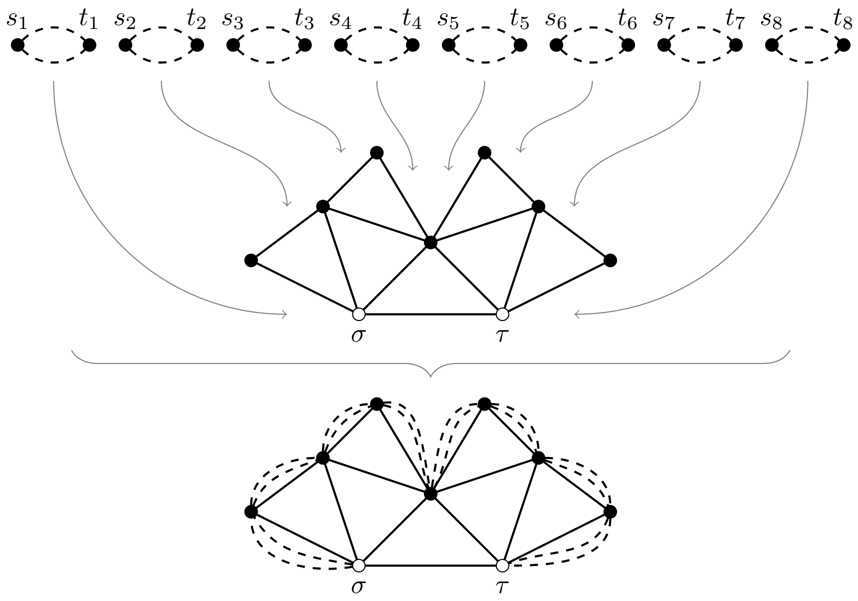


Figure 2.6.: Illustration of Construction 2.12 with $p = 2^3 = 8$. The vertices $s_1, \dots, s_8$ and $t_1, \dots, t_8$ indicate the source and sink vertices in the eight input instances, respectively. We use dashed lines to sketch the input graphs. Below the curved brace, we sketch the resulting graph of the target instance.

σ-τ edge cut in $\triangle_q^c$ *selects* an instance $\mathcal{I}$ if the edge cut corresponds to the root-leaf path with the leaf corresponding to instance $\mathcal{I}$.

Observation 2.13. *Every minimum edge cut C in $\triangle_q^c$ selects exactly one instance $\mathcal{I}$. Conversely, every instance $\mathcal{I}$ can be selected by exactly one minimum edge cut.*

Moreover, the graph obtained from Construction 2.12 has treewidth bounded in the maximum input instance size.

Observation 2.14. *Let $n_{\max} := \max_{i \in \{1,\dots,p\}} |V(G_i)|$, where G_i is the graph in instance $\mathcal{I}_i$, $i \in \{1, \dots, p\}$, from Construction 2.12 and let G be the obtained graph. Then the treewidth of G is $\operatorname{tw}(G) \leq 2 + n_{\max}$.*

Proof. By Observation 2.1, we know that the treewidth of the T-fractal is at most two. Moreover, we know that the treewidth of the modified T-fractal is at most two (see Remark 2.2). Considering a tree decomposition of the modified T-fractal, we replace each bag corresponding to an edge e of the outer boundary by the set containing all vertices of the instance appended on edge e. Hence, we obtain a tree decomposition of G of width at most $2 + n_{\max}$. ☐

If Construction 2.12 works as an OR-cross-composition, then Observation 2.14 gives the following (employing Proposition 1.3).

Proposition 2.15. *Unless coNP $\subseteq$ NP$_{/poly}$, any parameterized problem P that admits an OR-cross-composition for some NP-hard problem L by using Construction 2.12 does not admit a polynomial kernel with respect to the parameter treewidth* tw*.*

Using the same ideas as above and transferring them to the directed case yields the following construction with analogous properties.

Construction 2.16. Given $p = 2^q$ instances $\mathcal{I}_1, \ldots, \mathcal{I}_p$ of an NP-hard problem on directed acyclic graphs, where each instance $\mathcal{I}_i$ has a unique source vertex s_i and a unique sink vertex t_i.

(i) Equip $\vec{\triangle}_q^c$ with some "appropriate" edge cost $c \in \mathbb{N}$, where σ is the vertex with no incoming arc.

(ii) Let $v_0, \ldots, v_p$ be the vertices of the boundary B_q, labeled by their distances to σ in the σ-τ path corresponding to B_q (observe that $v_0 = \sigma$ and $v_p = \tau$).

(iii) Incorporate each of the p directed acyclic graphs of the input instances into $\vec{\triangle}_q^c$ as follows: for each $i \in \{1, \ldots, p\}$, identify s_i with vertex v_{i-1} in $\vec{\triangle}_q^c$ and identify t_i with vertex v_i in $\vec{\triangle}_q^c$.

In Chapter 3, we use Constructions 2.12 and 2.16 in OR-cross-compositions to rule out the existence of polynomial kernels. We baptize this approach *fractalism*. In particular, we provide the source and the target problem, appropriate values for the edge cost c and the budget in the composed instance, and the required distance between the special vertices σ and τ. Observe that the directed graph obtained from Construction 2.16 is acyclic. Hence, by Construction 2.16 we can apply OR-cross-compositions for problems on directed acyclic graphs.

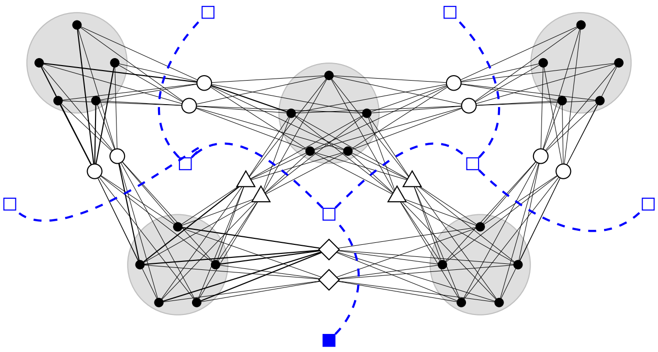


Figure 2.7.: The vertex deletion variant $\triangle_2^{2;5}$ of T-fractals. Vertex types: empty diamonds belong to the boundary B_0, empty triangles belong to the boundary B_1, empty circles belong to the boundary B_2. The squares and dashed lines indicate the dual structure, where the filled square corresponds to the root. We highlighted vertices in gray-filled circles that correspond to the vertices in the edge-deletion variant $\triangle_2$.

2.6. Vertex-Deletion Variants

We give another modification of the T-fractal such that vertex-deletion variants can be tackled. We obtain the vertex-deletion variant $\triangle_q^{c;d}$ of the T-fractal from $\triangle_q^c$ as follows, where d denotes an additional *vertex cost*. Recall that $\triangle_q^c$ can be reduced to an unweighted, simple graph $\hat{\triangle}_q^c$. We first obtain $\hat{\triangle}_q^c = (V' \cup V'', E')$ from $\triangle_q^c$, where V' denote the vertices not being the product of a subdivision in the step from $\triangle_q^c$ to $\hat{\triangle}_q^c$. Next, we describe how to obtain $\triangle_q^{c;d}$ from $\hat{\triangle}_q^c$. To this end, we introduce the following notation: given a graph $G = (V, E)$ and $v \in V$, we say we *clone* vertex v if we add a new vertex v' to V and the edge set $\{\{v', w\} \mid \{v, w\} \in E\}$ to E. We obtain $\triangle_q^{c;d}$ from $\hat{\triangle}_q^c$ by cloning every vertex in V' $d - 1$ times (we refer to them as the *clones* in the following). We denote by $C_x \subseteq V(\triangle_q^{c;d})$ the set containing vertex $x \in V(\hat{\triangle}_q^c)$ and its clones. We refer to Figure 2.7 for an illustration of the vertex-deletion variant of T-fractal $\triangle_2$ with edge cost 2 and vertex cost 5.

The vertex cost can be interpreted as a tool to avoid deletion of clones. Herein, we can set the vertex cost larger than the budget for vertex-deletions in a given problem instance to avoid any deletion of clones. To this end, note that to essentially change the structure of the graph by deleting a vertex having clones, it is required to delete all clones of the vertex as well.

We remark that the vertex-deletion variant of the T-fractal can be directed in the same way as the edge-deletion variant of the T-fractal such that the obtained graph is acyclic. Moreover, we can transfer the notion of boundaries, now being a set of vertices instead, as well as the dual structure for the vertex-deletion variant of the T-fractal (see Figure 2.7). Note that in general $\triangle_q^{c;d}$ is not planar, for example for $c, d \geq 3$.

One can show that all properties of the edge-deletion variant also hold on the vertex-deletion variant, replacing edge cuts by vertex cuts (modulo some constants), while forbidding to delete clones. Again, the latter is reasonable since in any application we can set the vertex cost larger than the budget for vertex-deletions. For example, considering any minimum C_σ-C_τ vertex cut in $\triangle_q^{c;d}$, where every vertex in every C_x, $x \in V(\triangle_q)$, is not allowed to be deleted. One can show that it is of size $(q+1) \cdot c$, using a simple bijection of the edges in $\triangle_q^c$ and the corresponding vertex sets in $\triangle_q^{c;d}$.

In addition, one can modify Constructions 2.12 and 2.16 slightly to use the vertex-deletion variants for vertex-deletion problems. Herein, it is worth to mention how the merging of the source and sink vertices of the input instances works. Consider s_i and v_{i-1} as defined in Construction 2.12, and let $d \in \mathbb{N}$ be the vertex cost. Note that v_{i-1} is replaced by $C := C_{v_{i-1}}$ with $|C| = d$. We remove s_i and all incident edges of s_i, and add d copies $s_i^1, \ldots, s_i^d$ of s_i. In addition, if $\{s_i, x\}$ was an edge we deleted in the previous step, we add the edges $\{s_i^j, x\}$ for all $j \in \{1, \ldots, d\}$. Finally, we identify each s_i^j with one vertex in C in such a way that each vertex in C is identified exactly once. We apply an analogous procedure to the sink vertex t_i and C_{v_i}.

2.7. Concluding Remarks

With the T-fractals we introduced a family of graphs with the property that, for two special vertices σ and τ, the number of minimum σ-τ edge cuts is exponential in the cuts' size. As we will see in the next Chapter 3, this property will be crucial in working as an instance selector in distance-related cut problems. We remark that Zschoche [Zsc17] employed the fractalism technique to exclude, assuming coNP $\not\subseteq$ NP$_{/\text{poly}}$, the existence of polynomial kernels for the problem of finding s-t separators in *temporal* graphs. Moreover, Zschoche [Zsc17] proposed a gadget based on grid graphs that could ensure planarity for the vertex-deletion variant of the T-fractal. However, the existence of such a variant remains open.

Open Problem 1. Is there a planar vertex-deletion variant of T-fractals suitable for the fractalism technique?

In general, we are curious about more fractal-like graphs with interesting properties. In particular, we wonder whether fractal-like graphs are suitable to answer open problems like whether Directed Feedback Vertex Set parameterized by the solution size [Fel+12a] or Imbalance parameterized by the imbalance [LMS13] admit polynomial kernels or not (see also, e.g., [BFS10, Cyg+14]).

Chapter 3.

Distance-Related Cut Problems

We rule out the existence of polynomial kernels for several graph problems (and their planar and directed variants) including the Length-Bounded Edge-Cut problem assuming coNP $\not\subseteq$ NP$_{/\text{poly}}$. To this end, we employ our fractalism technique described in Chapter 2.

3.1. Introduction

Edge-cut problems are fundamental problems in graph-theoretic research since decades. One of them is the well-known, polynomial-time solvable Minimum Cut problem, where given a graph G with two designated vertices s and t and an integer k, the question is whether at most k edge deletions can disconnect s and t. In this chapter, we provide kernelization lower bounds for three NP-hard, yet fixed-parameter tractable edge-cut problems employing our fractalism technique described in the previous Chapter 2.

If k edge-deletions do not suffice for disconnecting s and t, then one possible goal is to increase the distance of a shortest path between s and t as much as possible. This problem then becomes NP-hard [IPS82], and is formally defined as follows.

This chapter is based on (parts of) *Fractals for kernelization lower bounds* by Till Fluschnik, Danny Hermelin, André Nichterlein, and Rolf Niedermeier (SIAM Journal on Discrete Mathematics [Flu+18a]).

Table 3.1.: Overview of our results (assuming that coNP $\nsubseteq$ NP$_{/\text{poly}}$). PK stands for polynomial kernel. † Even for directed acyclic graphs. ‡ Even when parameterized by $k+\ell+\text{tw}$, where tw denotes the treewidth.

Problem	Param.	directed	undirected		
		planar	planar	general	
LBEC	$k+\ell$	No PK† (Thm. 3.1(i))	No PK‡ (Thm. 3.1(ii))		(Sec. 3.2)
MDED	$k+\ell$	No PK (Thm. 3.9(i))	No PK‡ (Thm. 3.9(ii))		(Sec. 3.3)
DSCT	$k+\ell$	No PK (Thm. 3.15)	PK [XZ11]	*open*	(Sec. 3.4)

Length-Bounded Edge-Cut (LBEC)
Input: An undirected graph $G=(V,E)$ with two distinct vertices $s,t\in V$, and two integers $k,\ell\geq 0$.
Question: Is there a subset $F\subseteq E$ of cardinality at most k such that $\text{dist}_{G-F}(s,t)\geq \ell$?

Golovach and Thilikos [GT11] proved LBEC parameterized by $k+\ell$ to be fixed-parameter tractable and asked whether it admits a polynomial kernel, but its status concerning polynomial kernelization remained open since then.[1]

The further two edge-cut problems are Minimum Diameter Edge Deletion (MDED) and Directed Small Cycle Transversal (DSCT). MDED asks, given an undirected connected graph $G=(V,E)$ and two integers k,ℓ, whether there are at most k edge deletions such that the resulting graph remains connected and has diameter at least ℓ. DSCT asks, given a directed graph $G=(V,E)$ and two integers k,ℓ, whether there are at most k edge deletions such that the resulting graph has no cycle of length smaller than ℓ.

Using our fractalism technique, we show that LBEC, MDED, DSCT, and several of their variants (planar, directed), each parameterized by $k+\ell$, admit no polynomial kernels assuming coNP $\nsubseteq$ NP$_{/\text{poly}}$. Table 3.1 surveys our no-polynomial-kernel results and spots an open question. We remark that we also show that for the undirected (planar) variants, unless coNP $\subseteq$ NP$_{/\text{poly}}$, LBEC and MDED parameterized by $k+\ell+\text{tw}$ do not admit a polynomial kernel, where tw denotes the treewidth of the input graph. On the way to obtain our results, we also prove LBEC to be NP-hard even on planar graphs—a result that may be of independent interest.

[1]The question also appeared in the open problem list of the FPT School 2014, Będlewo, Poland [Cyg+14].

As a technical remark, to simplify the presentation we employ T-fractals with edge costs in our proofs; However, all proofs work also without edge costs (see Section 2.4) thereby introducing a factor of two for the lengths inside the T-fractals (in the definition of each distance parameter ℓ).

Note that we claim without proof that, except for the planar variants, our proofs also transfer to the vertex deletion case, both for directed and undirected graphs (see Section 2.6).

3.2. Length-Bounded Edge-Cut

Our first application of the fractalism technique is the Length-Bounded Edge-Cut problem [Bai+10], also known as the problem of finding bounded edge undirected cuts [GT11], or the Shortest Path Most Vital Edges problem [Baz+19, BNN15, MMG89].

Length-Bounded Edge-Cut is NP-complete [IPS82] and fixed-parameter tractable with respect to $k + \ell$ [GT11]. If the *budget* k, that is, the number k of edge deletions, is at least the size of any s-t edge cut, then the problem becomes polynomial-time solvable by simply computing a minimum s-t edge cut. Thus, throughout this section, we assume that k is smaller than the size of any minimum s-t edge cut. The generalized problem where each edge is equipped with a positive length remains NP-hard even on series-parallel and outerplanar graphs [Bai+10]. The directed variant with positive edge lengths remains NP-hard on planar graphs where the source and the sink vertex are incident to the same face [PS16]. Dvorák and Knop [DK18] proved the problem to be W[1]-hard when parameterized by the pathwidth and solvable in polynomial time on graphs of bounded treewidth (see also [Kno17]). Here, we prove the following.

Theorem 3.1. *Unless coNP $\subseteq$ NP$_{/poly}$,* Length-Bounded Edge-Cut *admits no polynomial kernel when*

(i) parameterized by $k + \ell + \mathrm{tw}$, even on planar undirected graphs;
(ii) parameterized by $k + \ell$, even on planar directed acyclic graphs.

To prove Theorem 3.1, we prove LBEC on planar graphs to be NP-hard (deferred to Section 3.2.1, see Theorem 3.6), where the planar variant of LBEC is defined as follows.

PLANAR LENGTH-BOUNDED EDGE-CUT (PLANAR-LBEC)
Input: An undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, and two integers $k, \ell \geq 0$, where G admits a planar embedding with s and t being incident to the outer face.
Question: Is there a subset $F \subseteq E$ of cardinality at most k such that $\operatorname{dist}_{G-F}(s,t) \geq \ell$?

Indeed, in Theorem 3.6, we prove PLANAR-LBEC to be NP-hard on undirected graphs as well as on directed acyclic graphs, where in both cases the source and sink vertices are incident to the outer face. The property that the source and the sink vertices are allowed to be incident with the same face in the input graph allows us to use Constructions 2.12 and 2.16 with a target problem on planar graphs. We first prove Theorem 3.1(i), that is, we prove the following.

Proposition 3.2. *Unless coNP $\subseteq$ NP$_{/poly}$,* PLANAR LENGTH-BOUNDED EDGE-CUT *parameterized by $k + \ell + \mathrm{tw}$ admits no polynomial kernel.*

We prove Proposition 3.2 using an OR-cross-composition. To this end, we firstly define a polynomial equivalence relation on PLANAR-LBEC as follows.

Definition 3.1. An instance $(G_i, s_i, t_i, k_i, \ell_i)$ of PLANAR-LBEC is called *malformed* if $\max\{k_i, \ell_i\} > |E(G_i)|$. Two instances $(G_i, s_i, t_i, k_i, \ell_i)$ and $(G_j, s_j, t_j, k_j, \ell_j)$ of PLANAR-LBEC are *$\mathcal{R}$-equivalent* if $k_j = k_i$ and $\ell_j = \ell_i$, or both are malformed instances.

We prove that Definition 3.1 gives a polynomial equivalence relation.

Lemma 3.3. *Relation $\mathcal{R}$ from Definition 3.1 is a polynomial equivalence relation on* PLANAR-LBEC.

Proof. We can decide whether two instances $\mathcal{I}$ and $\mathcal{I}'$ of PLANAR-LBEC are $\mathcal{R}$-equivalent in $O(|\mathcal{I}| + |\mathcal{I}'|)$ time. Let $S := S_G \uplus S_B \subseteq \Sigma^*$ be a set of instances, where S_G denotes the not-malformed and S_B denotes the malformed instances. Note that all instances in S_B fall into one equivalence class. Let S_G contain p instances $\mathcal{I}_1, \ldots, \mathcal{I}_p$, where $\mathcal{I}_i = (G_i, s_i, t_i, k_i, \ell_i)$ for each $i \in \{1, \ldots, p\}$. Let $k := \max_{i \in \{1,\ldots,p\}} k_i$ and let $\ell := \max_{i \in \{1,\ldots,p\}} \ell_i$. Clearly, $k + \ell \leq 2 \max_{\mathcal{I} \in S} |\mathcal{I}|$. Hence, there are at most $(k+1) \cdot (\ell+1)$ equivalence classes in S_G. It follows that there are most $(k+1) \cdot (\ell+1) + 1$ equivalence classes in S. ☐

We next give the construction for the OR-cross-composition, which is basically Construction 2.12 from our fractalism technique.

Construction 3.4. Let $\mathcal{I}_1, \ldots, \mathcal{I}_p$, where $\mathcal{I}_i = (G_i, s_i, t_i, k, \ell)$ for each $i \in \{1, \ldots, p\}$, be $p = 2^q$, $q \in \mathbb{N}$, $\mathcal{R}$-equivalent instances of Planar-LBEC, where $\mathcal{R}$ is as in Definition 3.1. We construct an instance $\mathcal{I} := (G, s, t, k', \ell')$ of Planar-LBEC with

$$\begin{aligned} k' &:= k^2 \cdot (\log(p) + 1) + k \text{ and} \\ \ell' &:= \ell + \log(p), \end{aligned}$$

as follows.

Let G denote the graph obtained from the application of Construction 2.12 to $\mathcal{I}_1, \ldots, \mathcal{I}_p$ with edge cost $c := k^2$. Set $s := \sigma$ and $t := \tau$. This finishes the construction. Note that G is planar, and by Observation 2.14, the treewidth $\operatorname{tw}(G)$ of G is at most $2 + \max_{i \in \{1,\ldots,p\}} |V(G_i)|$. ▲

It remains to prove that the instance obtained from Construction 3.4 is a yes-instance if and only if at least one of the input instances is a yes-instance. Thus, we are ready for the proof of Proposition 3.2.

Proof of Proposition 3.2. We OR-cross-compose $p = 2^q$, $q \in \mathbb{N}$, $\mathcal{R}$-equivalent instances $\mathcal{I}_1, \ldots, \mathcal{I}_p$, where $\mathcal{I}_i = (G_i, s_i, t_i, k, \ell)$ for each $i \in \{1, \ldots, p\}$, of Planar-LBEC into one instance of Planar-LBEC, where $\mathcal{R}$ is as in Definition 3.1. Due to Lemma 3.3, we know that $\mathcal{R}$ is a polynomial equivalence relation on Planar-LBEC. We remark that we can assume that $\ell \geq 3$, since otherwise Planar-LBEC is solvable in polynomial time by counting all edges connecting the source with the sink vertex. Let $\mathcal{I} := (G, s, t, k', \ell')$ be the instance of Planar-LBEC obtained from applying Construction 3.4 given $\mathcal{I}_1, \ldots, \mathcal{I}_p$. We show that $\mathcal{I}$ is a yes-instance if and only if there exists an $i \in \{1, \ldots, p\}$ such that $\mathcal{I}_i$ is a yes-instance.

($\Leftarrow$) Let $\mathcal{I}_i$, $i \in \{1, \ldots, p\}$, be a yes-instance. Following Observation 2.13 in Section 2.5, let C be the minimum s-t edge cut in $\triangle_q^c$ that selects instance $\mathcal{I}_i$. Recall that C is of size $q + 1$ and that the edge cost equals k^2. Thus, the minimum s-t edge cut C has cost $(q + 1) \cdot k^2 = (\log(p) + 1) \cdot k^2$.

Note that after deleting the edges in C, the vertices s and t are only connected via paths through the incorporated graph G_i. Since $\mathcal{I}_i$ is a yes-instance, we can delete k edges (equal to the remaining budget) such that the distance of s_i and t_i in G_i is at least ℓ. Together with Lemma 2.7 in Section 2.2.1, such an additional edge deletion increases the length of any shortest s-t path in G to at least $\ell + \log(p) = \ell'$. Hence, $\mathcal{I}$ is a yes-instance.

($\Rightarrow$) Let $\mathcal{I}$ be a yes-instance, that is, one can delete at most k' edges in G such that each s-t path is of length at least ℓ'. Since the budget allows for $\log(p)+1$ edge-deletions in $\triangle_q^c$, by Lemma 2.8 in Section 2.2.1, if we do not cut s and t in $\triangle_q^c$, then there is an s-t path of length $\log(p)+2$. Since $\ell \geq 3$, such an edge deletion does not yield a solution. Thus, in every solution of $\mathcal{I}$, a subset of the deleted edges forms a minimum s-t edge cut in $\triangle_q^c$ and thus, by Observation 2.13, selects an input instance.

Consider an arbitrary solution to $\mathcal{I}$, that is, an edge subset of $E(G)$ of cardinality at most k' whose deletion increases the shortest s-t path to at least ℓ'. Let $\mathcal{I}_i$, $i \in \{1,\ldots,p\}$, be the selected instance. Note that any shortest s-t path contains edges in the selected instance $\mathcal{I}_i$. By Lemma 2.7, we know that the length of the shortest s-s_i path and the length of the shortest t_i-t path sum up to exactly $\log(p)$. It follows that the remaining budget of k edge deletions is spent in G_i in such a way that there is no path from s_i to t_i of length smaller than ℓ in G_i. Hence, $\mathcal{I}_i$ is a yes-instance. $\square$

Golovach and Thilikos [GT11] showed that LBEC on directed acyclic graphs is NP-complete. We prove in Section 3.2.1, Theorem 3.6, that Planar-LBEC on directed acyclic graphs, where s and t are incident to the outer face, remains NP-hard. Using Construction 2.16 instead of Construction 2.12 with Planar-LBEC on directed acyclic graphs as source and target problem, the same arguments as in the proof of Proposition 3.2 give Theorem 3.1(ii).

Proposition 3.5. *Unless coNP $\subseteq$ NP$_{/poly}$,* Planar Length-Bounded Edge-Cut *on directed acyclic graphs parameterized by $k+\ell$ admits no polynomial kernel.*

3.2.1. NP-hardness of LBEC on Planar Graphs

In the following, we consider LBEC on planar graphs. To the best of our knowledge, it was open whether LBEC remains NP-hard on planar undirected graphs. This is what we will prove next. In fact, we prove the following, even stronger result.

Theorem 3.6. Length-Bounded Edge-Cut *is NP-hard even on planar undirected graphs as well as on planar directed acyclic graphs, where for both problems s and t are incident to the outer face.*

For our proof of the theorem, we need the following definitions of planar embeddings of graphs.

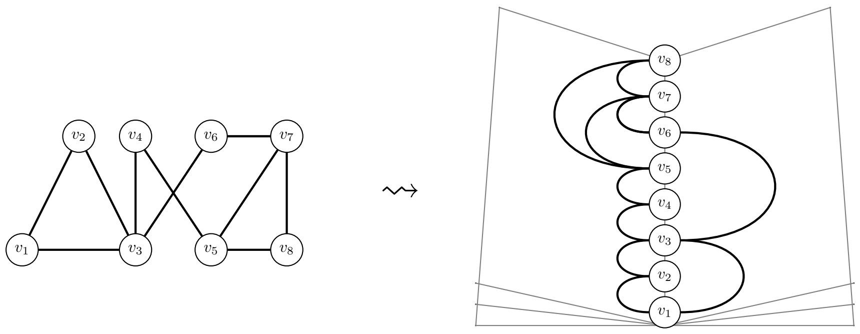


Figure 3.1.: A 2-page book embedding (right-hand side) for an example graph (left-hand side). Note that the example graph is planar and of maximum degree four.

Definition 3.2. A *page embedding* of a graph G is a planar embedding of G where all vertices lie on the real line and every edge lies in the upper half $\mathbb{R} \times \mathbb{R}^+$.

Definition 3.3. A graph $G = (V, E)$ admits a *k-page book embedding* if there is a partition $E_1, \dots, E_k$ of the edge set E such that $G_i := (V, E_i)$ admits a page embedding for all $i \in \{1, \dots, k\}$.

Intuitively, a k-page book embedding of a graph is a drawing of the graph where all vertices are drawn along the spine of the book, each edge is drawn on one of the k pages of the book, and each page is crossing-free (see Figure 3.1 for an illustration).

Our proof follows the same strategy as the proof due to Bar-Noy et al. [BKS95] for LBEC on general graphs, where Bar-Noy et al. [BKS95] reduce Vertex Cover to LBEC. We reduce from 3-Planar Vertex Cover, that is, Vertex Cover on planar graphs with maximum degree three, which remains NP-complete [Moh01]. Bekos et al. [BGR14] proved that any planar graph of maximum degree four allows for a 2-page book embedding. Moreover, Heath [Hea85] showed that a 2-page book embedding of any planar graph of maximum vertex degree three can be computed in time linear in the number of vertices of the input graph. We mainly copy the proof due to Bar-Noy et al. [BKS95] and, on the way, perform small changes on the gadgets and target parameters. We describe this in the following.

Construction 3.7. Let $\mathcal{I} = (G, k)$ be an instance of 3-Planar Vertex Cover. Since we can assume to have a 2-page book embedding, the vertices are drawn along the real line and connected by non-crossing edges lying in the lower and upper half. Further, we assume that the vertices are labeled from 1 to n, in the order along the real line. Refer to Figure 3.2 for an illustration of the following construction. We replace each vertex i by a gadget $\mathbf{i}$ as follows. The gadget $\mathbf{i}$ consists of two P_{2k}s, where P_{2k} denotes a simple path with $2k$ vertices, and one P_{2k+1}, all three identified at their endpoints. We denote the left and right (identified) endpoint of gadget $\mathbf{i}$ by s_i and t_i, respectively. One P_{2k} belongs to the upper half, the other to the lower half. The P_{2k+1} lies along the real line. We denote the two middle vertices of each of the two P_{2k}s by $x_i^u, x_i^\ell, y_i^u, y_i^\ell$, where x is left of y, and u and ℓ stand for "upper" and "lower". We identify t_i with s_{i+1} for all $i \in \{1, \ldots, n-1\}$. We set $s := s_1$ and $t := t_n$. Moreover, if two vertices $i < j$ are connected by an edge lying in the upper half, then we connect the vertex y_i^u with x_j^u via a path of length $(2k-1)(j-i)-2$ (analogously for edges in the lower half). We denote by G' the obtained graph. Observe that G' remains planar. Except for the edges $\{x_i^u, y_i^u\}, \{x_i^\ell, y_i^\ell\}$, $i \in \{1, \ldots, n\}$, there are no edges that are allowed to be deleted (see Bar-Noy et al. [BKS95]).

We set

$$\begin{aligned} k' &:= 2k \text{ and} \\ \ell' &:= k \cdot (2k) + (n-k) \cdot (2k-1). \end{aligned}$$

Let $\mathcal{I}' := (G', s, t, k', \ell')$ be the resulting instance of Planar-LBEC with forbidden edges. ▲

Proof of Theorem 3.6. Let $\mathcal{I} = (G, k)$ be an instance of 3-Planar Vertex Cover. Let $\mathcal{I}' := (G', s, t, k', \ell')$ be the instance of Planar-LBEC with forbidden edges obtained from $\mathcal{I}$ by applying Construction 3.7. We prove that $\mathcal{I}$ is a yes-instance of 3-Planar Vertex Cover if and only if $\mathcal{I}'$ is a yes-instance of Planar-LBEC with forbidden edges.

($\Rightarrow$) Let $\mathcal{I}$ be a yes-instance, that is, G admits a vertex cover of size at most k. Let $C \subseteq V(G)$ be a vertex cover of size k. We claim that the edge set $X := \{\{x_i^u, y_i^u\}, \{x_i^\ell, y_i^\ell\} \mid i \in C\}$ forms a solution to $\mathcal{I}'$.

Observe that any s-t path in $G' - X$ using only edges in the gadgets is of length at least ℓ'. To see this, consider a gadget $\mathbf{i}$ with $i \in C$. We know that $\{x_i^u, y_i^u\}, \{x_i^\ell, y_i^\ell\} \in X$. Hence, the only s_i-t_i path using only edges in the gadget $\mathbf{i}$ is of length $2k$ (this is the P_{2k+1} used in the construction). If no edge in a gadget $\mathbf{j}$ is deleted, then any shortest s_j-t_j path using only edges in the gadget $\mathbf{j}$ is of length $2k-1$ (those correspond to the P_{2k}s used in the

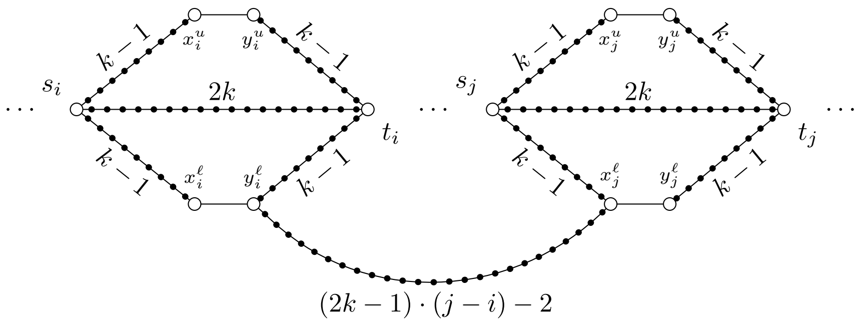


Figure 3.2.: Illustration of the gadgets in the proof of Theorem 3.6. Here, exemplified for two vertices $i, j \in V$ with $\{i, j\} \in E$, and the edge is embedded on the second (lower) page in the 2-page book embedding of the input graph $G = (V, E)$.

construction). Since $|C| = k$, any s-t path in $G' - X$ using only edges in the gadgets is of length at least $k \cdot (2k) + (n-k)\cdot(2k-1) = \ell'$.

We have to show that there is no shorter s-t path in $G' - X$ than any path using only edges in the gadgets. To this end, let $i, j \in V(G)$, $i < j$, be two adjacent vertices in G, that is, with $\{i, j\} \in E(G)$. Since C is a vertex cover, it follows that either $i \in C$ or $j \in C$. Let $i \in C$ and $j \notin C$ (the case with $j \in C$ and $i \notin C$ is symmetric). We consider the shortest path from s_i to t_j not going backwards, that is, not appearing in any gadget z with $z < i$ or $z > j$, and using the path connecting the gadgets of i and j. Let the path connecting the gadgets of i and j be a lower path, that is, the vertices y_i^ℓ and x_j^ℓ are connected by the path. Since the edges $\{x_i^\ell, y_i^\ell\}$ and $\{x_i^u, y_i^u\}$ are deleted, the shortest path from s_i to y_i^ℓ is of length $2k + (k-1)$. Then we take the path of length $(2k-1)(j-i)-2$ to get to the gadget of j. Finally, we take the path from x_j^ℓ via edge $\{x_j^\ell, y_j^\ell\}$ to t_i of length $(k-1)+1 = k$. In total, the path is of length

$$4k + (2k-1)(j-i) - 3, \tag{3.1}$$

and it is the shortest of its kind.

We compare this to the shortest path from s_i to t_j using only edges in the gadgets. The length of such a path is at most

$$2k(j-i) - (j-i-k) + (2k-1) \quad \text{if } j-i \geq k, \text{ and} \tag{3.2}$$

$$2k(j-i) + (2k-1) \quad \text{otherwise.} \tag{3.3}$$

We compare (3.1) with each of (3.2) and (3.3). Comparing (3.1) with (3.2), we have

$$4k + (2k-1)(j-i) - 3 - (2k(j-i) - (j-i-k) + (2k-1)) = k-2.$$

Comparing (3.1) with (3.3), we have (recall $j-i<k$)

$$4k + (2k-1)(j-i) - 3 - (2k(j-i) + (2k-1)) = 2k - (j-i) - 2 > k-2.$$

It follows that there is a path using only edges in the gadgets that is shorter than the shortest paths using at least one edge not appearing in the gadgets. Finally note that if both $i,j \in C$, then the difference of the path lengths is even bigger. Observe that using a path connecting gadget $\mathbf{i}$ with $\mathbf{j+1}$ (or $\mathbf{i-1}$ with $\mathbf{j}$) to get from s_i to t_j is longer by at least $k-1$ (or at least $k-3$), following from an analogous argumentation as above. Hence, the shortest path connecting s with t passes through the gadgets and is of length at least ℓ'.

($\Leftarrow$) Let $\mathcal{I}'$ be a yes-instance, that is, G' allows for $k' = 2k$ edge deletions such that any shortest s-t path is of length at least ℓ'. Our first observation is that in any solution to $\mathcal{I}$, either none or exactly two (of the allowed) edges are deleted in any gadget. Suppose that there is a gadget with only one edge deleted. Then a shortest path through this gadget is of length $2k-1$. Since $2k$ is the maximum increase of the passing length through a gadget, we get $(2k)(k-2) + (2k-1)(n-k+2) < (2k)\cdot k + (2k-1)(n-k) = \ell'$. Hence, in any gadget, either exactly two or zero edges are deleted. Let $C \subseteq V(G)$ be the set of vertices such that both edges are deleted in the corresponding gadgets. We claim that C is a vertex cover of size k in G.

Suppose that there are two gadgets $\mathbf{i}$ and $\mathbf{j}$, where $i<j$, not containing any deleted edge, that is, $\{i,j\} \cap C = \emptyset$, but $\{i,j\} \in E(G)$. Then the shortest s_i-t_j path using the path corresponding to edge $\{i,j\} \in E(G)$ is of length $2k + (2k-1)(j-i) - 2$. The shortest s_i-t_j path through the gadgets only is of length at least $2k-1+(2k-1)(j-i)$. Thus, the path using the path connecting the gadgets $\mathbf{i}$ and $\mathbf{j}$ is too short by exactly one, and hence, the shortest s-t path is of length smaller than ℓ'. This contradicts the fact that $\{\{x_i^u, y_i^u\}, \{x_i^\ell, y_i^\ell\} \mid i \in C\}$ forms a solution to $\mathcal{I}'$. It follows that for each edge $\{i,j\} \in E(G)$ we have $|C \cap \{i,j\}| > 0$. This is exactly the property of a vertex cover, and thus, C is a vertex cover in G of size k.

We have shown that Planar-LBEC with forbidden edges is NP-hard on planar, undirected graphs. In analogue with Bar-Noy et al. [BKS95], to reduce to Planar-LBEC (without types of edges), we replace each forbidden edge by $k'+1$ (parallel) edges, subdivide each edge, and double ℓ' accordingly.

Observe that we can direct all edges from "left to right". The planarity still holds, and we obtain a directed acyclic graph. Since we have shown in the proof that "going backwards" is never optimal, the proof can be easily adapted. Thus, the problem remains NP-hard on planar directed acyclic graphs. □

3.3. Minimum Diameter Edge Deletion

Our second fractalism application concerns about the following problem introduced by Schoone et al. [SBL87].

Minimum Diameter Edge Deletion (MDED)
Input: A connected, undirected graph $G = (V, E)$, and two integers $k, \ell \geq 0$.
Question: Is there a subset $F \subseteq E$ of cardinality at most k such that $G - F$ is connected and $\operatorname{diam}(G - F) \geq \ell$?

The problem was shown to be NP-complete, also on directed graphs [SBL87]. In their NP-hardness-proof for MDED, Schoone et al. [SBL87] reduce from Hamiltonian Path (HP) to MDED. The reduction does not modify the graph, that is, the input graph for HP remains the same for the MDED instance. Since HP remains NP-hard on planar graphs [GJS76], the reduction of Schoone et al. implies that MDED is NP-hard even on planar graphs.

A simple search tree algorithm yields fixed-parameter tractability with respect to $k + \ell$:

Theorem 3.8. Minimum Diameter Edge Deletion *can be solved in* $O((\ell-1)^k n^2(n+m))$ *time.*

Proof. We give a search tree algorithm branching over the possible edge deletions to prove that MDED is fixed-parameter tractable when parameterized by $k + \ell$. The key observation is that if some instance (G, k, ℓ) of MDED is a `yes`-instance, then there exists at least one pair $v, w \in V$ of vertices in the graph $G - X$ such that $\operatorname{dist}_{G-X}(v, w) \geq \ell$, where X is a solution to (G, k, ℓ). Hence, we will check whether the length of any shortest path between the chosen pair can be increased to at least ℓ by at most k edge deletions, where an edge is only deleted if the deletion leaves the graph connected.

To this end, for each pair, apply the branching algorithm provided by Golovach and Thilikos [GT11]: Find a shortest path and if its length is at most $\ell - 1$, then branch in all cases of deleting an edge on this path and decrease k by one. In each branch, check whether the graph is still connected. This can be done in $O(n+m)$ time with a simple depth/breadth first search. Hence, in total we obtain a branching algorithm running in $O(n^2 \cdot (\ell-1)^k(n+m))$ time. Thus, MDED parameterized by $k+\ell$ is fixed-parameter tractable. $\square$

Complementing the fixed-parameter tractability of MDED parameterized by $k+\ell$, we show the following.

Theorem 3.9. *Unless coNP $\subseteq$ NP$_{/poly}$,* Minimum Diameter Edge Deletion *admits no polynomial kernel when*

(i) parameterized by $k+\ell+\mathrm{tw}$, even on planar graphs;
(ii) parameterized by $k+\ell$, even on planar directed graphs.

To prove Theorem 3.9, we give OR-cross-compositions with Planar-LBEC as input problem using Construction 2.12 for Theorem 3.9(i) and Construction 2.16 for Theorem 3.9(ii). We first give the proof of Theorem 3.9(i).

Proposition 3.10. *Unless coNP $\subseteq$ NP$_{/poly}$,* Minimum Diameter Edge Deletion *on planar graphs parameterized by $k+\ell+\mathrm{tw}$ admits no polynomial kernel.*

In the proof of Proposition 3.10, we use the following modification of Construction 2.12.

Construction 3.11. Let $\mathcal{I}_1 = (G_1, s_1, t_1, k, \ell), \dots, \mathcal{I}_p = (G_p, s_p, t_p, k, \ell)$ be $p = 2^q$, $q \in \mathbb{N}$, $\mathcal{R}$-equivalent instances of the input problem Planar-LBEC on connected graphs, where $\mathcal{R}$ is defined as in Definition 3.1. Construct an instance $\mathcal{I} := (G, k', \ell')$ of MDED with

$$k' := k^2 \cdot (\log(p)+1) + k \text{ and}$$
$$\ell' := 2 \cdot L + \log(p) + \ell,$$

where

$$L := n_{\max} \cdot (2\log(p)+3)+1 \text{ and}$$
$$n_{\max} := \max_{i \in \{1,\dots,p\}} |V(G_i)|,$$

as follows (refer to Figure 3.3 for an exemplified illustration). Apply Construction 2.12 with edge cost $c := k^2$. Attach to σ as well as to τ a path of length L

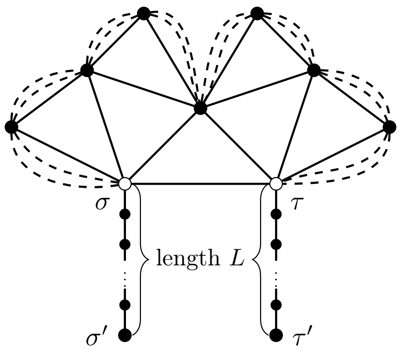


Figure 3.3.: Cross-composition (Construction 3.11) for Minimum Diameter Edge Deletion with $p = 8 = 2^3$, and $L = 9 \cdot n_{\max} + 1$. Dashed lines sketch the graphs in the p input instances.

each. Denote the endpoint of the path attached to σ by σ' (where $\sigma' \neq \sigma$), and let τ' be defined analogously. Let G denote the obtained graph. Note that the appended paths to the T-fractal do not increase the treewidth $\mathrm{tw}(G)$ of G, and hence by Observation 2.14, it holds that $\mathrm{tw}(G) \leq 2 + n_{\max}$. ▲

Proof of Proposition 3.10. We OR-cross-compose $p = 2^q$, $q \in \mathbb{N}$, instances of Planar-LBEC on connected graphs into one instance of MDED. Let $\mathcal{I}_1 = (G_1, s_1, t_1, k, \ell), \dots, \mathcal{I}_p = (G_p, s_p, t_p, k, \ell)$ be $p = 2^q$, $\mathcal{R}$-equivalent instances of Planar-LBEC on connected graphs, where $\mathcal{R}$ is defined as in Definition 3.1. Apply Construction 3.11 in polynomial time to obtain instance $\mathcal{I} := (G, k', \ell')$ of MDED. Finally, for each $i \in \{1, \dots, p\}$, add a path of length $\ell + 1$ and identify one endpoint with s_i and the other endpoint with t_i. We show that $\mathcal{I}$ is a yes-instance of MDED if and only if there exists an $i \in \{1, \dots, p\}$ such that $\mathcal{I}_i$ is a yes-instance of Planar-LBEC on connected graphs.

($\Leftarrow$) Let $\mathcal{I}_i$, $i \in \{1, \dots, p\}$, be a yes-instance of Planar-LBEC on connected graphs. Following Observation 2.13, we delete all edges in the minimum edge cut in $\triangle_q^c$ that selects instance $\mathcal{I}_i$. Then, we delete edges corresponding to a solution for $\mathcal{I}_i$ without disconnecting the graph G (note that in our final step in the construction, we added a path of length $\ell + 1$ connecting s_i and t_i). Let $X \subseteq E(G)$ be the set of deleted edges. The distance of σ and τ in $G - X$ is at least $\log(p) + \ell$, and thus, the distance of σ' and τ' is at least $2 \cdot L + \log(p) + \ell = \ell'$.

Hence, the diameter is at least ℓ' after k' edge deletions that leave the graph connected. It follows that $\mathcal{I}$ is a `yes`-instance.

($\Rightarrow$) Let $\mathcal{I}$ be a `yes`-instance of MDED, that is, G allows for k' edge deletions such that the remaining graph is connected and has diameter at least ℓ'. Let $X \subseteq E(G)$ be a solution. First observe that $G - X$ is connected. Consider the instances appended to the T-fractal as the artificial $(q+1)$st boundary of a $(q+1)$-T-fractal, where an edge in this boundary has length $n_{\max}$. Thus, we can apply Lemma 2.9(A) to this artificial $(q+1)$-T-fractal. Recall that our budget only allows for $\log(p)+1$ edge deletions (of cost k^2) in $\triangle_q^c$. Hence, we get that the distance to σ (and by symmetry to τ) of every vertex contained either in $\triangle_q^c$ or in any appended instance is at most $n_{\max} \cdot (\log(p) + \log(p) + 3) = L - 1$. It follows that $\operatorname{dist}_{G-X}(x, \sigma) \le \operatorname{dist}_{G-X}(\sigma, \sigma')$ and $\operatorname{dist}_{G-X}(x, \tau) \le \operatorname{dist}_{G-X}(\tau, \tau')$ for all $x \in V(G)$. Moreover, for all $x, y \in V(G)$ we have:

$$\begin{aligned}
\operatorname{dist}_{G-X}(x,y) &\le \operatorname{dist}_{G-X}(x,\sigma) + \operatorname{dist}_{G-X}(\sigma,\tau) + \operatorname{dist}_{G-X}(\tau,y) \\
&\le \operatorname{dist}_{G-X}(\sigma',\sigma) + \operatorname{dist}_{G-X}(\sigma,\tau) + \operatorname{dist}_{G-X}(\tau,\tau') \\
&= \operatorname{dist}_{G-X}(\sigma',\tau').
\end{aligned}$$

Hence, σ', τ' is the pair of vertices with the largest distance in $G - X$ and, thus, $\operatorname{dist}_{G-X}(\sigma', \tau') \ge \ell'$. Observe that $\operatorname{dist}_{G-X}(\sigma', \tau') \ge \ell'$ if and only if $\operatorname{dist}_{G-X}(\sigma, \tau) \ge \log(p) + \ell$ since every shortest σ'-τ' path contains both σ and τ. Following the argumentation in the correctness proof of Proposition 3.2, it follows that there is an instance $\mathcal{I}_i$, $i \in \{1, \ldots, p\}$, that is a `yes`-instance for Planar-LBEC on connected graphs. □

The diameter of a directed graph is defined as the maximum length of a shortest directed path over any two (ordered) vertices. The diameter of a directed graph that is not strongly connected equals infinity. Thus, Minimum Diameter Edge Deletion on directed graphs is referred to and defined as follows:

Minimum Diameter Arc Deletion (MDAD)
Input: A strongly connected directed graph $G = (V, E)$, and two integers $k, \ell \ge 0$.
Question: Is there is a subset $F \subseteq E$ of cardinality at most k such that $G - F$ is strongly connected and $\operatorname{diam}(G - F) \ge \ell$?

Observe that Minimum Diameter Arc Deletion on directed planar graphs parameterized by $k + \ell$ is fixed-parameter tractable, as a consequence of the proof of Theorem 3.8. Next we prove the problem to presumably admit no polynomial kernel regarding $k+\ell$, that is, we prove Theorem 3.9(ii).

Proposition 3.12. *Unless coNP $\subseteq$ NP$_{/poly}$,* MINIMUM DIAMETER ARC DELETION *on directed planar graphs parameterized by $k+\ell$ admits no polynomial kernel.*

The proof of Proposition 3.12 adapts the ideas of the proof of Proposition 3.10. Herein, we use the following modification of Construction 2.16.

Construction 3.13. Let $\mathcal{I}_1 = (G_1, s_1, t_1, k, \ell), \ldots, \mathcal{I}_p = (G_p, s_p, t_p, k, \ell)$ be $p = 2^q$, $q \in \mathbb{N}$, $\mathcal{R}$-equivalent instances of PLANAR-LBEC on directed acyclic graphs where $\mathcal{R}$ is defined as in Definition 3.1. Construct an instance $\mathcal{I} := (G, k', \ell')$ of MDAD with

$$\begin{aligned} k' &:= k^2 \cdot (\log(p)+1) + k \text{ and} \\ \ell' &:= 2 \cdot L + \log(p) + \ell, \end{aligned}$$

where

$$\begin{aligned} L &:= \ell \cdot n_{\max} \cdot (2\log(p)+3) + 1 \text{ and} \\ n_{\max} &:= \max_{i \in \{1,\ldots,p\}} |V(G_i)|, \end{aligned}$$

as follows.

First, we adjust the instances we compose in order to ensure that every vertex is reachable from the source and reaches the sink, and that we can delete all the arcs we want without destroying the aforementioned property. For each arc $(v,w) \in E(G_i)$, connect v and w by an additional path of length ℓ directed from v towards w. Apply this for each G_i, $i \in \{1,\ldots,p\}$, and let G_i' be the graph obtained from graph G_i. Note that the directed graph G_i' remains planar and acyclic. Observe that none of the introduced arcs will be in a minimal solution for the PLANAR-LBEC instance since they only occur in paths of length ℓ. Hence, $\mathcal{I}_i$ is a yes-instance of PLANAR-LBEC on directed acyclic graphs if and only if $(G_i', s_i, t_i, k, \ell)$ is a yes-instance of PLANAR-LBEC on directed acyclic graphs. Furthermore, in the composed MINIMUM DIAMETER ARC DELETION-instance, none of the introduced arcs will be deleted as this would introduce a vertex without in-going or without out-going arcs and this is not allowed in the problem setting.

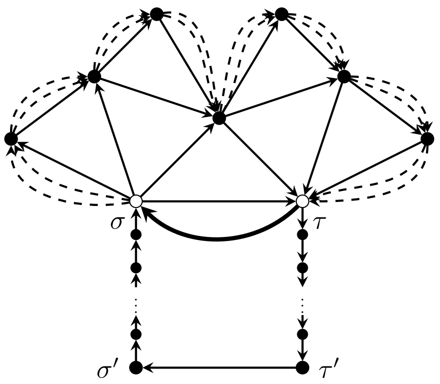


Figure 3.4.: Cross-composition (Construction 3.13) for Minimum Diameter Arc Deletion parameterized by $k + \ell$ with $p = 8 = 2^3$. Dashed lines sketch the graphs in the p input instances.

Apply Construction 2.16 with edge cost $c := k^2$ and the following additions (refer to Figure 3.4 for an exemplified illustration). Attach to σ as well as to τ a path of length L each. Denote the endpoint of the path attached to σ by σ' (where $\sigma' \neq \sigma$), and let τ' be defined analogously. Direct all edges in the paths from σ' towards to σ and from τ towards to τ', respectively. Moreover, add to the graph the arc (τ', σ'), and the arc (τ, σ), the latter with cost $k' + 1$.

Observe that G is planar, directed, and strongly connected. ▲

Proof of Proposition 3.12. We OR-cross-compose $p = 2^q$, $q \in \mathbb{N}$, $\mathcal{R}$-equivalent instances $\mathcal{I}_1 = (G_1, s_1, t_1, k, \ell), \ldots, \mathcal{I}_p = (G_p, s_p, t_p, k, \ell)$ of Planar-LBEC on directed acyclic graphs into one instance of MDAD on directed planar graphs, where $\mathcal{R}$ is defined as in Definition 3.1. Apply Construction 3.13 in polynomial time to obtain instance $\mathcal{I} := (G, k', \ell')$ of MDAD. We prove that $\mathcal{I}$ is a yes-instance of MDAD on directed planar graphs if and only if there exists an $i \in \{1, \ldots, p\}$ such that $\mathcal{I}_i$ is a yes-instance for Planar-LBEC on directed acyclic graphs.

($\Rightarrow$) Let $\mathcal{I}$ be a yes-instance of MDAD. Consider a solution $X \subseteq E(G)$ for the instance $\mathcal{I}$ of MDAD on directed planar graphs. The crucial observation is that for any two vertices x, y not contained in the attached paths with endpoints σ' on the one, and τ' on the other hand, the following holds: $\max\{\text{dist}_{G-X}(x, y), \text{dist}_{G-X}(y, x)\} \leq \text{dist}_{G-X}(\sigma', \tau')$. To see this, note that the arc (τ,σ) has cost $k'+1$ and thus $(\tau,\sigma)\notin X$. Since G is strongly connected, both x and y are reachable and reach σ and τ. Moreover, σ is reachable from τ via the arc (τ,σ). Without loss of generality, let $\operatorname{dist}_{G-X}(x,y) = \max\{\operatorname{dist}_{G-X}(x,y), \operatorname{dist}_{G-X}(y,x)\}$. It holds that

$$\begin{aligned}\operatorname{dist}_{G-X}(x,y) &\le \operatorname{dist}_{G-X}(x,\tau)+\operatorname{dist}_{G-X}(\tau,\sigma)+\operatorname{dist}_{G-X}(\sigma,y)\\ &\le \ell\cdot n_{\max}\cdot(2\log(p)+2)+1+\ell\cdot n_{\max}\cdot(2\log(p)+2)\\ &= 2\cdot\ell\cdot n_{\max}\cdot(2\log(p)+2)+1<\ell'.\end{aligned}$$

Herein, recall that we only allow $\log(p)+1$ arc deletions in $\vec{\triangle}^c_q$. The second inequality follows from Lemma 2.11 and the fact that in each graph $G_i - X$ the vertex s_i has distance at most $\ell\cdot n_{\max}$ to t_i.

As a consequence, the vertices at distance ℓ' appear in the paths appended on σ and τ. Among them, note that $\operatorname{dist}_{G-X}(\sigma',\tau')$ is maximal. Following the discussion in the proof of Proposition 3.10, the budget has to be spent in such a way that the arc deletions form a σ-τ arc cut in $\vec{\triangle}^c_q$, and the remaining budget must be spent in such a way that the instance $\mathcal{I}_i$ chosen by the cut allows no s_i-t_i path of length smaller than ℓ. Hence, $\mathcal{I}_i$ is a yes-instance.

($\Leftarrow$) Let $\mathcal{I}_i$ be a yes-instance of Planar-LBEC on directed acyclic graphs and let $X'\subseteq E(G_i)$ a minimum-size solution. We added to each arc of G_i a directed path of length ℓ and, as discussed above, none of the arcs in these paths is in X'. Hence, in $G_i - X'$ every vertex is still reachable from s_i and reaches t_i. Deleting in G the arcs in X' and the arcs corresponding to the cut selecting $\mathcal{I}_i$ preserves the strong connectivity of G. Let $X\subseteq E(G)$ be the set of deleted arcs. Following the discussion in the proof of Proposition 3.10, $\operatorname{dist}_{G-X}(\sigma',\tau')\ge\ell'$. It follows that $\mathcal{I}$ is a yes-instance of MDAD on directed planar graphs. □

3.4. Directed Small Cycle Transversal

Our third fractalism application concerns the following problem.

Directed Small Cycle Transversal (DSCT)
Input: A directed graph $G=(V,E)$, two integers $k,\ell\ge 0$.
Question: Is there a subset $F\subseteq E$ of cardinality at most k such that there is no induced directed cycle of length at most ℓ in $G-F$?

The problem is NP-hard [GL14], also on undirected graphs [Yan78]. The NP-completeness of DSCT follows by a simple reduction from the Feedback Arc

Set problem with an n-vertex graph, where we set $\ell := n$ and leave the graph unchanged in the reduction. We remark that the problem is also known as Cycle Transversal [Bod+16], or ℓ-(Directed)-Cycle Transversal [GL14]. The undirected variant is also known as Small Cycle Transversal [XZ11, XZ12].

As for the Minimum Diameter Edge Deletion problem, there is a simple search tree algorithm showing fixed-parameter tractability with respect to $k + \ell$.

Theorem 3.14. Directed Small Cycle Transversal *can be solved in* $O(\ell^k \cdot n \cdot (n + m))$ *time.*

Proof. We give a search tree algorithm branching over all possible edge deletions to prove that DSCT parameterized by $k + \ell$ is fixed-parameter tractable. Let (G, k, ℓ) be an instance of DSCT. To detect short cycles in G containing a vertex $v \in V(G)$, construct an auxiliary graph G_v as follows. Delete v (and all edges incident to v), and add v_{in} and v_{out}, and the arcs $\{(x, v_{\text{in}}) \mid (x, v) \in E(G)\}$, $\{(v_{\text{out}}, x) \mid (v, x) \in E(G)\}$ as well as the arc $(v_{\text{in}}, v_{\text{out}})$. Now to detect the shortest cycle in G containing v, compute a shortest v_{out}-v_{in} path in G_v. If a cycle is too short, then we branch into all possible, at most ℓ different deletions of an arc of the cycle (beside arc $(v_{\text{in}}, v_{\text{out}})$).

The depth of the search tree is at most k. Thus, we obtain an $O(\ell^k \cdot n \cdot (n+m))$-time algorithm since constructing for each $v \in V$ the auxiliary graph G_v and then finding a shortest path in unweighted graphs can be done in $O(n \cdot (n + m))$ time. □

However, the fractalism technique yields the following.

Theorem 3.15. *Unless* $\text{coNP} \subseteq \text{NP}_{/\text{poly}}$, Directed Small Cycle Transversal *on planar directed graphs parameterized by* $k + \ell$ *admits no polynomial kernel.*

Proof. We OR-cross-compose $p = 2^q$, $q \in \mathbb{N}$, $\mathcal{R}$-equivalent instances of Planar-LBEC on directed acyclic graphs into one instance of DSCT on planar directed graphs as follows, where $\mathcal{R}$ is defined as in Definition 3.1.

Construction: We apply Construction 2.16 with edge cost $c := k^2$. In addition, we add the edge (τ, σ) with edge cost $k' + 1$, where $k' := k^2 \cdot (\log(p) + 1) + k$. We denote by G the obtained graph. We refer to Figure 3.5 for an exemplified illustration of the construction. Observe that G is not acyclic, and the edge (τ, σ) participates in every cycle in G, that is, G without edge (τ, σ) is acyclic. Let (G, k', ℓ') be the target instance of DSCT with $\ell' := \ell + \log(p) + 1$.

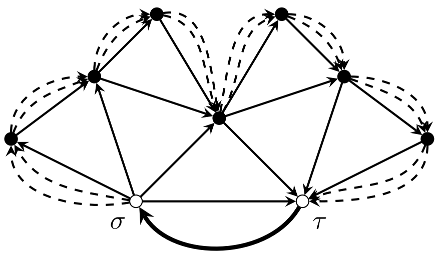


Figure 3.5.: Cross-composition (Theorem 3.15) for DSCT parameterized by k, ℓ with $p = 8 = 2^3$. Dashed lines sketch the graphs in the p input instances.

Correctness: Note that every cycle in G uses the edge (τ, σ). Since its edge cost equals $k' + 1$, the budget does not allow its deletion. Thus, the crucial observation is that the length of any shortest path from σ to τ must be increased to at least $\ell + \log(p) = \ell' - 1$. Hence, the correctness follows from Proposition 3.5. □

We remark that a straight-forward polynomial parameter transformation from Planar-LBEC on directed acyclic graphs to DSCT on planar directed graphs where we add a "too expensive" arc from the sink to the source vertex (which is possible since both vertices lie on the outer face) also proves Theorem 3.15.

3.5. Concluding Remarks

The fractalism technique makes it possible to prove kernelization lower bounds for parameterized distance-related cut-problems like Length-Bounded Edge-Cut parameterized by $k + \ell + \text{tw}$, even for restricted inputs like planar graphs. For the latter, we proved LBEC to be NP-hard on planar graphs. Building on this, Zschoche et al. [Zsc+18] proved that on planar graphs, the vertex-deletion variant of LBEC remains also NP-hard. Yet, polynomial kernelization for the vertex-deletion variant remains open.

Open Problem 2. Does the vertex-deletion variant of Length-Bounded Edge-Cut parameterized by $k + \ell$ admit a polynomial kernel on planar graphs?

Remarkably, Small Cycle Transversal parameterized by $k + \ell$ on planar undirected graphs admits a polynomial kernel [Bod+16, XZ11]. It remains open whether Small Cycle Transversal parameterized by $k + \ell$ admits a polynomial kernel in general undirected graphs.

Open Problem 3. Does Small Cycle Transversal parameterized by $k+\ell$ admit a polynomial kernel in general undirected graphs?

Part II.

Diminishers and Data Reduction inside P

All kernelization lower bounds derived via the (cross-)composition framework and polynomial parameter transformations rely on the assumption that $\mathrm{coNP} \not\subseteq \mathrm{NP}_{/\mathrm{poly}}$, an assumption that, while widely believed in the community, is a much stronger assumption than $\mathrm{P} \neq \mathrm{NP}$. Chen et al. [CFM11] were the first to observe a connection of kernelization lower bounds with the gold standard assumption of $\mathrm{P} \neq \mathrm{NP}$. These lower bounds apply to proper kernelizations, a variant of kernelization where the parameter value is not allowed to increase in the kernel. Chen et al. proved three parameterized problems, one of them being the CNF-SAT problem parameterized by the number n of variables, to admit no proper kernelization of polynomial size unless $\mathrm{P} = \mathrm{NP}$. Hence, a more restrictive kernelization is excluded under a weaker assumption. The key tool for proving these bounds are parameter-decreasing polynomial self-reductions [CFM11] that we will call *parameter diminishers* (diminishers for short). Roughly speaking, a diminisher is a polynomial-time algorithm that on any input instance outputs an equivalent instance where the parameter value is decreased. In the first chapter of this part, Chapter 4, we underline the applicability of the parameter diminisher framework. Therein, we prove the framework to apply to so-called *strict* kernelizations (where the parameter value is allowed to only increase by some general constant). We then show many parameterized problems to be diminishable—and hence, to not admit strict kernelizations of polynomial size unless $\mathrm{P} = \mathrm{NP}$.

Motivated by this finding, it is natural to ask about the limits and further applications of the diminisher framework. More specifically, we were curious about the following two questions. Firstly, can diminishers be strengthened to exclude "less" strict kernelizations (we call those *semi-strict* kernelization)? Secondly, can diminishers be employed to prove kernelization lower bounds for polynomial-time solvable, parameterized problems? In Chapter 5 we target these questions. To exclude semi-strict kernels of polynomial size, we introduce a strengthened variant of diminishers: *strong* diminishers. While we prove two problems to be strongly diminishable, for several, even diminishable problems, we prove that strong diminishability breaks the Exponential Time Hypothesis (ETH). Surprisingly, in turn, adapting the strong diminisher framework, we exclude several fast (e.g., linear-time) and small (e.g., quadratic-size) *proper* kernelizations for two polynomial-time solvable, parameterized problems. Herein, the lower bounds rely on popular conjectures like the APSP-conjecture, the

3SUM-conjecture, or the Strong Exponential Time Hypothesis (SETH).[2] However, the framework makes it possible to exclude fast proper kernelizations of *some* polynomial size, but not of *every* polynomial size. We complement our kernelization lower bounds with some straight-forward kernelization upper bounds (on this way also formalizing the concept of Turing kernelization for problems in P).

As said, our framework presented in Chapter 5 is not applicable for excluding fast kernelizations of *every* polynomial size. In Chapter 6, we study the polynomial-time solvable HYPERBOLICITY problem for which we prove fast kernelizations of any polynomial size to presumably not exist. The hyperbolicity of a graph is, roughly speaking, a number that measures how metrically similar the graph is to some tree. HYPERBOLICITY is the problem of computing the hyperbolicity of a given graph. We prove that HYPERBOLICITY parameterized by the vertex cover number admits a kernelization dichotomy: While it admits a linear-time computable kernelization of exponential size, it admits no kernelization of subexponential size running in truly subquadratic time assuming the SETH to hold. The latter implies, in particular, that there is no linear-time computable kernelization of *any* polynomial size assuming the SETH to hold. We complement our kernelization dichotomy by providing data-reduction-based parameterized linear-time algorithms.

[2]We point out the roles of ETH and SETH here: While we prove several problems to be not strongly diminishable assuming the ETH to hold, using strong diminishers we exclude fast and small proper kernelization assuming the SETH to hold.

Chapter 4.

Diminishers and Diminishable Problems

In this chapter we present the framework of *parameter diminishers*, an extension of *parameter-decreasing polynomial self-reductions* introduced by Chen et al. [CFM11]. We prove the framework's applicability to be of wider range, demonstrating its relevance for excluding more restrictive kernels of polynomial size under the assumption of P $\neq$ NP, an assumption weaker than coNP $\nsubseteq$ NP$_{/\text{poly}}$.

4.1. Introduction

In this chapter, we consider a more restrictive variant of kernelization, which we call *strict kernelization*, where we demand the output parameter k' to increase not more than by an additive constant.

Definition 4.1 (Strict kernel)**.** A *strict kernelization* for a parameterized problem L is a polynomial-time algorithm that on input $(x, k) \in \Sigma^* \times \mathbb{N}$ outputs $(x', k') \in \Sigma^* \times \mathbb{N}$, the *strict kernel*, satisfying:
(i) $(x, k) \in L \iff (x', k') \in L$,
(ii) $|x'| \leq f(k)$ for some function f, and
(iii) $k' \leq k + c$ for some constant $c \geq 0$.
We say that L admits a strict *polynomial* kernelization if $f(k) \in k^{O(1)}$.

A strict kernelization can be also understood as a slight relaxation of proper kernelization (Definition 1.7), where "$k' \leq k$" (Definition 1.7(iii)) is replaced

This chapter is based on (parts of) *Diminishable Parameterized Problems and Strict Polynomial Kernelization* by Henning Fernau, Till Fluschnik, Danny Hermelin, Andreas Krebs, Hendrik Molter, and Rolf Niedermeier (Computability [Fer+20]).

by "$k' \leq k + c$". While the term "strict" in the definition above makes sense mathematically, it is actually quite harsh from a practical perspective. Indeed, data reduction rules involved in known kernelizations rarely ever increase the parameter value (see, e.g., the surveys [GN07, Kra14, LMS12]). Furthermore, strict kernelization is clearly preferable to kernelizations that increase the parameter value in a dramatic way: Often a fixed-parameter algorithm on the resulting kernel is applied, whose running time highly depends on the value of the parameter, and so a kernelization that substantially increases the parameter value might in fact be useless. Finally, the equivalence with FPT is preserved: A decidable parameterized problem is solvable in $f(k) \cdot |x|^{O(1)}$ time if and only if it has a strict kernel of size $g(k)$ (where f and g are some computable functions only depending on k) [CFM11, Proposition 3.2].

Chen et al. [CFM11] developed the first framework, being the central framework of this chapter, to exclude polynomial proper kernels (which they call parameter non-increasing kernelization) under the assumption of P $\neq$ NP. The main concept behind the framework is that of a *parameter-decreasing polynomial self-reduction* (which we will call *parameter diminisher*): a polynomial-time algorithm that decreases the parameter value of any given instance by at least one. The crucial connection is that, for an NP-hard parameterized problem, the existence of a parameter diminisher and a proper (indeed, strict) polynomial kernel implies P = NP. Chen et al. [CFM11] showed that ROOTED PATH parameterized by the length of the path and CNF-SAT parameterized by the number of variables admit no proper polynomial kernel unless P = NP. The goal of this chapter is to show that the framework by Chen et al. [CFM11] applies to more parameterized problems, even when replacing proper by strict kernelization, while excluding strict polynomial kernelization is comparatively simple for these problems.

We remark that in addition to the results of Chen et al. [CFM11], there is a kernelization lower bound result by Cygan et al. [CPP16] that relies on the assumption of P $\neq$ NP: Cygan et al. proved that, unless P = NP, EDGE CLIQUE COVER (see Appendix A), when parameterized by the number k of cliques, admits no kernel of subexponential size.

Our Contributions. We build on the work of Chen et al. [CFM11], and further develop and widen the framework they presented for excluding strict polynomial kernels (Section 4.2). Using this extended framework, we show that several natural fixed-parameter tractable problems admit parameter diminishers and (hence) have no strict polynomial kernels unless P = NP (Section 4.3). The main result of our work reads as follows.[1]

Theorem 4.1. *Unless P = NP, none of the following fixed-parameter tractable problems admits a strict polynomial kernel:*

(i) CLIQUE *when parameterized by* $p \in \{\Delta, \mathrm{tw}, \mathrm{bw}, \mathrm{vs}, \mathrm{cw}\}$*;*

(ii) BICLIQUE *when parameterized by* $p \in \{\Delta, \mathrm{tw}, \mathrm{bw}, \mathrm{vs}, \mathrm{cw}\}$*;*

(iii) TERMINAL STEINER TREE *when parameterized by* $k + |T|$*;*

(iv) MULTICOLORED PATH *when parameterized by* $k \log(n)$ *and* COLORFUL GRAPH MOTIF *when parameterized by* k*;*

(v) MCA-DEFENSIVE ALLIANCE *and* MCA-VERTEX COVER *each when parameterized by* k*;*

(Herein, k *denotes the solution size,* n*,* Δ*,* tw*,* bw*,* vs*, and* cw *denote the number of vertices, the maximum vertex degree, the treewidth, bandwidth, vertex separation number, and cutwidth of the graph, respectively, and* T *denotes the set of terminals.)*

We remark that all parameterized problems stated in Theorem 4.1 are either known to be fixed-parameter tractable or we prove them to be fixed-parameter tractable. Moreover, for all of the problems one can exclude polynomial kernels under the assumption that $\mathrm{coNP} \not\subseteq \mathrm{NP}_{/\mathrm{poly}}$ using the (cross-)composition framework. Our results base on a weaker assumption, but exclude a more restricted version of polynomial kernels. On the contrary, the composition framework excludes a more general version of polynomial kernels but requires a stronger assumption. Hence, our results are incomparable with the existing no-polynomial-kernel results. However, the diminisher framework provides a simpler methodology and directly connects the exclusion of strict polynomial kernels to the assumption that P ≠ NP.

4.2. Diminisher Framework

In this section we present the general framework used in this chapter. Firstly, we define the central notion of a *parameter diminisher* extending the *parameter-decreasing polynomial self-reduction* introduced by Chen et al. [CFM11].

[1] Formal definitions of each of these parameterized problems and the used parameters are given in the following sections; see also Appendix A.

Definition 4.2 (Parameter diminisher)**.** A *parameter diminisher* for a parameterized problem L is a polynomial-time algorithm that maps any instance $(x, k) \in \Sigma^* \times \mathbb{N}$ of L to an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ of L such that

(i) $(x, k) \in L \iff (x', k') \in L$ and

(ii) $k' < k$.

A parameterized problem L is *diminishable* if there is a parameter diminisher for L.

Thus, a parameter diminisher (or diminisher for short) is an algorithm that is able to decrease the parameter of any given instance of a parameterized problem L in polynomial time. The algorithm is given freedom in that it can produce a completely different instance, as long as it is an equivalent one (with respect to L) and has a smaller parameter value. Note that, by definition, the parameter in a parameterized problem is a natural number, and thus the difference between the obtained parameter and the parameter in the input instance is at least one. The following theorem was proven first by Chen et al. [CFM11] (yet regarding polynomial proper kernels).

Theorem 4.2 ([CFM11])**.** *Let L be a parameterized problem such that its unparameterized version is NP-hard and $\{(x, k) \in L \mid k \leq c\} \in P$, for some constant c. If L is diminishable and admits a strict polynomial kernel, then $P = NP$.*

The idea behind Theorem 4.2 is to repeat the following two procedures until the parameter value drops below c (see Figure 4.1 for an illustration). First, apply the parameter diminisher a constant number of times so that when, second, the strict polynomial kernelization is applied, the parameter value is decreased. The strict polynomial kernelization keeps the instances small, hence the whole process runs in polynomial time.

The following type of reductions, which increases the parameter value not more than by a constant and runs in polynomial time, allows for transferring diminishability from one parameterized problem to another.

Definition 4.3 (Parameter-constant-increasing reduction)**.** Given two parameterized problems L with parameter k and L' with parameter k', a *parameter-constant-increasing reduction* from L to L' is a polynomial-time algorithm that maps each instance (x, k) of L to an instance (x', k') of L' such that

(i) $(x, k) \in L \iff (x', k') \in L'$, and

(ii) $k' \leq k + c$ for some constant $c \geq 0$.

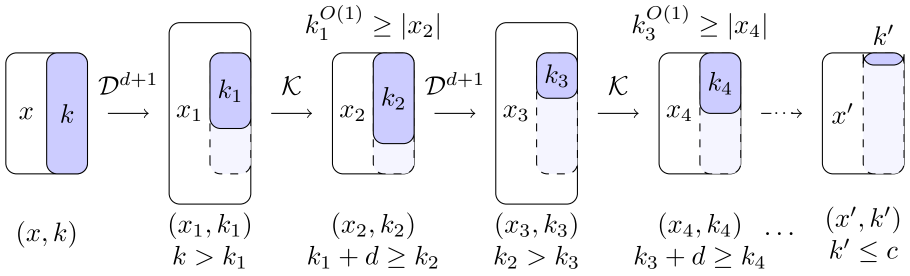


Figure 4.1.: Illustration for the proof of Theorem 4.2 with an input instance (x, k). Herein, $\mathcal{K}$ denotes the strict kernelization with additive constant d and $\mathcal{D}$ denotes the parameter diminisher, respectively. We represent each instance by boxes: the size of a box symbolizes the size of the instance or the value of the parameter (each dashed box refers to k).

Note that to transfer diminishability, we need parameter-constant-increasing reductions between two parameterized problems in *both* directions—a crucial difference to other reduction-based hardness results.

Lemma 4.3. *Let L_1 and L_2 be two parameterized problems such that there are parameter-constant-increasing reductions from L_1 to L_2 and from L_2 to L_1. Then L_1 is diminishable if and only if L_2 is diminishable.*

Proof. Let L_1 with parameter k_1 and L_2 with parameter k_2 be two parameterized problems. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be parameter-constant-increasing reductions from L_1 to L_2 with constant c_1 and from L_2 to L_1 with constant c_2, respectively. Let $\mathcal{D}_2$ be a parameter diminisher for L_2. Let (x_1, k_1) be an arbitrary instance of L_1.

Apply $\mathcal{A}_1$ to (x_1, k_1) to obtain the instance (x_2, k_2) of L_2 with $k_2 \leq k_1 + c_1$. Next, apply $\mathcal{D}_2$ to (x_2, k_2) $(c_1 + c_2 + 1)$-times to obtain the instance (x_2', k_2') of L_2 with $k_2' < k_2 - c_1 - c_2 \leq k_1 - c_2$. Finally, apply $\mathcal{A}_2$ to (x_2', k_2') to obtain the instance (x_1', k_1') of L_1 with $k_1' \leq k_2' + c_2 < k_1$. As $k_1' < k_1$, the above combination of $\mathcal{A}_1$, $\mathcal{D}_2$, and $\mathcal{A}_2$ forms a parameter diminisher for L_1. To get the reverse direction, exchange the roles of L_1 and L_2. □

Parameter-Decreasing Branching and Strict Composition. To construct parameter diminishers, it is useful to follow a "branch and compose" technique: Herein, first *branch* into several subinstances of the input instance while decreasing the parameter value in each, and then *compose* the subinstances into one instance without increasing the parameter value by more than an additive constant. We first give the definitions of parameter-decreasing branching rule and strict composition, and then show that both combined form a parameter diminisher.

Branching rules are highly common in parameterized algorithm design, and they are typically deployed when using depth-bounded search-trees or related techniques. Roughly speaking, in a *parameter-decreasing branching rule* one reduces the problem instance to several problem instances each with smaller parameter values such that at least one of these new instances is a `yes`-instance if and only if the original instance is a `yes`-instance.

Definition 4.4 (Parameter-decreasing branching rule)**.** A *parameter-decreasing branching rule* for a parameterized problem L is a polynomial-time algorithm that on input $(x, k) \in \Sigma^* \times \mathbb{N}$ outputs a sequence of instances $(y_1, k'), \ldots, (y_t, k') \in \Sigma^* \times \mathbb{N}$ such that

(i) $(x, k) \in L \iff (y_i, k') \in L$ for some $i \in \{1, \ldots, t\}$ and
(ii) $k' < k$.

Recall that composition is the core concept behind the standard kernelization lower bound framework. Here we use a more restrictive notion of this concept, where the parameter in the output instance is not allowed to increase by more than an additive constant to the input parameter:

Definition 4.5 (Strict composition)**.** A *strict composition* for a parameterized problem L is an algorithm that receives as input t instances $(x_1, k), \ldots, (x_t, k) \in \Sigma^* \times \mathbb{N}$, and outputs in polynomial time a single instance $(y, k') \in \Sigma^* \times \mathbb{N}$ such that

(i) $(y, k') \in L \iff (x_i, k) \in L$ for some $i \in \{1, \ldots, t\}$ and
(ii) $k' \leq k + c$ for some constant $c \geq 0$.

If we now combine a parameter-decreasing branching rule with a strict composition, then we get a parameter diminisher.

Lemma 4.4. *Let L be a parameterized problem. If L admits a parameter-decreasing branching rule and a strict composition, then it is diminishable.*

Proof. Let (x, k) be an instance of a parameterized problem L of size n, c be the constant associated with a strict composition for L, and $t \in O(n^d)$, $d \in \mathbb{N}$, be the number of instances computed by the parameter-decreasing branching rule. We recursively apply $c + 1$ times the parameter-decreasing branching rule for L to produce t^a instances (x_i, k^*), where a is a constant depending on c and d only. Note that in each application of the parameter-decreasing branching rule the parameter is decreased by at least one, and hence $k^* < k - c$. The strict composition receives t^a instances and produces in polynomial time an instance (y, k') with $k' \leq k^* + c < k$ of L which is a `yes`-instance if and only if (x, k) is a `yes`-instance. Hence, the whole procedure is a parameter diminisher for L. □

Remark 4.1. Lemma 4.4 also holds true if we require in Definitions 4.4 and 4.5 that the equivalence holds for *all* $i \in \{1, \ldots, t\}$. That is, for parameter-decreasing branching rule we replace (i) by "$(x, k) \in L \iff (y_i, k') \in L$ for all $i \in \{1, \ldots, t\}$", and for strict composition we replace (i) by "$(y, k') \in L \iff (x_i, k) \in L$ for all $i \in \{1, \ldots, t\}$".

As an example application of Lemma 4.4 above, we consider the parameter diminisher for the Rooted Path problem parameterized by the length k of the path due to Chen et al. [CFM11].

Rooted Path
Input: An undirected graph $G = (V, E)$, a vertex $r \in V$, and an integer $k \in \mathbb{N}$.
Question: Is there a path P with endpoint r of length at least k in G?

Let $v_1, \ldots, v_t$ be the neighbors of r in G. The parameter-decreasing branching rule for Rooted Path constructs from (G, r, k) the set of instances $(G - r, v_1, k - 1), \ldots, (G - r, v_t, k - 1)$. A strict composition for Rooted Path takes as input the instances $(G_1, r_1, k), \ldots, (G_t, r_t, k)$ and constructs the instance $(G', r, k + 1)$, where G' is the graph obtained by taking the disjoint union of all G_is and making all their roots adjacent to a new root vertex r. Combining these two algorithms—two applications of the parameter-decreasing branching rule and the strict composition—gives the parameter diminisher for Rooted Path.

4.3. Problems without Strict Polynomial Kernels

In this section we prove Theorem 4.1 based on several propositions to follow. We present parameter diminishers for all problems mentioned in Theorem 4.1, following the order in which the problems are stated in Theorem 4.1. Thus, we study Clique and Biclique in Section 4.3.1, Terminal Steiner Tree in Section 4.3.2, Multicolored Path and Colorful Graph Motif in Section 4.3.3, and Multi-Component Annotated Π in Section 4.3.4.

4.3.1. Clique and Biclique

We begin with the Clique problem.

Clique
Input: An undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Is there a vertex set $X \subseteq V$ of G such that $|X| \geq k$ and for all distinct $v, w \in X$ there is $\{v, w\} \in E$?

Since Clique parameterized by the solution size k is W[1]-complete [DF99], we focus on other parameterizations of Clique that yield fixed-parameter tractability, for instance the maximum degree Δ of the input graph, where Clique has a simple fixed-parameter algorithm (exhaustively search in the closed neighborhood of each vertex individually). Other parameterizations include treewidth $\mathrm{tw} = \mathrm{tw}(G)$, bandwidth $\mathrm{bw} = \mathrm{bw}(G)$, vertex separation $\mathrm{vs} = \mathrm{vs}(G)$, and the cutwidth $\mathrm{cw} = \mathrm{cw}(G)$ of the input graph (refer to Figure 1.4 in Section 1.2 for the parameters' relation). Our first main result of this section is the following.

Proposition 4.5. Clique *when parameterized by* $p \in \{\Delta, \mathrm{tw}, \mathrm{bw}, \mathrm{vs}, \mathrm{cw}\}$ *is diminishable.*

Observe that if $\mathrm{cw} \leq c$ for some constant c, then $\Delta \leq 2c$, and Clique is fixed-parameter tractable when parameterized by the maximum degree. We prove Proposition 4.5 via the "branch and compose" technique (see Section 4.2). For the parameter-branching rule, the following is the underlying construction.

Construction 4.6. Let $G = (V, E)$ be a graph. For each $v \in V$, construct the subgraph $G_v := G[N_G(v)]$ of G induced by the open neighborhood of v in G. Let $\mathcal{G}(G) := \{G_v \mid v \in V\}$ denote the set of all such graphs. ▲

In what follows, for each parameter stated in Proposition 4.5 we give its definition and prove that its value decreases by at least one in each graph in the set output by Construction 4.6. Note that for every graph the cutwidth cw upper-bounds each of the other parameters listed in Proposition 4.5. However, recall that diminishability of a parameter p does not necessarily transfer to parameters which are upper bounded by p. We begin with the maximum vertex degree Δ of the input graph.

Observation 4.7. *Let G be a graph and $\mathcal{G}(G)$ denote the set obtained from applying Construction 4.6 to G. Then $\max_{H\in\mathcal{G}(G)} \Delta(H) < \Delta(G)$.*

Proof. Observe that for all $v \in V(G)$, $G_v = G[N_G[v]] - v$, and hence we have that $\Delta(G_v) = \Delta(G[N_G[v]]) - 1 < \Delta(G)$. □

Next, we consider the parameter treewidth tw (see Definition 1.13).

Lemma 4.8. *Let G be a graph and $\mathcal{G}(G)$ denote the set of graphs obtained from applying Construction 4.6 to G. Then $\max_{H\in\mathcal{G}(G)} \operatorname{tw}(H) < \operatorname{tw}(G)$.*

Proof. For all $v \in V$, denote by G'_v the graph $G[N_G[v]]$. Note that for all $v \in V$, as $G'_v \subseteq G$, we have that $\operatorname{tw}(G'_v) \leq \operatorname{tw}(G)$. Let $\mathbb{T} = (T, (B_\alpha)_{\alpha\in V(T)})$ denote a tree decomposition of G of width $\operatorname{tw}(G)$. Note that $\mathbb{T}'_v = (T, (B'_\alpha)_{\alpha\in V(T)})$, where $B'_\alpha := B_\alpha \cap N_G[v]$ for all $\alpha \in V(T)$, is a tree decomposition of G'_v of width at most $\operatorname{tw}(G)$ [Die10]. We claim that the tree decomposition $\mathbb{T}_v = (T, (B^*_\alpha)_{\alpha\in V(T)})$, where $B^*_\alpha := B_\alpha \cap N_G(v)$ for all $\alpha \in V(T)$, of G_v has width at most $\omega(\mathbb{T}'_v) - 1$. Suppose not, that is, there is an $\alpha \in V(T)$ such that $|B^*_\alpha| = \omega(\mathbb{T}'_v) + 1$ (note that $B'_\alpha = B^*_\alpha$, and hence $v \notin B'_\alpha$). Let $\beta \in V(T)$ denote the node in T closest to α whose bag contains v in $\mathbb{T}'_v$, that is, $v \in B'_\beta$ and $\operatorname{dist}_T(\alpha, \beta) = \min_{\beta'\in V(T):v\in B'_{\beta'}} \operatorname{dist}_T(\alpha, \beta')$. Since v is adjacent to each vertex in $N_G(v)$, it holds true that for each $w \in B'_\alpha$, there is a $\gamma \in V(T)$ such that $\{v, w\} \subseteq B'_\gamma$. By property (iii) of tree decompositions and the choice of node β, it follows that $B'_\alpha \subseteq B'_\beta$. Since $v \in B'_\beta$, we have $|B'_\alpha| < |B'_\beta|$, contradicting the choice of α. Altogether, we have $\operatorname{tw}(G_v) \leq \omega(\mathbb{T}_v) < \omega(\mathbb{T}'_v) \leq \operatorname{tw}(G)$. □

For a vertex set V, a *vertex-ordering* is a bijection $\sigma\colon V \to \{1, \ldots, |V|\}$. The *bandwidth* of a graph $G = (V, E)$ is defined as the minimum value of $\max_{\{u,w\}\in E} |\sigma(u) - \sigma(w)|$ over all vertex-orderings $\sigma\colon V \to \{1, \ldots, |V|\}$.

Lemma 4.9. *Let G be a graph and $\mathcal{G}(G)$ denote the set of graphs obtained from applying Construction 4.6 to G. Then $\max_{H\in\mathcal{G}(G)} \operatorname{bw}(H) < \operatorname{bw}(G)$.*

Proof. For all $v \in V$, denote by G'_v the graph $G[N_G[v]]$. Note that for all $v \in V$, as $G'_v \subseteq G$, we have that $\operatorname{bw}(G'_v) \leq \operatorname{bw}(G)$ [DPS02]. Let $\sigma\colon V(G'_v) \to \{1, \ldots, |V(G'_v)|\}$ be a vertex-ordering for G'_v with $\max_{\{u,w\}\in E(G'_v)} |\sigma(u) - \sigma(w)|$ being equal to the bandwidth of G'_v, i.e., $\operatorname{bw}(G'_v) = \max_{\{u,w\}\in E(G'_v)} |\sigma(u) - \sigma(w)|$. Let σ' denote the vertex-ordering obtained from σ such that $\sigma'(w) := \sigma(w)$ if $\sigma(w) < \sigma(v)$, and $\sigma'(w) := \sigma(w) - 1$ if $\sigma(w) > \sigma(v)$ for all $w \in N_G(v)$. It follows that for every $\{x,y\} \in E(G'_v)$ with $\sigma(x) < \sigma(v) < \sigma(y)$, it holds true that $|\sigma'(x) - \sigma'(y)| = |\sigma(x) - \sigma(y)| - 1$. Observe that since v is adjacent to all vertices in $N_G(v)$, for every $\{x,y\} \in E(G'_v)$ with $v \notin \{x,y\}$ and $\sigma(y) - \sigma(x) = \max_{\{u,w\}\in E(G'_v)} |\sigma(u) - \sigma(w)|$ it holds true that $\sigma(x) < \sigma(v) < \sigma(y)$. Hence $\mathrm{bw}(G_v) = \mathrm{bw}(G'_v) - 1 < \mathrm{bw}(G)$. $\square$

The *vertex separation* of a graph $G = (V,E)$ is defined as the minimum value of $\max_{1\leq i\leq |V|} |F_{\sigma,i}|$ over all vertex-orderings $\sigma\colon V \to \{1,\dots,|V|\}$, where $F_{\sigma,i} := \{u \in V \mid \sigma(u) \leq i \wedge \exists \{u,w\} \in E : \sigma(w) > i\}$. Note that for every graph the vertex separation number and the pathwidth are equal [Kin92].

Lemma 4.10. *Let G be a graph and $\mathcal{G}(G)$ denote the set of graphs obtained from applying Construction 4.6 to G. Then $\max_{H\in\mathcal{G}(G)} \mathrm{vs}(H) < \mathrm{vs}(G)$.*

Proof. For all $v \in V$, denote by G'_v the graph $G[N_G[v]]$. Note that for all $v \in V$, as $G'_v \subseteq G$, we have that $\mathrm{vs}(G'_v) \leq \mathrm{vs}(G)$ [DPS02]. Let $\sigma\colon V(G'_v) \to \{1,\dots,n'_v\}$, where $n'_v := |V(G'_v)|$, be a vertex-ordering for G'_v with $\mathrm{vs}(G'_v) = \max_{i\in\{1,\dots,n'_v\}} |F_{\sigma,i}|$ such that $\sigma(v)$ is smallest among all such orderings. Define the set $I := \arg\max_{i\in\{1,\dots,n'_v\}} |F_{\sigma,i}|$. We claim that $\sigma(v) \leq i$ for all $i \in I$.

Suppose not, that is, there is an $i \in I$ such that $\sigma(v) > i$. Let $j \in I$ denote the largest index in I such that $\sigma(v) > j$. Observe that $j = \sigma(v) - 1$, as otherwise $|F_{\sigma,j}| < |F_{\sigma,\sigma(v)-1}|$ since all vertices in $V(G'_v) \setminus \{v\}$ are adjacent to v. Let $w \in V(G'_v)$ denote the vertex with $\sigma(w) = j$. Consider the vertex-ordering σ_{vw} with $\sigma_{vw}(v) := w$, $\sigma_{vw}(w) := v$, and $\sigma_{vw}(u) := u$ for all $u \in V(G'_v) \setminus \{v,w\}$. Observe that $|F_{\sigma',j}| \leq |F_{\sigma,j}|$ since every vertex u with $\sigma(u) \leq j$ is adjacent with v. Moreover, for all $i \in \{1,\dots,n'_v\} \setminus \{j\}$ it holds true that $|F_{\sigma_{vw},i}| = |F_{\sigma,i}|$ since $\{u \in V(G'_v) \mid \sigma(u) \leq i\} = \{u \in V(G'_v) \mid \sigma_{vw}(u) \leq i\}$. It follows that σ_{vw} is a vertex-ordering for G'_v with $\mathrm{vs}(G'_v) = \max_{i\in\{1,\dots,n'_v\}} |F_{\sigma_{vw},i}|$ such that $\sigma_{vw}(v) < \sigma(v)$, contradicting the choice of σ. Hence, we have that $\sigma(v) \leq i$ for all $i \in I$.

Let σ' denote the vertex-ordering obtained from σ such that $\sigma'(w) := \sigma(w)$ if $\sigma(w) < \sigma(v)$, and $\sigma'(w) := \sigma(w) - 1$ if $\sigma(w) > \sigma(v)$ for all $w \in N_G(v)$. Since $G_v \subseteq G'_v$, we have that $|F_{\sigma',\sigma'(w)}| \leq |F_{\sigma,\sigma(w)}|$ for all $w \in V(G_v)$ [DPS02]. Since for all $i \in I$ it holds true that $\sigma(v) \leq i$, we have that $|F_{\sigma',\sigma'(w)}| \leq |F_{\sigma,\sigma(w)}| - 1$ for all $w \in V(G_v)$ such that there is an $i \in I$ with $\sigma(w) = i$. Altogether, it follows that $\mathrm{vs}(G_v) \leq \max_{i\in\{1,\dots,n'_v\}} |F_{\sigma',i}| < \mathrm{vs}(G'_v) \leq \mathrm{vs}(G)$. $\square$

The *cutwidth* of a graph $G = (V,E)$ is defined as the minimum value of $\max_{i\in\{1,\dots,|V|\}} |E_{\sigma,i}|$ over all vertex-orderings $\sigma\colon V \to \{1,\dots,|V|\}$, where $E_{\sigma,i} := \{\{u,w\} \in E \mid \sigma(u) \leq i < \sigma(w)\}$.

Lemma 4.11. *Let G be a graph and $\mathcal{G}(G)$ denote the set of graphs obtained from applying Construction 4.6 to G. Then* $\max_{H \in \mathcal{G}(G)} \operatorname{cw}(H) < \operatorname{cw}(G)$.

Proof. For all $v \in V$, denote by G'_v the graph $G[N_G[v]]$. Note that for all $v \in V$, as $G'_v \subseteq G$, we have that $\operatorname{cw}(G'_v) \leq \operatorname{cw}(G)$ [DPS02]. Let $\sigma\colon V(G'_v) \to \{1, \ldots, n'_v\}$ be a vertex-ordering for G'_v with $\operatorname{cw}(G'_v) = \max_{1 \leq i \leq n'_v} |E_{\sigma,i}|$, where $n'_v := |V(G'_v)|$. Let σ' denote the vertex-ordering obtained from σ such that $\sigma'(w) := \sigma(w)$ if $\sigma(w) < \sigma(v)$, and $\sigma'(w) := \sigma(w) - 1$ if $\sigma(w) > \sigma(v)$ for all $w \in N_G(v)$. Note that σ' is an ordering on the vertices of G_v. Since v is adjacent to each vertex in $V(G'_v) \setminus \{v\}$, it holds that $|E_{\sigma',j'}| \leq |E_{\sigma,j}| - 1$ for each $j \in \{1, \ldots, n'_v\}$ and $j' = j$, if $j < \sigma(v)$, and $j' = j - 1$, otherwise. It follows that $\operatorname{cw}(G_v) \leq \max_{1 \leq i \leq |V(G_v)|} |E_{\sigma',i}| < \max_{1 \leq i \leq n'_v} |E_{\sigma,i}| = \operatorname{cw}(G'_v) \leq \operatorname{cw}(G)$. □

Construction 4.6 yields a parameter diminisher for parameters tw, bw, vs, cw and k. Leaving the parameter k aside (for which CLIQUE is W[1]-complete), we prove our first main result of this section.

Proof of Proposition 4.5. Let $(G = (V, E), k)$ be an instance of CLIQUE. The following is a parameter-decreasing branching rule for $(G = (V, E), k)$: Apply Construction 4.6 to G, and construct for each $G_v \in \mathcal{G}(G)$ the instance $(G_v, k - 1)$.

We prove that G has a clique of size k if and only if G_v has a clique of size $k - 1$ for some $v \in V$. Let C be a clique in G with k vertices and $v \in V(C)$. Let $C' := C - v$ denote the clique of size $k - 1$ obtained from C by deleting v. Since for every $w \in V(C')$ it holds that $w \in N_G(v)$, we have $C' \subseteq G_v$, and the claim follows. Conversely, let $v \in V$ such that G_v contains a clique C' of size $k - 1$. By construction, $v \notin V(C')$ as $C' \subseteq G_v$, and v is adjacent to all vertices in G_v. It follows that $V(C') \cup \{v\}$ forms a clique of size k in G.

Finally, due to Observation 4.7 and Lemmas 4.8 to 4.11, we know that in each instance the corresponding parameter value decreased.

For the composition step take the disjoint union of all graphs. Formally, on input instances $(G_1, k), \ldots, (G_t, k)$, $t \in \mathbb{N}$, compute the instance $(G_1 \uplus \cdots \uplus G_t, k)$. Note that a graph has a clique of size k if and only if one of its connected components has a clique of size k. Moreover, for all stated parameters it holds true that their value equals the maximum value of the connected components of the input graph [Chv+75, DPS02]. Applying Lemma 4.4 completes the proof. □

Next we show that the parameter diminisher presented for Clique can be adapted to the Biclique problem:

Biclique
Input: An undirected bipartite graph $G = (V = A \uplus B, E)$ and an integer k.
Question: Is there a vertex set $X \subseteq V$ of G such that $|X \cap A| = |X \cap B| = k$ and each vertex in $X \cap A$ is adjacent to each vertex in $X \cap B$?

Thus, we have our second main result of this section:

Corollary 4.12. Biclique *when parameterized by* $p \in \{\Delta, \mathrm{tw}, \mathrm{bw}, \mathrm{vs}, \mathrm{cw}\}$ *is diminishable.*

Proof sketch. We only present the parameter-decreasing branching rule. The rest of the proof is analogous to the proof of Proposition 4.5. Let $(G = (A \uplus B, E), k)$ be an instance of Biclique. The following is a parameter-decreasing branching rule for $(G = (A \uplus B, E), k)$: For each $\{v, w\} \in E$, construct the instance $(G_{v,w}, k - 1)$ where $G_{v,w} := G[(N_G(v) \setminus \{w\}) \cup (N_G(w) \setminus (v))]$. □

Corollary 4.12 and Proposition 4.5 together with Theorem 4.2 now prove Theorem 4.1(i) and (ii).

4.3.2. Terminal Steiner Tree

The well-known Steiner Tree problem is defined as follows: given an undirected graph $G = (V, E)$ with $V = N \uplus T$ (T is called the terminal set) and an integer $k \in \mathbb{N}$, decide whether there is a subgraph $H \subseteq G$ with at most $k + |T|$ vertices such that H is a tree containing all vertices in T. In this section, we consider the variant Terminal Steiner Tree [BMS15, LX02] of Steiner Tree, which additionally demands the terminal set T to be a subset of the set of leaves of the tree H.

Terminal Steiner Tree (TST)
Input: An undirected graph $G = (V = N \uplus T, E)$ and an integer k.
Question: Is there a subgraph H of G such that H is a tree with T being its set of leaves?

TST is proven to be NP-complete. We are not aware of any parameterized complexity study for TST. Hence, for the sake of completeness, in the following two lemmas we show that TST, when parameterized by the size $k + |T|$ of the terminal Steiner tree in question, is fixed-parameter tractable and admits no polynomial kernel unless $\mathrm{coNP} \subseteq \mathrm{NP}_{/\mathrm{poly}}$.

Lemma 4.13. Terminal Steiner Tree *parameterized by $k + |T|$ is fixed-parameter tractable.*

Proof. We give a parameterized reduction from TST to Steiner Tree, each parameterized by the order of the tree. As Steiner Tree parameterized by $k + |T|$ is fixed-parameter tractable [DW71], the claim follows (recall Lemma 1.5).

Let $(G = (N \uplus T, E), k)$ be an instance of TST. We construct an equivalent instance $(G' = (N' \uplus T, E'), k')$ of Steiner Tree as follows. Let G' be initially a copy of G. For each $t \in T$, apply the following. For each edge $\{v, t\} \in E$, remove $\{v, t\}$ from G' and add a path of length $2(k+|T|)$ to G' with endpoints v and t. Set $k' = |T| \cdot (2(k + |T|) - 1) + k$. This finishes the reduction. Clearly, the construction can be done in polynomial time.

We show that $(G = (N \uplus T, E), k)$ is a yes-instance of TST if and only if $(G' = (N' \uplus T, E'), k')$ is a yes-instance of Steiner Tree.

($\Rightarrow$) Let H be a terminal Steiner tree in G with at most $k + |T|$ vertices. We construct a Steiner tree H' in G' from H with at most $k' + |T|$ vertices as follows. Recall that each $t \in T$ has exactly one neighbor v_t in H. Hence, obtain H' by replacing for each $t \in T$ the edge $\{v_t, t\} \in E(H)$ by the path of length $2(k + |T|)$ connecting v_t with t. It is not difficult to see that H' is a Steiner tree in G'. Moreover,

$$\begin{aligned} |V(H')| &= |V(H)| + |T|(2(k + |T|) - 1) \\ &\leq k + |T| + |T|(2(k + |T|) - 1) = k' + |T|. \end{aligned}$$

($\Leftarrow$) Let H' be a minimum Steiner tree in G' with at most $k'+|T|$ vertices. We state some first observations on H'. Observe that no inner vertex of the paths added in the construction step from G to G' is a leaf of H' (as otherwise H' is not minimum). Moreover, as H' contains each $t \in T$, H' contains a path of length $2(k + |T|)$ for each $t \in T$. Suppose that there is a terminal $t \in T$ such that t is not a leaf in H'. Then the number of vertices of H' is

$$\begin{aligned} |V(H')| &\geq |T| + (|T| + 1) \cdot (2(k + |T|) - 1) \\ &> |T| \cdot (2(k + |T|) - 1) + 2(k + |T|) - 1 \\ &> k' + (k + |T|) - 1 \\ &\geq k' + |T|, \end{aligned}$$

yielding a contradiction. Hence, each terminal $t \in T$ forms a leaf in H'. We show how to obtain a terminal Steiner tree H in G from H' with at most $k+|T|$

vertices. As each terminal $t \in T$ forms a leaf in H', there is exactly one neighbor $v_t \in N_G(t)$ such that H' contains the path of length $2(k + |T|)$ connecting v_t with t. Replace for each $t \in T$ the path of length $2(k + |T|)$ connecting v_t with t in H' by the edge $\{v_t, t\} \in E$ to obtain H from H'. Note that H is a terminal Steiner tree. Moreover,

$$\begin{aligned}|V(H)| &= |V(H')| - |T| \cdot (2(k + |T|) - 1)\\ &\le |T| + |T| \cdot (2(k + |T|) - 1) + k - |T| \cdot (2(k + |T|) - 1)\\ &= k + |T|.\end{aligned}$$ □

Note that Steiner Tree parameterized by $k + |T|$ is WK[1]-complete [Her+15] and hence admits no polynomial kernel unless coNP $\subseteq$ NP$_{/\text{poly}}$.

Lemma 4.14. *Unless coNP $\subseteq$ NP$_{/poly}$,* Terminal Steiner Tree *admits no polynomial kernel.*

Proof. We give a polynomial parameter transformation from Steiner Tree to TST. Let $(G = (N \uplus T, E), k)$ be an instance of Steiner Tree. We construct instance $(G' = (N' \uplus T', E'), k')$ of TST as follows. Let G' be initially a copy of G. Next, for each terminal vertex $t \in T$, add a vertex v_t to G' and make it adjacent only with t. Denote by $T' := \{v_t \mid t \in T\}$ the set of vertices added in the previous step. Set $N' := N \cup T$ and $k' := k + |T|$. This finishes the construction of $(G' = (N' \uplus T', E'), k')$. We claim that $(G = (N \uplus T, E), k)$ admits a Steiner tree of size $k + |T|$ if and only if $(G' = (N' \uplus T', E'), k')$ admits a terminal Steiner tree of size $k' + |T'|$.

($\Rightarrow$) Let S be a Steiner tree in G of size $k + |T|$. It is not difficult to see that

$$S' := (V(S) \cup T', E(S) \cup \{\{v_t, t\} \in E(G') \mid t \in T\})$$

is a terminal Steiner tree in G' of size $k + |T| + |T| = k' + |T'|$.

($\Leftarrow$) Let S' denote a terminal Steiner tree in G' of size $k' + |T'|$. Note that $T' \subseteq V(S)$ and as each vertex $v_t \in T'$ has exactly one neighbor $t \in T$, it follows that

$$S := (V(S) \setminus T', E(S') \setminus \{\{v_t, t\} \in E(G') \mid t \in T\})$$

is a Steiner tree in G of size $k' + |T'| - |T'| = k' = k + |T|$. □

Notably, the proofs of Lemmas 4.13 and 4.14 imply that Terminal Steiner Tree is complete for WK[1] when parameterized by $k + |T|$.

Finally, we prove our diminishability result for Terminal Steiner Tree.

Proposition 4.15. TERMINAL STEINER TREE *parameterized by* $k + |T|$ *is diminishable.*

In our proof of Proposition 4.15, we use the following parameter-decreasing branching rule.

Construction 4.16. Let $(G = (N \uplus T, E), k)$ be an instance of TST with G being connected, $|T| \geq 3$, $N_G(t) \nsubseteq T$ for $t \in T$, and there is no vertex $v \in N$ such that $T \subseteq N_G(v)$. Select a terminal $t^* \in T$, and let $v_1, \ldots, v_d$ denote the neighbors of t^* in $G-(T\setminus\{t^*\})$. Construct d instances $(G_1, k-1), \ldots, (G_d, k-1)$ as follows. Define G_i, $i \in \{1, \ldots, d\}$, by $G_i := G - v_i$. Turn the vertices in $N_G(v_i)$ in G_i into a clique, that is, for each distinct vertices $v, w \in N_G(v_i)$ add the edge $\{v, w\}$ if not yet present. This finishes the construction of G_i. It is not hard to see that the construction can be done in polynomial time. ▲

Lemma 4.17. *Construction 4.16 is a parameter-decreasing branching rule for* TST *parameterized by* $k + |T|$*.*

Proof. Let $(G = (N \uplus T, E), k)$ and $(G_1, k-1), \ldots, (G_d, k-1)$ be as in Construction 4.16. We show that G has a terminal Steiner tree of size $k + |T|$ if and only if there is an $i \in \{1, \ldots, d\}$ such that G_i admits a terminal Steiner tree of size $k - 1 + |T|$.

($\Rightarrow$) Suppose that G has a terminal Steiner tree H of size $k+|T|$. As t^* is a leaf in H, there is exactly one neighbor v_i, $i \in \{1, \ldots, d\}$, being the neighbor of t^* in H. Let w be a neighbor of v_i in $H-T$ and let $A := N_H(v_i)$ (note that $t^* \in A$). Let H_i be the tree obtained from H by deleting v_i and connecting w with all vertices in A. Then H_i forms a terminal Steiner tree in G_i. Moreover, H_i is of size $k - 1 + |T|$.

($\Leftarrow$) Let G_i admit a terminal Steiner tree H_i of size $k - 1 + |T|$. As t^* is a leaf in H_i, there is exactly one vertex w being the neighbor of t^* in H_i. We obtain a terminal Steiner tree H in G from H_i as follows. If every edge in H_i is also present in G, then $H := H_i$ also forms a terminal Steiner tree in G. Otherwise, there is an inclusion-wise maximal edge set $E' \subseteq E(H_i)$ such that $E' \cap E(G) = \emptyset$. Observe that by construction, the set of endpoints of E' forms a subset of $N_G(v_i)$. Let initially H be a copy of H_i. Delete from H all edges in E', add vertex v_i to H, and for each $\{x, y\} \in E'$, add the edges $\{x, v_i\}$ and $\{y, v_i\}$. Note that H remains connected after this step, and the set of leaves remains unchanged. Finally, compute a minimum feedback edge set in H if necessary. Observe that since $V(H) = V(H_i) \cup \{v_i\}$, H forms a terminal Steiner tree of size $k + |T|$ in G. □

We are set to prove Proposition 4.15.

Proof of Proposition 4.15. We give a parameter-decreasing branching rule and a strict composition for Terminal Steiner Tree. Together with Lemma 4.4, the claim then follows. Let $(G = (N \uplus T, E), k)$ be an instance of TST (we can assume that G has a connected component containing T). We make several assumptions first. We can assume that $|T| \geq 3$ (otherwise a shortest path is the optimal solution). Additionally, we assume that for all terminals $t \in T$ it holds that $N_G(t) \nsubseteq T$ (as otherwise the instance is a no-instance and we output a trivial no-instance). Moreover, we can assume that there is no vertex $v \in N$ such that $T \subseteq N_G(v)$, as otherwise we immediately output a trivial yes-instance if $k \geq 1$ or a trivial no-instance otherwise.

For the parameter decreasing branching rule, use Construction 4.16. The strict composition for TST is as follows. Given the instances $(G_1, k), \dots, (G_d, k)$, compute an instance (G', k) as follows. Let G' be initially the disjoint union of $G_1, \dots, G_d$. For each $t \in T$, identify its copies in $G_1, \dots, G_d$, say $t_1, \dots, t_d$, with one vertex t' corresponding to t. This finishes the construction of G'. Note that for every $i, j \in \{1, \dots, d\}$, $i \neq j$, any path between a vertex in G_i and a vertex in G_j contains a terminal vertex. Hence, any terminal Steiner tree in G' contains non-terminal vertices only in G_i for exactly one $i \in \{1, \dots, d\}$. Thus, (G', k) is a yes-instance if and only if one of the instances $(G_1, k), \dots, (G_d, k)$ is a yes-instance. □

4.3.3. Multicolored Graph Problems

In this section, we present a diminisher for the Multicolored Path and Colorful Graph Motif problem. First, we present a diminisher for Multicolored Path:

Multicolored Path (MCP)
Input: An undirected graph $G = (V, E)$ and a vertex coloring col: $V \to \{1, \dots, k\}$.
Question: Is there a simple path P in G that contains exactly one vertex of each color?

MCP is NP-complete as it generalizes Hamiltonian Path, and fixed-parameter tractable (solvable in $2^{O(k)}n^2$ time [AYZ95]) and WK[1]-complete [Her+15] when parameterized by k. We prove the problem to be diminishable regarding k.

Proposition 4.18. Multicolored Path *parameterized by k is diminishable.*

Proof. We give a parameter-decreasing branching rule and a strict composition for MCP. The result then follows from Lemma 4.4. Let $(G = (V, E), \mathrm{col})$ be an instance of MCP. Our parameter-decreasing branching rule for $(G = (V, E), \mathrm{col})$ computes an instance $(G_{(v_1,v_2,v_3)}, \mathrm{col}')$ for each ordered triplet (v_1, v_2, v_3) of pairwise distinct vertices of V such that v_1, v_2, v_3 forms a multicolored path in G. The graph $G_{(v_1,v_2,v_3)}$ is constructed from G as follows: Delete from G all vertices $w \in V \setminus \{v_2, v_3\}$ with $\mathrm{col}(w) \in \{\mathrm{col}(v_1), \mathrm{col}(v_2), \mathrm{col}(v_3)\}$. Following this, only vertices of $k - 1$ colors remain, and v_2 and v_3 are the only vertices colored $\mathrm{col}(v_2)$ and $\mathrm{col}(v_3)$, respectively. Then delete all edges incident with v_2, apart from $\{v_2, v_3\}$, and relabel all colors so that the image of col for $G_{(v_1,v_2,v_3)}$ is $\{1, \ldots, k - 1\}$.

Clearly our parameter-decreasing branching rule can be performed in polynomial time. Furthermore, the parameter decreases in each output instance. We show that the first requirement of Definition 4.4 holds as well: Indeed, suppose that G has a multicolored path $v_1, v_2, \ldots, v_k$ of length k. Then $v_2, \ldots, v_k$ is a multicolored path of length $k - 1$ in $G_{(v_1,v_2,v_3)}$ by construction. Conversely, suppose that there is a multicolored path $u_2, \ldots, u_k$ of length $k - 1$ in some $G_{(v_1,v_2,v_3)}$. Then since v_2 is the only vertex of color $\mathrm{col}(v_2)$ in $G_{(v_1,v_2,v_3)}$, and since v_2 is only adjacent to v_3, without loss of generality it must be that $u_2 = v_2$ and $u_3 = v_3$. Hence, since v_1 is adjacent to v_2 in G, and no vertices of $u_2, \ldots, u_k$ have color $\mathrm{col}(v_1)$ in G, the sequence of $v_1, u_2, \ldots, u_k$ forms a multicolored path of length k in G.

Our strict composition for MCP is as follows. Given a sequence of inputs $(G_1, \mathrm{col}_1), \ldots, (G_t, \mathrm{col}_t)$, the strict composition constructs the disjoint union G and the coloring function col of all graphs G_i and coloring functions col_i, $1 \leq i \leq t$. Clearly, (G, col) contains a multicolored path of length k if and only if there is a multicolored path of length k in some (G_i, col_i). The result thus follows directly from Lemma 4.4. □

We are set to prove the first part of Theorem 4.1(iv).

Proposition 4.19. *Unless P = NP,* Multicolored Path *parameterized by* $k \log(n)$ *has no strict polynomial kernel.*

Proof. Chen et al. [CFM11, Proposition 3.10] proved that if L is a parameterized problem which can be solved in $2^{k^{O(1)}}|x|^{O(1)}$ time on any input (x, k), then L parameterized by k has a polynomial kernel if and only if L parameterized by $k \log(|x|)$ has a polynomial kernel. It is easy to verify that their proof also holds for strict polynomial kernels. Thus, as Multicolored Path can be solved in $2^{k^{O(1)}} n^{O(1)}$ time [AYZ95], the result follows from Proposition 4.18. ☐

The Colorful Graph Motif problem asks, given an undirected graph $G = (V, E)$ and a vertex coloring function $\mathrm{col} : V \to \{1, \ldots, k\}$, whether there exists a connected subgraph of G containing exactly one vertex of each color. Colorful Graph Motif is known to be fixed-parameter tractable when parameterized by k [Bet+11a] and has been used to show that several problems in degenerate graphs have no polynomial kernels unless $\mathrm{coNP} \subseteq \mathrm{NP}_{/\mathrm{poly}}$ [Cyg+12]. The idea used in the parameter diminisher for MCP can also be applied to Colorful Graph Motif.

Proposition 4.20. Colorful Graph Motif *parameterized by k is diminishable.*

Proof. We show that there is a parameter-decreasing branching rule and a strict composition for Colorful Graph Motif. Let $(G = (V, E), \mathrm{col})$ be an instance of Colorful Graph Motif. Assume that there are only edges between differently colored vertices. For each $\{v, w\} \in E$, the parameter-decreasing branching rule computes an instance $(G_{\{v,w\}}, \mathrm{col}')$ where the graph $G_{\{v,w\}}$ is a copy of G where all vertices of colors $\mathrm{col}(v)$ and $\mathrm{col}(w)$ are removed. Furthermore, a new vertex v^* is added with color $\mathrm{col}(v^*) = \mathrm{col}(v)$ and with edges to all vertices in $N_G(v) \cup N_G(w)$ that have not been removed. Clearly, $G_{\{v,w\}}$ only contains vertices of $k-1$ different colors and is computable in polynomial time. Also, if G contains a colorful motif $\{v_1, v_2\} \cup \{v_3, \ldots, v_k\}$ where, without loss of generality, $\{v_1, v_2\} \in E$, then $G_{\{v_1,v_2\}}$ contains the colorful motif $\{v^*\} \cup \{v_3, \ldots, v_k\}$. Conversely, if a graph $G_{\{v,w\}}$ contains a colorful motif, then it has to contain v^* since it is the only vertex of its color. Let $\{v^*\} \cup \{v_3, \ldots, v_k\}$ be a colorful motif in $G_{\{v,w\}}$, then $\{v, w\} \cup \{v_3, \ldots, v_k\}$ is a colorful motif in G since v^* is adjacent to some vertex v_i in the motif and hence, by construction, v_i is adjacent to v or to w and there is an edge between v and w.

The strict composition constructs the disjoint union of the sequence of inputs. Clearly, the disjoint union has a colorful motif if and only if one of the input graphs has a colorful motif. Lemma 4.4 now yields the result. ☐

4.3.4. Component-Wise Annotated Graph Problems

In the following, we prove that the following family of problems is diminishable.

Multi-Component Annotated Π (MCA-Π)

Input: An undirected graph $G = (V, E)$, a vertex subset $D \subseteq V$, and an integer k.

Question: Is there a vertex set S in a connected component $G' = (V', E')$ of G such that $(D \cap V') \subseteq S \subseteq V'$, $|S \setminus (D \cap V')| \leq k$, and S fulfills property Π in G'?

Herein, we refer to the set D as the annotated set. The basic idea behind the diminisher for MCA-Π is that we can branch over all possible vertices contained in a solution and add them to the annotated set. That is, we increase the annotated set in favor of decreasing the required solution size. We say that a property Π can be verified in polynomial time, if there is an algorithm that on any input graph $G = (V, E)$ with vertex subset $S \subseteq V$ decides in time polynomial in the size of G whether S fulfills property Π in G.

Lemma 4.21. *If* Π *can be verified in polynomial time, then* Multi-Component Annotated Π *is diminishable.*

Proof. The main idea of the parameter diminisher is to extend the set D of annotated vertices by each possible vertex in the graph. Formally, given an instance $(G = (V, E), D, k)$, in polynomial time we either return a trivial yes- or no-instance equivalent to (G, D, k) or compute an equivalent instance $(G^*, D^*, k-1)$ as follows. For each connected component $G' = (V', E')$, check whether $D \cap V'$ fulfills property Π, and if so, return a trivial yes-instance.

Otherwise, if $D = V$, then return a trivial no-instance. If $D \neq V$, then we construct the equivalent instance $(G^*, D^*, k-1)$. Let G^* and D^* be initially empty. For each connected component $G' = (V', E')$, consider two cases. If $D \cap V' = V'$, then add a copy of G' to G^*. Otherwise, if $D \cap V' \subsetneq V'$, do the following. Let the vertices in $V' \setminus D$ be enumerated as $v_1, \ldots, v_\ell$, where $\ell = |V' \setminus D|$. For each vertex $i \in \{1, \ldots, \ell\}$, add a copy G'_i of G' to G^*. Denote by D^i the copy of D in G'_i, and by v^i_j, $j \in \{1, \ldots, \ell\}$, the copies of the vertices in $V' \setminus D$. Add $D^i \cup \{v^i_i\}$ to D^*. This finishes the construction. We claim that (G, D, k) is a yes-instance if and only if $(G^*, D^*, k-1)$ is a yes-instance.

($\Rightarrow$) Let $S \subseteq V'$ be a solution for (G, D, k), where $G' = (V', E')$ forms a connected component in G. Let $v_i \in S \setminus D$ (note that $S \setminus D \neq \emptyset$). Then there is a connected component $G'_i = (V'_i, E'_i)$ in G^* such that $D^i \cup \{v^i_i\}$ is the set of annotated vertices in G'_i. Let S^* denote the copies of S in G'_i. As G'_i is isomorphic to G', S^* fulfills property Π in G'_i. Moreover, $|S^* \setminus (D^* \cap V'_i)| = |S \setminus ((D \cap V') \cup \{v_i\})| = |S \setminus (D \cap V')| - 1 \leq k - 1$.

($\Leftarrow$) Let $S^* \subseteq V_i'$ be a solution for $(G^*, D^*, k-1)$, where $G_i' = (V_i', E_i')$ forms a connected component in G^*. Let $G' = (V', E')$ be the connected component in G isomorphic to G_i'. Let S be the set of vertices whose copy in G_i' is S^*. Clearly, S fulfills property Π in G'. Note that there is exactly one vertex $v_i^i \in S^* \cap D^*$ such that for its origin $v_i \in V'$ holds $v_i \notin S \cap D^*$. It follows that $|S \setminus (D \cap V')| = |S^* \setminus (D^* \cap V_i') \cup \{v_i^i\}| = |S^* \setminus (D^* \cap V_i')| + 1 \leq k - 1 + 1 = k$. □

Our first application of Lemma 4.21 is Π being a *defensive alliance* [FR07], a notion introduced by Kristiansen et al. [KHH04] (see also [OST18] for a survey on alliances): Given an undirected graph $G = (V, E)$, a vertex set S is called a defensive alliance if for all $s \in S$ it holds that S forms a majority in the neighborhood of s, that is, $|N_G[s] \cap S| \geq |N_G[s] \setminus S|$. The problem Defensive Alliance, where given an undirected graph G and an integer k, the question is whether G contains a defensive alliance of size at most k, is NP-complete and fixed-parameter tractable when parameterized by the solution size k [Fer17, FR07]. Defensive alliances have the property that if S is a defensive alliance, then each $S' \subseteq S$ forming a maximally connected subgraph is a defensive alliance. Hence, the problem variant MCA-Defensive Alliance is a natural generalization of Defensive Alliance (for the generalization, set $D = \emptyset$). Via small modifications, one can prove that MCA-Defensive Alliance remains fixed-parameter tractable when parameterized by the solution size k. As a result, MCA-Defensive Alliance is contained in NP, and hence by Lemma 4.21 we obtain the following.

Proposition 4.22. MCA-Defensive Alliance *parameterized by k is diminishable.*

Another application of Lemma 4.21 is Π being a vertex cover. We point out that the classic Vertex Cover problem admits a quadratic kernel when parameterized by the solution size k [BG93, DF99]. Note that MCA-Vertex Cover remains trivially NP-complete and fixed-parameter tractable when parameterized by the solution size k. However, by Lemma 4.21 and Theorem 4.2, its kernelizability changes due to the following.

Proposition 4.23. MCA-Vertex Cover *parameterized by k is diminishable.*

4.4. Concluding Remarks

Based on results of Chen et al. [CFM11], we proved their basic ideas to be extendable to a larger class of problems than they dealt with. We showed that for several natural problems a strict polynomial kernel is as likely as P = NP. Since basically all observed (natural and practically relevant) polynomial kernels are strict, this reveals that the existence of valuable kernels may be tighter connected to the P vs. NP problem than previously expected. As a remark, Fernau et al. [Fer+18, Proposition 7] proved an adaption of the diminisher framework for excluding *non-uniform* strict polynomial kernels assuming $\mathrm{NP} \not\subseteq \mathrm{P}_{/\mathrm{poly}}$, where $\mathrm{NP} \subseteq \mathrm{P}_{/\mathrm{poly}}$ implies that the polynomial hierarchy collapses to its second level (see Lemma 1.7). This indicates that the framework adapts to notions of kernelizations where different running times are required.

The diminisher framework leaves several challenges for future work. Are there natural problems where the presented framework is able to refute strict polynomial kernels while the (cross-)composition framework is not? This possibly also ties in with the question whether there are "natural" parameterized problems that admit a polynomial kernel but no strict polynomial kernel.[2]

We finally list some concrete open problems. We proved Terminal Steiner Tree (TST) to be diminishable. A kind of "dual" problem to TST is the Internal Steiner Tree problem [Hua+13] (where the terminals are not allowed to be leaves, see Appendix A).

Open Problem 4. Is Internal Steiner Tree parameterized by $k + |T|$ diminishable?

We proved that Multicolored Path parameterized by the solution size k is diminishable. For graph problems, a vertex-coloring seems to help to construct diminishers. The diminishability of the uncolored version of the problem, and also of its directed variant, remains open.

Open Problem 5. Is Longest Path parameterized by the solution size k diminishable?

Finally, we ask the following.

Open Problem 6. Is Connected Vertex Cover parameterized by k or Hitting Set parameterized by n diminishable?

[2] Note that Chen et al. [CFM11, Proposition 3.3] presented an artificial parameterized problem admitting a polynomial kernel but no strict polynomial kernel.

Whether one can exclude some relaxation of strict (polynomial) kernelization using a strengthened form of diminishers is addressed in the next Chapter 5.

Chapter 5.

Strong Diminisher: Limits and Applications inside P

In this chapter, we develop strong diminishers in order to exclude less strict kernelizations. We prove that several problems admit no strong diminishers unless the Exponential Time Hypothesis breaks. However, we prove that strong diminishers can be used to obtain kernelization lower bounds for polynomial-time solvable problems.

5.1. Introduction

The diminisher framework (see preceding Chapter 4) applies to a wider range of parameterized problems than previously known. That is, several parameterized problems admit no strict polynomial kernels unless P = NP. In this chapter, we study two adaptions of the diminisher framework.

Firstly, observe that under the weak assumption of P $\neq$ NP, the diminisher framework only excludes polynomial kernelization where the parameter value is allowed to only increase by some constant addend. Hence, we ask for the following first adaption:

(1) Can we adapt the diminisher framework to exclude "less" strict kernels (which we will call *semi-strict* kernels), where we allow the parameter value to increase by only a constant *factor*, assuming P $\neq$ NP?

This chapter is based on (parts of) *Diminishable Parameterized Problems and Strict Polynomial Kernelization* by Henning Fernau, Till Fluschnik, Danny Hermelin, Andreas Krebs, Hendrik Molter, and Rolf Niedermeier (Computability [Fer+20]) and *Kernelization Lower Bounds for Finding Constant-Size Subgraphs* by Till Fluschnik, George B. Mertzios, and André Nichterlein (Computability in Europe (CiE'18) [FMN18]).

Secondly, the idea behind the diminisher framework to work is that a diminisher and a strict polynomial kernelization for a parameterized problem together form a polynomial-time algorithm deciding every input instance of the problem. Indeed, if the running times of the diminisher and of the kernelization are known, then one can derive the running time of the algorithm. Hence, intuitively, if a problem presumably admits no polynomial-time algorithm for some degree of the polynomial, yet admits a "fast" diminisher, then the problem also presumably admits no fast and small kernelization. This observation leads us to problems that are solvable in polynomial time, where *conditional* lower bounds on the running times, relying on popular conjectures like the SETH, the 3SUM-, or the APSP-conjecture, form an active research field (see, e.g, [AGV15, AV14, AVW16, Bri14]). Consequently, we ask for the following second adaption:

(2) Can we adapt the diminisher framework to prove conditional kernelization lower bounds for parameterized, polynomial-time solvable problems?

Although studied mostly for NP-hard problems, it is natural to apply kernelization also to polynomial-time solvable problems as done e.g. for finding maximum matchings [MNN17], and hence is part of the field "FPT in P" [GMN17] dealing with parameterized algorithms and complexity for problems in P (see also, e.g., [AVW16, Flu+17b, Fom+17b, Fom+18]). It is thus also important to know the *limits* of kernelization for problems in P.

As every decision problem in P admits a kernelization which simply solves the input instance and produces a kernel of size $O(1)$ (encoding the `yes`/`no` answer), it is crucial to investigate the *trade-off* between (i) the size of the kernel and (ii) the running time of the kernelization algorithm. The following notion captures this trade-off: An (a, b)-*kernelization* for a parameterized problem L is an algorithm that, given any instance $(x, k) \in \Sigma^* \times \mathbb{N}$, computes in $O(a(|x|))$ time an instance (x', k') such that (i) $(x, k) \in L \iff (x', k') \in L$ and (ii) $|x'| + k' \in O(b(k))$. In fact, in this chapter we will study *proper* (a, b)-kernelization, that is, where the parameter value is not allowed to increase.

Our Contributions. We adapt the diminisher framework (see Chapter 4) for semi-strict kernelization (Theorem 5.1). Crucial in the adaption is our notion of *strong diminishers* (Definition 5.2): diminishers that decrease the parameter value by some constant factor. On the one hand, we prove two parameterized problems to be strongly diminishable and hence exclude semi-strict polynomial kernelization (Theorem 5.2). On the other hand, we prove several parameterized

Table 5.1.: Overview of our results. Here, k is interchangeably the order of the largest connected component, the degeneracy, or the maximum degree.

	Negative Weight Triangle (NWT)	Triangle Collection (TC)
lower bounds (Thm. 5.11)	No proper (n^α, k^β)-kernelization with $\alpha, \beta \geq 1$ and $\alpha \cdot \beta < 3$, assuming: the APSP-conjecture to hold.	the SETH, 3SUM-, or APSP-conjecture to hold.
kernel (Thm. 5.26)	Proper $(n^{(3+\varepsilon)/(1+\varepsilon)}, k^{1+\varepsilon})$-kernelization for every $\varepsilon > 0$, e.g., proper $(n^{5/3}, k^3)$-kernelization.	

problems to admit no strong diminisher unless the *Exponential Time Hypothesis (ETH)* (see Hypothesis 1.10) breaks (Theorem 5.5), and thus answering our first question (1) in some negative.

We further adapt the diminisher and strong diminisher framework for proper (a, b)-kernelization of polynomial-time solvable problems. Our results concern the H-Subgraph Isomorphism *(H-SI)* problem, where, given an undirected graph $G = (V, E)$, the question is whether G contains H as a subgraph, for *constant-sized connected* graphs H. As a running example, we focus on the fundamental case where H is a triangle. We present diminishers (along with conditional kernelization lower bounds) for the following weighted and colored variants of the problem (our results are summarized in Table 5.1):

Negative Weight Triangle (NWT)
Input: An undirected graph G with edge weights $w\colon E(G) \to \mathbb{Z}$.
Question: Is there a triangle T in G with $\sum_{e \in E(T)} w(e) < 0$?

Triangle Collection (TC)
Input: An undirected graph G with surjective coloring $\mathrm{col} : V(G) \to \{1, \ldots, f\}$.
Question: Does there for all color-triples $C \in \binom{\{1,\ldots,f\}}{3}$ exist a triangle with vertex set $T = \{x, y, z\}$ in G such that $\mathrm{col}(T) = C$?

We assume the edge weights for NWT and the values of the coloring for TC to be upper-bounded polynomially in the number of vertices of the input graph (and hence of logarithmic encoding length). We consider three parameters for NWT and TC (in decreasing order): (i) order of the largest connected component, (ii) maximum degree, and (iii) degeneracy. We prove that both NWT and TC admit a strong linear-time diminisher for each of these three parameters. Together with the conditional hardness of NWT and TC (see below), we then obtain lower bounds on strict kernelization. Thus, we answer our second question (2) in some affirmative.

NWT and TC are conditionally hard in the following sense: If NWT admits a *truly* subcubic algorithm, that is, with running time $O(n^{3-\varepsilon})$ for some $\varepsilon > 0$, then All Pairs Shortest Paths (APSP) also admits a truly subcubic algorithm, breaking the APSP-conjecture [VW18] (see Conjecture 1.12). A truly subcubic algorithm for TC breaks the SETH (see Hypothesis 1.10), the 3SUM-conjecture (see Conjecture 1.13), and the APSP-conjecture [AVY18].

Finally, complementing our lower bounds, we prove some proper (Turing) kernelization upper bounds (refer to Table 5.1).

5.2. Semi-Strict Kernels and Strong Diminishers

As strict kernels only allow an increase of the parameter value by an additive constant (Definition 4.1), one may ask whether one can exclude less restrictive versions of strict kernels for parameterized problems using the concept of parameter diminishers. Targeting this question, in this section we study scenarios with a multiplicative (instead of additive) parameter value increase by a constant. That is, property (iii) in Definition 4.1 is replaced by $k' \leq c{\cdot}k$, for some constant c. We refer to this as *semi-strict kernels*.

Definition 5.1 (Semi-strict kernel)**.** A *semi-strict kernelization* for a parameterized problem L is a polynomial-time algorithm that on input instance $(x,k) \in \Sigma^* \times \mathbb{N}$ outputs an instance $(x',k') \in \Sigma^* \times \mathbb{N}$, the *semi-strict kernel*, satisfying:
(i) $(x,k) \in L \iff (x',k') \in L$,
(ii) $|x'| \leq f(k)$, for some function f, and
(iii) $k' \leq c \cdot k$, for some constant c.
We say that L admits a semi-strict *polynomial* kernelization if $f(k) \in k^{O(1)}$.

On the one hand, every strict kernelization with constant c is a semi-strict kernelization with constant $c+1$. On the other hand, if a parameterized problem L admits a semi-strict kernel with constant c, then there is not necessarily a constant c' such that for *every* input instance (x,k) the obtained parameter value k' of the output instance (x',k') is upper-bounded by $k+c'$. Hence, L

does not necessarily admit a strict kernelization. In this sense, Definition 5.1 generalizes strict kernelizations.

To exclude semi-strict kernels of polynomial size under the assumption P $\neq$ NP, we prove an analogue of Theorem 4.2 for semi-strict kernelization. To this end, we introduce a stronger version of our parameter diminisher: Formally, we replace property (ii) in Definition 4.2 by $k' \leq k/c$, for some constant $c > 1$. We refer to this as *strong parameter diminishers.*

Definition 5.2 (Strong parameter diminisher)**.** A *strong parameter diminisher* for a parameterized problem L is a polynomial-time algorithm that on input instance $(x, k) \in \Sigma^* \times \mathbb{N}$ outputs an instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that

(i) $(x, k) \in L \iff (x', k') \in L$, and

(ii) $k' \leq k/c$, for some constant $c > 1$.

We call a strong parameter diminisher also a strong diminisher for short, and a parameterized problem admitting a strong diminisher to be strongly diminishable.

Remark 5.1*.* To simplify arguments in subsequent proofs, we often assume without loss of generality that the constant of any strong diminisher is at least two. Consider a strong parameter diminisher $\mathcal{D}$ with constant $1 < c < 2$. Let $\mathcal{D}'$ be the repetition of $\mathcal{D}$ exactly $\lceil \log_c(2) \rceil$ times. Then $\mathcal{D}'$ is a strong parameter diminisher with constant $c' := c^{\lceil \log_c(2) \rceil} \geq 2$.

Next, we show an analogue of Theorem 4.2 for semi-strict polynomial kernelizations and strong parameter diminishers.

Theorem 5.1. *Let L be a parameterized problem such that its unparameterized version is NP-hard and $\{(x, k) \in L \mid k \leq c\} \in P$, for some constant $c \geq 1$. If L is strongly diminishable and admits a semi-strict polynomial kernel, then $P = NP$.*

Proof. Let L be a parameterized problem whose unparameterized version is NP-hard and it holds that $\{(x, k) \in L \mid k \leq c\} \in$ P, for a constant $c \geq 1$. Let $\mathcal{D}$ be a strong parameter diminisher for L with constant $c_d \geq 2$ and let $\mathcal{A}$ be a semi-strict polynomial kernelization for L with constant $c_a > 1$. We show that we can solve any instance (x, k) of L, with k being the parameter, in polynomial time. Let (x, k) be an instance of L. Apply $\mathcal{D}$ on (x, k) exactly $c_r := \lceil \log_{c_d}(c_a + c_d) \rceil$ times to obtain an equivalent instance $\mathcal{D}^{c_r}(x, k) = (x', k')$ with $k' \leq k/c_d^{c_r} \leq k/(c_a + c_d)$. Observe that the size of (x', k') is still polynomial in the size of (x, k) as c_r is a constant. Next, apply $\mathcal{A}$ on (x', k') to obtain an equivalent instance (x'', k'') with $|x''| + k'' \leq k'^{c'}$, $c' \geq 1$, and $k'' \leq c_a \cdot k' \leq$

$c_a \cdot k/(c_a + c_d) < k$. Repeating the described procedure at most k times produces an instance (y, k_y) of L with $k_y \leq c$, solvable in polynomial time. □

By Theorem 5.1, if we can give a strong diminisher for a parameterized problem, then it admits no semi-strict polynomial kernel, unless P = NP.

5.2.1. Two Strongly Diminishable Problems

We next study the Set Cover and the Hitting Set problem (see Appendix A for the problem definitions). We show that Set Cover parameterized by $k\log(n)$, where n denotes the size of the universe, and Hitting Set parameterized by $k\log(m)$, where m denotes the size of the set of subsets of the universe, are strongly diminishable. Hence, due to Theorem 5.1, both admit no semi-strict polynomial kernelizations unless P = NP.

Theorem 5.2. *Unless P = NP, none of the following admits a semi-strict polynomial kernel:*
(i) Set Cover *parameterized by* $k\log(n)$*;*
(ii) Hitting Set *parameterized by* $k\log(m)$*.*

Note that Hermelin et al. [Her+15] studied these two parameterized problems in the context of lower bounds regarding Turing kernelization.

Proposition 5.3. Set Cover *parameterized by* $k\log(n)$ *is strongly diminishable.*

Proof. Let $(U, \mathcal{F} = \{F_1, \ldots, F_m\}, k)$ be an instance of Set Cover and assume that $k \geq 2$ and $n = |U| \geq 5$. If k is odd, then we add a unique element to U, a unique set containing only this element to $\mathcal{F}$, and we set $k := k + 1$. Hence, we assume that k is even. The following procedure is a strong parameter diminisher for the problem parameterized by $k\log(n)$.

Let $U' = U$ and for all F_i, F_j create $F'_{\{i,j\}} = F_i \cup F_j$. Let $\mathcal{F}' = \{F'_{\{i,j\}} \mid i \neq j\}$ and set $k' = k/2$. This yields in polynomial time the instance $(U', \mathcal{F}', k')$ of Set Cover. In the following we show that $(U, \mathcal{F}, k)$ is a yes-instance if and only if $(U', \mathcal{F}', k')$ is a yes-instance. Furthermore, we argue that $k'\log(n') < (k\log(n))/c$ for some constant $c > 1$, where $n' = |U'|$.

($\Rightarrow$) Assume that there is a set cover $\mathcal{C} = \{C_1, C_2, \ldots, C_k\} \subseteq \mathcal{F}$ for U of size k. Let $\mathcal{C}' = \{C_1 \cup C_2, C_3 \cup C_4, \ldots, C_{k-1} \cup C_k\}$. Then clearly $\mathcal{C}' \subseteq \mathcal{F}'$ is a set cover for U' of size $k/2$.

($\Leftarrow$) Assume that there is a set cover $\mathcal{C}' = \{C'_1, C'_2, \dots, C'_{k/2}\}$ for U' of size $k/2$, where $C'_i = C_i \cup C_{i'}$ for every $i \in \{1, \dots, k/2\}$. Let $\mathcal{C} = \{C_i, C_{i'} \mid 1 \leq i \leq k/2\}$. Then clearly $\mathcal{C} \subseteq \mathcal{F}$ is a set cover for U of size at most k.

Furthermore, we have that $m' := |\mathcal{F}'| = \binom{m}{2}$. It follows that

$$k' \log(n') \leq \frac{k+1}{2} \log(n+1) \overset{k\geq 2}{\leq} \frac{3k}{4} \log(n+1) \leq \frac{\sqrt{3}}{2} k \log((n+1)^{\sqrt{3}/2}) \overset{n\geq 5}{\leq} \frac{\sqrt{3}}{2} k \log(n).$$

Note that in the first inequality, we consider the cases in which the instance was modified such that k is even. It follows that for $k > 2$ the parameter decreases by at least a factor of $\sqrt{3}/2$ and for $k = 2$ the parameter diminisher produces either a trivial `yes`- or a trivial `no`-instance (each with a constant number of vertices). □

A strong parameter diminisher for Hitting Set parameterized by $k \log(m)$ can be constructed in a similar way as we did for Set Cover. However, since Hitting Set and Set Cover are dual in the sense of parameter-constant-increasing reductions (Definition 4.3), we get the following.

Proposition 5.4. Hitting Set *parameterized by* $k \log(m)$ *is strongly diminishable.*

5.2.2. Problems without Strong Diminishers

In Chapter 4, we proved several problems to be diminishable and hence to exclude strict polynomial kernelization assuming P $\neq$ NP. In order to exclude semi-strict polynomial kernels, one may wonder whether these problems are also strongly diminishable. In the following, we prove that the latter is presumably not the case, that is, assuming the ETH to hold, there are natural diminishable problems that admit no strong parameter diminishers. This proves the limits of strong diminishers.

Theorem 5.5. *Assuming the ETH to hold, none of the following is strongly diminishable:*

(i) CNF-Sat *parameterized by number of variables;*
(ii) Rooted Path *parameterized by the path length;*
(iii) Clique *parameterized by* $p \in \{\Delta, \mathrm{tw}, \mathrm{bw}\}$.

The following lemma is the key tool for excluding strong parameter diminishers assuming the ETH to hold. Roughly, it states that a strong parameter diminisher can improve the running time of existing algorithms.

Lemma 5.6. *Let L be a parameterized problem. If there is an algorithm that solves any instance (x, k) of L in $2^{O(k)} \cdot |x|^{O(1)}$ time and L is strongly diminishable, then there is an algorithm that solves (x, k) of L in $2^{O(k/f(x,k))} \cdot |x|^{f(x,k)^{O(1)}} + T_f(x, k)$ time, where f is a T_f-time-computable function mapping instances (x, k) of L to the natural numbers with the following property: For every constant c there is a natural number n such that for all instances (x, k) of L we have that $|x| \geq n$ implies that $f(x, k) \geq c$.*

Proof. Let L be a parameterized problem. Let $\mathcal{A}$ be an algorithm that solves any instance (x, k) of L in $2^{c_1 \cdot k} \cdot |x|^{c_2}$ time with constants $c_1, c_2 > 0$ and let $\mathcal{D}$ be a strong parameter diminisher for L with constant $d \geq 2$. Recall that by definition of a strong parameter diminisher, the size of the instance grows at most polynomially each time $\mathcal{D}$ is applied. Let $b \geq 1$ be a constant such that the size of the instance obtained by applying $\mathcal{D}$ once to (x, k) is upper-bounded by $|x|^b$. We set $c := \min\{2, b\}$. Let f be a T_f-time-computable function such that $f(x, k) \geq c$ for all instances (x, k) of L with $|x| \geq n_0$ for some $n_0 \in \mathbb{N}$.

Let (x', k') be the instance of L obtained by applying $\mathcal{D}$ for $\lceil \log_c(f(x, k)) \rceil$ times, where $f(x, k)$ is computed in $T_f(x, k)$ time, to instance (x, k) of L with $|x| \geq n_0$. We obtain

$$|x'| \leq |x|^{b^{\lceil \log_c(f(x,k)) \rceil}} \leq |x|^{b^{2 \log_c(f(x,k))}} \leq |x|^{f(x,k)^{c_3}}, \text{ for some constant } c_3 \geq 1.$$

Furthermore, the parameter decreases by the constant factor d each time the diminisher is applied, hence

$$k' = k/d^{\lceil \log_c(f(x,k)) \rceil} \leq k/d^{\log_c(f(x,k))} \leq k/d^{\log_d(f(x,k))} \leq k/f(x, k).$$

Finally, applying $\mathcal{A}$ on (x', k') solves (x', k') in time

$$2^{c_1 \cdot k'} \cdot |x'|^{c_2} \leq 2^{c_1 \cdot k/f(x,k)} \cdot |x|^{c_2 \cdot f(x,k)^{c_3}} \in 2^{O(k/f(x,k))} \cdot |x|^{f(x,k)^{O(1)}}. \qquad \square$$

We apply Lemma 5.6 to exclude the existence of strong parameter diminishers assuming the ETH to hold as follows. Consider a problem where we know a running time lower bound based on the ETH and we also know an algorithm that matches this lower bound. Then, due to Lemma 5.6, for many problems a strong parameter diminisher and a suitable choice for the function f would imply the existence of an algorithm whose running time breaks the lower bound.

Chen et al. [CFM11] showed that CNF-Sat parameterized by the number n of variables and Rooted Path parameterized by k are diminishable. We show that we cannot obtain strong diminishability for these problems unless the ETH breaks. Recall CNF-Sat parameterized by n, the parameterized problem of deciding whether a given Boolean formula with n variables in conjunctive normal form is satisfiable.

Proposition 5.7. *Assuming the ETH to hold,* CNF-Sat *parameterized by n is not strongly diminishable.*

Proof. CNF-Sat can be solved in $2^n(n+m)^{O(1)}$ time via a brute-force algorithm $\mathcal{A}$, but admits no $2^{o(n)}$ time algorithm assuming the ETH to hold. By Lemma 5.6 with algorithm $\mathcal{A}$ and $f(\phi) = \log(n)$, CNF-Sat parameterized by n admits no strong parameter diminisher unless the ETH breaks. □

Proposition 5.8. *Assuming the ETH to hold,* Rooted Path *parameterized by k is not strongly diminishable.*

Proof. Hamiltonian Path on an n-vertex graph reduces trivially to Rooted Path by adding a universal vertex and taking it as the root and setting the length of the path $k = n$. Assuming the ETH to hold, as Hamiltonian Path admits no $2^{o(n)}$ time algorithm [LMS11], so does Rooted Path. However, there is an algorithm solving Rooted Path in $2^{O(k)}n^{O(1)}$ time [AYZ95]. Let $(G = (V,E), k)$ be an instance of Rooted Path and set $f((G,k)) := \log(|V|) = \log(n)$. By Lemma 5.6 we get an algorithm for Rooted Path running in $2^{O(k/\log(n))} \cdot |G|^{(\log(n))^{O(1)}} \in 2^{o(n)}$ time. Hence, Rooted Path parameterized by k admits no strong parameter diminisher unless the ETH breaks. □

Next, we show that Clique for most parameterizations we considered in Chapter 4, admits no strong parameter diminisher unless the ETH breaks.

Proposition 5.9. *Assuming the ETH to hold,* Clique *parameterized by $p \in \{\Delta, \mathrm{tw}, \mathrm{bw}\}$ is not strongly diminishable.*

Proof. Let $p \in \{\Delta, \mathrm{tw}, \mathrm{bw}\}$. Clique can be solved in $2^p \cdot n^{O(1)}$ time via a dynamic programming (brute-force) algorithm $\mathcal{A}$, but admits no $2^{o(n)}$ time algorithm unless the ETH breaks [LMS11]. Note that $p \in O(n)$. By Lemma 5.6 with algorithm $\mathcal{A}$ and $f(G,k) = \log(|V|) = \log(n)$, Clique parameterized by p admits no strong parameter diminisher unless the ETH breaks. □

Note that we do not obtain this result for Clique parameterized by the cutwidth cw, since $\mathrm{cw}(G) \in O(n^2)$, where n is the number of vertices of G. Hence, we leave open whether Clique parameterized by cw admits a strong diminisher.

Finally, note that it is not hard to observe that if we can exclude a strong parameter diminisher for a problem L parameterized by k assuming the ETH to hold, then we can exclude a parameter diminisher for L parameterized by $\log(k)$ assuming the ETH to hold. Thus, it would be interesting to know whether there is a way to exclude the existence of parameter diminishers avoiding this exponential gap between the parameterizations.

5.3. Strong Diminisher and Kernelization in P

In contrast to NP-hard problems, only little is known about kernelization lower bounds for problems in P. To the best of our knowledge all known kernelization lower bounds follow trivially from the corresponding lower bounds of the running time: For instance, in the next Chapter 6, we prove that assuming the SETH to hold, the hyperbolicity of a graph cannot be computed in $2^{o(k)} \cdot n^{2-\varepsilon}$ time for any $\varepsilon > 0$, where k denotes the vertex cover number. Abboud et al. [AVW16] proved a similar result for computing the diameter of a graph: assuming the SETH to hold, the diameter of a graph cannot be computed in $2^{o(k)} \cdot n^{2-\varepsilon}$ time for any $\varepsilon > 0$, where k denotes the treewidth of the graph. It follows that both problems admit no $(n^{2-\varepsilon}, 2^{o(k)})$-kernelization—a kernel with $2^{o(k)}$ vertices computable in $O(n^{2-\varepsilon})$ time—since such a kernelization yields an algorithm running in $O(2^{o(k)} + n^{2-\varepsilon})$ time.

Next we initiate a systematic approach to derive kernelization lower bounds for problems in P regarding "fast and small" proper kernelization.

Definition 5.3 (Proper (a, b)-kernelization)**.** A *proper* (a, b)*-kernelization* for a parameterized problem L is an algorithm that given any instance $(x, k) \in \Sigma^* \times \mathbb{N}$ computes in $O(a(|x|))$ time an instance (x', k') such that

(i) $(x, k) \in L \iff (x', k') \in L$,

(ii) $|x'| + k' \in O(b(k))$, and

(iii) $k' \leq k$.

Recall that for problems in P, both the size of the kernel *and* the kernelization running time are important.

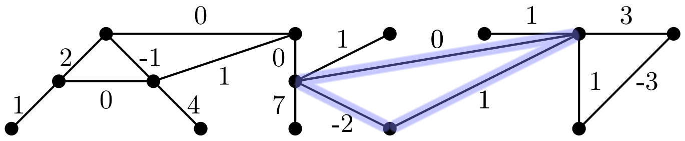


Figure 5.1.: An example undirected graph with edge weights. The highlighted edges form a triangle of negative (total) weight.

5.3.1. Adapting the Framework

To adapt the framework of (strong) diminishers to exclude proper polynomial kernelization, we need to adapt the notion of (strong) diminishers.

Definition 5.4 (a-diminisher)**.** An *a-diminisher* for a parameterized problem L is an algorithm that given any instance $(x, k) \in \Sigma^* \times \mathbb{N}$ in $O(a(|x|))$ time either decides whether $(x, k) \in L$ or computes an instance (x', k') such that

(i) $(x, k) \in L \iff (x', k') \in L$, and
(ii) $k' < k$.

A *strong a-diminisher* for L is an a-diminisher for L with $k' < k/c$ for some constant $c > 1$.

We use the problem Negative Weight Triangle (NWT) (see Figure 5.1 for an illustrative example), parameterized by the order k of the largest component, as a running example for the adapted framework of using a-diminishers to exclude proper kernels of polynomial size. Recall that the unparameterized version of this problem is as hard as APSP [VW18]. We wonder whether there is a proper $(n + m, k)$-kernelization for NWT parameterized by the size k of the largest component. Given an input $(G = (V, E), w, k)$ of NWT such a proper kernelization produces in $O(n + m)$ time an equivalent instance (G', w', k') with $|G'| + |w'| + k' \in O(k)$ and $k' \leq k$. We will prove that such a proper kernelization would yield a truly subcubic algorithm for APSP.

Now assume that NWT parameterized by k admits a proper $(n + m, k)$-kernelization *and* a strong $(n + m)$-diminisher. The basic idea of the whole approach is the same as before: alternately apply the diminisher and the kernel. While the diminisher will halve the size of the connected components at the cost of increasing the size of the instance, the proper kernel bounds the size of the instance in $O(k)$ without increasing k. Thus, after $\log(k)$ rounds of applying a strong diminisher and a proper kernel we arrive at an instance $\mathcal{I}$ with constant-size connected components. Then, we can use even a simple brute-force algorithm to solve each connected component in $O(1)$ time which gives an $O(n+m)$-time algorithm to solve the instance $\mathcal{I}$. Altogether, with $\log(k) \leq \log(n)$ rounds, each requiring $O(n+m)$ time, we arrive at an $O((n+m)\log(n))$-time algorithm for NWT. This implies a truly subcubic algorithm for APSP, thus contradicting the APSP-conjecture [VW18]. In general, we have the following.

Theorem 5.10. *Let L be a parameterized problem with parameter k such that each instance with parameter $k \leq c$ for some constant $c > 0$ is a trivial instance of L. If L with parameter k admits a proper (a,b)-kernelization and*

(i) an α-diminisher, then any instance (x,k) is solvable in $O(k \cdot (a(\alpha(b(k))) + a(|x|))$ time;

(ii) a strong α-diminisher, then any instance (x,k) is solvable in $O(\log(k) \cdot (a(\alpha(b(k))) + a(|x|))$ time.

Proof. Let (x,k) be an instance of L with parameter k. Let $\mathcal{K}$ be a proper (a,b)-kernelization and $\mathcal{D}$ be an α-diminisher. Apply $\mathcal{K}$ on (x,k) to obtain an instance (x',k') with $|x'| + k' \leq b(k)$ and $k' \leq k$. This step requires $O(a(|x|))$ time. Next, until $k' \leq c$, apply $\mathcal{K} \circ \mathcal{D}$ iteratively. Each iteration requires at most $O(\alpha(b(k)))$ time for the α-diminisher, and since the size of the resulting instance is upper-bounded by $O(\alpha(b(k)))$, the subsequent kernelization requires $O(a(\alpha(b(k))))$ time. Since in each iteration, the value of k' decreases by one, there are at most k iterations. (If $\mathcal{D}$ is a strong α-diminisher, then the number of rounds is $\log_c(k) \in O(\log(k))$.) Finally, if $k' \leq c$, then the algorithm decides the obtained instance in constant time. Hence, the algorithm requires $O(k \cdot a(\alpha(b(k))) + a(|x|))$ time to decide (x,k). (If $\mathcal{D}$ is a strong α-diminisher, then the algorithm requires $O(\log(k) \cdot a(\alpha(b(k))) + a(|x|))$ time to decide (x,k).) □

Again, as for the (strong) diminisher framework in the context of NP-hard problems, a-diminishability does not necessarily propagate through the parameter hierarchy.

Reductions for Transferring Kernels. There are two issues when using the strategy of *polynomial parameter transformations* to transfer results of Theorem 5.10 along polynomial-time solvable problems: First, we need to require the transformation to be computable "fast" enough and that the parameter does not increase ($k' \leq k$). Second, in order to transfer a proper kernel we need to show a reverse transformation from L' to L which again is computable "quickly" enough and does not increase the parameter. Hence, we essentially need to show that the two problems L and L' are *equivalent* under these restrictive transformations.

5.3.2. Applications of Strong Diminishers Inside P

In this section, we present (strong) diminishers for H-Subgraph Isomorphism (H-SI) for connected H with respect to the structural parameters (i) order ξ of the largest connected component, (ii) maximum degree Δ, and (iii) degeneracy d. Observe that $d \leq \Delta \leq \xi$ in every graph. These (strong) diminishers lead to our following main result.

Theorem 5.11. *If NWT (TC) parameterized by k being the*
(i) order ξ of the largest connected component,
(ii) maximum degree Δ, or
(iii) degeneracy d
admits a proper (n^α, k^β)-kernel for constants $\alpha, \beta \geq 1$ with $\alpha \cdot \beta < 3$, then the APSP-conjecture (the SETH, the 3SUM-, and the APSP-conjecture) breaks.

Parameter Order of the Largest Connected Component. In the following, we prove a linear-time strong diminisher regarding the parameter order of the largest connected component for problems of finding constant-size subgraphs (with some specific property).

Proposition 5.12. *NWT and TC parameterized by the order ξ of the largest connected component admit a strong $(n + m)$-diminisher.*

The idea behind our diminisher is depicted as follows: for each connected component, partition the connected component into small parts and then take the union of not too many parts to construct new (connected) components (see Figure 5.2 for an illustration of the idea with H being a triangle). Formally, we employ the following.

Construction 5.13. Let H be an arbitrary but fixed connected constant-size graph of order $c > 1$. Let $G = (V, E)$ be a graph with the largest connected component being of order ξ. First, compute in $O(n + m)$ time the connected components $G_1, \ldots, G_r$ of G. Then, construct a graph G' as follows.

Let G' be initially the empty graph. If $\xi \leq 4c$, then set $G' := G$. Otherwise, if $\xi > 4c$, then construct G' as follows. For each connected component $G_i =$

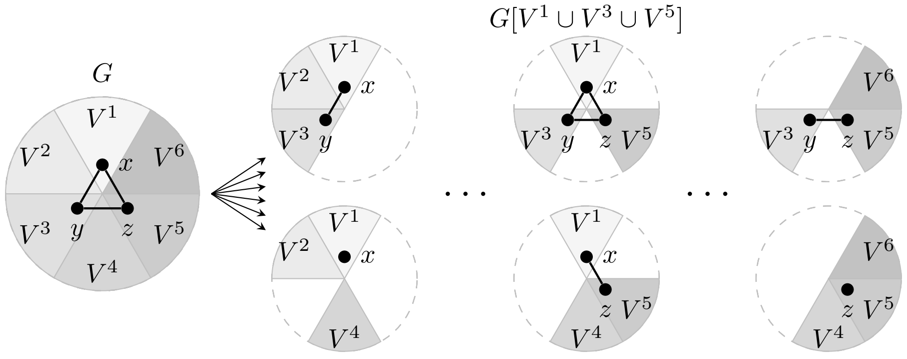


Figure 5.2.: Schematic illustration of the idea behind our diminisher for the parameter order of the largest connected component (see Construction 5.13). On the left-hand side, a connected graph G is depicted whose vertex set is partitioned into six parts $V^1, \ldots, V^6$ and that contains a triangle on the vertices $x \in V^1$, $y \in V^3$, and $z \in V^5$. Right of G, the induced subgraphs $G[V^{a_1} \cup V^{a_2} \cup V^{a_3}]$ for some $\{a_1, a_2, a_3\} \in \binom{\{1,\ldots,6\}}{3}$ are depicted.

(V_i, E_i), do the following. If the connected component $G_i = (V_i, E_i)$ is of order at most $\xi/2$, then add G_i to G'. Otherwise, if $n_i := |V_i| > \xi/2$, then we partition V_i as follows. Without loss of generality let V_i be enumerated as $V_i = \{v_i^1, \ldots, v_i^{n_i}\}$. For every $p \in \{1, \ldots, 4c\}$, define

$$V_i^p := \{v_i^q \in V_i \mid q \bmod 4c = p - 1\}.$$

This defines the partition $V_i = V_i^1 \uplus \cdots \uplus V_i^{4c}$. Then, for each $\{a_1, \ldots, a_c\} \in \binom{\{1,\ldots,4c\}}{c}$, add the graph $G[V_i^{a_1} \cup \ldots \cup V_i^{a_c}]$ to G'. This completes the construction. ▲

For the following two lemmas, let H be an arbitrary but fixed connected constant-size graph with $c > 1$ vertices and let $G = (V, E)$ be a graph with the largest connected component having order ξ.

Lemma 5.14. *Construction 5.13 outputs in $O(n + m)$ time a graph G' with connected components of order at most* $\max\{\xi/2, 4c\}$.

Proof. If $\xi > 4c$, then note that $\lfloor n_i/(4c) \rfloor \leq |V_i^p| \leq \lceil n_i/(4c) \rceil$ for all $p \in \{1, \ldots, 4c\}$. Moreover, $|V_i^{a_1} \cup \ldots \cup V_i^{a_c}| \leq c \cdot \lceil n_i/(4c) \rceil \leq \xi/4 + c < \xi/2$. The size of G' is $O(\binom{4c}{c}(n+m)) = O(n+m)$ as c is constant. It is not difficult to see that G' can be constructed in $O(n+m)$ time. $\square$

Lemma 5.15. *Graph G contains a subgraph $F = (V_F, E_F)$ isomorphic to H if and only if G', returned by Construction 5.13, contains a subgraph $F' = (V'_F, E'_F)$ isomorphic to H, where V'_F and E'_F are copies of V_F and E_F in G', respectively.*

Proof. Clearly, as G' is a disjoint collection of induced subgraphs of G and H is connected, if G' contains a subgraph isomorphic to H, then also G does.

Let G contain a subgraph F isomorphic to H. If $\xi \leq 4c$, then $G' = G$ contains F. Otherwise, if $\xi > 4c$, then consider the following two cases. If F is contained in a connected component in G of size at most $\xi/2$, then F is also contained in G'. Otherwise, F is contained in a connected component G_i of size larger than $\xi/2$. Let $V(F) \subseteq V_i^{a_1} \cup \ldots \cup V_i^{a_c}$ for some $\{a_1, \ldots, a_c\} \subseteq \binom{\{1,\ldots,4c\}}{c}$ (recall that F contains c vertices). Then F is a subgraph of $G[V_i^{a_1} \cup \ldots \cup V_i^{a_c}] \subseteq G'$. $\square$

With H being a triangle ($c = 3$) while asking for negative weight, due to Lemmas 5.14 and 5.15, we get a strong $(n+m)$-diminisher for NWT. When asking for a specific vertex-coloring, this also yields a strong $(n+m)$-diminisher for TC.

Proof of Proposition 5.12. Given an edge-weighted graph $G = (V, E, w)$, we apply Construction 5.13 to G with H being a triangle (note that $c = 3$) to obtain G'. We introduce the edge-weights w' to G' by assigning for each edge $e \in E$ its weight to all of its copies $e' \in E(G')$. By Lemma 5.14, G' is constructed in linear time. By Lemma 5.15 and the definition of w', G' contains a negative weight triangle if and only if G does. Hence, this procedure is a strong linear-time diminisher with respect to the order ξ of the largest connected component, as (by Lemma 5.14) either $\xi' \leq \xi/2$, or $\xi' \leq 4c$ (implying $G' = G$), where in the latter case our strong diminisher decides whether G' contains a triangle of negative weight in $O(n)$ time.

For TC, the proof works analogously except that for each vertex $v \in V$, we color its copies in G' with the color of v. $\square$

There is a straight-forward $O(\xi^2 \cdot n)$-time algorithm for NWT and TC: Check for each vertex all pairs of other vertices in the same connected component. However, assuming the APSP-conjecture to hold (and the SETH to hold for TC) there are no $O(n^{3-\varepsilon})$-time algorithms for any $\varepsilon > 0$ [AVY18, VW18].

Combining this with our diminisher in Proposition 5.12 we can exclude certain proper kernels as shown next.

Proof of Theorem 5.11(i). By Proposition 5.12, we know that NWT parameterized by ξ admits a strong $(n+m)$-diminisher. Suppose that NWT admits a proper (n^α, ξ^β)-kernel for $\alpha \geq 1, \beta \geq 1$ with $\alpha \cdot \beta = 3 - \varepsilon_0$, $\varepsilon_0 > 0$. It follows by Theorem 5.10 that NWT is solvable in $t(n,\xi) \in O(\xi^{\beta \cdot \alpha} \log(\xi) + n^\alpha)$ time. Observe that $\log(\xi) \in O(\xi^{\varepsilon_1})$ for $0 < \varepsilon_1 < \varepsilon_0$. Together with $\xi \leq n$ and $\alpha \cdot \beta = 3 - \varepsilon_0$ we get $t(n,\xi) \in O(n^{3-\varepsilon})$ with $\varepsilon = \varepsilon_0 - \varepsilon_1 > 0$. Hence, the APSP-conjecture breaks [VW18].

The proof for TC works analogously. □

Parameter Maximum Degree. The following is what we prove next.

Proposition 5.16. *NWT and TC parameterized by maximum degree Δ admit a strong $(n+m)$-diminisher.*

The diminisher described in Construction 5.13 does not necessarily decrease the maximum degree of the graph. We thus adapt the diminisher to partition the edges of the given graph (using a (not necessarily proper) edge-coloring) instead of its vertices. Furthermore, if H is of order c, then H can have up to $\binom{c}{2} \leq c^2$ edges. Thus, our diminisher considers all possibilities to choose c^2 (instead of c) parts of the partition. For the partitioning step, we need the following.

Lemma 5.17. *Let $G = (V,E)$ be a graph with maximum degree Δ and let $b \in \mathbb{N}$. One can compute a (not necessarily proper) edge-coloring* $\operatorname{col}\colon E \to \mathbb{N}$ *with less than $2b$ colors*

- *in $O(b(n+m))$ time such that*
- *each vertex is incident to at most $\lceil \Delta/b \rceil$ edges of the same color.*

Proof. The (not necessarily proper) edge-coloring can be computed in $O(b(n+m))$ time with a simple generalization of a folklore greedy algorithm to compute a proper edge-coloring ($b = \Delta$): Consider the edges one by one and assign each edge the first available color. Observe that at any considered edge each of the two endpoints can have at most $b-1$ unavailable colors, that is, colors that are used on $\lceil \Delta/b \rceil$ other edges incident to the respective vertex. Hence, the greedy algorithm uses at most $2b-1$ colors. The algorithm stores at every vertex an array of length $b-1$ to keep track of the number of edges with the respective colors. Thus, the algorithm can for each edge simply try all colors at each edge in $O(b)$ time. Altogether, this gives $O(b(n+m))$ time to compute the edge-coloring. $\square$

Construction 5.18. Let H be an arbitrary but fixed connected constant-size graph of order $c > 1$. Let $G = (V, E)$ be a graph with maximum degree Δ. First, employ Lemma 5.17 to compute a (not necessarily proper) edge-coloring $\mathrm{col}\colon E \to \mathbb{N}$ with $4c^2 \leq f < 8c^2$ many colors (without loss of generality we assume $\Im(\mathrm{col}) = \{1, \dots, f\}$) such that each vertex is incident to at most $\lceil \Delta/(4c^2) \rceil$ edges of the same color.

Now, construct a graph G' as follows (see Figure 5.3 for an illustration). Let G' be initially the empty graph. If $\Delta \leq 4c^2$, then set $G' := G$. Otherwise, if $\Delta > 4c^2$, then construct G' as follows. We first partition E: Let E^p be the edges of color p for every $p \in \{1, \dots, f\}$. Clearly, $E = E^1 \uplus \dots \uplus E^f$. Then, for each $\{a_1, \dots, a_{c^2}\} \in \binom{\{1,\dots,f\}}{c^2}$, add the graph $(V, E^{a_1} \cup \dots \cup E^{a_{c^2}})$ to G'. This completes the construction. ▲

For the following two lemmas, let H be an arbitrary but fixed connected constant-size graph with $c > 1$ vertices and let $G = (V, E)$ be a graph with maximum degree Δ.

Lemma 5.19. *Construction 5.18 outputs a graph G' in $O(n+m)$ time with maximum degree $\Delta(G') \leq \max\{\Delta/2, 4c^2\}$.*

Proof. If $\Delta > 4c^2$, then each vertex is incident to at most $\lceil \Delta/(4c^2) \rceil$ edges of E^p for all $p \in \{1, \dots, f\}$. Thus, in $(V, E^{a_1} \cup \dots \cup E^{a_{c^2}})$ the maximum degree is at most $c^2 \cdot \lceil \Delta/(4c^2) \rceil \leq \Delta/4 + c^2 < \Delta/2$. Using Lemma 5.17 with $b = 4c^2 \in O(1)$, it is not difficult to see that G' is constructed in $O(n+m)$ time. $\square$

Lemma 5.20. *Graph G contains a subgraph $F = (V_F, E_F)$ isomorphic to H if and only if G', returned by Construction 5.18, contains a subgraph $F' = (V'_F, E'_F)$ isomorphic to H, where V'_F and E'_F are copies of V_F and E_F in G', respectively.*

Proof. Clearly, as G' is a disjoint collection of subgraphs of G and H is connected, if G' contains a subgraph isomorphic to H, then also G does. Let G contain a subgraph F isomorphic to H. If $\Delta \leq 4c^2$, then $G' = G$ contains F. Otherwise, if $\Delta > 4c^2$, then let $E(F) \subseteq E^{a_1} \cup \dots \cup E^{a_{c^2}}$ for some $\{a_1, \dots, a_{c^2}\} \subseteq \binom{\{1,\dots,f\}}{c^2}$ (recall that F contains at most c^2 edges). Then F is a subgraph of $(V, E^{a_1} \cup \dots \cup E^{a_{c^2}}) \subseteq G'$. $\square$

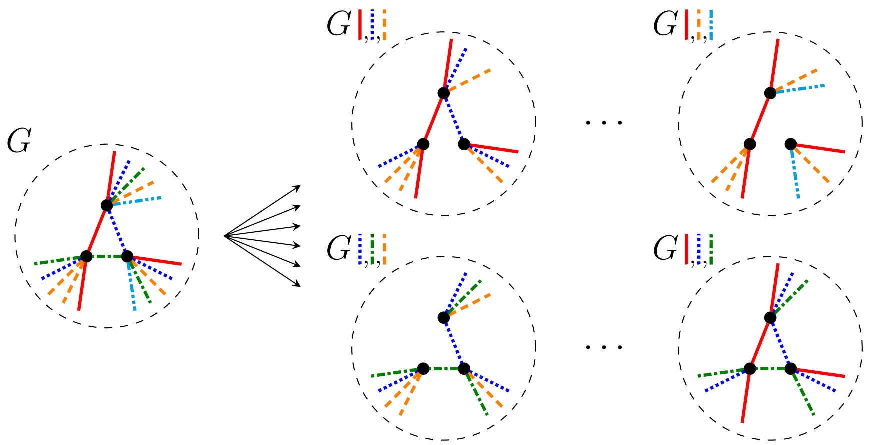


Figure 5.3.: Schematic illustration of the idea behind our diminisher for the parameter maximum degree. Line patterns indicate edge colors (see Construction 5.18). On the left-hand side, a graph G is depicted containing a triangle on three differently colored edges. Right of G, several subgraphs of G are depicted, where G followed by three lines of pairwise different patterns indicates the subgraph of G obtained by removing all edges with color different to each of the three colors corresponding to the line patterns.

Proof of Proposition 5.16. Given an edge-weighted graph $G = (V, E, w)$, we apply Construction 5.18 to G with H being a triangle (note that $c = 3$) to obtain G'. We introduce the edge-weights w' to G' by assigning for each edge $e \in E$ its weight to all of its copies $e' \in E(G')$. By Lemma 5.19, G' is constructed in linear time. By Lemma 5.20 and the definition of w', G' contains a negative weight triangle if and only if G does. Hence, this procedure is a strong linear-time diminisher with respect to the maximum degree, as (by Lemma 5.19) either $\Delta(G') \leq \Delta/2$, or $\Delta(G') \leq 4c^2$, where in the latter case our strong diminisher decides whether G' contains a triangle of negative weight in $O(n)$ time.

For TC, the proof works analogously except that for each vertex $v \in V$, we color its copies in G' with the color of v. $\square$

The proof of Theorem 5.11(ii) finally works analogously to the proof of Theorem 5.11(i).

Parameter Degeneracy. For a graph G, the degeneracy $d = \operatorname{dgn}(G)$ of G is the smallest number such that there is an ordering of the vertices with each vertex v having at most d neighbors ordered after v. Such an ordering is called *degeneracy ordering*, and can be computed in linear time [MB83]. For the parameter degeneracy, the diminisher follows the same idea as the diminisher for the parameter maximum degree (see Construction 5.18). The only difference between the two diminishers is how the partition of the edge set is obtained.

Construction 5.21. Let H be an arbitrary but fixed constant-size graph of order $c > 1$. Let $G = (V, E)$ be a graph with degeneracy d. First, compute a degeneracy ordering σ in $O(n + m)$ time. Construct a graph G' as follows.

Let G' be initially the empty graph. If $d \leq 4c^2$, then set $G' := G$. Otherwise, if $d > 4c^2$, then construct G' as follows. First, for each vertex $v \in V$, we partition the edge set $E_v := \{\{v, w\} \in E \mid \sigma(v) < \sigma(w)\}$ going to the right of v with respect to σ into $4c^2$ parts. Let E_v be enumerated as $\{e_1, \ldots, e_{|E_v|}\}$. For each v, we define $E_v^p := \{e_i \in E_v \mid i \bmod 4c^2 = p - 1\}$ for every $p \in \{1, \ldots, 4c^2\}$. Clearly, $E_v = E_v^1 \uplus \cdots \uplus E_v^{4c^2}$. Next, we define $E^p := \bigcup_{v \in V} E_v^p$ for every $p \in \{1, \ldots, 4c^2\}$. Clearly, $E = \biguplus_{1 \leq p \leq 4c^2} E^p = \biguplus_{1 \leq p \leq 4c^2} \biguplus_{v \in V} E_v^p$. Then, for each $\{a_1, \ldots, a_{c^2}\} \in \binom{\{1, \ldots, 4c^2\}}{c^2}$, add the graph $(V, E^{a_1} \cup \ldots \cup E^{a_{c^2}})$ to G'. This completes the construction. ▲

Proposition 5.22. *NWT and TC parameterized by degeneracy admit a strong $(n + m)$-diminisher.*

For the following two lemmas, let H be an arbitrary but fixed connected constant-size graph of order $c > 1$ and let $G = (V, E)$ be a graph with degeneracy d.

Lemma 5.23. *Construction 5.21 outputs a graph G' in $O(n + m)$ time with degeneracy at most* $\max\{d/2, 4c^2\}$.

Proof. If $d > 4c^2$, then for each $p \in \{1, \ldots, 4c^2\}$, the degeneracy of $F := (V, E^p)$ is at least $\lfloor d/(4c^2) \rfloor$ and at most $\lceil d/(4c^2) \rceil$. To see this, consider F with ordering σ (computed in the construction) on its vertices $V(F)$. Then, for each $v \in V(F)$, exactly $\lfloor |E_v|/(4c^2) \rfloor \leq |E_v^p| \leq \lceil |E_v|/(4c^2) \rceil$ vertices w with $\sigma(w) > \sigma(v)$ are incident with v in F. As $|E_v| \leq d$, the claim follows. Moreover, the degeneracy of $(V, E^{a_1} \cup \ldots \cup E^{a_{c^2}})$ is at most $c^2 \cdot \lceil d/(4c^2) \rceil \leq d/4 + c^2 < d/2$. It is not difficult to see that G' is constructed in $O(n + m)$ time. □

Lemma 5.24. *Graph G contains a subgraph $F = (V_F, E_F)$ isomorphic to H if and only if G', returned by Construction 5.21, contains a subgraph $F' = (V'_F, E'_F)$ isomorphic to H, where V'_F and E'_F are copies of V_F and E_F in G', respectively.*

Proof. Clearly, as G' is a disjoint collection of subgraphs of G, if G' contains a subgraph isomorphic to H, then also G does. Let G contain a subgraph F isomorphic to H. If $d \leq 4c^2$, then $G' = G$ contains F. Otherwise, if $d > 4c^2$, then let $E(F) \subseteq E^{a_1} \cup \ldots \cup E^{a_{c^2}}$ for some $\{a_1, \ldots, a_{c^2}\} \subseteq \binom{\{1,\ldots,4c^2\}}{c^2}$ (recall that F contains at most c^2 edges). Then F is a subgraph of $(V, E^{a_1} \cup \ldots \cup E^{a_{c^2}}) \subseteq G'$. ☐

Proof of Proposition 5.22. Given an edge-weighted graph $G = (V, E, w)$, we apply Construction 5.21 to G with H being a triangle (note that $c = 3$) to obtain G'. We introduce the edge-weights w' to G' by assigning for each edge $e \in E$ its weight to all of its copies $e' \in E(G')$. By Lemma 5.23, G' is constructed in linear time. By Lemma 5.24 and the definition of w', G' contains a negative weight triangle if and only if G does. Hence, this procedure is a strong linear-time diminisher with respect to degeneracy, as (by Lemma 5.23) either $d' \leq d/2$, or $d' \leq 4c^2$, where in the latter case our strong diminisher decides whether G' contains a triangle of negative weight in $O(n)$ time.

For TC, the proof works analogously except that for each vertex $v \in V$, we color its copies in G' with the color of v. ☐

The proof of Theorem 5.11(iii) finally works analogously to the proof of Theorem 5.11(i).

5.3.3. (Turing) Kernelization Upper Bounds

We complement our results on kernelization lower bounds by showing straightforward (proper) kernelization results for H-Subgraph Isomorphism for connected constant-size H to show the limits of any approach showing kernelization lower bounds.

Turing Kernelization. For the parameters order of the largest connected component and maximum degree, we present (a, b)-Turing kernelizations:

Definition 5.5. An (a, b)-*Turing kernelization* for a parameterized problem L is an algorithm that decides every input instance (x, k) in time $O(a(|x|))$ given access to an oracle that decides whether $(x', k') \in L$ for every instance (x', k')

with $|x'| + k' \leq b(k)$ in constant time. We call an (a, b)-Turing kernelization *proper* if the oracle decides whether $(x', k') \in L$ for every instance (x', k') with $|x'| + k' \leq b(k)$ and $k' \leq k$ in constant time.

Note that the diminisher framework in its current form cannot be applied to exclude (proper) (a, b)-Turing kernelizations. In fact, it is easy to see that H-Subgraph Isomorphism for connected constant-size H parameterized by the order ξ of the largest connected component admits a proper $(n + m, \xi^2)$-Turing kernelization, as each oracle call is on a connected component (which contains $O(\xi)$ vertices and at most $O(\xi^2)$ edges) of the input graph. We present a proper (a, b)-Turing kernelization for H-SI for connected constant-size H parameterized by maximum degree Δ.

Proposition 5.25. *H-Subgraph Isomorphism for connected H of order c parameterized by maximum degree Δ admits a proper $(n \cdot \Delta \cdot (\Delta - 1)^{\lfloor c/2 \rfloor}, \Delta \cdot (\Delta - 1)^{\lfloor c/2 \rfloor})$-Turing kernelization, where n denotes the number of vertices in the input graph.*

Proof. Let $(G = (V, E))$ be an input instance of H-Subgraph Isomorphism and let Δ denote the maximum degree in G. For each vertex $v \in V$, we create the subgraph G_v that is the subgraph induced by the closed $\lfloor c/2 \rfloor$-neighborhood $N_G^{\lfloor c/2 \rfloor}[v]$ of v (we refer to these as subinstances). In each subinstance the graph is of size at most $2\Delta \cdot (\Delta - 1)^{\lfloor c/2 \rfloor}$, is of maximum degree at most Δ, and each subinstance can be constructed in time linear in its size.

The algorithm outputs `yes` if and only if there is at least one subinstances containing H. This results in a total running time of $O(n \cdot \Delta \cdot (\Delta - 1)^{\lfloor c/2 \rfloor})$.

Finally, we prove that G contains H if and only if there exists a vertex $v \in V$ such that G_v contains H.

($\Leftarrow$) Since G_v is an induced subgraph of G for every $v \in V$, if G_v contains a subgraph isomorphic to H, so does G.

($\Rightarrow$) Recall that H is connected and of order c. Hence, there is a vertex $u \in V(H)$ such that $\text{dist}_H(u, w) \leq \lfloor c/2 \rfloor$ for every $w \in V(H)$. Let v be the vertex in G that corresponds to u in H. Then G_v contains H since G_v is induced on all vertices in G that are of distance at most $\lfloor c/2 \rfloor$ from v. $\square$

Running-time Related Proper Kernelization. For NP-hard problems, it is well-known that a decidable problem is fixed-parameter tractable if and only if it admits a kernel [DF13]. In the proof of the *only if*-statement, one derives a kernel of size only depending on the running time of a fixed-parameter algorithm solving the problem in question. We adapt this idea to derive a proper kernel where the running time and size admit such running time dependencies.

Theorem 5.26. *Let L be a parameterized problem admitting an algorithm solving each instance (x, k) of L in $k^c \cdot |x|$ time for some constant $c > 0$. Then for every $\varepsilon > 0$, L admits a proper $(|x|^{1+c/(1+\varepsilon)}, k^{1+\varepsilon})$-kernel.*

Proof. Let $\varepsilon > 0$ arbitrary but fixed. If $k^{1+\varepsilon} \geq |x|$, then the size of the instance is upper-bounded by $k^{1+\varepsilon} + k$. Otherwise, if $k^{1+\varepsilon} < |x|$, then we can compute a constant-size kernel (trivial `yes`-/`no`-instance) in $k^c \cdot |x| < |x|^{c/(1+\varepsilon)} \cdot |x| = |x|^{1+c/(1+\varepsilon)}$ time, where the first inequality follows from the fact that $k^{1+\varepsilon} < |x| \iff k < |x|^{1/(1+\varepsilon)}$. □

NWT and TC are both solvable in $O(k^2 \cdot n)$ time (k being the order ξ of the largest connected component, the maximum degree Δ, or the degeneracy d [CN85]). Together with Theorem 5.26 we obtain several kernelization results for NWT and TC, for instance, with $\varepsilon = 2$:

Corollary 5.27. *NWT admits a proper $(n^{5/3}, d^3)$-kernel when parameterized by the degeneracy d of the input graph.*

5.4. Concluding Remarks

We showed how different strong diminishers behave for NP-hard and polynomial-time solvable problems:

(a) While for some NP-hard problems, we can exclude strong diminishers for several parameterizations assuming the ETH to hold,

(b) for some polynomial-time solvable problems, using strong diminishers we can exclude proper (a, b)-kernelizations for several parameterizations assuming the APSP-conjecture, the 3SUM-conjecture, or the SETH to hold.

Regarding (a), we proved Clique to be not strongly diminishable for several parameters (assuming the ETH to hold), but we left open the following case.

Open Problem 7. Assuming the ETH to hold, does Clique parameterized by the cutwidth cw admit a strong diminisher?

Our framework and results regarding (b) are a first step for studying kernelization lower bounds for polynomial-time solvable parameterized problems, and hence contributes to the (young) field of "FPT in P". However, our framework leaves wide space for future work as discussed in the following.

Recall that for Negative Weight Triangle parameterized by the degeneracy d, we proved a proper (n^α, d^β)-kernel with $\alpha \cdot \beta = 5$ (Corollary 5.27) and the lower bound of $\alpha \cdot \beta < 3$ (Theorem 5.11(iii)). Future work could be to close this gap.

Open Problem 8. Is there a proper (n^α, d^β)-kernel with $3 \leq \alpha \cdot \beta < 5$ for Negative Weight Triangle or Triangle Collection each parameterized by the degeneracy d?

Moreover, our framework does not provide to exclude kernelizations where we allow the parameter value to increase, and, possibly more evidently, of *any* polynomial size in some specific, say linear, running time. In the next chapter, we prove a problem to admit an exponential-size linear-time kernelization but no subexponential-size linear-time kernelization assuming the SETH to hold. The latter is derived "indirectly" from an SETH-based running-time lower bound. We wonder whether there is a (modification of our) "direct" framework, possibly an adaption of the (cross-)composition framework, to provide such lower bounds?

Chapter 6.

Data Reduction Inside P: Hyperbolicity

In this chapter, we study the polynomial-time solvable Hyperbolicity problem. We prove a kernelization dichotomy of the problem when parameterized by the vertex cover number: in linear time, we can compute a kernel of exponential size, while unless the SETH breaks, no subexponential-sized kernel can be computed in subquadratic time. We complement our study on Hyperbolicity with three parameterized algorithms each with a running time depending linearly on the input size yet polynomially on the parameter.

6.1. Introduction

Gromov hyperbolicity [Gro87] is a popular attempt to capture and measure how *metrically* close a graph is to being a tree. For a graph, the (Gromov) hyperbolicity is a non-negative number δ that can be defined via a four-point condition: Considering a size-four subset $\{a, b, c, d\}$ of the vertex set of a graph, one takes the (non-negative) difference between the two largest of the three sums $\overline{ab} + \overline{cd}$, $\overline{ac} + \overline{bd}$, and $\overline{ad} + \overline{bc}$, where $\overline{uv}$ denotes the length of a shortest path between vertices u and v in the given graph. The hyperbolicity δ is the maximum of these differences over all size-four subsets of the vertex set of the graph. For every tree, this maximum over the differences is zero, and $\delta = 0$ means that the graph metric indeed is a tree metric (see Figure 6.1 for two graphs with $\delta = 0$). Hence, the smaller δ is, the more metrically tree-like the graph is.

This chapter is based on (parts of) *When Can Graph Hyperbolicity Be Computed in Linear Time?* by Till Fluschnik, Christian Komusiewicz, George B. Mertzios, André Nichterlein, Rolf Niedermeier, and Nimrod Talmon (Algorithmica [Flu+19a]).

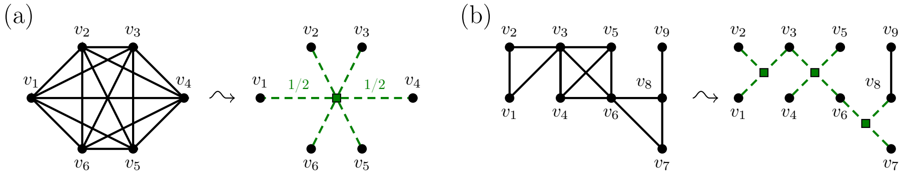


Figure 6.1.: Illustration of metric-likeness to trees of two graphs ((a) a clique and (b) some *block* graph). Right of each of the two graphs, the tree describing the graph's metric is depicted. Dashed edges (each having length 1/2) and rectangular nodes are additionally present in the trees.

The study of hyperbolicity is motivated by the fact that many real-world graphs are tree-like from a distance metric point of view [AD16, Bor+15]. This is due to the fact that many of these graphs (including Internet application networks or social networks) possess certain geometric and topological characteristics. Hence, for many applications (see, e.g., Borassi et al. [Bor+15]), including the design of efficient algorithms, it is useful to know the hyperbolicity of a graph. Typical hyperbolicity values for real-world graphs are below five [AD16]. Notably, the graph parameter treewidth—measuring tree-likeness in a non-metrical way—is unrelated to the hyperbolicity of a graph.

For an n-vertex graph, the definition of hyperbolicity via the four-point condition directly implies a simple (brute-force) $O(n^4)$-time algorithm to compute its hyperbolicity. It has been observed that this running time is too slow for computing the hyperbolicity of large graphs as occurring in applications [AD16, BCH16, Bor+15, FIV15]. On the theoretical side, it was shown that relying on some (rather impractical) matrix multiplication results, one can improve the upper bound to $O(n^{3.69})$ [FIV15]. Moreover, roughly quadratic running time lower bounds are known [BCH16, FIV15]. It is also known [CD14] that the problem of deciding whether a graph is 1-hyperbolic and the problem of deciding whether a graph contains a cycle of length four either both admit an $O(n^{3-\varepsilon})$-time algorithm, for some $\varepsilon > 0$, or neither does. The best known practical algorithm, however, still has an $O(n^4)$-time worst-case bound but uses several clever tricks when compared to the straightforward brute-force algorithm [Bor+15]. Indeed, empirical studies with this algorithm suggest an $O(mn)$ running time behavior, where m denotes the number of edges in the graph. Furthermore, there are heuristics for computing the hyperbolicity of a given graph [CCL15]. Cohen et al. [Coh+17] studied computing the hyperbolicity with a given clique-decomposition. In this context, they proved that computing the hyperbolicity of the subgraphs induced by the parts of the clique-decomposition yields a 1-additive approximation. Moreover, they proved that the hyperbolicity of an outerplanar graph can be computed in linear time.

The guiding principle in this chapter is to explore the possibility of data reduction for faster algorithms for computing the hyperbolicity in relevant special cases. To this end, we employ the framework "FPT in P" [GMN17, MNN17] of parameterized complexity analysis applied to the polynomial-time solvable hyperbolicity problem. In the "FPT in P" program, one often aims for developing *linear-time parameterized algorithms*, that is, algorithms with running times of the form $f(k) \cdot |x|$, where $|x|$ denotes the input size and $f(k)$ is some computable function only depending on the parameter k (referred to as L-FPT running time). A linear-time parameterized algorithm is *parameter-polynomial* if $f(k) \in k^{O(1)}$ (referred to as PL-FPT running time). Note that for the metric parameters diameter and hyperbolicity, linear-time algorithms are unlikely for any dependency on the parameter [BCH16].

Our Contributions. Our main result is the following kernelization dichotomy for HYPERBOLICITY regarding the parameter vertex cover number:

Theorem 6.1. HYPERBOLICITY *admits*
(i) a $2^{O(k)}$-size $O(n+m)$-time kernelization, and
(ii) no $2^{o(k)}$-size $O(n^{2-\varepsilon})$-time kernelization, for any $\varepsilon > 0$, unless the SETH breaks,

where k denotes the vertex cover number.

Building on Theorem 6.1(ii), we also prove that there is no kernel computable in truly subquadratic time of *any* size upper-bounded by a function in the maximum vertex degree, again assuming the SETH to hold. We point out that our kernelization lower bound regarding vertex cover number implies lower bounds for many other well-known graph parameters such as feedback vertex number, pathwidth, and treewidth, which can be much smaller than the vertex cover number (see Figure 1.4 in Chapter 1).

On the positive side, we prove for three natural graph parameters PL-FPT running times through data reduction (see Table 6.1 for an overview). These three graph parameters are (i) the *covering path number*, that is, the minimum number of paths which cover all vertices, where only the endpoints have degree greater than two, (ii) the *feedback edge number*, and (iii) the number of graph

Table 6.1.: Summary of our algorithmic results. Herein, k denotes the parameter and n and m denote the number of vertices and edges, respectively. RR abbreviates "Reduction Rule" used in the respective result. †"together with Reduction Rule 6.1".

Parameter	Running time	
covering path no.	$O(k(n+m)) + k^4(\log(n))^{O(1)}$	(Thm. 6.19, RR 6.3†)
feedback edge no.	$O(k(n+m)) + k^4(\log(n))^{O(1)}$	(Thm. 6.20, RR 6.3†)
no. of $\geq$ 3-deg vertices	$O(k^2(n+m)) + k^8(\log(n))^{O(1)}$	(Thm. 6.22, RR 6.4†)

vertices of *degree at least three*. Note that these three parameters are unrelated to the vertex cover number and can be arbitrarily larger than the vertex cover number (consider any biclique $K_{2,n}$, any biclique $K_{3,n}$, and any cycle C_n).

6.2. Definitions and First Observations

Let $G = (V, E)$ be graph and $a, b, c, d \in V$. We define $D_1 := \overline{ab} + \overline{cd}$, $D_2 := \overline{ac} + \overline{bd}$, and $D_3 := \overline{ad} + \overline{bc}$ (referred to as *distance sums*). Moreover, we define $\delta(a, b, c, d) := |D_i - D_j|$ if $D_k \leq \min\{D_i, D_j\}$, for pairwise distinct $i, j, k \in \{1, 2, 3\}$. If any two vertices of the quadruple $\{a, b, c, d\}$ are not connected by a path, then we set $\delta(a, b, c, d) := 0$.[1] The *hyperbolicity* of $G = (V, E)$ is defined as $\delta(G) := \max_{a,b,c,d \in V}\{\delta(a, b, c, d)\}$. We refer to Figure 6.2 for two illustrative examples. Note that by our definition, if G is not connected, then $\delta(G)$ computes the maximal hyperbolicity over all connected components of G. We say that the graph is *δ-hyperbolic* for some $\delta \in \mathbb{N}$ if it has hyperbolicity at most δ. That is, a graph is δ-hyperbolic[2] if for each quadruple $a, b, c, d \in V$ we have

$$\overline{ab} + \overline{cd} \leq \max\{\overline{ac} + \overline{bd}, \overline{ad} + \overline{bc}\} + \delta.$$

Formally, the HYPERBOLICITY problem is defined as follows.

HYPERBOLICITY
Input: An undirected graph $G = (V, E)$ and a positive integer δ.
Question: Is G δ-hyperbolic?

[1] This case is often left undefined in the literature. Our definition, however, allows to consider also disconnected graphs.

[2] Note that there is also a slightly different definition where graphs that we call δ-hyperbolic are called 2δ-hyperbolic [CCL15, MP14]; we follow the definition of Brinkmann et al. [BKM01].

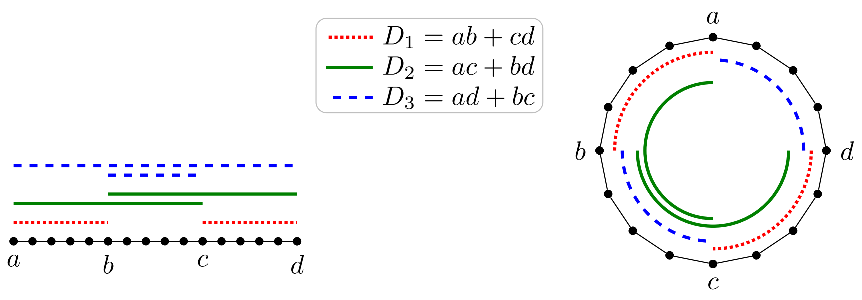


Figure 6.2.: Two illustrative examples for the four-point condition. On the left-hand side, a path P_{16} with 16 vertices is depicted. On the right-hand side, a cycle C_{16} with 16 vertices is depicted. For each of the two graphs, some quadrupel $\{a, b, c, d\}$ and the corresponding distance sums (dotted, solid, and dashed lines) are illustrated. Indeed, $\delta(P_{16}) = 0$, here indicated by the equally largest two distance sums D_2 (solid) and D_3 (dashed). The situation changes in the case of a C_{16} (that is, the P_{16} with one additional edge connecting the endpoints). We have $\delta(C_{16}) = 8$, here indicated by the largest distance sum D_2 (solid) and the two equally smallest distance sums D_1 (dotted) and D_3 (dashed).

The following Lemmas 6.2 to 6.4 and Reduction Rule 6.1 will be useful later. For any quadruple $\{a, b, c, d\}$, Lemma 6.2 upper-bounds $\delta(a, b, c, d)$ by twice the distance between any pair of vertices of the quadruple. Lemma 6.3 considers graphs for which the hyperbolicity equals the diameter. Reduction Rule 6.1 deletes degree-one vertices, and its correctness relies on Lemma 6.4.

Lemma 6.2 ([CCL15, Lemma 3.1])**.** $\delta(a, b, c, d) \leq 2 \cdot \min_{u \neq v \in \{a,b,c,d\}}\{\overline{uv}\}$

An implicit proof of the following lemma is given by Mitsche and Pralat [MP14]. We provide a direct proof of our particular statement.

Lemma 6.3. *Let G be a graph with diameter h and $\delta(G) = h$. Then for each quadruple $a, b, c, d \in V(G)$ with $\delta(a, b, c, d) = h$, it holds that exactly two disjoint pairs are at distance h and all the other pairs are at distance $h/2$.*

Proof. Let $a, b, c, d \in V(G)$ be some arbitrary quadruple with $\delta(a, b, c, d) = h$. Without loss of generality, assume $D_1 = \overline{ab} + \overline{cd}$ and $D_1 \geq \max\{D_2, D_3\}$. By Lemma 6.2 we have $\min_{u \neq v \in \{a,b,c,d\}}\{\overline{uv}\} \geq h/2$, and hence $\max\{D_2, D_3\} \geq h/2 + h/2 = h$. It follows that $h = D_1 - \max\{D_2, D_3\} \leq D_1 - h$ and thus $D_1 \geq 2h$.

Since G is of diameter h, we get $\overline{ab} = \overline{cd} = h$. Moreover, $\max\{D_2, D_3\} = h$ and, together with $\min_{u \neq v \in \{a,b,c,d\}}\{\overline{uv}\} \geq h/2$, we obtain that each other distance equals $h/2$. □

The following lemma immediately follows from a result due to Cohen et al. [Coh+17, Theorem 5].

Lemma 6.4. *Let $G = (V, E)$ be a graph with $|V| > 4$ and with a vertex $v \in V$ such that the number of connected components in $G - \{v\}$ is larger than in G. Let $A_1, \ldots, A_\ell$ denote the connected components in $G - \{v\}$. Then there is an $i \in \{1, \ldots, \ell\}$ such that $\delta(G) = \delta(G - V(A_i))$.*

Lemma 6.4 gives rise to the following degree-1 data reduction rule.

Reduction Rule 6.1. *As long as there are more than four vertices, remove vertices of degree at most one.*

Lemma 6.5. *Reduction Rule 6.1 is correct and can be exhaustively applied in linear time.*

Proof. Lemma 6.4 immediately proves the correctness of Reduction Rule 6.1. We apply Reduction Rule 6.1 in linear time as follows. First, as long as there are more than four vertices, delete degree-zero vertices. Second, collect all vertices with degree at most one in linear time in a list L. Then, as long as there are more than four vertices, iteratively delete degree-one vertices and put their neighbor into L if it has degree at most one after the deletion. Each iteration can be applied in constant time. Thus, Reduction Rule 6.1 can be applied exhaustively in linear time. □

We call a graph *reduced* if Reduction Rule 6.1 is not applicable.

6.3. A Kernelization Dichotomy regarding Vertex Cover Number

In this section, we study Hyperbolicity with respect to the parameter vertex cover number. We prove Theorem 6.1 by providing an exponential-size linear-time kernelization (Section 6.3.1) and a subexponential-size subquadratic-time kernelization lower bound, assuming the SETH to hold (Section 6.3.2). Building on the latter, we prove no subquadratic-time kernelization of *any* size in the maximum vertex degree to exist, assuming the SETH to hold (Theorem 6.13).

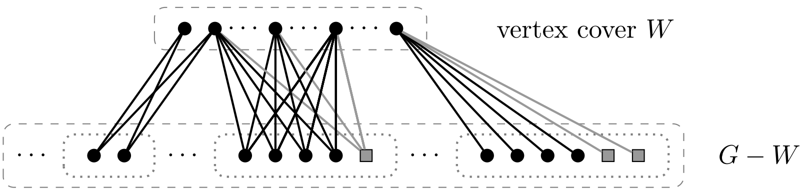


Figure 6.3.: Illustration to Reduction Rule 6.2 with a vertex cover W and the independent set $G - W$ (enclosed by dashed rectangles). The partition of $G - W$ regarding the neighborhoods in W is sketched by enclosing dotted rectangles. Here, each rectangular-shaped vertex and its incident edges (gray) will be deleted by an application of Reduction Rule 6.2.

6.3.1. An Exponential-Size Linear-Time Kernelization

We prove that HYPERBOLICITY can be solved in time linear in the size of the graph and exponential in the size k of a vertex cover. This result is based on a linear-time computable kernel consisting of $O(2^k)$ vertices. Note that there is a simple greedy linear-time algorithm computing a vertex cover of size at most twice the vertex cover number (see, e.g., [PS82]).

Proposition 6.6. *There is an algorithm that maps any instance of* HYPERBOLICITY *with n vertices, m edges, and vertex cover number k in $O(n+m)$ time to an equivalent instance of* HYPERBOLICITY *of size $2^{O(k)}$.*

Proposition 6.6 can be obtained by exhaustively applying the following data reduction rule (see Figure 6.3 for an illustration).

Reduction Rule 6.2. *If there are $\ell > 4$ vertices $v_1, \ldots, v_\ell \in V$ with the same (open) neighborhood $N(v_1) = N(v_2) = \ldots = N(v_\ell)$, then delete $v_5, \ldots, v_\ell$.*

We next show that the above reduction rule is correct, can be applied in linear time, and leads to a kernel for the parameter vertex cover number.

Lemma 6.7. *Reduction Rule 6.2 is correct and can be applied exhaustively in linear time. Furthermore, if Reduction Rule 6.2 is not applicable, then the graph contains at most $k + 4 \cdot 2^k$ vertices and $O(k \cdot 2^k)$ edges, where k is the vertex cover number.*

Proof. Let $G = (V, E)$ be the input graph with a vertex cover $W \subseteq V$ of size k and let $v_1, \ldots, v_\ell \in V$, $\ell > 4$, be vertices with the same open neighborhood.

First, we prove the correctness of Reduction Rule 6.2, that is, that $\delta(G[V \setminus \{v_5, \dots, v_\ell\}]) = \delta(G)$. Consider two vertices v_i, v_j with the same open neighborhood, and consider any other vertex u. The crucial observation is that $\overline{uv_i} = \overline{uv_j}$. This means that the two vertices are interchangeable with respect to the hyperbolicity. In particular, if $v_i, v_j \in V$ have the same open neighborhood, then $\delta(v_i, x, y, z) = \delta(v_j, x, y, z)$ for every $x, y, z \in V \setminus \{v_i, v_j\}$. As the hyperbolicity is obtained from a quadruple, it is sufficient to consider at most four vertices with the same open neighborhood. We conclude that $\delta(G[V \setminus \{v_5, \dots, v_\ell\}]) = \delta(G)$.

Next we show how to exhaustively apply Reduction Rule 6.2 in linear time. To this end, we apply in linear time a *partition refinement* [HP10] to compute a partition of the vertices into twin classes. Then, for each twin class we remove all but four (arbitrary) vertices. Overall, this can be done in linear time.

Since the size of the vertex cover $|W| \leq k$, it follows that there are at most 2^k pairwise-different neighborhoods (and thus twin classes) in the independent set $V \setminus W$. Thus, if Reduction Rule 6.2 is not applicable, then the graph consists of the vertex cover W of size k plus at most $4 \cdot 2^k$ vertices in $V \setminus W$. Furthermore, since W is a vertex cover, it follows that the graph contains at most $k^2 + 4k \cdot 2^k$ edges. □

With Reduction Rule 6.2 we can compute in linear time an equivalent instance having a bounded number of vertices. Applying to this instance the trivial $O(n^4)$-time algorithm yields the following.

Corollary 6.8. Hyperbolicity *can be computed in $O(2^{4k} + n + m)$ time, where k denotes the size of a vertex cover of the input graph.*

6.3.2. SETH-based Subquadratic-Time Lower Bounds

We show that, unless the SETH breaks, the $2^{O(k)} + O(n + m)$-time algorithm of Corollary 6.8 cannot be improved to an algorithm even with running time $2^{o(k)} \cdot n^{2-\varepsilon}$ for any $\varepsilon > 0$. This also implies that, assuming the SETH to hold, there is no kernel with $2^{o(k)}$ vertices computable in $O(n^{2-\varepsilon})$ time, that is, the kernel obtained by applying Reduction Rule 6.2 presumably cannot be improved asymptotically. The proof follows by a linear-time many-one reduction from the following problem:

Orthogonal Vectors

Input: Two sets $\vec{A}$ and $\vec{B}$ each containing n binary vectors of length $\ell = O(\log n)$.

Question: Are there two vectors $\vec{a} \in \vec{A}$ and $\vec{b} \in \vec{B}$ such that $\vec{a}$ and $\vec{b}$ are orthogonal, that is, such that there is no position i for which $\vec{a}[i] = \vec{b}[i] = 1$?

Williams and Yu [WY14] proved that the SETH breaks if Orthogonal Vectors can be solved in $O(n^{2-\varepsilon})$ time for some $\varepsilon > 0$. We provide a linear-time reduction from Orthogonal Vectors to Hyperbolicity where the graph G constructed in the reduction contains $O(n)$ vertices and admits a vertex cover of size $O(\log(n))$ (and thus contains $O(n \cdot \log(n))$ edges). The reduction then implies that, unless the SETH breaks, there is no algorithm solving Hyperbolicity in time polynomial in the vertex cover number and linear in the size of the graph. We mention that Borassi et al. [BCH16] showed that, assuming the SETH to hold, Hyperbolicity cannot be solved in $O(n^{2-\varepsilon})$ time for some $\varepsilon > 0$. The instances constructed in their reduction, however, have a minimum vertex cover of size $\Omega(n)$. Note that our reduction is based on ideas from the reduction of Abboud et al. [AVW16, Theorem 1.7] for the Diameter problem.

Proposition 6.9. *Unless the SETH breaks,* Hyperbolicity *cannot be solved in $2^{o(k)} \cdot n^{2-\varepsilon}$ time, even on graphs with $O(n\log(n))$ edges, diameter four, and domination number three. Here, k denotes the vertex cover number of the input graph.*

Construction 6.10. Let $(\vec{A}, \vec{B})$ be an instance of Orthogonal Vectors. We construct an instance (G, δ) of Hyperbolicity in linear time, where graph G is constructed as follows (we refer to Figure 6.4 for an illustration of the construction).

Make each $\vec{a} \in \vec{A}$ into a vertex a and each $\vec{b} \in \vec{B}$ into a vertex b of G, and add two vertices for each of the ℓ dimensions, that is, add to G the vertex sets

$$A \coloneqq \{a \mid \vec{a} \in \vec{A}\}, \qquad C \coloneqq \{c_1, \ldots, c_\ell\}, \text{ and}$$
$$B \coloneqq \{b \mid \vec{b} \in \vec{B}\}, \qquad D \coloneqq \{d_1, \ldots, d_\ell\},$$

and make each of C and D a clique. Next, connect each $a \in A$ to the vertices of C in the natural way, that is, add an edge between a and c_i if and only if $\vec{a}[i] = 1$. Similarly, add an edge between $b \in B$ and $d_i \in D$ if and only

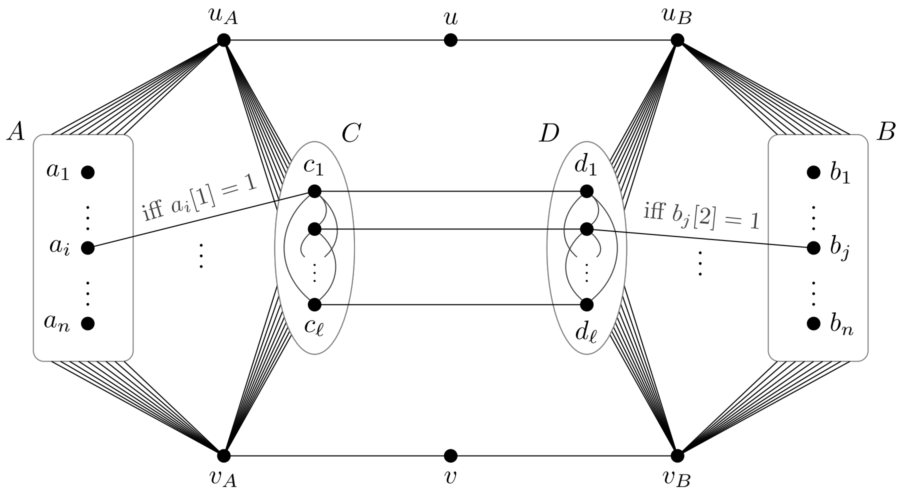


Figure 6.4.: Illustration of the construction described in the proof of Proposition 6.9. Ellipses indicate cliques, rectangles indicate independent sets. Multiple edges to an object indicate that the corresponding vertex is incident to each vertex enclosed within that object.

if $\vec{b}[i] = 1$. Moreover, add the edge set $\{\{c_i, d_i\} \mid i \in \{1, \ldots, \ell\}\}$. This part will constitute the central gadget of our construction.

Our aim is to ensure that the maximum hyperbolicity is reached for 4-tuples (a, b, c, d) such that $a \in A$, $b \in B$, and $\vec{a}$ and $\vec{b}$ are orthogonal vectors. The construction of G is completed by adding two paths (u_A, u, u_B) and (v_A, v, v_B), and making u_A and v_A adjacent to all vertices in $A \cup C$, and u_B and v_B adjacent to all vertices in $B \cup D$. ▲

We state two observations on the graph obtained from Construction 6.10. We first observe some structural properties of the graph. Recall that if every vertex of G is either contained in or adjacent to a vertex in a set X, then X forms a dominating set.

Observation 6.11. *Graph G obtained from Construction 6.10 (i) contains $O(n)$ vertices, $O(n \cdot \log(n))$ edges, (ii) has diameter four, (iii) has the dominating set $\{u_A, u_B, v\}$, and (iv) has the vertex cover $V \setminus (A \cup B)$ of size $O(\log(n))$.*

Our second observation is on specific distances in the graph regarding the vertices u, v and v_A, v_B, u_A, u_B.

Observation 6.12. *Let G be the graph obtained from Construction 6.10. Then (i) each vertex in $A \cup B \cup C \cup D$ is at distance two to each of u and v, (ii) v_A and v_B are at distance three to u, (iii) u_A and u_B are at distance three to v, and (iv) u and v are at distance four.*

Proof of Proposition 6.9. We reduce any instance $(\vec{A}, \vec{B})$ of Orthogonal Vectors to an instance (G, δ) of Hyperbolicity, where graph G is obtained from applying Construction 6.10.

By Observation 6.11, we know that G admits a vertex cover of size $O(\log(n))$. We complete the proof by showing that $(\vec{A}, \vec{B})$ is a yes-instance of Orthogonal Vectors if and only if G has hyperbolicity at least $\delta = 4$.

($\Rightarrow$) Let $(\vec{A}, \vec{B})$ be a `yes`-instance, and let $\vec{a} \in \vec{A}$ and $\vec{b} \in \vec{B}$ be a pair of orthogonal vectors. We claim that $\delta(a, b, u, v) = 4$. Since $\vec{a}$ and $\vec{b}$ are orthogonal, there is no $i \in \{1, \ldots, \ell\}$ with $\vec{a}[i] = \vec{b}[i] = 1$ and, hence, there is no path connecting a and b only containing two vertices in $C \cup D$, and it holds that $\overline{ab} = 4$. Moreover, we know that $\overline{uv} = 4$ and that $\overline{au} = \overline{bu} = \overline{av} = \overline{av} = 2$. Thus, $\delta(a, b, u, v) = 8 - 4 = 4$, and G is 4-hyperbolic.

($\Leftarrow$) Let $S = \{a, b, c, d\}$ be a set of vertices such that $\delta(a, b, c, d) \geq 4$. By Lemma 6.2, it follows that no two vertices of S are adjacent. Hence, we assume without loss of generality that $\overline{ab} = \overline{cd} = 4$. Observe that all vertices of C and D have distance at most three to all other vertices. Similarly, each vertex of $\{u_A, v_A, u_B, v_B\}$ has distance at most three to all other vertices. Consider for example u_A. By construction, u_A is a neighbor of all vertices in $A \cup C \cup \{u\}$ and, hence, u_A has distance at most two to v_A and to all vertices in D. Thus, u_A has distance at most three to v, B, u_B and v_B and therefore to all vertices of G. The arguments for v_A, u_B, and v_B are symmetric.

It follows that $S \subseteq A \cup B \cup \{u, v\}$, and therefore at least two vertices in S are from $A \cup B$. Thus, assume without loss of generality that a is contained in A. By the previous assumption, we have that $\overline{ab} = 4$. This implies that $b \in B$ and $\vec{a}$ and $\vec{b}$ are orthogonal vectors, as every other vertex in $V \setminus B$ is at distance three to a and each $b' \in B$ with $\vec{b'}$ being non-orthogonal to $\vec{a}$ is at distance three to a. Hence, $(\vec{A}, \vec{B})$ is a `yes`-instance. ☐

Remark 6.1. With Construction 6.10, the hardness also holds for the variants of Hyperbolicity in which we fix one vertex (u) or two vertices (u and v). The reduction also shows that approximating the hyperbolicity of a graph within a factor of $4/3 - \varepsilon$ cannot be done in strongly subquadratic time or in PL-FPT running time with respect to the vertex cover number.

Next, we adapt the above reduction to obtain the following hardness result on graphs of bounded maximum degree.

Theorem 6.13. *Unless the SETH breaks,* Hyperbolicity *cannot be solved in $f(\Delta) \cdot n^{2-\varepsilon}$ time, where Δ denotes the maximum degree of the input graph.*

We introduce the following notation that we will use in the proof of Theorem 6.13.

Definition 6.1 (Matching paths)**.** For two sets of vertices X and Y with $|X| = |Y|$, we say that we introduce *matching paths* if we connect the vertices in X with the vertices in Y with paths with no inner vertices from $X \cup Y$ such that for each $x \in X$, x is connected to exactly one $y \in Y$ via one path and for each $y \in Y$, y is connected to exactly one $x \in X$ via one path.

The following construction used in the proof of Theorem 6.13 is basically Construction 6.10 where several edges are replaced by binary trees, which are then connected via matching paths. The resulting graph will then have maximum degree five.

Construction 6.14. Let G' be the graph obtained from the graph from Construction 6.10 after deleting all edges. For each x_A, $x \in \{u, v\}$, add two binary trees, $T^A_{x_A}$ with n leaves and height at most $\lceil \log(n) \rceil$, and $T^C_{x_A}$ with ℓ leaves and height at most $\lceil \log(\ell) \rceil$. Connect each tree root by an edge with x_A. Next introduce matching paths between A and the leaves of $T^A_{x_A}$ such that each shortest path connecting a vertex in A with x_A is of length

$$h := 2(\lceil \log(n) \rceil + 1) + 1.$$

Similarly, introduce matching paths between C and the leaves of $T^C_{x_A}$ such that each shortest path connecting a vertex in C with x_A is of length h. Apply the same construction for x_B, $x \in \{u, v\}$, B, and D.

For $x \in A \cup B$, we denote by $|x|_1$ the number of 1's in the corresponding binary vector $\vec{x}$. Moreover, for $c_i \in C$, we denote by $|c_i|$ the number of vectors in A with a 1 as its ith entry. For $d_i \in D$, we denote by $|d_i|$ the number of vectors in B with a 1 as their ith entry.

For each vertex $a \in A$, add a binary tree with $|a|_1$ leaves and height at most $\lceil \log(|a|_1) \rceil$ and connect its root by an edge with a. For each $i \in \{1, \ldots, \ell\}$, add a binary tree with $|c_i|$ leaves and height at most $\lceil \log(|c_i|) \rceil$ and connect its root by an edge with c_i. Next, construct matching paths between the leaves of all binary trees introduced for the vertices in A on the one hand, and the leaves of all binary trees introduced for the vertices in C on the other hand, such that the following holds: (i) for each $a \in A$ and $c_i \in C$, there is a path only containing the vertices of the corresponding binary trees if and only if $\vec{a}[i] = 1$, and (ii) each of these paths is of length exactly h. Apply the same construction for B and D.

Next, for each $i \in \{1, \ldots, \ell\}$, add a binary tree with $\ell - 1$ leaves and height at most $\lceil \log(\ell - 1) \rceil$ and connect its root by an edge with c_i. Finally, add paths between the leaves of all binary trees introduced in this step such that (i) each leaf is incident to exactly one path, (ii) for each $i, j \in \{1, \ldots, \ell\}$, $i \neq j$, there is a path only containing the vertices of the corresponding binary trees, and (iii) each of these paths is of length exactly h. Apply the same construction for D.

Finally, for each $i \in \{1, \ldots, \ell\}$, connect c_i with d_i via a path of length h. Moreover, for $x \in \{u, v\}$, connect x_A with x and x with x_B each via a path of length h. This completes the construction of G.

Observe that the number of vertices in G is at most the number of vertices in the graph obtained from G' by replacing each edge with paths of length h. As G' contains $O(n \log(n))$ edges, the number of vertices in G is in $O(n \log^2(n))$. Finally, observe that the vertices in $C \cup D$ are the vertices of maximum degree which is five. ▲

Before we prove Theorem 6.13, we discuss some properties of G, the graph obtained from some instance $(\vec{A}, \vec{B})$ of Orthogonal Vectors using Construction 6.14.

Firstly, we discuss the distances of several vertices in G. Observe that u and v are at distance $4h$. For $x \in \{u, v\}$, the distance between x and x_A or x_B is h, and the distance between x_A and x_B is $2h$. The distance from any $c \in C$ to any $d \in D$ is at least h and at most $2h$. Moreover, the distance between any $a \in A$ and $b \in B$ is at least $3h$ and at most $4h$. We have the following

Lemma 6.15. *For any $a \in A$ and $b \in B$, $\overline{ab} = 4h$ if and only if $\vec{a}$ and $\vec{b}$ are orthogonal.*

Proof. ($\Leftarrow$) Let $\vec{a}$ and $\vec{b}$ be orthogonal. Suppose that there is a shortest path P between a and b of length smaller than $4h$. Observe that any shortest path between a and b containing u or v is of length $4h$. Hence, P contains vertices in $C \cup D$. As the shortest paths from a to C, C to D, and D to b are each of length h, the only shortest path containing vertices in $C \cup D$ of length smaller than $4h$ is of the form (a, c_i, d_i, b) for some $c_i \in C$ and $d_i \in D$ (recall that the shortest path between any two vertices in C or in D is of length h). Hence, $\vec{a}$ and $\vec{b}$ have both a 1 as their ith entry, and thus are not orthogonal. This contradicts the fact that $\vec{a}$ and $\vec{b}$ form a solution. It follows that $\overline{ab} = 4h$.

($\Rightarrow$) Let $\vec{a}$ and $\vec{b}$ be not orthogonal. Then there is an $i \in \{1, \dots, \ell\}$ such that $a[i] = b[i] = 1$. Hence, there is a path (a, c_i, d_i, b) of length $3h < 4h$. $\square$

Let P_x^Y denote the set of inner vertices of the shortest path connecting x and x_Y, for $x \in \{u, v\}$, $Y \in \{A, B\}$. Define

$$\begin{aligned} M &:= A \cup B \cup C \cup D \cup \{x, x_A, x_B \mid x \in \{u, v\}\} \text{ and} \\ M^* &:= \{p \in P_x^Y \mid x \in \{u, v\}, Y \in \{A, B\}\}. \end{aligned}$$

So far, we know that the only vertices that can be at distance $4h$ are those in $A \cup B \cup \{u, v\}$.

Lemma 6.16. *The vertex set $A \cup B \cup \{u, v\} \cup M^*$ is the only set containing vertices at distance $4h$. Moreover, G is of diameter $4h$.*

Proof. Consider any vertex $p \in V(G) \setminus M$. Then p is contained in a shortest path between two vertices x and y in M at distance h. Moreover, $\max\{\overline{px}, \overline{py}\} =: h' < h$.

We first discuss the case where $p \in M^*$. By symmetry, let $p \in P_u^A$. Observe that for $q \in P_v^B$ with $\overline{vq} = \overline{up}$ holds $\overline{pq} = 4h$.

Let $p \notin M \cup M^*$. We claim that for all vertices $q \in V(G)$ it holds that $\overline{pq} < 4h$. Suppose not, so that there is some $q \in V(G)$ with $\overline{pq} \geq 4h$. Observe that q is not contained in a shortest path between x and y. It follows that $\overline{xq} \geq 4h - h' > 3h$ or $\overline{yq} \geq 4h - h' > 3h$. Let $z \in \{x, y\}$ denote the vertex of minimal distance among the two, and let $\bar{z}$ denote the other one. Note that since h is odd, the distances to z and $\bar{z}$ are different.

Case 1: $q \in M$. Then $z, q \in A \cup B$, where z and q are not both contained in A or B. Recall that $p \notin M \cup M^*$ and, hence, the case $z, q \in \{u, v\}$ is not possible. By symmetry, assume $z \in A$ and $q \in B$. As $\overline{zq} > 3h$, it follows that $\bar{z} = c_i \in C$ for some $i \in \{1, \dots, \ell\}$ with $1 = \vec{z}[i] \neq \vec{q}[i]$, or $\bar{z} \in \{u_A, v_A\}$. Hence, the distance of $\bar{z}$ to q is at most the distance of z to q, contradicting the choice of z.

Case 2: $q \notin M$. Then q is contained in a shortest path between two vertices $x', y' \in M$ of length h. Moreover, $\max\{\overline{qx'}, \overline{qy'}\} =: h'' < h$. Consider a shortest path between p and q and notice that it must contain z and $z' \in \{x', y'\}$. It holds that $\overline{zz'} \geq 4h - h' - h'' > 2h$. By symmetry, assume $z \in A$, and $z' \in D \cup \{u_B, v_B\}$ (recall that $p \notin M \cup M^*$). Then $\bar{z}$ is in $C \cup \{u_A, v_A\}$, and hence of shorter distance to q, contradicting the choice of z.

We proved that $\overline{pq} < 4h$ for all $p \in V(G) \setminus (M \cup M^*)$, $q \in V(G)$. We conclude that the vertex set $A \cup B \cup \{u, v\} \cup M^*$ is the only set containing vertices at distance $4h$. Moreover, G is of diameter $4h$. □

We are set to prove our second main result.

Proof of Theorem 6.13. Let $(\vec{A}, \vec{B})$ be an instance of Orthogonal Vectors and let (G, δ) be the instance of Hyperbolicity obtained from $(\vec{A}, \vec{B})$ using Construction 6.14. We claim that $(\vec{A}, \vec{B})$ is a yes-instance of Orthogonal Vectors if and only if G has hyperbolicity at least $\delta = 4h$.

($\Rightarrow$) Let $\vec{a} \in \vec{A}$ and $\vec{b} \in \vec{B}$ be orthogonal. We claim that $\delta(a, b, u, v) = 4h$. Observe that $\overline{uv} = 4h$, and that $\overline{ab} = 4h$ by Lemma 6.15. The remaining distances are $2h$ by construction, and hence $\delta(G) = \delta(a, b, u, v) = 4h$.

($\Leftarrow$) Let $\delta(G) = 4h$ and let w, x, y, z be a quadruple with $\delta(w, x, y, z) = 4h$. By Lemma 6.3, we know that there are exactly two pairs of distance $4h$ and, hence, by Lemma 6.16 we have $\{w, x, y, z\} \subseteq A \cup B \cup \{u, v\} \cup M^*$. We claim that $|\{w, x, y, z\} \cap (M^* \cup \{u, v\})| \leq 2$. By Lemma 6.3, we know that, out of w, x, y, z, there are exactly two pairs at distance $4h$ and all other pairs have distance $2h$. Assume that $|\{w, x, y, z\} \cap (M^* \cup \{u, v\})| \geq 3$. Then, at least two vertices are in $P_v^A \cup P_v^B \cup \{v\}$ or in $P_u^A \cup P_u^B \cup \{u\}$. Observe that any two vertices in $P_v^A \cup P_v^B \cup \{v\}$ or in $P_u^A \cup P_u^B \cup \{u\}$ are at distance smaller than $2h$, but this contradicts the choice of the quadruple. It follows that $|\{w, x, y, z\} \cap (M^* \cup \{u, v\})| \leq 2$. We may thus assume without loss of generality that $w, x \in A \cup B$. As each vertex in A is at distance smaller than $3h$ to any vertex in $A \cup \{u, v\} \cup M^*$, it follows that the other vertex is in B. Applying Lemma 6.15, we have that w and x are at distance $4h$ if and only if $\vec{w}$ and $\vec{x}$ are orthogonal; hence, the statement of the theorem follows. □

We remark that Bentert et al. [Ben+19] define a parameterized problem to be *general problem hard* if, intuitively speaking, its unparameterized version is at least as hard as the problem for some constant values of the parameter. Hence in these terms, Proposition 6.9 and Theorem 6.13 prove Hyperbolicity to be general problem hard regarding the parameters diameter, minimum dominating set, and maximum degree.

6.4. Parameter-Polynomial Linear-Time Parameterized Algorithms

In this section, we provide parameter-polynomial linear-time parameterized algorithms with respect to the parameters *minimum maximal path cover number* (formally defined below), feedback edge number, and number of vertices with degree at least three. We first describe a parameter-polynomial linear-time parameterized algorithm for the minimum maximal path cover number. We then obtain parameter-polynomial linear-time parameterized algorithms for the other two parameters by proving that through specific data reduction rules, the number of maximal paths can be upper-bounded by a polynomial in the respective parameter value.

6.4.1. Minimum Maximal Path Cover Number

We first give the definition of maximal paths and then discuss graphs that can be covered by maximal paths.

Definition 6.2 (Maximal path)**.** Let G be a graph and P be a path in G. Then, P is a *maximal path* if the following hold: (1) P contains at least two vertices; (2) all inner vertices of P have degree two in G; (3) both endpoints of P have degree at least three in G.

The *minimum maximal path cover number* is the minimum number of maximal paths needed to cover the vertices of a given graph. A *pending cycle* in a graph is an induced cycle in G with at most one vertex of degree larger than two. A pending cycle with no vertex of degree larger than two is called *isolated*. While not all graphs can be covered by maximal paths (e.g., edgeless graphs), graphs which have minimum degree two and contain no pending cycles can be covered by maximal paths (this follows by, e.g., a greedy algorithm which iteratively starts a path with an arbitrary uncovered vertex and exhaustively extends it arbitrarily; since there are no isolated cycles and the minimum degree is two, we eventually hit at least one vertex of degree three). The next data reduction rule handles pending cycles.

Reduction Rule 6.3. *Let $\mathcal{I} = (G, \delta)$ be an instance of* Hyperbolicity *with $C \subseteq G$ being a pending cycle of G. If $\delta(C) > \delta$, then return that $\mathcal{I}$ is a* **no***-instance, and otherwise, delete from G all vertices $v \in V(C)$ with* $\deg(v) = 2$ *and their incident edges.*

The correctness of Reduction Rule 6.3 follows immediately from Lemma 6.4. We refer to a reduced graph G with no pending cycles as *cycle-reduced.* Based on Reduction Rule 6.3, we have the following.

Lemma 6.17. *There is a linear-time algorithm that given an instance $\mathcal{I} = (G, \delta)$ of* Hyperbolicity*, either decides $\mathcal{I}$ or computes an instance (G', δ) equivalent to $\mathcal{I}$ and a set $\mathcal{P}(G')$ such that $G' \subseteq G$ is cycle-reduced and $\mathcal{P}(G') \subseteq \mathcal{P}(G)$ is the set of all maximal paths in G' of length at least three.*

Proof. First, we apply Reduction Rule 6.1 to have a graph with no vertices of degree at most one. Next, we employ the linear-time algorithm by Bentert et al. [Ben+18, Lemma 6] towards computing the set of all pending cycles and the set of maximal paths. Herein, instead of storing the pending cycles, in each iteration of the algorithm where a pending cycle C_{4p+q} is found, we apply Reduction Rule 6.3. Note that for a cycle C_{4p+q} with $p \in \mathbb{N}$ and $q \in \{0, 1, 2, 3\}$ it holds true that $\delta(C_{4p+q}) = 2p - 1$ if $q = 1$, and $\delta(C_{4p+q}) = 2p$ otherwise [KM02]. If the deletion of the cycle causes a vertex to have degree one, then, starting at this vertex, we iteratively delete vertices of degree one along the path (as long as there are more than four vertices). Then we continue in the algorithm of Bentert et al. [Ben+18]. □

Based on the linear-time approximation algorithm given in the next lemma, we assume in the following that we are given a maximal path cover.

Lemma 6.18. *There is a linear-time 2-approximation algorithm for the minimum maximal path cover number for cycle-reduced graphs.*

Proof. The algorithm operates in two phases. In the first phase, we employ Lemma 6.17 to obtain set of all maximal paths of length at least two.

The second phase, when all vertices of degree two are already covered, ideally we would find a matching between those uncovered vertices of degree at least three. To get a 2-approximation we arbitrarily select a vertex of degree at least three, view it as a path of length zero, and arbitrarily extend it until it is maximal. This finishes the description of the linear-time algorithm.

For correctness of the first phase, the crucial observation is that each vertex of degree two has to be covered by at least one path. For the second phase, the factor two follows since each maximal path can cover at most two vertices of degree at least three. □

Now we are ready to present a parameter-polynomial linear-time parameterized algorithm for Hyperbolicity with respect to the minimum maximal path cover number.

Theorem 6.19. *Let $G = (V, E)$ be a cycle-reduced graph and k be its minimum maximal path cover number. Then,* Hyperbolicity *can be solved in $O(k(n+m)) + k^4(\log(n))^{O(1)}$ time.*

In the subsequent proofs, we denote for any path P with $u, v \in V(P)$ by $\overline{uv}|_P$ the distance of u to v on P.

Proof. We use Lemma 6.18 to get a set $\mathcal{P}$ of at most $2k$ maximal paths which cover G. By initiating a breadth-first search from each of the endpoints of those maximal paths, we can compute the pairwise distances between those endpoints in $O(k(n+m))$ time. Thus, for the rest of the algorithm we assume that we can access the distances between any two vertices which are endpoints of those maximal paths in constant time.

Let (a, b, c, d) be a quadruple such that $\delta(a, b, c, d) = \delta(G)$. Since the set $\mathcal{P}$ covers all vertices of G, each vertex of a, b, c, and d belongs to some path $P \in \mathcal{P}$. Since $|\mathcal{P}| \leq 2k$, there are $O(k^4)$ possibilities to assign the vertices a, b, c, and d to paths in $\mathcal{P}$. For each possibility we compute the maximum hyperbolicity respecting the assignment, that is, we compute the positions of the vertices on their respective paths that maximize $\delta(a, b, c, d)$. We achieve this by formulating an integer linear program (ILP) with a constant number of variables and constraints whose coefficients have value at most n.

To this end, denote by $P_x \in \mathcal{P}$ the path containing $x \in \{a, b, c, d\}$. We assume for now that these paths are different and discuss subsequently the case that one path contains at least two vertices from a, b, c, d. Let x_1 and x_2 be the endpoints of P_x for each $x \in \{a, b, c, d\}$. Furthermore, denote by m_P the length of a path $P \in \mathcal{P}$, that is, the number of its edges. Without loss of generality assume that $D_1 \geq D_2 \geq D_3$. We now compute the positions of the vertices on their respective paths that maximize $D_1 - D_2$ by solving an ILP. Recall that $\overline{v_1 v}|_{P_v}$ denotes the distance of v to v_1 on P_v. Thus, $\overline{v_1 v}|_{P_v} + \overline{v v_2}|_{P_v} = m_{P_v}$, and $\overline{v_1 v}|_{P_v} \geq 0$ and $\overline{v v_2}|_{P_v} \geq 0$. The following is a compressed description of the ILP containing the minimum function in some constraints. We describe hereafter how to remove it.

$$\begin{aligned}
\text{maximize:} \quad & D_1 - D_2 \\
\text{subject to:} \quad & D_1 = \overline{ab} + \overline{cd} \\
& D_2 = \overline{ac} + \overline{bd} \\
& D_3 = \overline{ad} + \overline{bc} \\
& D_1 \geq D_2 \geq D_3 \\
\forall x \in \{a,b,c,d\} : \quad & m_{P_x} = \overline{x_1 x}|_{P_x} + \overline{x_2 x}|_{P_x} && (6.1) \\
\forall x, y \in \{a,b,c,d\} : \quad & \overline{xy} = \min_{i,j \in \{1,2\}} (\overline{x_i x}|_{P_x} + \overline{x_i y_j} + \overline{y_j y}|_{P_y}) && (6.2)
\end{aligned}$$

First, observe that the ILP obviously has a constant number of variables: $\overline{xy}$, for all $x, y \in \{a,b,c,d\}$, $x \neq y$, $\overline{x_j x}|_{P_x}$, for all $x \in \{a,b,c,d\}$ and $j \in \{1,2\}$, and D_j, $j \in \{1,2,3\}$. The only constant coefficients are $\overline{x_i y_j}$ for $x, y \in \{a,b,c,d\}$ and $i, j \in \{1,2\}$ and obviously have value at most $n-1$. To remove the minimization function in (6.2), we use another case distinction: We simply try all possibilities of which value is the smallest one and adjust the ILP accordingly. For the case that for some $x, y \in \{a,b,c,d\}$ the minimum in (6.2) is $\overline{x_{i'} x}|_{P_x} + \overline{x_{i'} y_{j'}} + \overline{y_{j'} y}|_{P_y}$ with $i', j' \in \{1,2\}$, we replace this equation by the following:

$$\begin{aligned}
&\overline{xy} = \overline{x_{i'} x}|_{P_x} + \overline{x_{i'} y_{j'}} + \overline{y_{j'} y}|_{P_y} \\
&\overline{xy} \leq \overline{x_i x}|_{P_x} + \overline{x_i y_j} + \overline{y_j y}|_{P_y}, \qquad && i, j \in \{1,2\}, (i,j) \neq (i', j').
\end{aligned}$$

There are four possibilities of which value is the smallest one, and we have to consider each of them independently for each of the $\binom{4}{2} = 6$ pairs. Hence, for each assignment of the vertices a, b, c, and d to paths in $\mathcal{P}$, we need to solve $4 \cdot 6 = 24$ different ILPs in order to remove the minimization function. Since each ILP has a constant number of variables and constraints, solving them takes $L^{O(1)}$ time where $L = O(\log(n))$ is the total size of the ILP instance (e.g., by using an algorithm of Lenstra [Len83]).

It remains to discuss the case that at least two vertices of a, b, c, and d are assigned to the same path $P \in \mathcal{P}$. We show the changes exemplified for the case that a, b, and c are mapped to $P_a \in \mathcal{P}$. The adjustments for the other cases can be done in a similar fashion. We assume without loss of generality that the vertices a_1, a, b, c, a_2 appear in this order in P (allowing $a = a_1$ and $c = a_2$). The objective function as well as the first four lines of the ILP remain unchanged. Line (6.1) is replaced with the following:

$$\begin{aligned} m_{P_a} &= \overline{a_1 a}|_{P_a} + \overline{ab}|_{P_a} + \overline{bc}|_{P_a} + \overline{ca_2}|_{P_a} \\ m_{P_d} &= \overline{d_1 d}|_{P_d} + \overline{dd_2}|_{P_a} \end{aligned}$$

To ensure that (6.2) works as before, we add the following (recall that $P_a = P_b = P_c$):

$$\begin{aligned} \overline{aa_2}|_{P_a} &= \overline{ab}|_{P_a} + \overline{bc}|_{P_a} + \overline{ca_2}|_{P_a} \\ \overline{b_1 b}|_{P_b} &= \overline{a_1 a}|_{P_a} + \overline{ab}|_{P_a} \\ \overline{bb_2}|_{P_b} &= \overline{bc}|_{P_a} + \overline{ca_2}|_{P_a} \\ \overline{c_1 c}|_{P_c} &= \overline{a_1 a}|_{P_a} + \overline{ab}|_{P_a} + \overline{bc}|_{P_a} \\ \overline{cc_2}|_{P_c} &= \overline{ca_2}|_{P_a} \end{aligned}$$

□

Feedback Edge Number. We next present a parameter-polynomial linear-time parameterized algorithm with respect to the parameter feedback edge number k. The idea is to show that a graph that is cycle-reduced contains $O(k)$ maximal paths.

Theorem 6.20. Hyperbolicity *can be solved in* $O(k(n+m)) + k^4(\log(n))^{O(1)}$ *time, where* k *is the feedback edge number.*

Proof. The first step of the algorithm is to apply the algorithm of Lemma 6.17. After this step, if the input instance is not yet decided, we can assume our input graph to be cycle-reduced.

Denote by $X \subseteq E$ a minimum feedback edge set for the cycle-reduced graph $G = (V, E)$ and observe that $|X| \leq k$. We will show that the minimum maximal path cover number of G is in $O(k)$. More precisely, we show the slightly stronger claim that the number of maximal paths in G is in $O(k)$.

Observe that all vertices in G have degree at least two since G is cycle-reduced. Thus, every leaf of $G - X$ is incident with at least one edge in X, which implies that there are at most $2k$ leaves in $G - X$. Moreover, since $G - X$ is a forest, the number of vertices with degree at least three in $G - X$ is at most the number of leaves in $G - X$ and thus at most $2k$. This implies that the number of maximal paths in $G - X$ is at most $2k$ (each maximal path corresponds to an edge in the forest obtained from $G - X$ by contracting all degree-two vertices).

We now show that the number of maximal paths in G is linear in k by showing that an insertion of an edge into any graph H increases the number of maximal paths by at most three. First, note that each edge can be part of at most one maximal path in any graph. If each endpoint of the edge to insert is of degree two in H, then the number of maximal paths increases by three. In the case that at least one endpoint is of degree at least three or at most one the insertion increases the number of maximal paths by at most two. Thus G contains at most $5k$ maximal paths. The statement of the theorem now follows from Theorem 6.19. □

6.4.2. Number of Vertices with Degree at least Three

We show a parameter-polynomial linear-time parameterized algorithm with respect to the number k of vertices with degree at least three. To this end, we use the following data reduction rule additionally to the linear-time algorithm of Lemma 6.17 to upper-bound the number of maximal paths in the graph by $O(k^2)$ (with the goal to employ Theorem 6.19).

Reduction Rule 6.4. *Let $G = (V, E)$ be a graph, $u, v \in V_G^{\geq 3}$ be two vertices of degree at least three, and $\mathcal{P}_{uv}$ be the set of maximal paths in G with endpoints u and v. Let $\mathcal{P}_{uv}^9 \subseteq \mathcal{P}_{uv}$ be the set containing the shortest path, the four longest even-length paths, and the four longest odd-length paths in $\mathcal{P}_{uv}$. If $\mathcal{P}_{uv} \setminus \mathcal{P}_{uv}^9 \neq \emptyset$, then delete in G all inner vertices of the paths in $\mathcal{P}_{uv} \setminus \mathcal{P}_{uv}^9$.*

Lemma 6.21. *Reduction Rule 6.4 is correct and can be exhaustively applied in linear time.*

Proof. (Running time) In linear time we compute the set $V_G^{\geq 3}$ of all vertices with degree at least three. Then for each $v \in V_G^{\geq 3}$ we do the following. Starting from v, we perform a modified breadth-first search that stops at vertices in $V_G^{\geq 3}$. Let $R(V_G^{\geq 3}, v)$ denote the visited vertices and edges. Observe that $R(V_G^{\geq 3}, v)$ consists of v, some degree-two vertices, and all vertices of $V_G^{\geq 3}$ that can be reached from v via maximal paths in G. Furthermore, with the breadth-first search approach we can also compute for all $u \in R(V_G^{\geq 3}, v) \cap V_G^{\geq 3}$ with $u \neq v$ the number of maximal paths between u and v and their respective lengths. Then, in time linear in $|R(V_G^{\geq 3}, v)|$, we remove the paths in $\mathcal{P}_{uv} \setminus \mathcal{P}_{uv}^9$ for all $u \in$

$R(V_G^{\geq 3}, v) \cap V_G^{\geq 3}$. Thus, we can apply Reduction Rule 6.4 for each $v \in V_G^{\geq 3}$ in $O(|R(V_G^{\geq 3}, v)|)$ time. Altogether, the running time is in

$$O(\sum_{v \in V_G^{\geq 3}} |R(V_G^{\geq 3}, v)|) = O(n + m),$$

where the equality follows from the fact that each edge and each maximal path in G is visited twice by the modified breadth-first search.

(Correctness) Let $G = (V, E)$ be the input graph, let $P \in \mathcal{P}_{uv} \setminus \mathcal{P}_{uv}^9$ be a maximal path from u to v whose inner vertices are removed by the application of the data reduction rule, and let $G' = (V', E')$ be the resulting graph. We show that $\delta(G) = \delta(G')$. The correctness of Reduction Rule 6.4 follows then from iteratively applying this argument. First, observe that since $\mathcal{P}_{uv}^9$ contains the shortest maximal path of $\mathcal{P}_{uv}$, it follows that u and v have the same distance in G and in G'. Furthermore, it is easy to see that each pair of vertices $w, w' \in V'$ has the same distance in G and in G' (Reduction Rule 6.4 removes only paths and does not introduce degree-one vertices). Hence, we have that $\delta(G) \geq \delta(G')$ and it remains to show that $\delta(G) \leq \delta(G')$.

Let $a, b, c, d \in V$ be four vertices defining the hyperbolicity of G, that is, $\delta(G) = \delta(a, b, c, d)$. If P does not contain any of these four vertices as inner vertices, then we are done. Thus, assume that P *internally* contains at least one vertex from $\{a, b, c, d\}$. (We say in this proof that a path Q internally contains a vertex z if z is an inner vertex of Q.) We next distinguish four cases regarding the number of vertices of $\{a, b, c, d\}$ that are internally contained in P (we refer to Figure 6.5 for illustrations of the cases **I–III** and subcases therein).

Case I: P internally contains *one* vertex of $\{a, b, c, d\}$. Without loss of generality assume that P internally contains a. We show that we can replace a by another vertex a' in a path $P' \in \mathcal{P}_{uv}^9$ such that $\delta(a, b, c, d) = \delta(a', b, c, d)$. Since P internally contains a, we can choose P' as one of the four (odd/even)-length longest paths in $\mathcal{P}_{uv}^9$ such that

- $m_{P'} - m_P$ is non-negative and even (either both lengths are even or both are odd) and
- P' contains no vertex of $\{b, c, d\}$.

Since P is removed by Reduction Rule 6.4, it follows that $m_P \leq m_{P'}$. We choose a' on P' such that $\overline{ua'}|_{P'} = \overline{ua}|_P + (m_{P'} - m_P)/2$. Observe that this implies that $\overline{a'v}|_{P'} = \overline{av}|_P + (m_{P'} - m_P)/2$ and thus

$$\overline{ua}|_P - \overline{av}|_P = \overline{ua'}|_{P'} - \overline{a'v}|_{P'}.$$

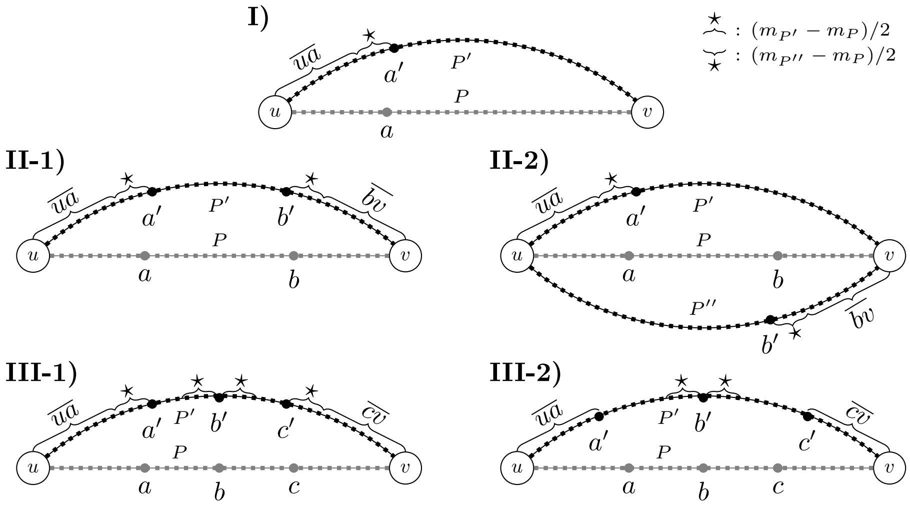


Figure 6.5.: Illustrations to the cases **I–III** and the subcases therein in the proof of Lemma 6.21.

Recall that

$$D_1 := \overline{ab} + \overline{cd}, \qquad D_2 := \overline{ac} + \overline{bd}, \text{ and} \qquad D_3 := \overline{ad} + \overline{bc}.$$

Denote with D_1', D_2', and D_3' the respective distance sums resulting from replacing a with a', for example $D_1' = \overline{a'b} + \overline{cd}$. Observe that by the choice of a' we increased all distance sums by the same amount, that is, for all $i \in \{1,2,3\}$ we have $D_i' = D_i + (m_{P'} - m_P)/2$. Since $\delta(a,b,c,d) = D_i - D_j$ for some $i, j \in \{1,2,3\}$, we have that

$$\delta(G') = \delta(a',b,c,d) = D_i' - D_j' = \delta(a,b,c,d) = \delta(G).$$

Case II: P internally contains *two* vertices of $\{a,b,c,d\}$. Without loss of generality, assume that P internally contains a and b but not c and d, and a is closer to u on P than b. We follow a similar pattern as in the previous case and again use the same notation. Let $P', P'' \in \mathcal{P}_{uv}^9$ be the two longest paths such that both P' and P'' do neither internally contain c nor d and both $m_{P'} - m_P$ and $m_{P''} - m_P$ are even. We distinguish two subcases.

Case II-1: D_1 is not the largest sum ($D_1 < D_2$ or $D_1 < D_3$). We replace a and b with a' and b' on P' such that $\overline{ua'}|_{P'} = \overline{ua}|_P + (m_{P'} - m_P)/2$ and $\overline{ub'}|_{P'} = \overline{ub}|_P + (m_{P'} - m_P)/2$. Thus, $D_1' = D_1$ since $\overline{ab} = \overline{a'b'}$. However, for $i \in \{2, 3\}$ we have $D_i' = D_i + (m_{P'} - m_P)/2$. Since either D_2 or D_3 was the largest distance sum, we obtain

$$\delta(G) = \delta(a, b, c, d) = D_i - D_j \leq D_i' - D_{j'}' = \delta(a', b', c, d) = \delta(G')$$

for some $i \in \{2, 3\}$, $j, j' \in \{1, 2, 3\}$, $i \neq j$, and $i \neq j'$.

Case II-2: D_1 is the largest sum ($D_1 \geq D_2$ and $D_1 \geq D_3$). We need another replacement strategy since we did not increase D_1 in case (**II-1**). In fact, we replace a and b with two vertices on different paths P' and P''. We replace a with a' on P' and b with b' on P'' such that $\overline{ua'}|_{P'} = \overline{ua}|_P + (m_{P'} - m_P)/2$ and $\overline{b'v}|_{P''} = \overline{bv}|_P + (m_{P''} - m_P)/2$. Observe that for $i \in \{2, 3\}$ it holds that

$$D_i' = D_i + (m_{P'} - m_P)/2 + (m_{P''} - m_P)/2.$$

Moreover, since a' and b' are on different maximal paths, we also have

$$\begin{aligned}\overline{ab} &\leq \min_{x \in \{u,v\}} \{\overline{xa}|_P + \overline{xb}|_P\} \\ &= \min_{x \in \{u,v\}} \{\overline{xa'}|_{P'} + \overline{xb'}|_{P''}\} - \frac{m_{P'} - m_P}{2} - \frac{m_{P''} - m_P}{2} \\ &= \overline{a'b'},\end{aligned}$$

where the last equality is due to the fact that $\{u, v\}$ forms an a'-b' separator in G'. Thus $D_1' \geq D_1 + (m_{P'} - m_P)/2 + (m_{P''} - m_P)/2$. It follows that

$$\delta(G) = \delta(a, b, c, d) = D_1 - D_j \leq D_1' - D_j' = \delta(a', b', c, d) = \delta(G')$$

for some $j \in \{2, 3\}$.

Case III: P internally contains *three* vertices of $\{a, b, c, d\}$. Without loss of generality, assume that P internally contains a, b, and c but not d and that among a, b, c vertex a is the closest vertex to u on P and c is the closest vertex to v on P (that is, a, b, c appear in this order on P). We distinguish two subcases.

Case III-1: $\overline{ac}|_P = \overline{ac}$. We follow a similar pattern as in case (**I**) and use the same notation. Again, there is a $P' \in \mathcal{P}_{uv}^9$ such that $m_{P'} - m_P$ is even and non-negative and P' does not contain d. We replace each vertex a, b, c as in case (**I**), that is, for each $x \in \{a,b,c\}$ we choose x' on P' such that $\overline{ux'}|_{P'} = \overline{ux}|_P + (m_{P'} - m_P)/2$. Observe that only the distances between d and the other three vertices change. Thus, we have again for all $i \in \{1,2,3\}$ that $D'_i = D_i + (m_{P'} - m_P)/2$ and hence $\delta(G) = \delta(G')$.

Case III-2: $\overline{ac}|_P > \overline{ac}$. We use again a similar strategy as in case (**I**) and use the same notation. Again, there is a $P' \in \mathcal{P}^9_{uv}$ such that $m_{P'} - m_P$ is even and non-negative and P' does not contain d. We replace the vertices a, b, c with a', b', c' on P' such that

- $\overline{au} = \overline{au}|_P = \overline{a'u}|_{P'} = \overline{a'u}$,
- $\overline{cv} = \overline{cv}|_P = \overline{c'v}|_{P'} = \overline{c'v}$,
- $\overline{b'u}|_{P'} = \overline{bu}|_P + (m_{P'} - m_P)/2$, and
- $\overline{b'v}|_{P'} = \overline{bv}|_P + (m_{P'} - m_P)/2$.

Note that since $\overline{ac}|_P > \overline{ac}$, it follows that the distances not involving b (respectively b') remain unchanged, that is, $\overline{ac} = \overline{a'c'}$, $\overline{ad} = \overline{a'd}$, and $\overline{cd} = \overline{c'd}$. Furthermore, all distances involving b (respectively b') increase by $(m_{P'} - m_P)/2$, that is, $\overline{bx} = \overline{b'x} - (m_{P'} - m_P)/2$ for each $x \in \{a,c,d\}$. Thus, we have again for all $i \in \{1,2,3\}$ that $D'_i = D_i + (m_{P'} - m_P)/2$ and hence $\delta(G) = \delta(G')$.

Case IV: P internally contains all *four* vertices of $\{a,b,c,d\}$. We consider two subcases.

Case IV-1: the union of the shortest paths between these four vertices induces a path. In this case, we have $\delta(G) = 0$ and thus trivially $\delta(G) \leq \delta(G')$.

Case IV-2: the union of the shortest paths between these four vertices induces a cycle C. As before, there is a path $P' \in \mathcal{P}^9_{uv}$ such that $m_{P'} - m_P$ is non-negative and even. Let Q denote the shortest path on C between u and v. Observe that Q contains no vertex in $\{a,b,c,d\}$ and is present in G'. Denote by C' the cycle formed by Q and P'. Note that $|C'| \geq |C|$ since $m_{P'} \geq m_P$. Moreover, $(|C'| - |C|) \bmod 2 = 0$ since $(m_{P'} - m_P) \bmod 2 = 0$, and hence the cases $C = C_{4p}$ and $C' = C_{4p+1}$ for some $p \in \mathbb{N}$ are excluded. Thus, it holds true that $\delta(C') \geq \delta(C)$. It follows that $\delta(G') \geq \delta(C') \geq \delta(C) = \delta(a,b,c,d) = \delta(G)$. □

Observe that if the graph G is reduced with respect to Reduction Rule 6.4 after Lemma 6.17 was applied, then for each pair $u, v \in V_G^{\geq 3}$ there exist at most nine maximal paths with endpoints u and v. Thus, G contains at most $O(k^2)$ maximal paths. Employing Theorem 6.19 we arrive at the following.

Theorem 6.22. Hyperbolicity *can be solved in* $O(k^2(n+m))+k^8(\log(n))^{O(1)}$ *time, where* k *is the number of vertices with degree at least three.*

6.5. Concluding Remarks

We proved Hyperbolicity parameterized by the vertex cover number to admit a (single-)exponential-size linear-time kernel, but no subexponential-size truly-subquadratic-time kernel unless the SETH breaks. The latter result is derived from a running time lower bound, which implies no kernelizations in linear time of any polynomial size. We wonder what is possible in quadratic time:

Open Problem 9. Does Hyperbolicity admit a problem kernel computable in quadratic time of size polynomial (or subexponential) in the vertex cover number?

We proved PL-FPT algorithms for Hyperbolicity regarding three parameters each being incomparable to the vertex cover number. Each of these algorithms bases on data reduction in the parameterized algorithmic sense, supporting the importance of this kind of preprocessing also for polynomial-time solvable problem.

Finally, Williams et al.'s [Vas+15] conjectured $\Omega(n^3)$-time lower bound for 4-Independent Set transfers to Hyperbolicity [Flu+19a]. Challenging the conjecture, we pose the following:

Open Problem 10. Is Hyperbolicity solvable in truly subcubic time?

Part III.

Losing Weights and Secluded Problems

In a given graph, finding some subgraph fulfilling some property is a basic algorithmic task. Motivated by security applications, one may wish to have only few vertices with direct access to the subgraph (that is, neighboring the vertices of the subgraph). When asking for the closed neighborhood to be small, this problem modification is known as *secluded*, coined by Chechik et al. [Che+17] for the modification of the problems of finding a two-terminal path, that is, a path connecting two terminals (Secluded Path), and of finding a Steiner tree (Secluded Steiner Tree). While Chechik et al. [Che+17] obtained complexity classification and approximation results for the two mentioned problems, Fomin et al. [Fom+17a] studied the two problems from a parameterized complexity perspective. Interestingly, while a shortest two-terminal path in a given graph can be found in polynomial time, Secluded Path is not only NP-hard, but also admits no kernel of size polynomial in the vertex cover number (unless $\text{coNP} \subseteq \text{NP}_{/\text{poly}}$). So, the following question arises: How do the computational and parameterized complexity—in particular, the kernelizability—of well-known graph problems like finding short paths, small separators, or small feedback vertex sets, change when one additionally seeks to restrict the neighborhood size of their solution vertex sets?

Asking for the closed neighborhood of a vertex set to be small gives no control on the size of the set or of its open neighborhood: finding a two-terminal path with small closed neighborhood may allow for a long path with small open neighborhood, or for a short path with large open neighborhood. By this observation, our first modification to the secluded setup yields the "small secluded" setup, where we demand the size of both the vertex set and its *open* neighborhood to be small. Chapter 8 is devoted to the Small Secluded Path problem, where kernelization lower and upper bounds regarding several (combined) parameters are provided. In Chapter 9, we elaborate more on the difference between the non-secluded, secluded, and small secluded setup. To this end, we apply the setups to the examples of finding two-terminal separators and feedback vertex sets.

In our study of Small Secluded Path, it turned out that the following strategy leads to efficient and effective preprocessing: We first shrink the number of vertices and edges of the graph in our input instance while introducing vertex weights. This hence gives us an instance of a vertex-weighted problem. We then, secondly, employ a technique of Frank and Tardos [FT87] to shrink these introduced vertex weights. Lastly, we employ that the class NP is closed under polynomial-time many-one reductions and map our vertex-weighted instance back to an instance without vertex weights, that is, of our original problem.

Our strategy hence uses the technique of Frank and Tardos [FT87] to obtain a polynomial kernelization for an unweighted problem. This, together with the work by Etscheid et al. [Ets+17] and Marx and Végh [MV15], underlines the applicability of the “losing-weight technique” of Frank and Tardos [FT87] for polynomial kernelization.

In Chapter 7, we describe the technique of Frank and Tardos [FT87]. Moreover, we show that the technique applies also for problems where the goal function is, at first glance, not linear, and hence enlarge the application domain of the technique.

Chapter 7.

Losing Weight for Polynomial Kernelization

In this chapter, we explain the technique of “losing weight” due to Frank and Tardos [FT87], discuss its role in parameterized algorithmics for obtaining kernels of polynomial size, and prove that it not only applies to problems with linear goal functions, but also to those with goal functions that are, at least at first glance, not linear.

7.1. Introduction

In the early eighties, Grötschel et al. [GLS81] employed the famous ellipsoid method by Khachiyan [Kha79, Kha80] for the Weighted Independent Set (WIS) problem: Given an undirected graph $G = (V, E)$ with vertex weights $w = (w(v))_{v \in V} \in \mathbb{Q}^{|V|}$, find a set $U \subseteq V$ such that U is an independent set and maximizes $\sum_{v \in U} w(v)$. Grötschel et al. [GLS81] proved WIS to be solvable in polynomial time on perfect graphs. The running time of their algorithm, however, depends on the length of (the encoding of) the maximum vertex-weight in the input. Hence, one may wonder: Is WIS on perfect graphs solvable in polynomial time where the running time is independent of the maximum vertex-weight in the input?[1]

Frank and Tardos [FT87] answered this question in the affirmative by proving a *losing-weight technique*. The technique employs a “preprocessing algorithm” that, exemplified for WIS, does the following: Compute in time polynomial in

[1]The question can be restated as follows: Given the *weakly* polynomial time algorithm of Grötschel et al. [GLS81], is WIS on perfect graphs solvable in *strongly* polynomial time?

the number n of graph vertices and (the encoding length of) the input maximum vertex-weight some vertex weights $\widehat{w}$ with small entries, that is,

(a) where the length of (the encoding of) the maximum entry in $\widehat{w}$ is polynomially upper-bounded in n,

such that the quality of all solutions and non-solutions is preserved, that is,

(b) for every two (independent) sets $U, U' \subseteq V$ we have that $\sum_{v \in U} w(v) \geq \sum_{v \in U'} w(v)$ if and only if $\sum_{v \in U} \widehat{w}(v) \geq \sum_{v \in U'} \widehat{w}(v)$.

Hence, when first applying the losing-weight technique and then the algorithm of Grötschel et al. [GLS81], WIS on perfect graphs is solved in time independent of the input maximum vertex-weight. Notably, the preprocessing algorithm makes use of the simultaneous Diophantine approximation algorithm due to Lenstra et al. [LLL82].

To the best of our knowledge, Frank and Tardos' technique appeared in parameterized algorithmics the first time in the work of Fellows et al. [Fel+08]. However, Fellows et al. employed the technique to obtain fixed-parameter algorithms running in polynomial space. Marx and Végh [MV15] firstly observed the connection of the losing-weight technique with polynomial kernelization. They proved a polynomial kernel utilizing the losing-weight technique for the Minimum-Cost Edge-Connectivity Augmentation by One problem, where, given an undirected, $(k-1)$-edge-connected graph $G = (V, E)$, edge set E^*, two weight functions $w : E^* \to \mathbb{N}$ and $c : E^* \to \mathbb{R}_+ \cup \{+\infty\}$, and $k, p \in \mathbb{N}$, the task is to find a set $F \subseteq E^*$ with $\sum_{e \in F} w(e) \leq p$ such that the graph $(V, E \cup F)$ is k-edge-connected and $\sum_{e \in F} c(e)$ is minimum. Interestingly, their kernelization first increases the size of the instance and introduces additional edge weights. Marx and Végh state about the technique of employing the losing-weight technique that

> *"[...] this technique seems to be an essential tool for kernelization of problems involving costs."*

Subsequently, Etscheid et al. [Ets+17] proved polynomial kernels for several weighted problems using the losing-weight technique, supporting Marx and Végh's statement by making the losing-weight technique an important tool for obtaining polynomial kernels for weighted problems. Notably, in Chapter 8 we apply the losing-weight technique to kernelize an *unweighted* problem. While such an application was seemingly not done before, our underlying approach of introducing weights is close to Marx and Végh's approach.

Our Contributions. We give a brief introduction into the losing-weight technique for polynomial kernelization. Moreover, we show that the technique applies to obtain polynomial kernels for problems with goal functions not linear in the solution size (note that for the three problems mentioned above, the goal function is linear in the solution size). To this end, we introduce the notion of *linearizable* functions.

7.2. The Losing-Weight Technique

The preprocessing algorithm of Frank and Tardos [FT87] provides the following central result of this chapter and key tool in the subsequent Chapter 8 (sign denotes the signum function, see Definition 1.11).

Proposition 7.1 ([FT87, Section 3])**.** *There is an algorithm that, on input $w \in \mathbb{Q}^d$ and integer N, computes in polynomial time a vector $\widehat{w} \in \mathbb{Z}^d$ with*
(i) $\|\widehat{w}\|_\infty \leq 2^{4d^3} N^{d(d+2)}$ *such that*
(ii) $\operatorname{sign}(w^\top b) = \operatorname{sign}(\widehat{w}^\top b)$ *for all $b \in \mathbb{Z}^d$ with $\|b\|_1 \leq N - 1$.*

Recall our description of the losing-weight technique in Section 7.1 on the example of WIS. We briefly explain how (a) and (b) from the description relate to Proposition 7.1. While the correspondence between (a) and Proposition 7.1(i) is immediate, that (b) corresponds to Proposition 7.1(ii) is less obvious: Let $u = (u(v))_{v\in V}, u' = (u'(v))_{v\in V} \in \{0,1\}^{|V|}$ be the vectors representing the sets U and U' respectively, that is, $u(v) = 1$ if and only if $v \in U$ (analogously for u' and U'). Then $\sum_{v\in U} w(v) = u^\top w$ and $\sum_{v\in U'} w(v) = u'^\top w$. By this, observe that the "if-and-only-if" statement in (b) can be rewritten as $u^\top w - u'^\top w \geq 0 \iff u^\top \widehat{w} - u'^\top \widehat{w} \geq 0$. With $b := u - u'$ (note that $\|b\|_1 \leq 2|V|$) the correspondence to Proposition 7.1(ii) now becomes clear.

Next, we give some first observations on and an example application of Proposition 7.1. First observe that the signum of weights is maintained.

Observation 7.2. *For $N \geq 2$, Proposition 7.1 gives* $\operatorname{sign}(w^\top \vec{e}_i) = \operatorname{sign}(\bar{w}^\top \vec{e}_i)$ *for each $i \in \{1, \ldots, d\}$, where $\vec{e}_i \in \mathbb{Z}^d$ is the vector that has 1 in the i-th entry and zeroes in the others. Thus, one has* $\operatorname{sign}(w_i) = \operatorname{sign}(\bar{w}_i)$ *for each $i \in \{1, \ldots, d\}$.*

Moreover, observe that the order relation between the weights is also maintained.

Observation 7.3. *For $N \geq 3$, Proposition 7.1 gives* $\operatorname{sign}(w^\top(\vec{e}_i - \vec{e}_j)) = \operatorname{sign}(\widehat{w}^\top(\vec{e}_i - \vec{e}_j))$ *for each $i, j \in \{1, \dots, d\}$. Thus, one has $w_i - w_j \geq 0 \iff \widehat{w}_i - \widehat{w}_j \geq 0$ for each $i \in \{1, \dots, d\}$.*

One may wonder whether Proposition 7.1 also works for decision rather than optimization problems. Indeed, the application to decision problems is a direct corollary, first stated by Marx and Végh [MV15] and then formalized by Etscheid et al. [Ets+17], by observing that the value given along in the description of the decision problem can be "attached" to the weight vector.

Corollary 7.4 ([Ets+17])**.** *There is an algorithm that, on input $(w, k) \in \mathbb{Q}^{d+1}$ with $w \in \mathbb{Q}^d$ and integer N, computes in polynomial time a vector $(\widehat{w}, \bar{k}) \in \mathbb{Z}^{d+1}$ with $\widehat{w} \in \mathbb{Z}^d$ and*

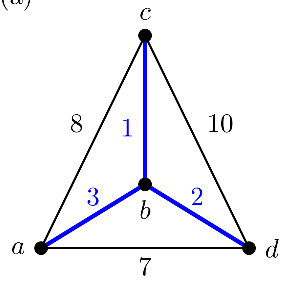
(i) $\|\widehat{w}\|_\infty, |\bar{k}| \leq 2^{4(d+1)^3} N^{(d+1)(d+3)}$ *such that*
(ii) $\operatorname{sign}(w^\top b - k) = \operatorname{sign}(\widehat{w}^\top b - \bar{k})$ *for all $b \in \mathbb{Z}^d$ with $\|b\|_1 \leq N - 2$.*

Moreover, whenever we are facing a weighted problem with a linear goal function, that is, for example finding some set S such that $\sum_{s \in S} w(s)$ is minimized (or maximized), the application of Proposition 7.1 is often immediate. So it is for the well-known Knapsack problem, as first proven by Etscheid et al. [Ets+17], yet solving a long-standing open question for this problem [Cyg+14].

Example 7.1. Recall the Knapsack problem: Given a set $X = \{1, \dots, n\}$ of n items with weights $w = (w_i)_{i \in \{1,\dots,n\}} \in \mathbb{Q}^n$ and values $v = (v_i)_{i \in \{1,\dots,n\}} \in \mathbb{Q}^n$, and rational numbers $k, \ell \in \mathbb{Q}$, the question is whether there is a subset $S \subseteq X$ of items such that $\sum_{i \in S} w_i \leq k$ and $\sum_{i \in S} v_i \geq \ell$. A direct application of Corollary 7.4 to each of (w, k) and (v, p) with $d = n$ and $N = n + 2$ yields a problem kernel of size polynomial in n. $\triangleleft$

One may wonder whether an application as outlined in Example 7.1 works also for goal functions that are stated in a non-linear way. Recall that b in Proposition 7.1 represents all solution candidates. However, if your goal function is not immediately of the form $w^\top b$, an application of Proposition 7.1 is not directly clear. Yet, we prove an application for some problems with goal functions stated in a non-linear way in the next two sections.

(a)



(b)

	a,b	a,c	a,d	b,c	b,d	c,d
$\vec{w}$	3	8	7	1	2	10
a	**1**	1	1	0	0	0
b	**1**	0	0	1	1	0
c	0	1	0	**1**	0	1
d	0	0	1	0	**1**	1
$\vec{x} :=$	**2**	0	0	**1**	**1**	0

Figure 7.1.: Illustrative example for MIN-POWER SYMMETRIC CONNECTIVITY and the application of Proposition 7.1. (a) depicts an edge-weighted undirected example graph, and (b) shows its incidence matrix (x, y is short for edge $\{x, y\}$), the vector $\vec{w}$ of edge-weights, and the vector $\vec{x}$ representing the solution indicated by thick edges in (a).

7.3. The Case of MIN-POWER SYMMETRIC CONNECTIVITY

Consider the following NP-hard optimization problem from survivable network design [Alt+06, CPS04].

MIN-POWER SYMMETRIC CONNECTIVITY (MIPOSYCO)
Input: A connected undirected graph $G = (V, E)$ and edge weights $w\colon E \to \mathbb{N}$.
Task: Find a connected spanning subgraph $T = (V, F)$ of G that minimizes

$$\sum_{v \in V} \max_{\{u,v\} \in F} w(\{u, v\}). \tag{7.1}$$

The goal function (7.1) is, at first sight, not linear in F in the following sense: Let $E = \{e_1, \ldots, e_m\}$ be enumerated and the weight w be represented as a vector $\vec{w} \in \mathbb{N}^m$ such that $\vec{w}_i = w(e_i)$. Let $b \in \{0, 1\}^m$ be the vector representing the edge set F of a solution $T = (V, F)$, that is, $b_i = 1$ if and only if $e_i \in F$. Then, the value $\vec{w}^\top b$ is *not* equal to $\sum_{v \in V} \max_{\{u,v\} \in F} w(\{u, v\})$. See Figure 7.1(a) for an example.

However, we can circumvent this issue (arising from the max-function in the goal function) and still apply Proposition 7.1. To this end, observe that we only need to change the representation of a solution. An edge $e \in F$ contributes its weight to (7.1) each time a vertex v incident to e "pays" for e, that is, e is of maximum weight among the edges in F for v. Hence, a solution can be represented as vector in $b \in \{0,1,2\}^m$, with $b_i = x$ if $x \in \{0,1,2\}$ of its incident vertices declare e_i to be the edge they pay the weight for, that is, which is incident and of maximum weight to them in the solution $T = (V,F)$. See Figure 7.1(b) for an example. Notably, this change of the representation of a solution only changes the domain of the vector b, and hence the value of N in the application of Proposition 7.1 by a factor of two. Eventually, we obtain the following.

Lemma 7.5. *There is an algorithm that, on any input instance $(G = (V,E), w)$ of* MiPoSyCo *with $m := |E|$, computes in time polynomial in $|(G,w)|$ an instance $(G, \widehat{w})$ of* MiPoSyCo *such that*

(i) $\|\widehat{w}\|_\infty \leq 2^{4m^3} \cdot (2m+2)^{m(m+2)}$ *and*

(ii) a connected subgraph $T = (V,F)$ of G is an optimal solution for (G,w) if and only if T is an optimal solution for $(G,\widehat{w})$.

Proof. Without loss of generality, we consider the edges of G as enumerated $E = \{e_1,\dots,e_m\}$ and the weight functions $w, \widehat{w}$ as (column) vectors in $\mathbb{N}^m$ such that $w_i = w(e_i)$ and $\widehat{w}_i = \widehat{w}(e_i)$ for all $i \in \{1,\dots,m\}$. We apply Proposition 7.1 with $d = m$ and $N = 2m+1$ to the weight vector w to obtain the weight vector $\widehat{w}$. From Proposition 7.1 immediately follows (i), that is, $\|\widehat{w}\|_\infty \leq 2^{4m^3} \cdot (2m+1)^{m(m+2)}$. Moreover, recall that $\widehat{w}_i > 0$ for all $i \in \{1,\dots,m\}$ due to Observation 7.2. Next, we prove that also (ii) holds true, that is, a connected graph $T = (V,F)$ is an optimal solution for (G,w) if and only if T is an optimal solution for $(G,\widehat{w})$.

Let $T = (V,F)$ be a connected subgraph of G and let $\phi_T \colon V \to F$ be a mapping such that $\phi_T(v) \in \arg\max_{\{u,v\}\in F} w(\{u,v\})$ for all $v \in V$. Observe that $\sum_{v\in V} \max_{\{u,v\}\in F} w(\{u,v\}) = \sum_{v \in V} w(\phi_T(v))$. Let vector $s \in \{0,1,2\}^m$ represent for each edge e the number of its endpoints mapped by ϕ to e. Formally, for each $i \in \{1,\dots,m\}$ we have

$$s_i = |\{v \in e_i \mid \phi(v) = e_i\}|.$$

For a connected subgraph $T' = (V,F')$ of G, let $\phi_{T'}$ and $s' \in \{0,1,2\}^m$ be derived analogously. Note that the cost of T and T' is $s^\top w$ and $(s')^\top w$, respectively. Define $b := s - s'$. Note that for each $i \in \{1, \dots, m\}$ it holds true that $-2 \leq b_i \leq 2$, and hence $\|b\|_1 \leq 2m = N-1$. Moreover, from Proposition 7.1 we have $\operatorname{sign}(b^\top w) = \operatorname{sign}(b^\top \widehat{w})$, or, equivalently,

$$\begin{aligned} s^\top w - (s')^\top w \leq 0 &\iff (s - s')^\top w \leq 0 \\ &\overset{\text{Prop. 7.1}}{\iff} (s - s')^\top \widehat{w} \leq 0 \iff s^\top \widehat{w} - (s')^\top \widehat{w} \leq 0. \end{aligned}$$

Finally, note that due to Observation 7.3, both T and T' are still correctly represented by s and s' given $\widehat{w}$, that is,

$$\begin{aligned} \sum_{v \in V} \max_{\{u,v\} \in F} \widehat{w}(\{u,v\}) &= \sum_{v \in V} \widehat{w}(\phi_T(v)) = s^\top \widehat{w}, \text{and} \\ \sum_{v \in V} \max_{\{u,v\} \in F'} \widehat{w}(\{u,v\}) &= \sum_{v \in V} \widehat{w}(\phi_{T'}(v)) = s'^\top \widehat{w} \end{aligned}$$ □

Remark 7.1. We can easily adapt Lemma 7.5 to shrink weights for an instance (G, w, k) of the decision variant of MiPoSyCo by employing Corollary 7.4.

Bentert et al. [Ben+17a] proved a polynomial-time algorithm that maps any instance of MiPoSyCo to an instance where the number of vertices and edges is linear in the feedback edge number fes, but (the encoding lengths of) the weights in the obtained instance are not necessarily upper-bounded (polynomially) in fes. Combining their result with Lemma 7.5 yields the following.

Corollary 7.6. MiPoSyCo *admits a kernel of size polynomial in the feedback edge number of the input graph.*

7.4. The Case of Small Set Expansion

Next, consider the following optimization problem [Ban+14, RS10].

Small Set Expansion (SSE)
Input: An undirected graph G with edge weights $w : E(G) \to \mathbb{Q}_+$.
Question: Find a non-empty subset $S \subseteq V(G)$ of size at most $|S| \leq n/2$ that minimizes

$$\frac{1}{|S|} \sum_{e \in (S, V(G) \setminus S)} w(e), \tag{7.2}$$

where $(S, V(G) \setminus S)$ denotes the set of all edges with exactly one endpoint in S.

Observe that the goal function (7.2) is not linear in a solution vertex set. However, the value of interest for a vertex set S can be represented by $w^\top s$ for a *fractional* vector $s \in \{0, \frac{1}{|S|}\}^{|E(G)|}$, where s is different to zero if and only if the corresponding edge is in the edge cut $(S, V(G) \setminus S)$. Yet, fractional numbers are not captured by Proposition 7.1. However, we can derive the following where we define

$$\mathbb{Q}_r := \{\frac{p}{q} \mid |p|, q \in \{0, \dots, r\}, q \neq 0\}. \tag{7.3}$$

Proposition 7.7. *There is an algorithm that, on input $w \in \mathbb{Q}^d$ and integer $r \in \mathbb{N}$, computes in polynomial time a vector $\widehat{w} \in \mathbb{Z}^d$ with*
(i) $\|\widehat{w}\|_\infty \leq 2^{4d^3}(r^2 \cdot d + 1)^{r \cdot d(d+2)}$ *such that*
(ii) $\operatorname{sign}(w^\top b) = \operatorname{sign}(\widehat{w}^\top b)$ *for all $b \in \mathbb{Q}_r^d$.*

Proof. Apply Proposition 7.1 with $N = r! \cdot r \cdot d + 1$ to obtain a vector $\widehat{w} \in \mathbb{Z}^d$ with

$$\|\widehat{w}\|_\infty \leq 2^{4d^3} N^{d(d+2)} = 2^{4d^3}(r! \cdot r \cdot d + 1)^{d(d+2)} \leq 2^{4d^3}(r^2 \cdot d + 1)^{r \cdot d(d+2)}$$

such that $\operatorname{sign}(w^\top b) = \operatorname{sign}(\widehat{w}^\top b)$ for all $b \in \mathbb{Z}^d$ with $\|b\|_1 \leq N - 1$. Let $b^* \in \mathbb{Q}_r^d$. We have

$$\begin{aligned} \operatorname{sign}(w^\top b^*) = \operatorname{sign}(\widehat{w}^\top b^*) &\iff r! \cdot \operatorname{sign}(w^\top b^*) = r! \cdot \operatorname{sign}(\widehat{w}^\top b^*) \\ &\iff \operatorname{sign}(w^\top (r! \cdot b^*)) = \operatorname{sign}(\widehat{w}^\top (r! \cdot b^*)) \\ &\iff \operatorname{sign}(w^\top b') = \operatorname{sign}(\widehat{w}^\top b'), \end{aligned}$$

where for $b' = r! \cdot b^*$ holds true that $b' \in \mathbb{Z}^d$ and $\|b'\|_1 \leq r! \cdot r \cdot d = N - 1$. □

Now, with Proposition 7.7, we get the following.

Lemma 7.8. *There is an algorithm that, on any input instance $(G = (V, E), w)$ of* SSE *with $n := |V|$ and $m := |E|$, computes in time polynomial in $|(G, w)|$ an instance $(G, \widehat{w})$ of* SSE *with $\widehat{w} \in \mathbb{N}^m$ such that*

(i) $\|\widehat{w}\|_\infty \leq 2^{4m^3} \cdot (n^4 \cdot m + 1)^{n^2 m(m+2)}$ *and*

(ii) *a set $S \subseteq V$ is an optimal solution for (G, w) if and only if S is an optimal solution for $(G, \widehat{w})$.*

Proof. Without loss of generality, we consider the edges of G as enumerated $E = \{e_1, \ldots, e_m\}$ and the weight functions w and $\widehat{w}$ as (column) vectors in $\mathbb{Q}_+^m$ and in $\mathbb{N}^m$, respectively, such that $w_i = w(e_i)$ and $\widehat{w}_i = \widehat{w}(e_i)$ for all $i \in \{1, \ldots, m\}$. We apply Proposition 7.7 with $d = m$ and $r = n^2$. Let $S \subseteq V$, and let $s \in \{0, \frac{1}{|S|}\}^m$ be the vector corresponding to the edges in the cut $(S, V \setminus S)$, that is, $s_i \neq 0$ if and only if $e_i \in (S, V \setminus S)$. Let $S' \subseteq V$ be another set, and let $s' \in \{0, \frac{1}{|S'|}\}^m$ with $s'_i \neq 0$ if and only if $e_i \in (S', V \setminus S')$. Define $b := s - s'$. Note that $|s_i - s'_i| = |\frac{|S'|s_i}{|S'|} - \frac{|S|s'_i}{|S|}| \in \{0, |\frac{|S'|-|S|}{|S|\cdot|S'|}|, \frac{1}{|S|}, \frac{1}{|S'|}\}$, and hence $b_i \in \mathbb{Q}_{n^2}$. We thus get

$$s^\top w - (s')^\top w \leq 0 \iff (s - s')^\top w \leq 0$$
$$\overset{\text{Prop. 7.7}}{\iff} (s - s')^\top \widehat{w} \leq 0 \iff s^\top \widehat{w} - (s')^\top \widehat{w} \leq 0. \qquad \square$$

We are not aware of any study on SSE from a parameterized algorithmics point of view.

7.5. Linearizable Functions

The cases of MiPoSyCo and SSE show that problems with non-linear goal functions still allow an application of the losing-weight technique. A natural question is what characterizes these goal functions. Both of our cases have in common that for any weight vector w, the goal function's value for every solution s can be represented as $b_s^\top w$ with b_s being a vector associated with s. Moreover, to apply the losing-weight technique, we also need that if we change the weight vector to a "smaller" weight vector $\widehat{w}$, then the goal function's value is still represented for solution s as $b_s^\top \widehat{w}$ and vice versa. That is, we want that the goal function's value for w is $b_s^\top w$ if and only if the goal function's value for $\widehat{w}$ is $b_s^\top \widehat{w}$. What we described so far is captured in the following (recall (7.3) for the definition of $\mathbb{Q}_r$).

Definition 7.1. A function $f\colon L \times \mathbb{Q}^d \to \mathbb{Q}$ with $L \subseteq \Sigma^*$ is α-linearizable, $\alpha \in \mathbb{N}$, if for all $w \in \mathbb{Q}^d$ it holds true that

(A) for all $x \in L$ there exists $b_x \in \mathbb{Q}_\alpha^d$ such that $f(x, w) = b_x^\top w$, and

(B) for all $w' \in \mathbb{Q}^d$ for which $\operatorname{sign}(\beta^\top w) = \operatorname{sign}(\beta^\top w')$ for all $\beta \in \mathbb{Q}_\alpha^d$ holds, it holds true that $f(x, w) = b_x^\top w \iff f(x, w') = b_x^\top w'$.

We prove next that the losing-weight technique applies to problems with α-linearizable goal functions, that is, for any weight vector we can compute in time polynomial in the input size a “smaller” weight vector such that solutions are preserved.

Proposition 7.9. *Let $f\colon L \times \mathbb{Q}^d \to \mathbb{Q}$ with $L \subseteq \Sigma^*$ be an α-linearizable function, and let $w \in \mathbb{Q}^d$. Then we can compute in time polynomial in the encoding length of w and α, a vector $\widehat{w} \in \mathbb{Z}^d$ such that*

(i) $\|\widehat{w}\|_\infty \leq 2^{4d^3}(4\alpha^4 \cdot d + 1)^{2\alpha^2 \cdot d(d+2)}$, *and*

(ii) $f(x, w) \geq f(y, w) \iff f(x, \widehat{w}) \geq f(y, \widehat{w})$ *for all $x, y \in L$.*

Proof. Apply Proposition 7.7 with $r = 2\alpha^2$ to obtain the vector $\widehat{w}$. Since f is α-linearizable, by (A) we know that for every $x, y \in L$ there are $b_x, b_y \in \mathbb{Q}_\alpha^d$ such that $f(x, w) = b_x^\top w$ and $f(y, w) = b_y^\top w$. Moreover, for $b := b_x - b_y$, we have $b \in \mathbb{Q}_{2\alpha^2}^d$. Next we have that

$$\begin{aligned} f(x, w) - f(y, w) \geq 0 &\overset{\text{(A)}}{\iff} (b_x - b_y)^\top w \geq 0 \\ &\overset{\text{Prop. 7.7}}{\iff} (b_x - b_y)^\top \widehat{w} \geq 0 \\ &\overset{\text{(B)}}{\iff} f(x, \widehat{w}) - f(y, \widehat{w}) \geq 0, \end{aligned}$$

where the last equivalence follows from the fact that for $\widehat{w}$, by Proposition 7.7, for all $\beta \in \mathbb{Q}_{2\alpha^2}^d$ we have $\operatorname{sign}(\beta^\top w) = \operatorname{sign}(\beta^\top \widehat{w})$, and hence from (B) we get $f(x, \widehat{w}) = b_x^\top \widehat{w}$ and $f(y, \widehat{w}) = b_y^\top \widehat{w}$. □

Intuitively, Proposition 7.9 yields the following: if we know that our goal function is α-linearizable, then we can employ the losing-weight technique where the encoding length of the computed weight vector is polynomially upper bounded in α and the dimension d. To easily employ Proposition 7.9, we only need to determine whether our goal function is α-linearizable, and, in particular, determine α. In fact, in what follows, we show that α-linearizable functions are functional *composable*: if a function is some specific function, say the maximum, of an α-linearizable function, then it is α'-linearizable for some α'. This allows for recognizing whether a function is α-linearizable by only looking at the functions it is composed of. In the following we define several of these functional compositions, and exemplify its usage on MiPoSyCo and SSE.

Revisiting the Case of MiPoSyCo. The goal function in MiPoSyCo is composed of a sum over maxima. We prove that these compositions preserve linearizability.

Lemma 7.10. *Let $f : L \times \mathbb{Q}^d \to \mathbb{Q}$, and $f' : L' \times \mathbb{Q}^d \to \mathbb{Q}$ be two functions where L' encodes some set U and L encodes the set $\{X \subseteq U \mid |X| \le n\}$ for some $n \in \mathbb{N}$. If f' is α-linearizable, then*

(i) $f(X,w) = \sum_{x\in X} f'(x,w)$ *is $n \cdot \alpha!\alpha$-linearizable;*

(ii) $f(X,w) = \max_{x\in X} f'(x,w)$ *is $2\alpha^2$-linearizable;*

(iii) $f(X,w) = \min_{x\in X} f'(x,w)$ *is $2\alpha^2$-linearizable.*

Proof. (i): Since f' is α-linearizable, by (A) for f' we know that for every $x \in L'$ there is $b'_x \in \mathbb{Q}^d_\alpha$ such that $f'(x,w) = b'^{\top}_x w$. Hence, we have

$$f(X,w) = \sum_{x\in X} f'(x,w) = \sum_{x\in X} b'^{\top}_x w = (\sum_{x\in X} b'_x)^\top w = b_X^\top w,$$

where $b_X \in \mathbb{Q}^d_{n\alpha!\alpha}$. That is, (A) holds for f. To prove (B) for f, let $\widehat{w} \in \mathbb{Q}^d$ such that $\operatorname{sign}(\beta^\top w) = \operatorname{sign}(\beta^\top \widehat{w})$ for all $\beta \in \mathbb{Q}^d_{n\alpha!\alpha}$. Note that due to (B) for f', we have

$$f'(x,w) = b'^{\top}_x w \iff f'(x,\widehat{w}) = b'^{\top}_x \widehat{w}, \tag{7.4}$$

since $\operatorname{sign}(\beta^\top w) = \operatorname{sign}(\beta^\top \widehat{w})$ holds for all $\beta \in \mathbb{Q}^d_\alpha \subseteq \mathbb{Q}^d_{n\alpha!\alpha}$. It follows that

$$\begin{aligned} f(X,w) = b_X^\top w \iff & \sum_{x\in X} f'(x,w) = \sum_{x\in X} b'^{\top}_x w \\ \overset{(7.4)}{\iff} & \sum_{x\in X} f'(x,\widehat{w}) = \sum_{x\in X} b'^{\top}_x \widehat{w} \iff f(X,\widehat{w}) = b_X^\top \widehat{w}, \end{aligned}$$

and hence (B) follows.

(ii): Since f' is α-linearizable, by (A) for f' we know that for every $x \in L'$ there is $b'_x \in \mathbb{Q}^d_\alpha$ such that $f'(x,w) = b'^{\top}_x w$. Hence, we have

$$f(X,w) = \max_{x\in X} f'(x,w) = \max_{x \in X} b'^{\top}_x w = b'^{\top}_{x^*} w = b^{\top}_X w,$$

where $x^* \in \arg\max_{x\in X} b'^{\top}_x w$ and $b_X \in \mathbb{Q}^d_\alpha$. That is, (A) holds for f. To prove that (B) holds for f, let $\widehat{w} \in \mathbb{Q}^d$ such that $\operatorname{sign}(\beta^\top w) = \operatorname{sign}(\beta^\top \widehat{w})$ for all $\beta \in \mathbb{Q}^d_{2\alpha^2}$. Note that due to (B) for f', we have

$$f'(x,w) = b'^{\top}_x w \iff f'(x,\widehat{w}) = b'^{\top}_x \widehat{w}, \tag{7.5}$$

since $\operatorname{sign}(\beta^\top w) = \operatorname{sign}(\beta^\top \widehat{w})$ holds for all $\beta \in \mathbb{Q}^d_\alpha \subseteq \mathbb{Q}^d_{2\alpha^2}$. Moreover, for $y \in X$ let $f'(y,w) = b'^{\top}_y w$. Let $b := b'_{x^*} - b'_y$ and note that $b \in \mathbb{Q}^d_{2\alpha^2}$. Hence, by the choice of $\widehat{w}$, it holds true that $b^\top w \geq 0$ if and only if $b^\top \widehat{w} \geq 0$. Thus, with (7.5) and by the choice of x^* we have $f'(x^*,w) \geq f'(y,w) \iff f'(x^*,\widehat{w}) \geq f'(y,\widehat{w})$ for all $y \in X$, and hence

$$\max_{x\in X} f'(x,w) = b'^{\top}_{x^*} w \iff \max_{x\in X} f'(x,\widehat{w}) = b'^{\top}_{x^*} \widehat{w}. \tag{7.6}$$

It follows that

$$\begin{aligned} f(X,w) = b^{\top}_X w &\iff \max_{x\in X} f'(x,w) = b'^{\top}_{x^*} w \\ &\overset{(7.6)}{\iff} \max_{x\in X} f'(x,\widehat{w}) = b'^{\top}_{x^*} \widehat{w} \iff f(X,\widehat{w}) = b^{\top}_X \widehat{w}, \end{aligned}$$

and hence (B) follows.

(iii): Follows analogously to (ii). □

We explain the use of our machinery for Min-Power Symmetric Connectivity. First observe that we can rewrite the goal function to fit our notion as follows. Let $F_v := \{e \in F \mid v \in e\}$ and $\mathcal{F} := \{F_v \mid v \in V\}$. Then

$$h(\mathcal{F},w) = \sum_{F_v \in \mathcal{F}} g(F_v,w), \qquad \text{with } g(F,w) = \max_{e\in F} w(e).$$

Clearly, with $E = \{e_1,\ldots,e_m\}$ the function $f : E \times \mathbb{Q}^m \to \mathbb{Q}$, $f(e_i,w) \mapsto w_i$ is 1-linearizable: On the one hand, we have that $f(e_i,w) = \vec{e}_i^{\,\top} w$ (recall that $\vec{e}_i$ denotes the unit vector with the ith entry being one). On the other hand, for all $w' \in \mathbb{Q}^m$ it holds true that $f(e_i, w) = \vec{e}_i^\top w \iff f(e_i, w') = \vec{e}_i^\top w'$ (in fact, this even holds true without any conditions on w').

Due to Lemma 7.10(ii), we know that the function $g(F, w) = \max_{e \in F} f(e, w)$ is 2-linearizable. Finally, due to Lemma 7.10(i) (with $L' = 2^E$ and $n = |V|$), we know that the function $h(\mathcal{F}, w) = \sum_{F_v \in \mathcal{F}} g(F_v, w)$ is $2! \cdot 2m$-linearizable. Employing Proposition 7.9, we get in polynomial time a vector $\widehat{w} \in \mathbb{Z}^m$ such that

- $\|\widehat{w}\|_\infty \in 2^{O(m^4 \log(m))}$, and
- for any two connected subgraphs $T = (V, F)$, $T' = (V, F')$ of G, we have

$$\sum_{v \in V} \max_{\{u,v\} \in F} w(\{u, v\}) \geq \sum_{v \in V} \max_{\{u,v\} \in F'} w(\{u, v\}) \iff$$
$$\sum_{v \in V} \max_{\{u,v\} \in F} \widehat{w}(\{u, v\}) \geq \sum_{v \in V} \max_{\{u,v\} \in F'} \widehat{w}(\{u, v\}),$$

that is, optimal solutions are preserved under $\widehat{w}$. Altogether, we reproved Lemma 7.5.

Revisiting the Case of SSE. The goal function in SSE is a multiplication of a number and a sum. By Lemma 7.10(i), we already know that the sum preserves linearizability. However, we also need to prove whether, and if how, linearizability is preserved under multiplying by some number.

Lemma 7.11. *Let $f, f' \colon L \times \mathbb{Q}^d \to \mathbb{Q}$ where L is equipped with some function $c \colon L \to \mathbb{Q}_n \setminus \{0\}$, where $n \in \mathbb{N}$. If f' is α-linearizable, then $f(x, w) = c(x) \cdot f'(x, w)$ is $\alpha \cdot n$-linearizable.*

Proof. Since f' is α-linearizable, by (A) for f' we know that for every $x \in L$ there is $b'_x \in \mathbb{Q}^d_\alpha$ such that $f'(x, w) = b'^\top_x w$. Hence, we have

$$f(x, w) = c(x) f'(x, w) = c(x) \cdot b'^\top_x w = (c(x) \cdot b'_x)^\top w = b_x^\top w,$$

where $b_x \in \mathbb{Q}^d_{n\alpha}$, proving (A). Let $\widehat{w} \in \mathbb{Q}^d$ such that $\operatorname{sign}(\beta^\top w) = \operatorname{sign}(\beta^\top \widehat{w})$ for all $\beta \in \mathbb{Q}^d_{n\alpha}$. Note that we have $f'(x, w) = (b_x / c(x))^\top w \iff f'(x, \widehat{w}) = (b_x / c(x))^\top \widehat{w}$. It follows that

$$\begin{aligned} f(x, w) = b_x^\top w &\iff f'(x, w) = (b_x / c(x))^\top w \\ &\iff f'(x, \widehat{w}) = (b_x / c(x))^\top \widehat{w} \\ &\iff f(x, \widehat{w}) = b_x^\top \widehat{w}, \end{aligned}$$

and hence (B) follows. □

We now explain the usage of our machinery for SSE. Let $E_S := (S, V \setminus S)$ for all $S \subseteq V$. Let $L = \{(S, E_S) \mid S \subseteq V, 1 \leq |S| \leq n/2\}$. Let $c : L \to \mathbb{Q}_n \setminus \{0\}, c : (S, E_S) \mapsto \frac{1}{|S|}$. Then

$$h((S, E_S), w) = \frac{1}{|S|} g((S, E_S), w), \qquad \text{with } g((S, E_S), w) = \sum_{e \in E_S} w(e)$$

We already now that $w : E \to \mathbb{Q}$ is 1-linearizable. Moreover, by Lemma 7.10(i) we know that g is m-linearizable. Finally, due to Lemma 7.11, we arrive at h being $n \cdot m$-linearizable. Finally employing Proposition 7.9 reproves Lemma 7.8.

A Brief Summary. We introduced α-linearizable functions (Definition 7.1). Due to Lemmas 7.10 and 7.11, we can recognize some α-linearizable functions by simply looking at how the functions are composed. Further, we proved that if a problem has an α-linearizable goal function, the losing-weight technique applies (Proposition 7.9). However, note that in Lemma 7.10(i), we obtain a factorial in the "α". This is not the case if we replace $\mathbb{Q}_\alpha$ in Definition 7.1 by $\mathbb{Z}_\alpha$, where

$$\mathbb{Z}_r := \{p \in \mathbb{Z} \mid |p| \in \{0, \ldots, r\}\}. \tag{7.7}$$

This replacement while more restrictive, appears to be often sufficient like in the case of MiPoSyCo, and also allows for "chaining up sums" while keeping α polynomially bounded.

The functional compositions preserving linearizability we detected are taking a sum, a maximum or minimum, or scaling (by some factor). We are curious about other compositions of functions preserving linearizability.

7.6. Concluding Remarks

The losing-weight technique due to Frank and Tardos [FT87] emerges as a key ingredient for obtaining polynomial kernelization for weighted parameterized problems. While Etscheid et al. [Ets+17] and Marx and Végh [MV15] proved the usefulness of the technique for several problems with linear goal functions, we proved the technique to be applicable to problems with non-linear goal functions, like the Min-Power Symmetric Connectivity problem. Moreover, in the next chapter we use the technique for an unweighted problem (via introducing weights similar to Marx and Végh [MV15], and hence reducing to a weighted problem).

As Etscheid et al. [Ets+17] already pointed out, one direction for future work could be to improve on the upper bound in Proposition 7.1(i). Another direction, seemingly not addressed so far, aims on the running time. Note that Frank and Tardos [FT87] state no explicit running time of their algorithm, and Lenstra et al. [LLL82, Proposition 1.26] state that their simultaneous Diophantine approximation algorithm, which forms a subroutine in Frank and Tardos' technique, runs in $O(d^6(\log(\|w\|_\infty))^{O(1)})$ time. Hence, we are curious about the following.

Open Problem 11. Can Proposition 7.1 be executed in quadratic, or even linear time?

Very recently, Eisenbrand et al. [Eis+19] reconsidered Frank and Tardos' technique in the context of integer programming. They give a non-constructive improvement on Proposition 7.1 shaving off to a $d\log(d)$ exponent in the upper bound. However, it is not explained how to use this for polynomial kernelization, since formally we have to *construct* the kernel (here, the weight vector). Eisenbrand et al. use some oracle-machinery and state that non-constructiveness suffices for this machinery. Inspired by that, we wonder whether Frank and Tardos' technique can be used in developing polynomial Turing kernelizations.

Chapter 8.

The Short Secluded Path Problem

We study the Short Secluded Path problem for the possibility of efficient data reduction regarding several (structural) parameters and prove several lower and upper bounds. Herein, we achieve two kernelization upper bounds via the losing-weight technique.

8.1. Introduction

Finding (the length of) a shortest path between two terminal vertices in an undirected graph is a fundamental problem. In addition to tasks like route planning, finding shortest paths is an inherent task in computing several graph measures, like centrality of a vertex or the diameter of a graph. It is folklore that a shortest path between two terminal vertices s and t can be computed in linear time. In this chapter, we study the following NP-hard variant of the classic problem of finding a shortest s-t path.

Short Secluded Path (SSP)
Input: An undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, and two integers $k \geq 2$ and $\ell \geq 0$.
Question: Is there an s-t path P in G such that $|V(P)| \leq k$ and $|N_G(V(P))| \leq \ell$?

This chapter is based on (parts of) *On the computational complexity of length- and neighborhood-constrained path problems* by Max-Jonathan Luckow and Till Fluschnik (Information Processing Letters [LF20]) and *Parameterized algorithms and data reduction for the short secluded s-t-path problem* by René van Bevern, Till Fluschnik, and Oxana Yu. Tsidulko (Networks [BFT20]).

In other words, we want to find a short path connecting the terminals s and t such that the number of vertices adjacent to the path is small. Thus, the problem is motivated by safe routing through transportation networks—taking a short route with few interventions is of smaller risk.

SSP is similar to the Secluded Path problem, introduced by Chechik et al. [Che+17], that, given an undirected graph $G = (V, E)$ with two designated vertices $s, t \in V$, vertex-weights $w : V \to \mathbb{N}$, and two integers $k, C \in \mathbb{N}$, asks whether there is an s-t path P such that the size of the *closed* neighborhood $|N_G[V(P)]| \leq k$ and the weight of the closed neighborhood $w(N_G[V(P)]) \leq C$. While SSP is unweighted, it distinguishes between the number of vertices in the path and in its *open* neighborhood, in contrast to Secluded Path.

In this chapter, we aim for efficient data reduction for SSP regarding the parameters k and ℓ, several structural graph parameters like treewidth, and their combinations.

Related Work. Luckow [Luc17] (see also [LF20]) first defined Short Secluded Path and proved it to be NP-hard, W[1]-hard regarding k, and para-NP-hard regarding ℓ.

Chechik et al. [Che+17] mostly studied Secluded Path in the context of approximation algorithms. In addition to Secluded Path, they introduced the Secluded Steiner Tree problem and proved it to be NP-complete. Fomin et al. [Fom+17a] studied the parameterized complexity of Secluded Path and Secluded Steiner Tree. They proved that Secluded Path admits kernels with size polynomial in the combination of k and the feedback vertex number. Moreover, they proved that Secluded Path admits no kernel with size polynomial in the vertex cover number.

Golovach et al. [Gol+17] studied the "small secluded" scenario for finding connected induced subgraphs with given properties. They proved that if the requested property is characterized through finitely many forbidden induced subgraphs, then the problem is fixed-parameter tractable when parameterized by the size ℓ of the open neighborhood. Their result does not generalize to SSP, since SSP is NP-hard even for $\ell = 0$ [LF20, Luc17].

Our Contributions. Our results and the outline of this chapter are summarized in Figure 8.1. We study four (structural) parameters, each of them combined with k, ℓ, and $k + \ell$. For two of our results, namely Theorems 8.25 and 8.31, we employ the losing-weight technique (Proposition 7.1) described

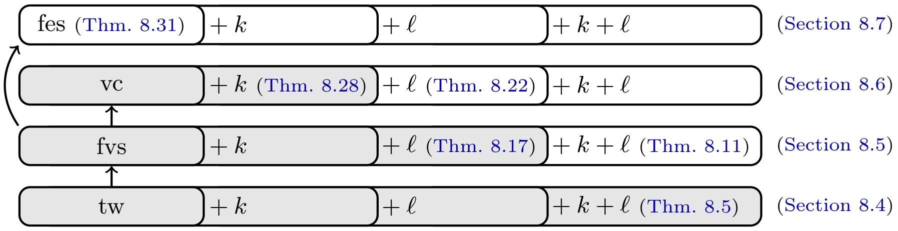


Figure 8.1.: Overview of the existence of polynomial kernelization (arrows relate pairs of parameters, cf. Figure 1.4). A white box depicts the existence of a polynomial kernel and a gray box depicts an exclusion of a polynomial kernel (unless $\text{coNP} \subseteq \text{NP}_{/\text{poly}}$).

in the previous chapter. Notably, in both cases, the technique is applied to an unweighted problem. We describe our application of the technique in more detail in Section 8.3.

Our negative results on kernelization for SSP regarding the vertex cover number vc are stronger than the corresponding result of Fomin et al. [Fom+17a] for SECLUDED PATH: we prove that SSP parameterized by vc$+r$ is WK[1]-hard and admits no polynomial kernel even in bipartite $K_{r,r}$-subgraph-free graphs.

8.2. Preliminaries on SSP

In this section, we first prove SSP to be fixed-parameter tractable when parameterized by the combination of the number of vertices in the solution path and its open neighborhood size.

Theorem 8.1. SHORT SECLUDED PATH *admits an* $O((k+\ell)^k) \cdot n^{O(1)}$*-time algorithm and hence is fixed-parameter tractable when parameterized by* $k+\ell$.

Proof. Let $\mathcal{I} = (G = (V, E), s, t, k, \ell)$ be an instance of SSP. We partition $V = R \uplus B$ such that $R := \{v \in V \mid \deg_G(v) \geq k + \ell + 1\}$. Clearly, no solution s-t path can contain any vertex from R. Apply the following branching tree algorithm. Starting at s, consider all neighbors of s and branch on vertices from B but not from R, that is, only on vertices of degree at most $k+\ell$, and proceed recursively. Stop branching at depth $k-1$ (s is by convention at depth zero). Observe that every s-t path with at most k vertices is found in the branching, and we can verify in polynomial time whether the size of the open neighborhood of the found path is at most ℓ (return that $\mathcal{I}$ is a yes-instance in this case). As we only branch on vertices from B, we have at most $(k+\ell)^k$ nodes in our branching tree. If the whole branching tree is explored without returning that $\mathcal{I}$ is a `yes`-instance, then return that $\mathcal{I}$ is a `no`-instance. Hence, we can decide $\mathcal{I}$ for SSP in $O((k+\ell)^k) \cdot n^{O(1)}$ time. □

Note that we will prove SSP to admit no kernel of size polynomial in $k+\ell$ unless coNP $\subseteq$ NP$_{/\text{poly}}$ (Theorem 8.5). Further, from now on we can assume that the input graph is connected due to the following.

Reduction Rule 8.1. *If G has more than one connected component, then delete all components except the one containing both s and t or, if such a component does not exist, return that the instance is a `no`-instance.*

8.3. Weighted SSP and Losing Weights

We achieve two of our polynomial kernels via an auxiliary vertex-weighted variant of SSP. Our vertex-weighted variant of SSP allows each vertex to have three weights, where $\kappa(v)$ contributes to the length of the path, and $\lambda(v)$ and $\eta(v)$ contribute to the number of neighbors. It is defined as follows.

VERTEX-WEIGHTED SHORT SECLUDED PATH (VW-SSP)
Input: An undirected graph $G=(V,E)$ with two distinct vertices $s,t \in V$, two integers $k \geq 2$ and $\ell \geq 0$, and vertex weights $\kappa : V \to \mathbb{N}$, $\lambda : V \to \mathbb{N}_0$, and $\eta : V \to \mathbb{N}_0$.
Question: Is there an s-t-path P with $\sum_{v\in V(P)} \kappa(v) \leq k$ and $\sum_{v\in V(P)} \eta(v) + \sum_{v\in N(V(P))} \lambda(v) \leq \ell$ in G?

Note that an instance of SSP can be considered to be an instance of VW-SSP with unit-weight functions κ and λ and the zero-weight function η. As a convention throughout this chapter, for any set $X \subseteq V$, we write $\gamma(X)$ for $\sum_{x\in X} \gamma(x)$ for every $\gamma \in \{\kappa, \lambda, \eta\}$.

Now, the idea behind the two polynomial kernels via VW-SSP can be summarized as follows (refer to Figure 8.2):

1. Shrink the number of vertices and edges while introducing vertex-weights and hence constructing a vertex-weighted instance.

2. Shrink the vertex-weights using the losing-weight technique (Proposition 7.1) such that their encoding lengths is upper-bounded polynomially by the number of vertices in the graph.

$$\begin{tikzcd}
\boxed{\mathcal{I}} \arrow[r, "\text{partial}", "\text{bikernelization}"'] & \boxed{\mathcal{I}_w} \arrow[r, "\text{kernelization}", "\text{via Prop. 7.1}"'] & \boxed{\mathcal{I}'_w} \arrow[r, "\text{many-one}", "\text{reduction}"'] & \boxed{\mathcal{I}'} \\
 & \#\text{ vertices} \in O(p) & |\mathcal{I}'_w| \in O(p^4) & |\mathcal{I}'| \in p^{O(1)}
\end{tikzcd}$$

Figure 8.2.: High-level sketch of our approach to achieve polynomial kernels. $\mathcal{I}$ and $\mathcal{I}'$ denote instances of SSP (white boxes), and $\mathcal{I}_w$ and $\mathcal{I}'_w$ denote instances of Vertex-Weighted Short Secluded Path (gray boxes).

3. Apply a polynomial-time many-one reduction to map the vertex-weighted instance to an instance of SSP.

Step 1 of our outlined algorithm consists of parameter-specific data reduction rules. We describe Steps 2 and 3 in more detail in the remainder of this section.

Step 2: Shrinking Weights. To shrink the weights of an VW-SSP instance, we apply Proposition 7.1 and Observation 7.2 to the weights of VW-SSP.

Lemma 8.2. *An instance $\mathcal{I} = (G = (V, E), s, t, k, \ell, \lambda, \kappa, \eta)$ of* Vertex-Weighted Short Secluded Path *with $n := |V|$ can be reduced in polynomial time to an instance $\mathcal{I}' = (G, s, t, k', \ell', \lambda', \kappa', \eta')$ of* VW-SSP *such that*

(i) $k', \kappa'(v), \ell', \lambda'(v), \eta'(v) \in \{0, \dots, 2^{4(2n+1)^3} \cdot (n+2)^{(2n+1)(2n+3)}\}$, for each vertex $v \in V$, and

(ii) $\mathcal{I}$ is a `yes`*-instance if and only if $\mathcal{I}'$ is a* `yes`*-instance.*

Proof. We denote the weight functions λ, λ', κ, κ', η, and η' as column vectors in $\mathbb{N}_0^n$ such that $\gamma_v = \gamma(v)$ for each $v \in V$ and $\gamma \in \{\kappa, \lambda, \eta\}$.

We apply Proposition 7.1 with $d = 2n + 1$ and $N = n + 2$ separately to the vectors $(\eta, \lambda, \ell) \in \mathbb{N}^{2n+1}$ and $(\kappa, \{0\}^n, k) \in \mathbb{N}^{2n+1}$ to obtain vectors $(\eta', \lambda', \ell') \in \mathbb{Z}^{2n+1}$ and $(\kappa', \{0\}^n, k') \in \mathbb{Z}^{2n+1}$ in polynomial time.

(i) This follows from Proposition 7.1, with $d = 2n + 1$ and $N = n + 2$, and from Observation 7.2 since (η, λ, ℓ) and $(\kappa, \{0\}^n, k)$ are vectors of non-negative numbers.

(ii) Consider an arbitrary s-t path P in G and two associated vectors $x, y \in \mathbb{Z}^n$, where

$$x_v = \begin{cases} 1 & \text{if } v \in V(P), \\ 0 & \text{otherwise,} \end{cases} \qquad y_v = \begin{cases} 1 & \text{if } v \in N(V(P)) \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Observe that $\|(x, \{0\}^n, -1)\|_1 \leq \|(x, y, -1)\|_1 \leq n + 1$. Since $n + 1 \leq N - 1$, Proposition 7.1 gives $\operatorname{sign}((x, y, -1)^\top(\eta, \lambda, \ell)) = \operatorname{sign}((x, y, -1)^\top(\eta', \lambda', \ell'))$, being equivalent to

$$\sum_{v \in V(P)} \eta(v) + \sum_{v \in N(V(P))} \lambda(v) \leq \ell \iff \sum_{v \in V(P)} \eta'(v) + \sum_{v \in N(V(P))} \lambda'(v) \leq \ell',$$

and $\operatorname{sign}((x, \{0\}^n, -1)^\top(\kappa, \{0\}^n, k)) = \operatorname{sign}((x, \{0\}^n, -1)^\top(\kappa', \{0\}^n, k'))$, being equivalent to

$$\sum_{v \in P} \kappa(v) \leq k \iff \sum_{v \in P} \kappa'(v) \leq k'. \qquad \square$$

Step 3: Reducing back. For reducing back, we give a polynomial-time many-one reduction from any instance of VW-SSP being of the following specific structure.

Definition 8.1. An instance $(G = (V, E), s, t, k, \ell, \lambda, \kappa, \eta)$ of VW-SSP is called *simplified* if there is a set $A \subseteq V$ such that

(i) $\kappa(s) = \kappa(t) = 1$,

(ii) $\lambda(v) = 1$ for all $v \in V$,

(iii) $\eta(v) > \ell$ and $\kappa(v) = 1$ for all $v \in A$, and

(iv) in $G - A$, every vertex v with $\kappa(v) > 1$ has exactly two neighbors u and w, with $\kappa(u) = \kappa(w) = 1$, and each having degree at most two and being distinct from s and t.

Next we show that for any given simplified instance of VW-SSP, we can compute in linear time an equivalent instance of SSP whose number of vertices only depends on κ and η.

Proposition 8.3. *Any simplified instance* $(G = (V, E), s, t, k, \ell, \lambda, \kappa, \eta)$ *of* VW-SSP *with given* $A \subseteq V$ *can be reduced to an equivalent instance of* SSP *with at most* $M := \kappa(V) + \eta(V)$ *vertices in time linear in* $M + |E|$.

To prove Proposition 8.3, we use the following construction.

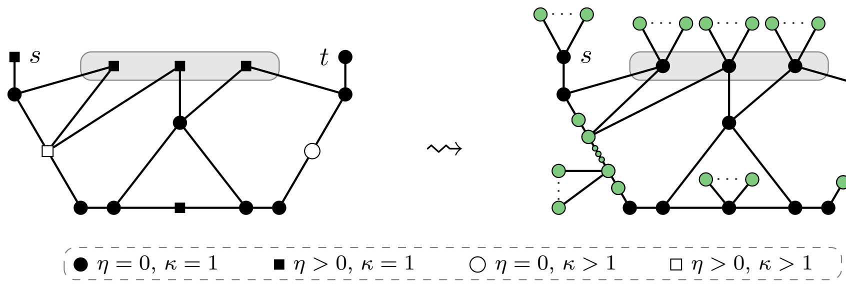

s
t
s
t
● $\eta = 0,\ \kappa = 1$
■ $\eta > 0,\ \kappa = 1$
○ $\eta = 0,\ \kappa > 1$
□ $\eta > 0,\ \kappa > 1$

Figure 8.3.: Illustrative example to Construction 8.4. On the left-hand side, an input graph with vertex weights (indicated by different vertex shapes) is depicted. on the right-hand side, the graph obtained after applying Construction 8.4 (green vertices indicate added vertices) is shown. Vertices enclosed in the gray solid rectangle form the set A.

Construction 8.4. Let $(G = (V, E), s, t, k, \ell, \lambda, \kappa, \eta)$ be a simplified instance of VW-SSP with given set $A \subseteq V$ as in Definition 8.1. Construct an instance (G', s', t', k, ℓ) of SSP as follows (see Figure 8.3 for an illustrative example). Let G' be initially a copy of G. For each $v \in V$ with $\kappa(v) > 1$, let $\{v', v''\} = N_{G-A}(v)$, replace v by a path P_v with $\kappa(v)$ vertices, make one endpoint adjacent to v', and the other endpoint adjacent to v''. Next, for each $v \in V$, add a set U_v of $\eta(v)$ vertices. If $\kappa(v) = 1$, then make each $u \in U_v$ only adjacent to v. If $\kappa(v) > 1$, then make each $u \in U_v$ only adjacent to some vertex x on P_v. Finally, for each $v \in V \setminus (A \cup \{s, t\})$ with $\kappa(v) > 1$ and $A_v := N_G(v) \cap A \neq \emptyset$, make each $w \in A_v$ adjacent with some vertex x on P_v. This finishes the construction of G'. Observe that the construction can be done in $O(M + |E|)$ time and (G', s, t, k, ℓ) consists of M vertices. ▲

Proof of Proposition 8.3. Let $\mathcal{I} = (G, s, t, k, \ell, \lambda, \kappa, \eta)$ be a simplified instance of VW-SSP with $G = (V, E)$ and given set $A \subseteq V$ as in Definition 8.1. Apply Construction 8.4 to compute instance $\mathcal{I}' := (G', s, t, k, \ell)$ of SSP with at most $M := \kappa(V) + \eta(V)$ vertices in time linear in $M + |E|$. We claim that $\mathcal{I}$ is a `yes`-instance if and only if $\mathcal{I}'$ is a `yes`-instance.

($\Rightarrow$) Let $\mathcal{I}$ be a `yes`-instance and $P := (v_1, v_2, \ldots, v_q)$ with $v_1 = s$ and $v_q = t$ be a solution s-t path. Let $W \subseteq V(P)$ denote the vertices in P with $\kappa(v) > 1$. We claim that the path P' obtained from P by replacing each vertex $v \in W$ by P_v

is a solution s-t path to $\mathcal{I}'$. First, observe that $|V(P')| = |V(P)\setminus W| + \kappa(W) \leq k$. It remains to prove (recall that $\lambda(v) = 1$ for all $v \in V$)

$$|N_{G'}(V(P'))| = |N_G(V(P))| + \sum_{v \in V(P')} |U_v| = |N_G(V(P))| + \sum_{v \in V(P)} \eta(v) \leq \ell.$$

To this end, it is enough to prove $N_{G'}(V(P')) = N_G(V(P)) \uplus \biguplus_{v \in V(P')} U_v$. First observe that no vertex in A is in $V(P)$ since $\eta(v) > \ell$ for all $v \in A$. Thus, no vertex in A is contained in $V(P')$. For each $v \in W$ let v' and v'' be the only two neighbors of v in $G - A$. Then, for each $v \in W$, we have $N_{G'}(V(P_v)) \setminus \{v', v''\} = N_G(v) \setminus \{v', v''\}$, since the neighbors of v in A coincide with the neighbors of $V(P_v)$ in A. Thus,

$$\begin{aligned}
N_{G'}(V(P')) = {} & \left(N_{G'}(V(P) \setminus W) \setminus \bigcup_{v \in W} V(P_v) \right) \cup \bigcup_{v \in W} (N_{G'}(V(P_v)) \setminus \{v', v''\}) \\
& \uplus \biguplus_{v \in V(P')} U_v \\
= {} & (N_G(V(P) \setminus W) \setminus W) \cup \bigcup_{v \in W} (N_G(v) \setminus \{v', v''\}) \uplus \biguplus_{v \in V(P')} U_v \\
= {} & N_G(V(P)) \uplus \biguplus_{v \in V(P')} U_v.
\end{aligned}$$

($\Leftarrow$) Let $\mathcal{I}'$ be a `yes`-instance and let P' be a solution s-t path. Note that all vertices in P_v for $v \in V$ with $\kappa(v) > 1$ are of degree two in $G' - (A \cup U_v)$ and distinct from s and t. Hence, if a vertex of P_v is contained in P', then all vertices from P_v are contained in P'. Let $W \subseteq V$ denote the set of vertices v with $\kappa(v) > 1$ such that P_v is a subpath of P'. We claim that the path P obtained from P' by replacing each path P_v by $v \in W$ is a solution s-t path for $\mathcal{I}$. First, observe that $\kappa(V(P)) = |V(P')| - \sum_{v \in W} |V(P_v)| + \kappa(W) = |V(P')| \leq k$. Second, similarly as in the above direction, since $\mathcal{I}$ is simplified, we have

$$\begin{aligned}
|N_G(V(P))| + \sum_{v \in V(P)} \eta(v) &= |N_G(V(P))| + \sum_{v \in V(P)} |U_v| \\
&= |N_{G'}(V(P'))| \leq \ell. \qquad \square
\end{aligned}$$

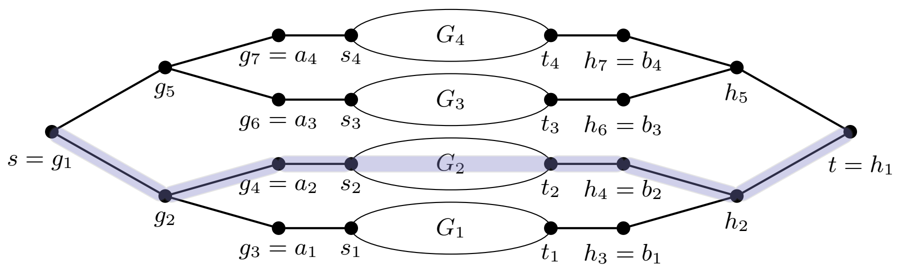


Figure 8.4.: Illustrative example of Construction 8.6 with $p = 4$ instances with graphs $G_1, \ldots, G_4$.

8.4. Treewidth

We proved SSP parameterized by $k + \ell$ to be fixed-parameter tractable (Theorem 8.1). Complementing this, we prove that, unless coNP $\subseteq$ NP$_{/\text{poly}}$, SSP admits no kernel of size polynomial in $k + \ell$. In fact, this holds true even when the parameter is additionally combined with the treewidth of the input graph:

Theorem 8.5. *Unless coNP $\subseteq$ NP$_{/poly}$,* Short Secluded Path *admits no kernel with size polynomial in* $\text{tw} + k + \ell$*, even on planar graphs with maximum degree six.*

We prove Theorem 8.5 using an OR-cross-composition. The construction is as follows (refer to Figure 8.4 for an illustrative example).

Construction 8.6. Let p be a power of two, and let $\mathcal{I}_i = (G_i, s_i, t_i, k_i, \ell_i)$, $i \in \{1, \ldots, p\}$, be instances of SSP such that each G_i is a planar graph of maximum degree five and has a planar embedding with s_i and t_i on the outer face. Without loss of generality, the vertex sets of the graphs $G_1, \ldots, G_p$ are pairwise disjoint, and, for all $i \in \{1, \ldots, p\}$, we have $|V(G_i)| = n$, $\ell_i = \ell$, $k_i = k$, and $k, \ell \leq n$ (this is a polynomial equivalence relation). We construct an instance $\mathcal{I} = (G, s, t, k', \ell')$ of SSP, where

$$k' := k + 2\log(p) + 2,$$
$$\ell' := \ell + 2\log(p),$$

and the graph G is as follows. Graph G consists of $G_1, \ldots, G_p$ and two rooted balanced binary trees T_s and T_t with roots s and t, respectively, each having p

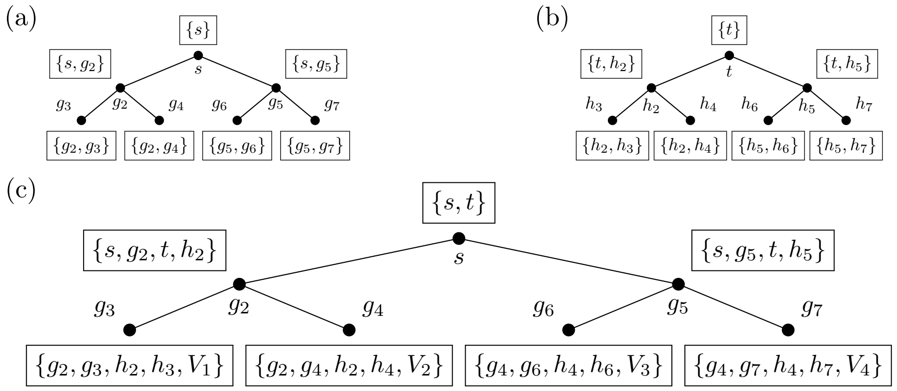


Figure 8.5.: Overview of the tree decompositions (sets in boxes refer to the bags), exemplified for $p = 4$ input instances. (a) and (b) display the tree decomposition for T_s and T_t, respectively. (c) displays the tree decomposition $\mathbb{T}$ (and $\mathbb{T}_{st}$ when removing V_i for $i \in \{1, \ldots, 4\}$). Here, V_i represents the set of vertices in the input graph G_i.

leaves. Let $g_1, \ldots, g_{2p-1}$ and $h_1, \ldots, h_{2p-1}$ denote the vertices of T_s and T_t enumerated by a depth-first search starting at s and t, respectively. Moreover, let $a_1, \ldots, a_p$ and $b_1, \ldots, b_p$ denote the leaves of T_s and T_t as enumerated in each depth-first search mentioned before. Then, for each $i \in \{1, \ldots, p\}$, graph G contains the edges $\{a_i, s_i\}$ and $\{b_i, t_i\}$. This finishes the construction. ▲

We first prove that the graph computed in Construction 8.6 has treewidth at most $n + 3$. We then prove that the instance computed in Construction 8.6 is a yes-instance if and only if one of the input instances is a yes-instance.

Lemma 8.7. *Let $\mathcal{I}_i$, $i \in \{1, \ldots, p\}$, and $\mathcal{I} = (G, s, t, k', \ell')$ be as in Construction 8.6. Then* $\operatorname{tw}(G) \leq n + 3$.

Proof. We give a tree decomposition (see Definition 1.13) of width at most $n+3$ for G as illustrated in Figure 8.5:

First, we construct a tree decomposition $\mathbb{T}_s = (T_s, \beta_s)$ of T_s with bags as follows. Let $\text{parent}_{T_s}(v)$ denote the parent of $v \in V(T_s)$ (where $\text{parent}_{T_s}(s) = s$). For each $v \in V(T_s)$, let $\beta_s(v) := \{v\} \cup \{\text{parent}_{T_s}(v)\}$. Then $\mathbb{T}_s$ is a tree decomposition of width one. Let $\mathbb{T}_t = (T_t, \beta_t)$ be the tree decomposition for T_t constructed analogously.

We now construct a tree decomposition $\mathbb{T}_{st} = (T, \beta_{st})$ for the disjoint union of T_s and T_t as follows: take $T = T_s$ and, for each $i \in \{1, \ldots, 2p-1\}$, let $\beta_{st}(g_i) := \beta_s(g_i) \cup \beta_t(h_i)$, where g_i and h_i are the vertices of T_s and T_t according to the depth-first search labeling in Construction 8.6. As $\mathbb{T}_s$ and $\mathbb{T}_t$ are tree decompositions of two vertex-disjoint trees T_s and T_t, respectively, and $\{g_i, g_j\}$ is an edge of T_s if and only if $\{h_i, h_j\}$ is an edge of T_t, $\mathbb{T}_{st}$ is a tree decomposition for the disjoint union of T_s and T_t. The width of $\mathbb{T}_{st}$ is three.

Now, recall that, for $i \in \{1, \ldots, p\}$, the graph G_i in G is adjacent to exactly one leaf a_i of T_s and one leaf b_i of T_t. Hence, by adding $V(G_i)$ to bag $\beta(a_i)$, containing both a_i and b_i for each $i \in \{1, \ldots, p\}$, we obtain a tree decomposition $\mathbb{T}$ of G from $\mathbb{T}_{st}$ of width at most $n + 3$. Hence, we have $\operatorname{tw}(G) \leq n + 3$. $\square$

Next we prove Construction 8.6 to construct an equivalent instance.

Lemma 8.8. *Let $\mathcal{I}_i$, $i \in \{1, \ldots, p\}$, and $\mathcal{I}$ be as in Construction 8.6. Then $\mathcal{I}$ is a `yes`-instance if and only if $\mathcal{I}_i$ is a `yes`-instance for some $i \in \{1, \ldots, p\}$.*

In the proof of Lemma 8.8, we will use the following observation on the graph obtained from Construction 8.6.

Observation 8.9. *Let G with terminals s and t be the graph obtained from Construction 8.6. Then every s-t path contains at most one terminal pair s_i, t_i for some $i \in \{1, \ldots, p\}$.*

Proof. We prove the statement by induction on q, where $p = 2^q$. For $q = 1$, the observation holds immediately. For the induction step, assume that the statement holds for $q-1 > 1$. Let G with terminals s and t be the graph obtained from Construction 8.6 with $p = 2^q$. Recall that we labeled both rooted balanced binary trees by a depth-first search and connected the leaves with the instances in the same order in each tree. Hence, $G - \{s, t\}$ contains two components G^1 and G^2, where G^1 and G^2 are the graphs obtained from Construction 8.6 with $p = 2^{q-1}$. Let s^1, t^1 denote the terminals for G^1, and s^2, t^2 denote the terminals for G^2. By the inductive assumption, every s^1-t^1 path contains at most one terminal pair s_i, t_i in G^1, and every s^2-t^2 path contains at most one terminal pair s_i, t_i in G^2. Since every s-t path in G contains either s^1 (and then also t^1) or s^2 (and then also t^2), every s-t path contains a subpath which is either an s^1-t^1 path or an s^2-t^2 path. Thus, the claim follows. $\square$

Proof of Lemma 8.8. ($\Leftarrow$) Let $\mathcal{I}_i$, $i \in \{1, \ldots, p\}$, be a `yes`-instance, and let P_i be a solution s_i-t_i path in G. Let $P_{s,i}$ denote the unique path with endpoints s

and a_i in T_s. Note that $|V(P_{s,i})| = \log(p) + 1$. Similarly, let $P_{t,i}$ denote the unique path with endpoints t and b_i in T_t. Note that $|N_{T_s}(P_{s,i})| = \log(p)$, as each vertex in $P_{s,i}$ except s and σ_i are of degree three in T_s, and s has one unique neighbor not in $P_{s,i}$. With the same argument, we have $|N_{T_t}(P_{i,t})| = \log(p)$. Let

$$\begin{aligned}
V_{P_i} &:= V(P_i) \cup V(P_{s,i}) \cup V(P_{i,t}) \text{ and} \\
E_{P_i} &:= E(P_i) \cup E(P_{s,i}) \cup E(P_{i,t}) \cup \{\{a_i, s_i\}, \{t_i, b_i\}\}.
\end{aligned}$$

We claim that the path $P = (V_{P_i}, E_{P_i})$ is a solution s-t path in G. By construction, P contains at most k' vertices: Note that $V(P_i) \cup V(P_{s,i}) \cup V(P_{i,t})$ contains at most $k + 2\log(p) + 2$ vertices. Moreover, we have $|N_G(P)| = |N_{T_s}(P_{s,i})| + |N_{T_t}(P_{i,t})| + |N_{G_i}(P_i)| \leq 2\log(p) + \ell = \ell'$.

($\Rightarrow$) Let $\mathcal{I}$ be a yes-instance, and let P be a solution s-t path in G. We claim that there is a subpath $P_i \subseteq P$ such that P_i is a solution s_i-t_i path in G_i, for some $i \in \{1, \ldots, p\}$. Observe that P must contain at least one leaf in T_s and one leaf in T_t. Hence, $|V(P) \cap V(T_s)| \geq \log(p) + 1$ and $|V(P) \cap V(T_s)| \geq \log(p) + 1$. Moreover, $s_i \in V(P)$ if and only if $t_i \in V(P)$, as P has only endpoints s and t, and $\{s_i, t_i\}$ separates $V(G_i) \setminus \{s_i, t_i\}$ from $V(G) \setminus V(G_i)$. Hence, let $i \in \{1, \ldots, p\}$ such that $a_i \in V(P)$ (and hence $b_i \in V(P)$). Let P_i be the subpath of P with endpoints s_i and t_i. Clearly, $V(P_i) \subseteq V(G'_i)$. We claim that P_i is a solution s_i-t_i path in G_i. First, suppose $|V(P_i)| > k$. Then we have

$$\begin{aligned}
|V(P)| &\geq |V(P) \cap V(T_s)| + |V(P) \cap V(T_t)| + |V(P_i)| \\
&> k + 2(\log(p) + 1) = k',
\end{aligned}$$

contradicting the fact that P is a solution s-t path in G.

Due to Observation 8.9, we know that s_i and t_i are the only terminals contained in P. It follows that $T_s[V(P) \cap V(T_s)]$ is the unique path in T_s with endpoints s and a_i, and $T_t[V(P) \cap V(T_t)]$ is the unique path in T_t with endpoints t and b_i. Moreover, $|N_{T_s}(V(P))| = |N_{T_t}(V(P))| = \log(p)$. Finally, suppose that $|N_{G_i}(V(P_i))| > \ell$. Then we have

$$\begin{aligned}
|N_G(V(P))| &= |N_{T_s}(V(P))| + |N_{T_t}(V(P))| + |N_{G_i}(V(P_i))| \\
&> \ell + 2\log(p) = \ell',
\end{aligned}$$

contradicting the fact that P is a solution s-t path in G. We conclude that P_i is a solution s_i-t_i path in G_i, and hence, $\mathcal{I}_i$ is a yes-instance. $\square$

To prove Theorem 8.5, we only need that the variant of SSP described in Construction 8.6 as input problem is NP-hard.

Theorem 8.10 (Luckow and Fluschnik [LF20]). Short Secluded Path *is NP-complete even on graphs of maximum degree five that admit a planar embedding with terminals s and t being on the outer face.*

Lemmas 8.7 and 8.8 and Theorem 8.10 at hand, we are ready to prove that Construction 8.6 is an OR-cross-composition, yielding Theorem 8.5.

Proof of Theorem 8.5. For $i \in \{1, \ldots, p\}$, p being a power of two, let $I_i = (G_i, s_i, t_i, k_i, \ell_i)$ be instances of SSP such that each G_i is a planar graph of maximum degree five and has a planar embedding with s_i and t_i on the outer face, and for all $i \in \{1, \ldots, p\}$, $|V(G_i)| = n$, $k_i = k$, $\ell_i = \ell$, and $k, \ell \leq n$. Apply Construction 8.6 to obtain instance $\mathcal{I} := (G, s, t, k', \ell')$ in time polynomial in $\sum_{i=1}^{p} |\mathcal{I}_i|$. Due to Lemma 8.7, we know that $\operatorname{tw}(G) + k' + \ell' \in (n + \log(p))^{O(1)}$. Due to Lemma 8.8, we know that $\mathcal{I}$ is a `yes`-instance if and only if $\mathcal{I}_i$ is a `yes`-instance for some $i \in \{1, \ldots, p\}$. Hence, Construction 8.6 is an OR-cross-composition. Finally, from Theorem 8.10 together with Proposition 1.3, the statement follows. □

As a final remark, SSP is fixed-parameter tractable when parameterized by the treewidth [BFT20].

8.5. Feedback Vertex Number

In this section, we study the parameter feedback vertex number fvs combined with ℓ and $k+\ell$. We prove in Section 8.5.1 kernels with size polynomial in $\text{fvs}+k+\ell$ via our technique described in Section 8.3. Moreover, we prove in Section 8.5.2 that dropping k is presumably impossible for polynomial kernelization, that is, unless $\text{coNP} \subseteq \text{NP}_{/\text{poly}}$ there is no kernel of size polynomial in $\text{fvs} + \ell$. Note that in Section 8.6.3, we prove that, unless $\text{coNP} \subseteq \text{NP}_{/\text{poly}}$, no kernel of size polynomial in the vertex cover number exists, implying the same for $\text{fvs} + k$.

8.5.1. A Polynomial Kernel with $O(\text{fvs} \cdot (k+\ell)^2)$ Vertices

We show that SSP admits a kernel with a number of vertices cubic in the parameter $\text{fvs} + \ell + k$.

Theorem 8.11. Short Secluded Path *admits a kernel of size polynomial in* $\mathrm{fvs} + k + \ell$ *with* $O(\mathrm{fvs} \cdot (k + \ell)^2)$ *vertices.*

In a nutshell, we will construct a simplified instance of VW-SSP by shrinking the number and sizes of the trees in the graph after removing a feedback vertex set. Herein, we store information on each shrinking step in the vertex weights. To shrink the sizes of the trees, we delete leaves (as their number upper-bounds the number of vertices of degree at least three) and replace maximal paths consisting of degree-two vertices by shorter paths. The number of vertices and edges as well as the vertex weights will be upper-bounded polynomially in $\mathrm{fvs} + k + \ell$. Finally, we employ Proposition 8.3 on the simplified instance of VW-SSP to compute an instance of SSP where the number of vertices and edges is upper-bounded polynomially in $\mathrm{fvs} + k + \ell$.

Let $G = (V, E)$ be the input graph with $V = F \uplus W$ such that F is a feedback vertex set with $s, t \in F$ (hence, $G[W]$ is a forest). Let $\beta := |F|$. We distinguish the following types of vertices of G (see Figure 8.6 for an illustration).

$R \subseteq F$ is the subset of vertices in F with more than $k + \ell$ neighbors or more than $\ell + 2$ degree-one neighbors. Since no vertex of R is part of any solution path, we refer to the vertices in R as *forbidden.*

$Y \subseteq W$ is the subset of vertices in W containing all vertices that have at least one neighbor in F that is not forbidden, that is, all vertices v with $N(v) \cap F \nsubseteq R$. We call the vertices in Y *good.*

$\mathcal{T}$ is the set of connected components of $H := G[W]$, all of which are trees.

Towards proving Theorem 8.11, we will first prove the following, and then strip the weights using Proposition 8.3.

Proposition 8.12. *For any instance of* SSP *we can compute in polynomial time an equivalent simplified instance of* VW-SSP *with* $O(\mathrm{fvs} \cdot (k + \ell))$ *vertices,* $O(\mathrm{fvs}^2 \cdot (k + \ell))$ *edges, and vertex weights in* $O(k + \ell)$*.*

We will interpret the input SSP instance as an instance of VW-SSP with unit-weight functions κ and λ and the zero-weight function η. For an exemplified illustration of the following Reduction Rules 8.3 to 8.5, we refer to Figure 8.6. The first reduction rule ensures that each forbidden vertex remains forbidden throughout our application of all reduction rules. It is clearly applicable in linear time.

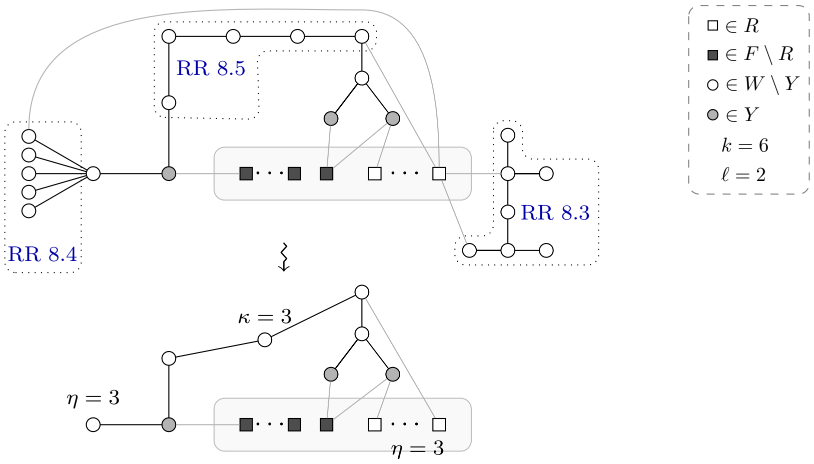


Figure 8.6.: Exemplified illustration of the partition into forbidden (white square, adjacent degree-one vertices are omitted) and good (gray round) vertices, and for the application of Reduction Rules 8.3 to 8.5 (abbreviated by RR; indicated by dotted boxes). The vertices enclosed in the light-gray rectangle are all vertices in the feedback vertex set F.

Reduction Rule 8.2. *For each $v \in R$, set $\eta(v) = \ell + 1$.*

Since each vertex in $F \setminus R$ has degree at most $k + \ell$, by the definition of good vertices, we have the following.

Observation 8.13. *The number of good vertices is $|Y| \leq \beta(k + \ell)$.*

Since a solution path has neither vertices nor neighbors in any tree $T \in \mathcal{T}$ containing no vertex of Y, we delete such trees.

Reduction Rule 8.3. *Delete all trees $T \in \mathcal{T}$ with $V(T) \cap Y = \emptyset$.*

Note that if Reduction Rule 8.3 is not applicable, then each tree in $\mathcal{T}$ contains a vertex from Y, which gives $|\mathcal{T}| \leq \beta(k + \ell)$ due to Observation 8.13.

The following data reduction rule deletes degree-one vertices in trees that are not in Y, since they cannot be part of a solution path (yet can neighbor it).

Reduction Rule 8.4. *If there is a tree $T \in \mathcal{T}$ and $v \in V(T) \setminus Y$ with $N_T(v) = \{w\}$, then set $\eta(w) := \min\{\ell + 1, \eta(w) + 1\}$ and delete v.*

Updating $\eta(w)$ to the minimum of $\ell+1$ and $\eta(w)+1$ is correct due to the following: if a vertex has any weight at least $\ell+1$, then the vertex is equally excluded from any solution path as having weight $\ell+1$.

Correctness proof. Let $\mathcal{I} = (G, s, t, k, \ell, \lambda, \kappa, \eta)$ be an instance of VW-SSP and let $\mathcal{I}' = (G', s, t, k, \ell, \lambda, \kappa, \eta')$ be the instance of VW-SSP obtained from applying Reduction Rule 8.4. Let $v \in V(T) \setminus Y$ with $T \in \mathcal{T}$ be the vertex deleted by the application of Reduction Rule 8.4. We claim that $\mathcal{I}$ is a yes-instance if and only if $\mathcal{I}'$ is a yes-instance.

($\Rightarrow$) Let P be a solution s-t path in G. We know that v is different from s and t, and, since $v \notin Y$, that $\{w\} = N_T(v) = N_G(v) \setminus R$. Hence, $v \notin V(P)$. If $w \notin V(P)$, then P is a solution s-t path in G'. If $w \in V(P)$, then $\eta'(w) = \eta(w) + 1 \leq \ell$ and

$$\begin{aligned}\sum_{x \in V(P)} \eta'(x) + |N_{G'}(V(P))| &= \sum_{x \in V(P) \setminus \{w\}} \eta(x) + \eta(w) + 1 + |N_G(V(P))| - 1 \\ &\leq \ell.\end{aligned}$$

Hence, P is a solution s-t path in G'.

($\Leftarrow$) Let P be a solution s-t path in G'. Since $G' = G - \{v\}$, we know that P is an s-t path in G with $\sum_{v \in V(P)} \kappa(v) \leq k$. If $w \notin V(P)$, then P is a solution s-t path in G. If $w \in V(P)$, then $\eta(w) + 1 = \eta'(w) \leq \ell$ and $\sum_{x \in V(P)} \eta(x) + |N_G(V(P))| = \sum_{x \in V(P)} \eta'(x) + 1 + |N_{G'}(V(P))| - 1 \leq \ell$. Hence, P is a solution s-t path in G. $\square$

Lemma 8.14. *Reduction Rules 8.3 and 8.4 are exhaustively applicable in linear time.*

Proof. For each tree T in $G - F$, do the following. As long as there is a not-good degree-one vertex $v \in V(T) \setminus Y$, delete v. This is clearly doable in $O(|V(T)|)$ time. When no vertex remains, apply Reduction Rule 8.3. Otherwise, Reduction Rule 8.4 is exhaustively applied on T. Since $\sum_{T \in \mathcal{T}} |V(T)| = |W|$, the claim follows. $\square$

If none of Reduction Rules 8.3 and 8.4 is applicable, the leaves of each tree are all good vertices. Recall that the number of leaves in a tree upper-bounds the number of vertices of degree at least three in the tree. Hence, to upper-bound the number of vertices in the trees, it remains to upper-bound the number of degree-two vertices in the tree. The next data reduction rule deletes these degree-two vertices by shrinking so-called *maximal-edgy paths.*

Definition 8.2. We call an a-b path in a tree T *edgy* if it contains no good vertex and no vertex w with $\deg_T(w) \geq 3$. We call an a-b path Q *maximal-edgy* if there is no edgy path containing Q with more vertices than Q.

Reduction Rule 8.5. *Let $T \in \mathcal{T}$ and let $Q \subseteq T$ be a maximal-edgy a-b path in T with $|V(Q)| > 3$. Let $K := V(Q) \setminus \{a, b\}$. Then, add a vertex x and the edges $\{x, a\}$ and $\{x, b\}$. Set $\kappa(x) := \min\{k + 1, \kappa(K)\}$ and $\eta(x) := \min\{\ell + 1, \eta(K)\}$. For each $w \in N_G(K) \cap R$, add the edge $\{x, w\}$. Delete all vertices in K.*

Correctness proof. Let $\mathcal{I} = (G, s, t, k, \ell, \lambda, \kappa, \eta)$ be an instance of VW-SSP and let $\mathcal{I}' = (G', s, t, k, \ell, \lambda, \kappa', \eta')$ be the instance of VW-SSP obtained from applying Reduction Rule 8.5. Let $T \in \mathcal{T}$ and let $Q \subseteq T$ be the maximal-edgy a-b path in T being changed to the maximal-edgy a-b path Q' by the application of Reduction Rule 8.5. We claim that $\mathcal{I}$ is a yes-instance if and only if $\mathcal{I}'$ is a yes-instance.

($\Rightarrow$) Let $\mathcal{I}$ be a yes-instance and P be a solution path in G. Note that, by construction of G', for each $X \subseteq V(G) \setminus (R \cup V(Q))$, we have $N_G(X) = N_{G'}(X)$. Thus, if $V(Q) \cap V(P) = \emptyset$, then P is also a solution path in G'. Hence, assume that $V(Q) \cap V(P) \neq \emptyset$. Since Q contains no good vertex and no vertex of degree at least three in T, it follows that $V(Q) \subseteq V(P)$. Moreover, we have $\kappa'(x) = \kappa(K) \leq k$ and $\eta'(x) = \eta(K) \leq \ell$. For the path P' in G' obtained from P by replacing $V(Q)$ by $V(Q')$, we have

$$\begin{aligned}
\kappa'(V(P')) &= \kappa'(V(P') \setminus \{x\}) + \kappa'(x) = \kappa(V(P) \setminus K) + \kappa(K) = \kappa(V(P)),\\
N_{G'}(V(P')) &= (N_{G'}(V(P') \setminus \{x\}) \setminus \{x\}) \cup (N_{G'}(x) \setminus \{a, b\})\\
&= (N_G(V(P) \setminus K) \setminus K) \cup (N_G(K) \setminus \{a, b\}) = N_G(V(P)), \text{ and}\\
\eta'(V(P')) &= \eta'(x) + \eta'(V(P') \setminus \{x\}) = \eta(K) + \eta(V(P) \setminus K) = \eta(V(P)).
\end{aligned} \tag{8.1}$$

($\Leftarrow$) Let $\mathcal{I}'$ be a yes-instance and P' be a solution path in G'. If $V(Q') \cap V(P') = \emptyset$, then P' is also a solution path in G. Hence, assume that $V(Q') \cap V(P') \neq \emptyset$. Since Q' contains no good vertex and no vertex of degree at least three in T, it follows that $V(Q') \subseteq V(P')$. Moreover, we have $\kappa'(x) = \kappa(K) \leq k$ and $\eta'(x) = \eta(K) \leq \ell$. Let P be the path in G obtained from P' by replacing $V(Q')$ by $V(Q)$. We have $\kappa(V(P)) = \kappa'(V(P'))$, $N_G(V(P)) = N_{G'}(V(P'))$, and $\eta(V(P)) = \eta'(V(P'))$ by (8.1). □

Lemma 8.15. *If Reduction Rules 8.3 and 8.4 are not applicable, then Reduction Rule 8.5 is exhaustively applicable in linear time. Moreover, no application of Reduction Rule 8.5 makes Reduction Rule 8.3 or Reduction Rule 8.4 applicable again.*

Proof. If Reduction Rules 8.3 and 8.4 are not applicable, then every maximal path of degree-two vertices in $G - F$ containing no good vertices is a maximal-edgy path. Hence, employ the following. Let Z be the set of all degree-two vertices in $G - F$ and Z' be a working copy of Z. As long as $Z' \neq \emptyset$, do the following. Select any vertex $v \in Z'$ and start a breadth-first search that stops when a good vertex or a vertex of degree at least three is found. Apply Reduction Rule 8.5 on the just identified maximal-edgy path (if it contains more than three vertices). Delete all the vertices found in the iteration from Z'.

Since no application of Reduction Rule 8.5 deletes a good vertex or creates a vertex of degree one, no application of Reduction Rule 8.5 makes Reduction Rule 8.3 or Reduction Rule 8.4 applicable. □

We prove next that if none of Reduction Rules 8.3 to 8.5 is applicable, the trees are small in the sense that the number of vertices in the tree is linear in the number of good vertices.

Lemma 8.16. *Let $T \in \mathcal{T}$ be such that none of Reduction Rules 8.3 to 8.5 is applicable. Let $Y_T := Y \cap V(T)$ denote the set of good vertices in T. Then T has $O(|Y_T|)$ vertices, each of weight in $O(k + \ell)$.*

Proof. We first show that, if none of Reduction Rules 8.3 and 8.4 is applicable, then $V(T) = Y_T \uplus V_3 \uplus \biguplus_{Q \in \mathcal{Q}} V(Q)$, where V_3 denotes the set of all vertices w not in Y_T with $\deg_T(w) \geq 3$ and $\mathcal{Q}$ denotes the set of all maximal-edgy paths in T. Note that the sets Y_T, V_3, and $\biguplus_{Q \in \mathcal{Q}} V(Q)$ are pairwise disjoint (by Definition 8.2, no edgy path contains a good vertex or a vertex of degree at least three). Suppose $V(T) = Y_T \uplus V_3 \uplus \biguplus_{Q \in \mathcal{Q}} V(Q) \uplus X$. We show that $X = \emptyset$. Due to Reduction Rules 8.3 and 8.4, the only vertices in T of degree one are good vertices. It follows that X contains only degree-two vertices, none of which are good. Since every vertex in $V(T) \setminus Y_T$ of degree two is contained in a maximal-edgy path, it follows that X is empty. It follows that $V(T) = Y_T \uplus V_3 \uplus \biguplus_{Q \in \mathcal{Q}} V(Q)$. To finish the proof we upper-bound the number of vertices in V_3 and in all paths in $\mathcal{Q}$ linearly in $|Y_T|$.

Again, due to Reduction Rules 8.3 and 8.4, every degree-one vertex is in Y_T. Hence there are at most $|Y_T|$ degree-one vertices in T, and thus $|V_3| \leq |Y_T|$.

Moreover, $|\mathcal{Q}| \leq 2|Y_T|$. Due to Reduction Rule 8.5, for every $Q \in \mathcal{Q}$ we have $|V(Q)| \leq 3$. It follows that $|V(T)| \leq 2|Y_T| + 6|Y_T| = 8|Y_T|$. Due to Reduction Rules 8.4 and 8.5, each vertex v in T has $\kappa(v) \leq k+1$ and $\eta(v) \leq \ell+1$, and hence is of weight in $O(k + \ell)$. □

We are now ready to prove Proposition 8.12. In a nutshell, we approximate a minimum feedback vertex set in linear time [Bar+98], then apply Reduction Rules 8.3 to 8.5 exhaustively in linear time (Lemmas 8.14 and 8.15), and finish the proof using Lemma 8.16.

Proof of Proposition 8.12. Compute a feedback vertex set F of size $\beta \leq 4 \cdot \mathrm{fvs}$ in linear time [Bar+98]. Then apply Reduction Rules 8.3 and 8.4 exhaustively in linear time (Lemma 8.14), and finally Reduction Rule 8.5 exhaustively in linear time (Lemma 8.15).

Now, consider a graph G to which no data reduction rules are applicable and let $T_1, \ldots, T_h$ denote the trees in $G - F$. By Lemma 8.16, each T_i has $O(|Y_{T_i}|)$ vertices, each of maximal weight in $O(k + \ell)$, where $Y_{T_i} = Y \cap V(T_i)$. Thus, the number of vertices and edges in $G - F$ is

$$\sum_{i=1}^{h} O(|Y_{T_i}|) = \sum_{i=1}^{h} O(|Y \cap V(T_i)|) = O(|Y|) \subseteq O(\beta \cdot (k + \ell)),$$

where the last inclusion follows from Observation 8.13. It follows that there are $O(\beta^2 \cdot (k + \ell))$ edges in G. Altogether, G has $O(\beta \cdot (k + \ell))$ vertices, each of weight in $O(k + \ell)$, and $O(\beta^2 \cdot (k + \ell))$ edges. Moreover, the obtained instance is simplified (with $A = R$, see Definition 8.1). □

Combining Proposition 8.12 with Proposition 8.3, we now prove the main result of this section.

Proof of Theorem 8.11. Let $\mathcal{I} = (G, s, t, k, \ell)$ be an instance of SSP. Employ first Proposition 8.12 to obtain a simplified instance $\mathcal{I}' := (G', s, t, k, \ell, \lambda, \kappa, \eta)$ of VW-SSP, and then Proposition 8.3 to obtain instance $\mathcal{I}'' := (G'', s', t', k', \ell)$ of SSP. We know that G' has $O(\mathrm{fvs} \cdot (k + \ell))$ vertices, $O(\mathrm{fvs}^2 \cdot (k + \ell))$ edges, and vertex weights in $O(k + \ell)$. Hence,

$$\begin{aligned} |V(G'')| = \kappa(V(G')) + \eta(V(G')) &\in O(\mathrm{fvs} \cdot (k + \ell) \cdot k + \mathrm{fvs} \cdot (k + \ell) \cdot \ell) \\ &\subseteq O(\mathrm{fvs} \cdot (k + \ell)^2). \end{aligned}$$ □

8.5.2. Polynomial Kernelization Lower Bounds regarding $\mathrm{fvs} + \ell$

In the previous section, we proved a kernel for SSP with size polynomial in $\mathrm{fvs} + k + \ell$. We will see that, unless $\mathrm{coNP} \subseteq \mathrm{NP}_{/\mathrm{poly}}$, we cannot drop ℓ here, as a kernel with size polynomial in $\mathrm{fvs} + k$ would also be polynomial in vc (Theorem 8.28, see also Remark 8.1). In this section, we prove that, unless $\mathrm{coNP} \subseteq \mathrm{NP}_{/\mathrm{poly}}$, we cannot drop k either:

Theorem 8.17. *Unless coNP $\subseteq$ NP$_{/poly}$,* Short Secluded Path *admits no kernel with size polynomial in* $\mathrm{fvs} + \ell$*.*

To prove Theorem 8.17, we OR-cross-compose (see Definition 1.5) the Multicolored Clique problem into SSP.

Multicolored Clique (MCC)
Input: An undirected k-partite graph $G = (V = V_1 \uplus \ldots \uplus V_k, E)$.
Question: Is there an vertex set $X \subseteq V$ of size k in G with $|X \cap V_i| = 1$ for all $i \in \{1, \ldots, k\}$ and all vertices in X are pairwise adjacent in G?

In fact, we will cross-compose the NP-hard [Cyg+15, Fel+09] special case of Multicolored Clique where instances $G = (V = V_1 \uplus \ldots \uplus V_k, E)$ with $E_{\{i,j\}} := \{\{u, v\} \in E \mid u \in V_i, v \in V_j\}$ satisfy

- $|V_i| = |V_j|$ for $1 \le i < j \le k$,
- $|E_{\{i,j\}}| = |E_{\{i',j'\}}|$ for all $1 \le i < j \le k$ and $1 \le i' < j' \le k$, and
- have at least $k + 1$ vertices.

For the OR-cross-composition, we use the following polynomial equivalence relation on instances of Multicolored Clique.

Lemma 8.18. *Let two* Multicolored Clique *instances $G = (V = V_1 \uplus \ldots \uplus V_k, E)$ and $G' = (V' = V'_1 \uplus \ldots \uplus V'_{k'}, E')$ be $\mathcal{R}$-*equivalent *if and only if $|V| = |V'|$, $|E| = |E'|$, and $k = k'$. Then, $\mathcal{R}$ is a polynomial equivalence relation.*

Proof. Deciding whether G and G' are $\mathcal{R}$-equivalent is doable in $O(|V| + |V'| + |E| + |E'|)$ time. Now, let $S \subseteq \Sigma^*$ be a set of instances and $n := \max_{x \in S} |x|$. There are at most $n^{O(1)}$ different vertex set sizes, edge set sizes, and partition sizes of the vertex sets, resulting in at most $n^{O(1)}$ equivalence classes. □

We next describe the OR-cross-composition.

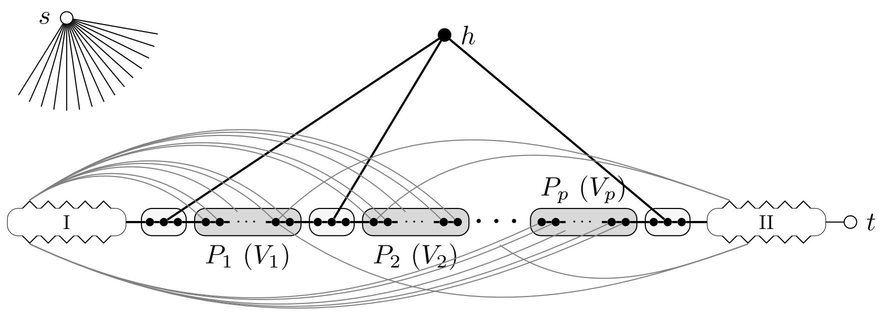


Figure 8.7.: A high-level sketch of Construction 8.19. Vertices enclosed in rectangles are added in step 3, vertex h is added in step 4, vertices s and t are added in step 7, and some illustrative edges between the gadgets are drawn which are introduced in steps 5 and 6 of Construction 8.19. For illustrations and details of the gadgets labeled I and II, see Figure 8.8(a) and (b), respectively.

Construction 8.19. Let $G_1 = (V_1 = V_{1,1} \uplus \ldots \uplus V_{1,k}, E_1), \ldots, G_p = (V_p = V_{p,1} \uplus \ldots \uplus V_{p,k}, E_p)$ be $p = 2^q$, $q \in \mathbb{N}$, $\mathcal{R}$-equivalent Multicolored Clique instances. Let n denote the number of vertices and by m denote the number of edges in each instance. Moreover, let $V_{a,i} = \{v^1_{a,i}, \ldots, v^r_{a,i}\}$, and $E_a = \biguplus_{1 \le i < j \le k} E_{a,\{i,j\}}$ with $E_{a,\{i,j\}} = \{e^1_{a,\{i,j\}}, \ldots, e^x_{a,\{i,j\}}\}$ for all $a \in \{1, \ldots, p\}$. Construct the following SSP instance (G, s, t, k', ℓ) with graph G (refer to Figures 8.7 and 8.8 for an illustration). Let G be initially empty, and let

$$K := \binom{k}{2},$$
$$M := k + |E| - K + q + 2, \text{ and}$$
$$L := p \cdot n \cdot m + 2 \cdot (M + K \cdot M^2 + q \cdot M^2).$$

We build G step by step:

1. Add $q+1$ paths $A_1, \ldots, A_{q+1}$, where $V(A_y) = \{a_{y,1}, \ldots, a_{y,L}\}$ with $a_{y,1}$ and $a_{y,L}$ being the endpoints. For each $y \in \{1, \ldots, q\}$, add the vertex set $U_y = \{u_{y,0}, u_{y,1}\}$, and make each vertex of U_y adjacent to $a_{y,L}$ and $a_{y+1,1}$. Define $U := \bigcup_{y=1}^{q} U_y$. See Figure 8.8(a).

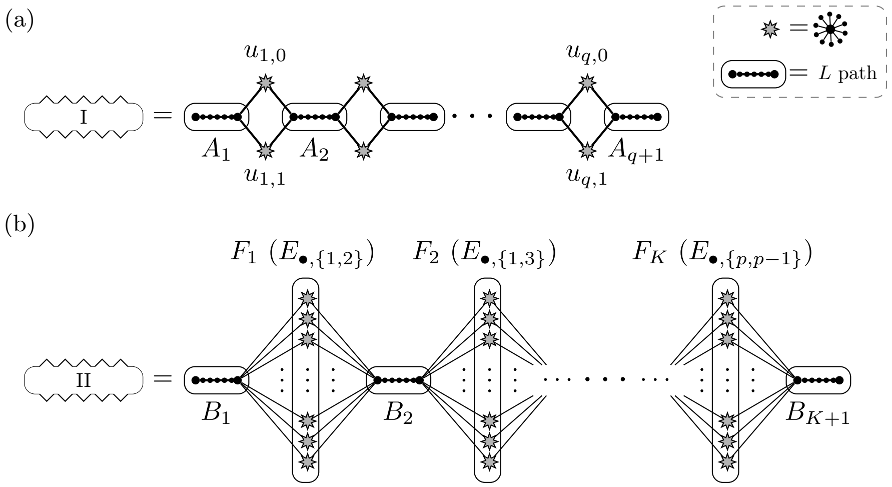


Figure 8.8.: Illustrations and details for the gadgets labeled I and II in Figure 8.7. (a) Details for the gadget labeled I which is introduced in step 1 of Construction 8.19. (b) Details for the gadget labeled II which is introduced in step 2 of Construction 8.19. For P_a and F_y, in parentheses we indicate to which sets of the input graphs they correspond (where $\bullet$ is a placeholder for every element in $\{1, \dots, p\}$). Each star shape in (a) and (b) depicts a star graph which is introduced in step 8 of Construction 8.19.

2. Add $K + 1$ paths $B_1, \dots, B_{K+1}$, where $V(B_z) = \{b_{z,1}, \dots, b_{z,L}\}$ with $b_{z,1}$ and $b_{z,L}$ being the endpoints. For each $z \in \{1, \dots, K\}$, add the vertex set $F_z = \{e_z^1, \dots, e_z^x\}$ and make each vertex in F_z adjacent to $b_{z,L}, b_{z+1,1}$. Define $F := \bigcup_{z=1}^{K} F_z$. Choose an arbitrary bijection $\pi\colon \{1, \dots, K\} \to \{\{i, j\} \mid 1 \le i < j \le k\}$. We say that e_y^z corresponds to the z-th edge $e^z_{a,\pi(y)} \in E_{a,\pi(y)}$ for all $a \in \{1, \dots, p\}$. See Figure 8.8(b).

3. Add p paths $P_1, \dots, P_p$ such that P_a has vertex set

$$\begin{aligned}
V(P_a) &= \{v_{a,i}^d \mid i \in \{1, \dots, k\},\ d \in \{1, \dots, r\}\} \text{ and edge set}\\
E(P_a) &= \{\{v_{a,i}^r, v_{a,i+1}^1\} \mid i \in \{1, \dots, k-1\}\}\\
&\qquad \cup \bigcup_{1 \le i \le k} \{\{v_{a,i}^d, v_{a,i}^{d+1}\} \mid d \in \{1, \dots, r-1\}\}.
\end{aligned}$$

We say P_a corresponds to the vertices in V_a in the a-th graph G_a. Next, for each $a \in \{1, \dots, p+1\}$, add a path of three vertices $w_{a,1}, w_{a,2}, w_{a,3}$ with edges $\{w_{a,1}, w_{a,2}\}, \{w_{a,2}, w_{a,3}\}$. Make $w_{1,1}$ adjacent to $a_{q+1,L}$, and $w_{p+1,3}$ adjacent to $b_{1,1}$. For each $1 < a \leq p+1$, make $w_{a,1}$ adjacent to $v^r_{a-1,k}$. For each $1 \leq a < p+1$, make $w_{a,3}$ adjacent to $v^1_{a,1}$.

4. Add one vertex h and for each $a \in \{1, \dots, p+1\}$ make $w_{a,2}$ adjacent to h.

5. For each $a \in \{1, \dots, p\}$, make each $v \in V(P_a)$ adjacent to the vertex in F corresponding to an incident edge. That is, if $v^d_{a,i}$ is incident with edge $e^{x'}_{a,\{i,j\}}$, then make $v^d_{a,i}$ adjacent to vertex $e^{x'}_z$ where $z = \pi^{-1}(\{i,j\})$.

6. For each $a \in \{1, \dots, p\}$, make each $v \in V(P_a)$ adjacent to the vertices in U as follows: Let $a_1 a_2 \cdots a_q$ be the 0-1-string of length q encoding the number $a-1$ in binary. Then, make each $v \in V(P_a)$ adjacent to each vertex in the set $\{u_{i,a_i} \mid i \in \{1, \dots, q\}\}$. Note that for each $u \in U$, we have that if $N(u) \cap V(P_a) \neq \emptyset$ for some $a \in \{1, \dots, p\}$, then $N(u) \supseteq V(P_a)$. Moreover, for each $u \in U$ we have $|\{a \in \{1, \dots, p\} \mid N(u) \cap V(P_a) \neq \emptyset\}| = p/2$.

7. Add s and t. Make t adjacent to $b_{K+1,L}$. Make s adjacent to all vertices except the vertices in $\bigcup_{a=1}^{p} V(P_a)$.

8. For each vertex $v \in F \cup U$, add M^2 vertices only adjacent to v.

Finally, set $k' := (q+K+2)L + q + (p-1)n + 3(p+1) + K + 1$ and $\ell := M + K \cdot M^2 + q \cdot M^2$. ▲

Before we prove that the instance $\mathcal{I}$ obtained from Construction 8.19 is a yes-instance if and only if at least one input instance is a yes-instance, we prove some crucial properties of solutions to $\mathcal{I}$ in the case that $\mathcal{I}$ is a yes-instance.

Lemma 8.20. *Let (G, s, t, k', ℓ) be the* SSP*-instance obtained from Construction 8.19 and let (G, s, t, k', ℓ) be a* yes*-instance. Let P be a solution s-t path in G. Then the following hold:*

(i) P contains each path $Q \in \{A_2, \dots, A_{q+1}, B_1, \dots, B_{K+1}\}$ and a subpath of A_1 as subpath. Moreover, the first vertex on P after s is in $V(A_1) \setminus \{a_{1,L}\}$.

(ii) $|V(P) \cap U_y| = |V(P) \cap F_z| = 1$ *for all* $y \in \{1, \dots, q\}$, $z \in \{1, \dots, K\}$.

(iii) *Let* $v \in U_y$ *for some* $y \in \{1, \dots, q\}$ *be contained in the path* P*, and let* $(\{v', v, v''\}, \{\{v', v\}, \{v, v''\}\})$ *be a subpath of* P *where the distance from* v' *to* s *in* P *is smaller than the one from* v *or* v''*. Then* $v' = a_{y,L}$ *and* $v'' = a_{y+1,1}$.

(iv) *Let* $v \in F_z$ *for some* $z \in \{1, \dots, K\}$ *be contained in the path* P*, and let* $(\{v', v, v''\}, \{\{v', v\}, \{v, v''\}\})$ *be a subpath of* P *where the distance from* v' *to* s *in* P *is smaller than the one from* v *or* v''*. Then* $v' = b_{z,L}$ *and* $v'' = b_{z+1,1}$.

Proof. (i): From each path $Q \in \{A_2, \dots, A_{q+1}, B_1, \dots, B_{K+1}\}$, at least $L - \ell > \ell$ vertices must be contained. Since the inner vertices of Q are only adjacent to vertices in Q and s, it follows that Q is a subpath of P. Moreover, also at least $L - \ell > \ell$ vertices from A_1 must be contained in P. Hence, a subpath of A_1 is a subpath of P. From the latter, we observe that the first vertex on P after s is in $V(A_1) \setminus \{a_{1,L}\}$.

(ii): From (i), we know that each path $Q \in \{A_2, \dots, A_{q+1}, B_1, \dots, B_{K+1}\}$ is a subpath of P, and the first vertex on P after s is in $V(A_1) \setminus \{a_{1,L}\}$. If $Q = A_y$, $2 \le y \le q + 1$, then we know that $a_{y,1}$ is only incident with vertices in $U_{y-1} \cup \{s\} \cup \{a_{y,2}\}$. It follows that for each U_y at least one vertex is contained in P. If $Q = B_z$, $2 \le z \le K+1$, then we know that $b_{z,1}$ is only incident with vertices in $F_{z-1} \cup \{s\} \cup \{b_{z,2}\}$. It follows that for each F_z at least one vertex is contained in P. Suppose there is a set $X \in \{U_1, \dots, U_q, F_1, \dots, F_K\}$ such that at least two vertices from X are contained in P. Recall that, by construction, each vertex in $U \cup F$ has M^2 degree-one neighbors. Then P has at least $M^2 \cdot \binom{k}{2} + M^2 \cdot q + M^2 > \ell$ neighbors, yielding a contradiction. Hence, we know that for each U_i and F_j exactly one vertex is contained in P.

(iii): Let $v \in U_y$ for some $y \in \{1, \dots, q\}$. Suppose that $v' \neq a_{y,L}$ (for v'', this works analogously). We know that P contains A_i as a subpath. Hence, $a_{y,L}$ is adjacent to the other vertex in $U_y \setminus \{v\}$ on P, yielding a contradiction to (ii).

(iv): In the same way as (iii), we can prove the claim for $v \in F_z$ for some $z \in \{1, \dots, K\}$. □

We next prove that the instance obtained from Construction 8.19 is a yes-instance if and only if at least one input instance is a yes-instance.

Lemma 8.21. *Let* $G_1, \dots, G_p$ *be* $p = 2^q$ *instances of* Multicolored Clique *that are* $\mathcal{R}$*-equivalent, where* $q \in \mathbb{N}$*. Let* (G, s, t, k', ℓ) *be the* SSP*-instance* *obtained from Construction 8.19. Then at least one instance G_a is a* `yes`*-instance if and only if (G, s, t, k', ℓ) is* `yes`*-instance for* SSP*.*

Proof. ($\Rightarrow$) Let G_a be a `yes`-instance for some $a \in \{1, \ldots, p\}$ and let C be a clique of order k in G_a. Construct an s-t path P as follows: P starts at s, then goes to $a_{1,1}$, follows along the vertices only in $A_1, \ldots, A_{q+1}$ and U until $a_{q+1,L}$, while selecting the vertices in U such that only the vertices corresponding to $V(G_a)$ are not in the neighborhood yet. This is possible since, for each $b \in \{1, \ldots, p\}$, only one of $u_{y,0}$ and $u_{y,1}$ is adjacent to the vertices in $V(P_b)$. Next, follow the vertices in $V(P_1), \ldots, V(P_p)$, avoiding the vertices in $V(P_a)$ by using $w_{a,2}, h, w_{a+1,2}$. Then follow, starting at $b_{1,1}$ towards $b_{K+1,L}$ and then to t by only selecting the vertices corresponding to the edges in C. This path contains

2	vertices s and t,
$(q+1) \cdot L$	vertices which are all vertices from the set $A_1 \uplus \ldots \uplus A_{q+1}$,
q	vertices from the set U,
$(p-1) \cdot n$	vertices which are all vertices from $\biguplus_{b \in \{1,\ldots,p\} \setminus \{a\}} V(P_a)$,
$3(p+1) - 1$	vertices which are all vertices from $w_{a,1}$, $w_{a,2}$, h, $w_{a+1,2}$, $w_{a+1,3}$, and $\biguplus_{b \in \{1,\ldots,p\} \setminus \{a,a+1\}} \{w_{b,1}, w_{b,2}, w_{b,3}\}$,
$(K+1) \cdot L$	vertices which are all vertices from the set $B_1 \uplus \ldots \uplus B_{K+1}$, and
K	vertices, one from each F_z, $z \in \{1, \ldots, K\}$.

That is, P contains

$$2 + (q+1) \cdot L + q + (p-1) \cdot n + (3(p+1) - 1) + (K+1) \cdot L + K \leq k'$$

vertices. Moreover, path P is neighboring

$q \cdot M^2$	degree-one vertices neighboring U, i.e., M^2 degree-one vertices from each of the q vertices from U in P,
$K \cdot M^2$	degree-one vertices neighboring F, i.e., M^2 degree-one vertices from each of the K vertices from F in P,
k	vertices on the path P_a (those corresponding to the vertices of clique C),
$\lvert E \rvert - K$	vertices in F,
q	vertices from U, and
2	vertices $w_{a,3}$ and $w_{a+1,1}$.

That is, P is neighboring

$$q \cdot M^2 + K \cdot M^2 + k + |E| - K + q + 2 = q \cdot M^2 + K \cdot M^2 + M \leq \ell$$

vertices. Hence, P is a solution s-t path in G.

($\Leftarrow$) Let (G,s,t,k',ℓ) be a yes-instance for SSP. Let P be a solution s-t path. We claim that if P contains a vertex in $V(P_a)$ for some $a \in \{1,\dots,p\}$, then it contains all vertices in $V(P_a)$. Suppose not, that is, there is an $a \in \{1,\dots,p\}$ such that $1 \leq |V(P) \cap V(P_a)| < n$. Note that $N(V(P_a)) \subseteq U \cup F \cup \{w_{a,3}, w_{a+1,1}\}$. Since $1 \leq |V(P) \cap V(P_a)| < n$, there is a vertex $v \in V(P_a) \cap V(P)$ such that at least one of its neighbors in $V(P_a)$ is not contained in $V(P)$. It follows that in P, v is adjacent to a vertex in $U \cup F$. This contradicts Lemma 8.20(iii).

From Lemma 8.20(ii) and (iii), we know that P contains $|E| - \binom{k}{2} + q + M^2 \cdot \binom{k}{2} + M^2 \cdot q$ neighbors not contained in $A_1 \cup \bigcup_{a=1}^{p} V(P_a)$. By the values of k' and ℓ, we know that either exactly one P_a is not contained in P, or there are $n+2$ vertices from A_1 being not contained in P. In the latter case, we have at least

$$n + 2 + |E| - \binom{k}{2} + q + M^2 \cdot \binom{k}{2} + M^2 \cdot q > M + M^2 \cdot \binom{k}{2} + M^2 \cdot q = \ell$$

neighbors (recall that $n > k$), yielding a contradiction. It follows the former case: there is exactly one P_a being not contained in P. It follows that $h \in V(P)$ and $w_{a,3}, w_{a+1,1} \in N(V(P))$.

By Lemma 8.20(ii), from each F_z there is exactly one vertex contained in P. Moreover, for each $z \in \{1,\dots,K\}$ and for each $v \in F_z$ it holds true that $|V(P_a) \cap N(v)| \geq 2$. Hence, $|N(V(P)) \cap V(P_a)| \geq k$, as K edges cannot be distributed among fewer than k vertices. It follows that A_1 is a subpath of P.

Since $|N(V(P)) \setminus V(P_a)| = |E| - \binom{k}{2} + q + M^2 \cdot \binom{k}{2} + M^2 \cdot q + 2$, it follows that there must be exactly k vertices in $V(P_a)$ neighboring P. This witnesses a clique of order k in G_a, and the statement follows. □

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

Proof of Theorem 8.17. Due to Lemma 8.18, we know that $\mathcal{R}$ is a polynomial equivalence relation on the instances of Multicolored Clique. Let $G_1,\dots,G_p$ be $p = 2^q$, $q \in \mathbb{N}$, $\mathcal{R}$-equivalent instances of Multicolored Clique. We construct an instance (G,s,t,k',ℓ) of SSP by applying Construction 8.19 in time polynomial in $\sum_{a=1}^{p} |G_a|$. By Lemma 8.21, we have that (G,s,t,k',ℓ) is a yes-instance if and only if G_a is a yes-instance for some $a \in \{1,\dots,p\}$. The set $W := U \cup F \cup \{s,h,t\}$ forms a feedback vertex set with $|W| \leq 2\log p + K \cdot x$, that is, $|W|$ is upper-bounded by a polynomial in $|G_a| + \log p$ for any $a \in \{1,\dots,p\}$. Moreover, $\ell = M + M^2 \cdot \binom{k}{2} + M^2 \log p$, where $M := k + |E| - \binom{k}{2} + \log p + 2$ is upper-bounded by a polynomial in $|G_a| + \log p$. Altogether, we described an

OR-cross-composition from Multicolored Clique into SSP parameterized by $\mathrm{fvs} + \ell$, and the statement follows. □

8.6. Vertex Cover Number

In the previous section, we proved a kernel for Short Secluded Path with number of vertices being cubic in $\mathrm{fvs} + k + \ell$. This gives a kernel for SSP with number of vertices cubic in $\mathrm{vc}+\ell$ (recall that vc denotes the vertex cover number): We have $\mathrm{fvs} \leq \mathrm{vc}$, and it is not hard to see that one can assume $k \leq 2\mathrm{vc} + 1$.

Remark 8.1. A vertex cover contains at least $\lfloor k/2 \rfloor$ vertices of a path with k vertices. Thus, a polynomial parameter transformation of SSP parameterized by vc to SSP parameterized by $\mathrm{vc} + k$ can safely reduce k so that $k \leq 2\mathrm{vc} + 1$.

First, in Section 8.6.1, we strengthen the previously mentioned (cubic) kernel and prove a kernel with number of vertices quadratic in $\mathrm{vc} + \ell$. Secondly, in Section 8.6.2, we prove a polynomial kernel whenever the input graph is $K_{r,r}$-subgraph-free, with r being a constant. Finally, in Section 8.6.3, we prove that presumably, we can drop neither ℓ from the parameterization $\mathrm{vc} + \ell$ nor the restriction on the input graph ($K_{r,r}$-subgraph-freeness) to obtain polynomial kernelization regarding vc.

8.6.1. A Polynomial Kernel with $O(\mathrm{vc} \cdot (\mathrm{vc} + \ell))$ Vertices

In this section, we prove the following polynomial kernel.

Theorem 8.22. Short Secluded Path *admits a linear-time computable kernel with $O(\mathrm{vc} \cdot (\mathrm{vc} + \ell))$ vertices.*

Let $(G = (V, E), s, t, k, \ell)$ be an instance of SSP. Let $C \subseteq V$ be a vertex cover of size vc. Define the set

$$R := \{v \in V \mid \deg_G(v) \geq \ell + 2\mathrm{vc} + 2\} \subseteq V$$

of vertices $v \in V$ of degree at least $\ell + 2\mathrm{vc} + 2$ in the instance graph. Note that $R \subseteq C$ and hence $|R| \leq \mathrm{vc}$. Note that no vertex from R can appear in a solution path, hence we can do the following (which we will use later on).

Reduction Rule 8.6. *For each $v \in R$, add a set W_v of $\ell + 1$ vertices and make each only adjacent with v.*

Note that we can apply Reduction Rule 8.6 in linear time. The correctness of Reduction Rule 8.6 results immediately from the following.

Observation 8.23. *Before and after the application of Reduction Rule 8.6, no vertex in R is contained in an s-t path P with $|V(P)| \leq k$ and $|N(V(P))| \leq \ell$.*

Proof. Let G and G' denote the graph before and after the application of Reduction Rule 8.6, respectively. Let $v \in R$. Since $\deg_G(v) \geq \ell + 2\mathrm{vc} + 2$ and $k \leq 2\mathrm{vc} + 1$, for every s-t path P with $|V(P)| \leq k$ and $v \in V(P)$ it holds true that $|N_G(P)| \geq |N_G(v)| - |V(P)| \geq \ell + 1$. In G', v has $\ell + 1$ degree-1 neighbors all distinct to s and t. Hence, for every s-t path P with $v \in V(P)$ it holds true that $|N_{G'}(P)| \geq \ell + 1$. It follows that every vertex of R is excluded from every s-t path P with $|V(P)| \leq k$ and $|N(V(P))| \leq \ell$ in G or in G'. ☐

Clearly, if a vertex is only neighboring vertices excluded from every solution path (that is, contained in R), then this vertex is also excluded from every solution path. We make use of this observation in the following way.

Reduction Rule 8.7. *If there is a vertex $v \in V \setminus \bigcup_{v \in R}(\{s, t, v\} \cup W_v)$ with $N(v) \subseteq R$, then delete v.*

Lemma 8.24. *Reduction Rule 8.7 is correct and can be exhaustively applied in linear time.*

Proof. *(Correctness)* Let $G = (V, E)$ and G' denote the graph before and after application of Reduction Rule 8.7, respectively, and let $v \in V \setminus \bigcup_{v \in R}(\{s, t, v\} \cup W_v)$ such that $v \in V(G) \setminus V(G')$. We claim that (G, s, t, k, ℓ) is a `yes`-instance of SSP if and only if (G', s, t, k, ℓ) is a `yes`-instance of SSP.

($\Rightarrow$) Let P be a solution s-t path in G. Since $R \cap V(P) = \emptyset$ (Observation 8.23) and $N_G(v) \subseteq R$, we have $N_G(v) \cap V(P) = \emptyset$ and hence $v \notin V(P)$. Thus, P is an s-t path in G'. Moreover, since $N_G(v) \cap V(P) = \emptyset$, we have $|N_G(V(P))| = |N_{G'}(V(P))|$. Hence, P is also a solution s-t path in G'.

($\Leftarrow$) Let P be a solution s-t path in G'. Since $V(G') \subset V(G)$, P is also an s-t path in G. Moreover, since $R \cap V(P) = \emptyset$ (Observation 8.23) and $N_G(v) \subseteq R$, we have $N_G(v) \cap V(P) = \emptyset$ and hence $|N_G(V(P))| = |N_{G'}(V(P))|$. It follows that P is also a solution s-t path in G.

(Running time) We can find and delete all vertices $v \in V \setminus \bigcup_{v \in R}(\{s, t, v\} \cup W_v)$ with $N(v) \subseteq R$ in linear time. Since deleting any of these vertices only affects vertices in R, Reduction Rule 8.7 is not applicable anymore. ☐

Proof of Theorem 8.22. Let $\mathcal{I} = (G = (V, E), s, t, k, \ell)$ be an instance of SSP and let $\mathrm{vc} := \mathrm{vc}(G)$. If for $v \in \{s, t\}$ we have that $v \in R$ or $N(v) \subseteq R$, then output a trivial `no`-instance (due to Observation 8.23). Let $\mathcal{I}' = (G' = (V', E'), s, t, k, \ell)$ be the instance obtained from $\mathcal{I}$ by applying first Reduction Rule 8.6 and then Reduction Rule 8.7 exhaustively. Due to the correctness of Reduction Rules 8.6 and 8.7, we know that $\mathcal{I}$ is a `yes`-instance of SSP if and only if $\mathcal{I}'$ is a `yes`-instance of SSP. It remains to prove that the number of vertices of G' is in $O(\mathrm{vc} \cdot (\mathrm{vc} + \ell))$.

Note that $R \subseteq V(G')$. Let $W := \bigcup_{v \in R} W_v$ and let C' be a minimum-cardinality vertex cover for $G' - W$. Note that $|C'| \leq \mathrm{vc}$ since $G' - W \subseteq G$. Due to Reduction Rule 8.7, each vertex in $V' \setminus (R \cup W)$ has at least one neighbor not contained in $R \cup W$, Hence, each vertex in the independent set $V' \setminus (C' \cup W)$ has a neighbor in $C' \setminus R$. We have:

$$\begin{aligned} |V' \setminus (C' \cup W)| &\leq |C' \setminus R| \cdot (2\mathrm{vc} + \ell + 1) \leq \mathrm{vc} \cdot (2\mathrm{vc} + \ell + 1), \\ |(C' \cup W)| &\leq |C'| + |R| \cdot (\ell + 1) \leq \mathrm{vc} + \mathrm{vc} \cdot (\ell + 1), \end{aligned}$$

and finally, $|V'| = |V' \setminus (C' \cup W)| + |(C' \cup W)| \in O(\mathrm{vc} \cdot (\mathrm{vc} + \ell))$. □

8.6.2. A Polynomial Kernel for Planar Graphs

In this section, we show that we can reduce any instance of SSP in $K_{r,r}$-subgraph-free graphs to an equivalent instance with size polynomial in the vertex cover number of the input graph. In the next section, we prove that this does not generalize to general graphs.

Theorem 8.25. *For each constant $r \in \mathbb{N}$,* Short Secluded Path *in $K_{r,r}$-subgraph-free graphs admits a kernel with size polynomial in the vertex cover number of the input graph.*

Note that SSP is trivially polynomial-time solvable in $K_{1,1}$-subgraph-free graphs. The proof of Theorem 8.25 for $r \geq 2$ consists of basically the same three steps our technique from Section 8.3 consists of (see Figure 8.2). However, note that herein, we will not construct and reduce back from a simplified instance of VW-SSP. We explain the three steps briefly.

1. In linear time, we transform an n-vertex instance of SSP into an equivalent instance of VW-SSP with $O(\mathrm{vc}^r)$ vertices.

2. Using Lemma 8.2, in polynomial time, we shrink the vertex weights to $2^{O(\mathrm{vc}^{3r})}$ so that their encoding length is in $O(\mathrm{vc}^{3r})$.

3. Since SSP is NP-complete in $K_{r,r}$-subgraph-free graphs with $r \geq 2$ [BFT20, LF20] we can, in polynomial time, reduce the shrunk instance back to an instance of the unweighted SSP in $K_{r,r}$-subgraph-free graphs. Since the reduction runs in polynomial time, the obtained instance is of size polynomial in vc.

Our data reduction will be based on removing twins. As the first step towards proving Theorem 8.25, we will show that the following data reduction rule, when applied to an instance of SSP with a $K_{r,r}$-subgraph-free graph for constant r, gives an instance of VW-SSP with $O(\mathrm{vc}^r)$ vertices.

Reduction Rule 8.8. *Let $(G = (V, E), s, t, k, \ell, \kappa, \lambda, \eta)$ be an instance of* VW-SSP *with unit weights κ and λ, and zero weights η, and G being a $K_{r,r}$-subgraph-free graph. For each maximal set $U \subseteq V \setminus \{s, t\}$ of twins such that $|U| > r$, delete $|U| - r$ vertices of U from G, and, for an arbitrary remaining vertex $v \in U$, set $\lambda(v) := |U| - r + 1$ and $\kappa(v) := k + 1$.*

Lemma 8.26. *Reduction Rule 8.8 is correct and can be applied in linear time.*

Proof. All maximal sets of twins can be computed in linear time [HPV98]. It is now easy to check which of them has size larger than r and to apply Reduction Rule 8.8.

To prove that Reduction Rule 8.8 is correct, we prove that its input instance $\mathcal{I} = (G, s, t, k, \ell, \kappa, \lambda, \eta)$ is a yes-instance if and only if its output instance $\mathcal{I}' = (G', s, t, k, \ell, \kappa', \lambda', \eta)$ is a yes-instance. Herein, note that η is the zero function, so we will ignore it in the rest of the proof.

($\Rightarrow$) Let $\mathcal{I}$ be a yes-instance and let P be a solution s-t path such that $\sum_{v \in V(P)} \kappa(v) \leq k$ and $\sum_{v \in N(V(P))} \lambda(v) \leq \ell$ in G. Let $U \subseteq V \setminus \{s, t\}$ be an arbitrary set of twins with $|U| > r$. Since G is $K_{r,r}$-subgraph-free, it holds true that $|N(U)| \leq r - 1$. Thus, P contains at most $|N(U)| - 1 \leq r - 2$ vertices of U. Reduction Rule 8.8 reduces U to a set U' with r vertices, where only one of the vertices $v \in U'$ has weight $\kappa'(v) > 1$. Thus, without loss of generality, we can assume that P uses only the $r - 1$ vertices $v \in U \cap U'$ with $\kappa'(v) = 1$. Hence,

$$\sum_{v \in V(P) \cap U} \kappa(v) = \sum_{v \in V(P) \cap U} \kappa'(v) = |V(P) \cap U|. \tag{8.2}$$

Moreover, if P uses a vertex of U, then it also uses a vertex of $N(U)$ and, hence, $U \setminus V(P) \subseteq N(V(P))$. Thus,

$$\sum_{v\in N_G(V(P))\cap U} \lambda(v) = \sum_{v\in U\setminus V(P)} \lambda(v) = \sum_{v\in U'\setminus V(P)} \lambda'(v) = \sum_{v\in N_{G'}(V(P))\cap U'} \lambda'(v) \tag{8.3}$$

since $|U \setminus U'| = |U| - r$ and there is a vertex $v \in U' \cap U$ that has $\lambda(v) = 1$ on the left-hand side of (8.3) but $\lambda'(v) = |U| - r + 1$ on the right-hand side of (8.3). From (8.2), (8.3), and the arbitrary choice of U, it follows that P is an s-t path with $\sum_{v\in V(P)} \kappa'(v) \le k$ and $\sum_{v\in N(V(P))} \lambda'(v) \le \ell$ in G'. Thus, $\mathcal{I}'$ is a yes-instance.

($\Leftarrow$) Let $\mathcal{I}'$ be a yes-instance and let P be a solution s-t path such that $\sum_{v\in V(P)} \kappa'(v) \le k$ and $\sum_{v\in N(V(P))} \lambda'(v) \le \ell$ in G'. Let $U \subseteq V \setminus \{s,t\}$ be a set of twins in G reduced to a subset U' in G' by Reduction Rule 8.8. The only vertex $v \in U'$ with weight $\kappa'(v) > 1 = \kappa(v)$ has $\kappa'(v) = k+1$ and thus is not on P. Yet, if P uses vertices of U', then $v \in U' \setminus V(P) \subseteq N_{G'}(V(P))$ and $U \setminus V(P) \subseteq N_G(V(P))$. Thus, (8.2) and (8.3) apply and, together with the arbitrary choice of U, show that P is an s-t path with $\sum_{v\in V(P)} \kappa(v) \le k$ and $\sum_{v\in N(V(P))} \lambda(v) \le \ell$ in G and, thus, $\mathcal{I}$ is a yes-instance. $\square$

We now prove the upper bound on the number of vertices that remain after Reduction Rule 8.8.

Proposition 8.27. *Applied to an instance of* SSP *with a $K_{r,r}$-subgraph-free graph with vertex cover number* vc, *Reduction Rules 8.1 and 8.8 yield an instance of* VW-SSP *on at most* $(\mathrm{vc}+2) + r \cdot (\mathrm{vc}+2)^r$ *vertices in linear time.*

Proof. Let $(G' = (V', E'), s, t, k, \ell, \lambda', \kappa', \eta)$ be the instance obtained from applying Reduction Rules 8.1 and 8.8 to an instance $(G, s, t, k, \ell, \lambda, \kappa, \eta)$.

Let C be a minimum-cardinality vertex cover for G' that contains s and t, and let $Y = V' \setminus C$. Since G' is a subgraph of G, one has $|C| \le \mathrm{vc}(G') + 2 \le \mathrm{vc}(G) + 2 = \mathrm{vc} + 2$. It remains to upper-bound $|Y|$. To this end, we upper-bound the number of vertices of degree at least r in Y and of degree exactly i in Y for each $i \in \{0, \dots, r-1\}$. Note that vertices in Y have neighbors only in C.

Since Reduction Rule 8.1 has been applied, there are no vertices of degree zero in Y. Since Reduction Rule 8.8 has been applied, for each $i \in \{1, \dots, r-1\}$ and each subset $C' \subseteq C$ with $|C'| = i$, we find at most r vertices in Y whose neighborhood is C'. Thus, for each $i \in \{1, \dots, r-1\}$, the number of vertices with degree i in Y is at most $r \cdot \binom{|C|}{i}$.

Finally, since G is $K_{r,r}$-subgraph-free, any r-sized subset of the vertex cover C has at most $r-1$ common neighbors. Hence, since vertices in Y have neighbors only in C, the number of vertices in Y of degree at least r is at most $(r-1)\cdot\binom{|C|}{r}$. We conclude that

$$|V'| \leq |C| + (r-1)\cdot\binom{|C|}{r} + r\cdot\sum_{i=1}^{r-1}\binom{|C|}{i} \leq (\mathrm{vc}+2) + r(\mathrm{vc}+2)^r. \qquad \square$$

To finish the proof of Theorem 8.25, it remains to first shrink the weights and then reduce VW-SSP back to SSP on $K_{r,r}$-subgraph-free graphs.

Proof of Theorem 8.25. Since SSP is trivially polynomial-time solvable in $K_{1,1}$-subgraph-free graphs, we assume $r \geq 2$ in the following. Using Proposition 8.27 and Lemma 8.2, we reduce any instance $\mathcal{I}$ of SSP on a $K_{r,r}$-subgraph-free n-vertex graph for constant r with vertex cover number vc to an equivalent instance $\mathcal{I}'$ of VW-SSP on $O(\mathrm{vc}^r)$ vertices whose weights are upper-bounded by $2^{O(\mathrm{vc}^{3r})}$. Thus, the overall encoding length of $\mathcal{I}'$ is $O(\mathrm{vc}^{4r})$. Since SSP is NP-complete even in planar graphs [LF20] and thus in $K_{3,3}$-subgraph-free graphs, and in $K_{2,2}$-subgraph-free graphs [BFT20], we can in polynomial time reduce $\mathcal{I}'$ to an equivalent instance $\mathcal{I}^*$ of SSP on $K_{r,r}$-subgraph-free graphs. Since the running time of the reduction is polynomial, the size of $\mathcal{I}^*$ is polynomial in the size of $\mathcal{I}'$ and, hence, polynomial in vc. $\square$

Finally, observe that r appears in the degree of the polynomial upper-bounding the size of the kernel and hence, we have shown polynomial kernels for SSP parameterized by vc in $K_{r,r}$-subgraph-free graphs only for constant r. Indeed, in the next section we prove that, unless coNP $\subseteq$ NP$_{/\mathrm{poly}}$, there is no kernel of size polynomial in both vc and r.

8.6.3. Polynomial (Turing) Kernelization Lower Bounds

In the preceding sections, we have seen that SSP allows for kernels of size polynomial in $\mathrm{vc}+\ell$ (Section 8.6.1) and in vc if the input graph is $K_{r,r}$-subgraph-free for some *constant* r (Section 8.6.2). A natural question is whether one can loosen the requirement of combining with ℓ or of r being constant. We will answer in the negative. Indeed, the following shows that, unless every parameterized problem contained in the complexity class WK[1] admits a polynomial Turing kernelization [Her+15], SSP admits no Turing kernelization of size polynomial in $\mathrm{vc}+r$.

Theorem 8.28. *Even in bipartite graphs,* Short Secluded Path *is WK[1]-hard when parameterized by* vc*, where* vc *is the vertex cover number of the input graph.*

Remark 8.2. Since every graph is $K_{r,r}$-subgraph-free for $r > \text{vc}$, from Theorem 8.28 it follows that for SSP in $K_{r,r}$-subgraph-free graphs, there is no kernel with size polynomial in $\text{vc} + r$ unless $\text{coNP} \subseteq \text{NP}_{/\text{poly}}$.

Theorem 8.28 also holds regarding the parameter $\text{vc} + k$ (see Remark 8.1). However, recall that a kernel of size polynomial in $\text{vc} + \ell$ exists (Section 8.6.1).

To prove Theorem 8.28, we use a polynomial parameter transformation (Definition 1.6) of Multicolored Clique parameterized by $k \log n$ [Her+15] into SSP parameterized by vc. Our polynomial parameter transformation of Multicolored Clique into SSP uses the following gadget.

Definition 8.3 (z-binary gadget)**.** A *z-binary gadget* for some power z of two is a set $B = \{u_1, u_2, \dots, u_{2\log(z)}\}$ of vertices. We say that a vertex v is *p-connected to B* for some $p \in \{0, \dots, z-1\}$ if v is adjacent to $u_q \in B$ if and only if there is a "1" in position q of the string that consists of the binary encoding of p followed by its complement.

Example 8.1. The binary encoding of 5 followed by its complement is 101010. Thus, a vertex v is 5-connected to an 8-binary gadget $\{u_1, \dots, u_6\}$ if and only if v is adjacent to exactly u_1, u_3, and u_5. Also observe that, if a vertex v is q-connected to a z-binary gadget B, then v is adjacent to exactly half of the vertices of B, that is, to $\log z$ vertices of B.

Construction 8.29. Let $G = (V_1, V_2, \dots, V_k, E)$ be an instance of Multicolored Clique with n vertices. Without loss of generality, assume that $V_i = \{v_i^1, v_i^2, \dots, v_i^{\tilde{n}}\}$ for each $i \in \{1, \dots, k\}$, where $\tilde{n}$ is some power of two (we can guarantee this by adding isolated vertices to G). We construct an equivalent instance (G', s, t, k', ℓ') of SSP, where

$$k' := 2 \cdot \binom{k}{2} + 1, \qquad \ell' := |E| - \binom{k}{2} + k \log \tilde{n},$$

and graph $G' = (V', E')$ is as follows (see Figure 8.9 for an illustration). Vertex set V' consists of vertices s, t, a vertex v_e for each edge $e \in E$, vertices w_h for each $h \in \{1, \dots, \binom{k}{2} - 1\}$, and mutually disjoint $\tilde{n}$-binary vertex

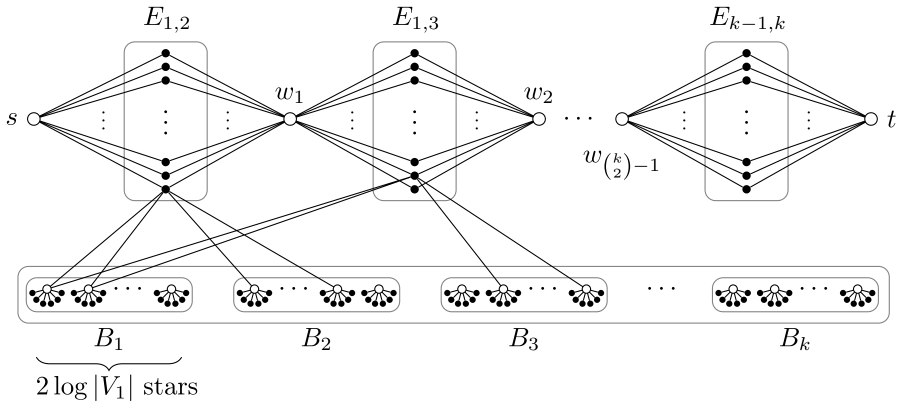


Figure 8.9.: Illustration of the polynomial parameter transformation. White vertices indicate the vertices in the vertex cover.

gadgets $B_1, \dots, B_k$, each vertex in which has $\ell' + 1$ neighbors of degree one. We set

$$\begin{aligned} B &:= B_1 \uplus B_2 \uplus \dots \uplus B_k, \\ E^* &:= \{v_e \in V' \mid e \in E\}, \\ E_{i,j} &:= \{v_{\{x,y\}} \in E^* \mid x \in V_i, y \in V_j\}, \text{ and} \\ W &:= \{w_h \mid 1 \leq h \leq \tbinom{k}{2} - 1\}. \end{aligned}$$

The edges of G' are as follows. For each edge $e = \{v_i^p, v_j^q\} \in E$, vertex $v_e \in E_{i,j}$ of G' is p-connected to B_i and q-connected to B_j. Vertex $s \in V'$ is adjacent to all vertices in $E_{1,2}$ and vertex $t \in V'$ is adjacent to all vertices in $E_{k-1,k}$. Finally, to describe the edges incident to vertices in W, consider the lexicographic ordering of the pairs $\{(i,j) \mid 1 \leq i < j \leq k\}$. Then, vertex $w_h \in W$ is adjacent to all vertices in $E_{i,j}$ and to all vertices in $E_{i',j'}$, where (i,j) is the h-th pair in the ordering and (i',j') is the $(h+1)$-st. This finishes the construction. The construction clearly can be done in polynomial time. ▲

We prove that Construction 8.29 is a polynomial parameter transformation.

Lemma 8.30. *Construction 8.29 is a polynomial parameter transformation from* Multicolored Clique *parameterized by* $k \log n$ *to* SSP *parameterized by* vc.

Proof. Let $\mathcal{I}' := (G', s, t, k', \ell')$ be the SSP instance created by Construction 8.29 from a Multicolored Clique instance $G = (V_1 \uplus \ldots \uplus V_k, E)$. We show that $\mathrm{vc} \in (k \log(n))^{O(1)}$. The vertex set of G' partitions into two independent sets

$$X := \{s, t\} \cup W \cup B \text{ and}$$
$$Y := N_{G'}(B) \cup E^*.$$

Hence, X is a vertex cover of G'. Its size is $2k \log(n) + \binom{k}{2} + 2$. It remains to show that G is a yes-instance if and only if $\mathcal{I}'$ is a yes-instance.

($\Rightarrow$) Let $E(C)$ be the edge set of a clique C of order k in G. For each $1 \le i < j \le k$, $E(C)$ contains exactly one edge e between V_i and V_j. Thus, $E_C := \{v_e \in E^* \mid e \in E(C)\}$ is a set of $\binom{k}{2}$ vertices—exactly one vertex of $E_{i,j}$ for each $1 \le i < j \le k$. Thus, by Construction 8.29, G' contains an s-t-path $P = (V_P, E_P)$ with $|V_P| \le k'$: its inner vertices are $E_C \cup W$, alternating between the sets E_C and W. To show that (G', s, t, k', ℓ') is a yes-instance, it remains to show $|N_{G'}(V_P)| \le \ell'$.

Since P contains all vertices of W, one has $N_{G'}(V_P) \subseteq B \cup (E^* \setminus E_C)$, where $|E^* \setminus E_C| = |E| - \binom{k}{2}$. To show $|N_{G'}(V_P)| \le \ell'$, it remains to show that $|N_{G'}(V_P) \cap B| \le k \log(\tilde{n})$. To this end, we show that $|N_{G'}(V_P) \cap B_i| \le \log(\tilde{n})$ for each $i \in \{1, \ldots, k\}$.

The vertices in $W \cup \{s, t\}$ have no neighbors in B. Thus, let $i \in \{1, \ldots, k\}$ be fixed and consider arbitrary vertices $v_{e_1}, v_{e_2} \in E_C$ such that $N_{G'}(v_{e_1}) \cap B_i \ne \emptyset$ and $N_{G'}(v_{e_2}) \cap B_i \ne \emptyset$ (possibly, $e_1 = e_2$). Then, $e_1 = \{v_i^p, v_j^q\}$ and $e_2 = \{v_i^{p'}, v_{j'}^{q'}\}$. Since C is a clique, e_1 and e_2 are incident to the same vertex of V_i. Thus, we have $p = p'$. Both v_{e_1} and v_{e_2} are thus p-connected to B_i and hence have the same $\log(\tilde{n})$ neighbors in B_i. It follows that $|N_{G'}(V_P)| \le \ell'$ and, hence, that $\mathcal{I}'$ is a yes-instance.

($\Leftarrow$) Let $P = (V_P, E_P)$ be an s-t path in G' with $|V_P| \le k'$ and $|N_{G'}(V_P)| \le \ell'$. The path P contains no vertex of B, since each of them has $\ell' + 1$ neighbors of degree one. Thus, the inner vertices of P alternate between vertices in W and in E^* and we get $N_{G'}(V_P) = (E^* \setminus V_P) \cup (N_{G'}(V_P) \cap B)$. Since P contains one vertex of $E_{i,j}$ for each $1 \le i < j \le k$, we know $|E^* \setminus V_P| = |E| - \binom{k}{2}$. Thus, since $|N(V_P)| \le \ell'$, we have $|N(V_P) \cap B| \le k \log(\tilde{n})$. We show that the set $E(C) := \{e \in E \mid v_e \in V_P \cap E^*\}$ is the edge set of a clique C in G. To this end, we show that, for each $i \in \{1, \ldots, k\}$, any two edges $e_1, e_2 \in E(C)$ with $e_1 \cap V_i \ne \emptyset$ and $e_2 \cap V_i \ne \emptyset$ have the same endpoint in V_i: then $E(C)$ is a set of $\binom{k}{2}$ edges on k vertices and thus C forms a clique of order k in G.

For each $1 \leq i < j \leq k$, P contains exactly one vertex $v \in E_{i,j}$, which has exactly $\log(\tilde{n})$ neighbors in each of B_i and B_j. Thus, from $|N_{G'}(V_P) \cap B| \leq k \log(\tilde{n})$ it follows that $|N_{G'}(V_P) \cap B_i| = \log(\tilde{n})$ for each $i \in \{1, \dots, k\}$. It follows that, if two vertices v_{e_1} and v_{e_2} on P both have neighbors in B_i, then both are p-connected to B_i for some p, implying that the edges e_1 and e_2 of G share endpoint v_i^p. We conclude that C is a clique of order k in G. Hence, G is a `yes`-instance. □

We are set to prove Theorem 8.28.

Proof of Theorem 8.28. Multicolored Clique parameterized by $k \log(n)$ is known to be WK[1]-complete [Her+15] and, hence, to admit no polynomial kernel unless coNP $\subseteq$ NP$_{/\text{poly}}$. Since Construction 8.29 is a polynomial parameter transformation from Multicolored Clique parameterized by $k \log(n)$ to SSP parameterized by vc (Lemma 8.30), it thus follows that SSP parameterized by vc is WK[1]-hard and admits no polynomial kernel unless coNP $\subseteq$ NP$_{/\text{poly}}$. □

8.7. Feedback Edge Set Number

In this section, we show that we can reduce any instance of SSP to an equivalent instance of Vertex-Weighted Short Secluded Path (VW-SSP) with number of vertices linear in the feedback edge number fes. We hereby allow a trade-off between the running time and the size of the resulting instance. The first reduction runs in linear time and creates vertices with weights in $O(k + \ell)$. The second reduction, following our technique described in Section 8.3, runs in polynomial time and creates vertex weights that can be encoded using $O(\text{fes}^3)$ bits. Thus, when finally reducing back to SSP using Proposition 8.3, we obtain a problem kernel of size $O(\text{fes} \cdot (k + \ell))$ using the first reduction and a problem kernel of size polynomial in fes using the second reduction.

Theorem 8.31. Short Secluded Path *admits a kernel*

(i) with size $O(\text{fes} \cdot (k + \ell))$ computable in linear time, and

(ii) with size polynomial in fes *computable in polynomial time.*

We first prove the following intermediate result.

Proposition 8.32. *For any instance of* SSP *we can compute in linear time an equivalent simplified instance of* VW-SSP *with* $16\text{fes}+9$ *vertices,* $17\text{fes}+8$ *edges and vertex weights in $O(k + \ell)$.*

Our proof of Proposition 8.32 is similar to the proof of the kernel of size polynomial in $\text{fvs} + k + \ell$ (Proposition 8.12).

Let F be a feedback edge set of size fes in $G = (V, E)$. By Reduction Rule 8.1, we may assume G to be connected. Thus, $T := G - F$ is a tree. Let $Y := \{v \in V \mid v \in e \in F\} \cup \{s, t\}$ denote the set of vertices containing s and t and all endpoints of the edges in F. We call the vertices in Y *good*. In the following, we will interpret the input SSP instance as an instance of VW-SSP with unit weight functions κ and λ and the zero weight function η. Our two reduction rules we state next are simplified versions of Reduction Rules 8.4 and 8.5 (see Section 8.5.1) with $\mathcal{T} = \{T\}$ and $R = \emptyset$.

Reduction Rule 8.9. *If there is a vertex $v \in V(T) \setminus Y$ with $N_T(v) = \{w\}$, then set $\eta(w) := \min\{\ell + 1, \eta(w) + 1\}$ and delete v.*

Reduction Rule 8.10. *Let $Q \subseteq T$ be a maximal-edgy a-b path with $|V(Q)| > 3$ in T and let $K := V(Q) \setminus \{a, b\}$. Then, add a vertex x and the edges $\{x, a\}$ and $\{x, b\}$. Set $\kappa(x) := \min\{k + 1, \kappa(K)\}$ and $\eta(x) := \min\{\ell + 1, \eta(K)\}$. Delete all vertices in K.*

The correctness of Reduction Rules 8.9 and 8.10 follows immediately from the correctness of Reduction Rules 8.4 and 8.5. Moreover, due to Lemmas 8.14 and 8.15 in Section 8.5.1, we can first apply Reduction Rule 8.9 exhaustively in linear time, and then apply Reduction Rule 8.10 exhaustively in linear time without making Reduction Rule 8.9 applicable again. After applying Reduction Rules 8.9 and 8.10 exhaustively, we have the following.

Observation 8.33. *Let T be such that none of Reduction Rules 8.9 and 8.10 is applicable. Then G has at most $8|Y| - 7$ vertices and $8|Y| - 8 + |F|$ edges, where each vertex is of weight in $O(k + \ell)$.*

Proof. Due to Reduction Rule 8.9, every leaf of T is in Y. Hence, there are at most $2|Y| - 1$ vertices in T of degree not equal to two. Since T is a tree, there are at most $2|Y| - 2$ paths connecting two vertices being good or of degree at least three. Due to Reduction Rule 8.10, these paths contain at most three vertices. It follows that there are at most $8|Y| - 7$ vertices in T, each of weight in $O(k + \ell)$, and, consequently, at most $8|Y| - 8$ edges in T. As T only differs from G by F, it follows that G has at most $8|Y| - 8 + |F|$ edges. $\square$

We are set to prove Proposition 8.32.

Proof of Proposition 8.32. Let $\mathcal{I} = (G, s, t, k, \ell)$ be an instance of SSP. Compute a minimum feedback edge set F of size $\text{fes} := |F|$ in G in linear time (just take the complement of a spanning tree). Compute the set Y of good vertices. First apply Reduction Rule 8.9 exhaustively in linear time. Next, apply Reduction Rule 8.10 exhaustively in linear time. Let $\mathcal{I}' := (G', s, t, k, \ell, \lambda, \kappa, \eta)$ denote the obtained instance of VW-SSP. Observe that due to Reduction Rule 8.10, $\mathcal{I}'$ is simplified (with $A = \emptyset$, see Definition 8.1). Due to Observation 8.33, we know that G' has at most $8|Y| - 7$ vertices and $8|Y| - 8 + \text{fes}$ edges, where each vertex is of weight in $O(k + \ell)$. Note that $|Y| \leq 2\text{fes} + 2$. Hence, G' has at most $16\text{fes} + 9$ vertices, $17\text{fes} + 8$ edges, and vertex weights in $O(k + \ell)$. □

Having shown Proposition 8.32, we can now prove Theorem 8.31. We will employ Proposition 8.3 for Theorem 8.31(i) and Lemma 8.2 for Theorem 8.31(ii).

Proof of Theorem 8.31. Let $\mathcal{I} = (G, s, t, k, \ell)$ be an instance of SSP. Employ Proposition 8.32 to obtain a simplified instance $\mathcal{I}' = (G', s, t, k, \ell, \lambda, \kappa, \eta)$ of VW-SSP, where G' has at most $O(\text{fes})$ vertices and edges, where each vertex is of weight in $O(k + \ell)$. Employing Proposition 8.3 yields an instance $\mathcal{I}'' = (G'', s', t', k'', \ell'')$ of SSP in time

$$\kappa(V(G')) + \eta(V(G')) + |E(G')| \in O(\text{fes} \cdot (k + \ell)).$$

Due to Proposition 8.3, it follows that G'' has at most M vertices, yielding (i).

For statement (ii), apply Lemma 8.2 (instead of Proposition 8.3) to obtain from $\mathcal{I}'$ an instance $\mathcal{I}^* = (G', s, t, k', \ell', \lambda', \kappa', \eta')$ of VW-SSP with k', ℓ', and all weights encoded with $O(\text{fes}^3)$ bits. Since VW-SSP is NP-complete, there is a polynomial-time many-one reduction to SSP. Employing such a polynomial-time many-one reduction on instance $\mathcal{I}^*$ yields statement (ii). □

8.8. Concluding Remarks

When not only asking for a two-terminal path to be short, but additionally to have few vertices neighboring it, one turns a polynomial-time solvable problem into an NP-hard problem. For SSP, we proved a polynomial kernelization hierarchy (see Figure 8.1) regarding the combination of its problem-specific parameters (numbers k and ℓ of vertices in the path and neighboring it, respectively) and four structural parameters (treewidth, feedback vertex and edge number, and vertex cover number).

Table 8.1.: Overview of Luckow and Fluschnik's [LF20] results (two included here in this chapter): W[1]/W[2]-h., p-NP-h., noPK abbreviate W[1]/W[2]-hard, para-NP-hard, no polynomial kernel unless coNP $\subseteq$ NP$_{/\text{poly}}$, respectively. [a] (even on planar graphs) [b] (even on planar graphs with maximum vertex degree seven)

Problem	Parameterized Complexity		
	k	ℓ	$k + \ell$
SHORT SECLUDED PATH	XP, W[1]-h.	p-NP-h.[a]	FPT /noPK[b] (Thms. 8.1 and 8.5)
LONG SECLUDED PATH	p-NP-h.[a]	p-NP-h.[a]	p-NP-h.[a]
SHORT UNSECLUDED PATH	XP, W[2]-h.	*open*	FPT /noPK[b]
LONG UNSECLUDED PATH	p-NP-h.[a]	p-NP-h.[a]	*open*/noPK[b]

Interestingly, our hierarchy suggests that combining with ℓ is more powerful for polynomial kernelization than combining with k. However, we wonder whether there is a (natural) parameter p "between" feedback vertex number and feedback edge number such that a polynomial kernel with this parameter is presumably excluded but any combination with k or ℓ allows for a polynomial kernelization.

Future work could also perform a study of other restrictions on the numbers of vertices in the path and neighboring it. Next to SSP, three more variants of the two-terminal path problem, that is, asking for long paths and large neighborhoods, are studied [LF20] (see Table 8.1; refer to Appendix A for problem definitions). Remarkably, all four variants are proven to be NP-complete, and hence, indistinguishable regarding their classic computational complexity. Yet, they seem to be distinguishable through their parameterized complexities regarding k, ℓ, and their combination $k + \ell$. Answering the two open questions in Table 8.1 would settle whether these pairwise different (complexity-theoretic) fingerprints exist.

Open Problem 12. What is the parameterized complexity of SHORT UNSECLUDED PATH parameterized by ℓ and of LONG UNSECLUDED PATH parameterized by $k + \ell$?

Notably, the pairwise different fingerprints would already exist if the two parameterized problems were contained in XP. Besides, we wonder how SHORT

Unsecluded Path, for instance, classifies regarding structural parameters possibly combined with k and ℓ.

Chapter 9.

Secluded Graph Problems: Data Reduction with Neighborhoods

In this chapter we continue the study of polynomial kernelization for more classic graph problems when additionally demanding the size of the neighborhood of the solution set to be small, that is, in the secluded setup. Herein, we focus on parameterizations that are given with each problem definition, more precisely, on the parameters size of the solution, size of the closed or open neighborhood, and their combination.

9.1. Introduction

In the previous Chapter 8, we studied the problem of finding a short s-t path when additionally limiting the *exposure* of the solution as measured by the size of the neighborhood. We can also limit the exposure of a solution of several other optimization problems on graphs where one searches for a minimum or maximum cardinality subset of vertices and edges satisfying certain properties. Limiting the exposure is motivated by safely sending sensitive information through a network [Che+17] or by the search for segregated communities in social networks [Gae04, IIO05]. In addition to being a natural constraint in the above applications, restricting the exposure of the solution may also yield more efficient algorithms [HKS15, Hüf+09, IIO05, Kom+09]. Our aim in this chapter

This chapter is based on (parts of) *The parameterized complexity of finding secluded solutions to some classical optimization problems on graphs* by René van Bevern, Till Fluschnik, George B. Mertzios, Hendrik Molter, Manuel Sorge, and Ondřej Suchý (Discrete Optimization [Bev+18]).

is to study the classic computational and parameterized complexity of secluded variants of classic combinatorial optimization problems in graphs.

Following Chechik et al. [Che+17], the first way we measure the exposure of a solution S is by the size of the closed neighborhood $N_G[S]$ of S in the input graph G. Given a predicate $\Pi(G, S)$ that determines whether S is a solution for input graph G, we study the following general problem.

Secluded Π
Input: An undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Is there a vertex subset $S \subseteq V$ such that S satisfies $\Pi(G, S)$ and $|N_G[S]| \leq k$?

In some cases, it may be necessary to control the size of the solution and its neighborhood independently, as we did for Short Secluded Path in Chapter 8. Hence, the second way we measure the exposure of the solution is the size of the open neighborhood $N_G(S) = N_G[S] \setminus S$. We thus introduce and study the complexity of the following problem.

Small Secluded Π
Input: An undirected graph $G = (V, E)$ and two integers $k \geq 1, \ell \geq 0$.
Question: Is there a vertex subset $S \subseteq V$ such that S satisfies $\Pi(G, S)$, $|S| \leq k$, and $|N_G(S)| \leq \ell$?

In this chapter, we study Secluded Π and Small Secluded Π with Π being the problems of finding a small s-t separator (s-t Separator) and a small feedback vertex set (Feedback Vertex Set).

Our Contributions. Our results are summarized in Table 9.1. We prove Secluded s-t Separator and Secluded Feedback Vertex Set to remain polynomial-time solvable and NP-hard, respectively, and the latter to admit a polynomial kernel regarding k. Small Secluded s-t Separator, however, we prove to be NP-hard, to presumably admit no polynomial kernel regarding $k + \ell$, and to be W[1]-hard when parameterized by k or by ℓ, where W[1]-hardness regarding ℓ also holds true for Small Secluded Feedback Vertex Set.

Related Work. In Chapter 8, we already reason on the problems of finding a short s-t path in the secluded setup (see Section 8.1).

The small secluded concept can be found in the context of separator problems in graphs [FGK13, Mar06]. One of these problems is Cutting k Vertices, where, given an undirected graph $G = (V, E)$ and two integers $k \geq 1$ and $\ell \geq 0$,

Table 9.1.: Overview of the classic and parameterized complexity of our (secluded) problems. P, NP-c., FPT, PK, noPK, and W[1]-h. stand for containment in the class P, NP-complete, fixed-parameter tractable, a polynomial kernel exists, no polynomial kernel exists unless coNP $\subseteq$ NP$_{/\text{poly}}$, and W[1]-hard, respectively. † [Bev+18]

	Complex.	Parameterized Complexity		
		k	ℓ	$k+\ell$
Secluded				
s-t Separator	P			
Feedback Vertex set	NP-c.	FPT, PK		
Small Secluded				
s-t Separator	NP-c.	W[1]-h.	W[1]-h.	FPT†, noPK
Feedback Vertex set	NP-c.	*open*	W[1]-h.	*open*

the question is whether there is a non-empty set $S \subseteq V$ such that $|S| = k$ and $|N_G(S)| \leq \ell$. The problem is NP-hard [BJ92] and W[1]-hard regarding $k + \ell$ [Mar06]. If set S must induce a connected subgraph in G, then the problem becomes fixed-parameter tractable regarding $k + \ell$ while staying W[1]-hard regarding k and regarding ℓ [Mar06]. Fomin et al. [FGK13] studied the variant of Cutting k Vertices where one requires $|S| \leq k$ (resembling our small secluded concept). This variant is W[1]-hard regarding k but fixed-parameter tractable regarding ℓ [FGK13]. We remark that we observed the latter for none of our studied small secluded problems.[1] The problem variant Cutting at Most k Vertices with Terminal, where S has to contain a given vertex $s \in V$, is W[1]-hard regarding k and regarding ℓ yet fixed-parameter tractable regarding $k + \ell$ [FGK13].

The concept of *isolation* states that the solution (vertex set) should have few edges to (instead of few neighbors in) the rest of the graph. The concept is studied for edge-weighted graphs [Dow+03], for finding (isolated) dense subgraphs like cliques [IIO05], and also from a parameterized algorithmics point of view [HKS15, Hüf+09, Kom+09].

Small Secluded Π can be seen as special cases of Fixed Cardinality Optimization [Bru+06, Cai08, CCC06, KS15]. Hence, from the literature

[1]Note that for short or long paths with small or large open neighborhood, fixed-parameter tractability regarding ℓ is neither observed, see Table 8.1 in Chapter 8.

on Fixed Cardinality Optimization [Bev+18] one can derive results for secluded problems.

Secluded Concepts and their Relations. Concerning the classic computational complexity, the Small Secluded variant of a problem is at least as hard as the non-secluded problem, by a simple reduction in which we set $\ell = n$, where n denotes the number of graph vertices. Since this reduction is a parameterized reduction with respect to k, parameterized hardness regarding k also transfers. Furthermore, observe that hardness also transfers from Secluded Π to Small Secluded Π for all problems Π, since Secluded Π allows for a parameterized Turing reduction to Small Secluded Π: try out all k' and ℓ' with $k = k' + \ell'$.

Observation 9.1. Secluded Π *parameterized by k is parameterized Turing reducible to* Small Secluded Π *parameterized by $k + \ell$ for all predicates* Π.

Additionally, tractability results (in particular polynomial-time solvability and fixed-parameter tractability) transfer from Small Secluded Π parameterized by $k + \ell$ to Secluded Π parameterized by k. Thus, for the Small Secluded variant of the problems, interesting cases are those where the base problem (deciding whether input graph G contains a vertex set S of size k that satisfies $\Pi(G, S)$) is tractable or where the size ℓ of the open neighborhood is a parameter.

9.2. s-t Separator with Small Neighborhood

In this section, we show that Secluded s-t Separator is in P (Section 9.2.1), while Small Secluded s-t Separator is NP-hard and W[1]-hard when parameterized by the size k of the solution or by the size ℓ of the open neighborhood (Section 9.2.2). Moreover, when parameterized by $k + \ell$, while being fixed-parameter tractable [Bev+18], we prove Small Secluded s-t Separator to not allow for polynomial kernels (unless coNP $\subseteq$ NP$_{/\text{poly}}$).

9.2.1. Secluded s-t Separator

In this section we prove the following problem to be polynomial-time solvable.

Secluded s-t Separator (S$st$S)
Input: An undirected graph $G = (V, E)$, two distinct vertices $s, t \in V$, and an integer $k \geq 0$.
Question: Is there an s-t separator $S \subseteq V \setminus \{s, t\}$ such that $|N_G[S]| \leq k$?

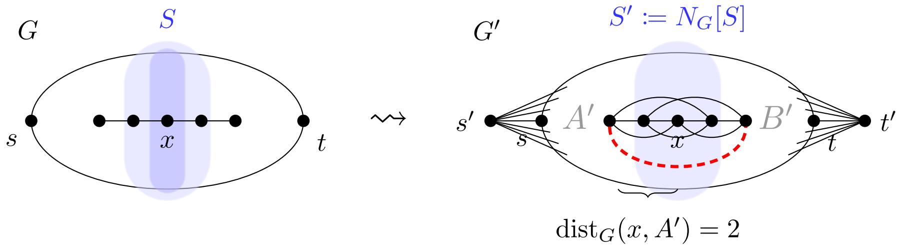


Figure 9.1.: Illustration to Lemma 9.3 with input graph G on the left-hand side and the constructed graph G' on the right-hand side (both graphs are sketched by ellipses). An s-t separator S containing a vertex x and its open neighborhood is indicated for G. The s'-t' separator S' corresponding to S is indicated for G', separating into the parts A' and B'.

Theorem 9.2. Secluded s-t Separator *is solvable in polynomial time.*

In order to prove Theorem 9.2, we show that it is enough to compute a separator of size k in the third power of a graph that is obtained from G by adding two new terminals and making one adjacent to s and one to t. The x-th power of a graph G is obtained by adding edges between vertices that are at distance at most x in G (this can be done in polynomial time), formally:

Definition 9.1. For $x \in \mathbb{N}$ the x-th power of a graph $G = (V, E)$ is a graph $G' = (V, E')$ where for each pair of distinct vertices $u, v \in V$ we have $\{u, v\} \in E'$ if and only if $\mathrm{dist}_G(u, v) \leq x$.

The following is the key ingredient behind the proof of Theorem 9.2.

Lemma 9.3. *Let $G = (V, E)$ be an undirected graph with two distinct vertices $s, t \in V$. Let G' be the third power of the graph G'', where G'' is obtained from G by adding two vertices s', t' and two edges $\{s', s\}, \{t, t'\}$. Then there is an s-t separator S in G with $|N[S]| \leq k$ if and only if there is an s'-t' separator S' with $|S'| \leq k$ in G'.*

We refer to Figure 9.1 accompanying (the proof of) Lemma 9.3.

Proof. ($\Rightarrow$) Let S be an s-t separator in G with $|N_G[S]| \leq k$. Observe that S is also an s'-t' separator in G'' as every path in G'' from s' must go through s and every path to t' must go through t. We claim that $S' = N_G[S]$ is an

s'-t' separator in G'. Suppose towards a contradiction that there is an s'-t' path $P = (p_0, p_1, \dots, p_q)$ in $G' - S'$. Let A' be the set of vertices of the connected component of $G''-S$ containing s' and let a be the largest index such that $p_a \in A'$ (note that $p_0 = s' \in A'$ and $p_q = t' \notin A'$ by definition). It follows that $p_{a+1} \notin A'$ and, since $\{p_a, p_{a+1}\} \in E'$, there is a p_a-p_{a+1} path P' in G'' of length at most three. As we have $p_a \in A'$ and $p_{a+1} \in V \setminus (A' \cup S')$ and $G[A']$ is a connected component of $G'' - S$, there must be a vertex $x \in S$ on P'. Since neither p_a nor p_{a+1} is in $S' = N_G[S]$, it follows that $\operatorname{dist}_G(p_a, x) \geq 2$ and $\operatorname{dist}_G(p_{a+1}, x) \geq 2$. This contradicts P' having length at most 3.

($\Leftarrow$) Let S' be an s'-t' separator in G' of size at most k. Let A' be the vertex set of the connected component of $G' - S'$ containing s'. Consider the set $S = \{v \in S' \mid \operatorname{dist}_{G''}(v, A') = 2\}$. We claim that S is an s-t separator in G and, moreover, that $N_G[S] \subseteq S'$ and, hence, $|N_G[S]| \leq k$. As to the second part, we have $S \subseteq S'$ by definition. Suppose towards a contradiction that there is a vertex $u \in N_G(S) \setminus S'$ that is a neighbor of $v \in S$. Then, since $\operatorname{dist}_{G''}(v, A') = 2$, we have $\operatorname{dist}_{G''}(u, A') \leq 3$. Thus, u has a neighbor in A' in G', and hence u is in A'. This implies that $\operatorname{dist}_{G''}(v, A') = 1$, contradicting the choice of v. Hence, $N_G[S] \subseteq S'$ and thus $|N_G[S]| \leq k$.

It remains to show that S is an s-t separator in G. For this, we prove that S is an s'-t' separator in G''. Note that S contains neither s nor t, since otherwise $S' \supseteq N_G[S]$ contains s' or t', contradicting S' being a subset of $V(G') \setminus \{s', t'\}$. It follows that S' must be also an s-t separator in G. Assume towards a contradiction that there is an s'-t' path in $G'' - S$. This implies that there is a path from A' to t' in $G'' - S$. Let $q := \operatorname{dist}_{G''-S}(t', A')$ and let P be a corresponding shortest path in $G'' - S$. Let us denote $P = (p_0, \dots, p_q)$ with $p_q = t'$ and $p_0 \in A'$. If $\operatorname{dist}_{G''}(t', A') \leq 3$, then t' has a neighbor in A' in G', and thus it is in A' contradicting the fact that S' is an s'-t' separator in G'. As $t' = p_q$, we have $q > 3$. Since $\operatorname{dist}_{G''}(p_0, A') = 0$, $\operatorname{dist}_{G''}(p_q, A') > 3$, and $\operatorname{dist}_{G''}(p_{i+1}, A') \leq \operatorname{dist}_{G''}(p_i, A')+1$ for every $i \in \{0, \dots q-1\}$, there is an index a such that $\operatorname{dist}_{G''}(p_a, A') = 2$. Note that each vertex v with $\operatorname{dist}_{G''}(v, A') \leq 3$ is either in A' or in S'. If p_a is not in S', then p_a is in A', contradicting our assumptions on P and q as $a \geq 2$. Hence we have $\operatorname{dist}_{G''}(p_a, A') = 2$ and p_a is in S'. It follows that p_a is in S, contradicting the choice of P. □

Proof of Theorem 9.2. By Lemma 9.3, we know that we can decide S$st$S on graph G by computing the size of a minimum s'-t' separator in G' (see Lemma 9.3 for the description of G'). A minimum two-terminal separator can be computed in polynomial time using standard methods like network flows (see, e.g., [KT06]), for instance. Thus, Theorem 9.2 follows. □

9.2.2. Small Secluded s-t Separator

In this section we prove two hardness results for the following problem:

Small Secluded s-t Separator (SS$st$S)
Input: An undirected graph $G = (V, E)$, two distinct vertices $s, t \in V$, and two integers $k \geq 0$, $\ell \geq 0$.
Question: Is there an s-t separator $S \subseteq V \setminus \{s, t\}$ such that $|S| \leq k$ and $|N_G(S)| \leq \ell$?

We show that, in contrast to Secluded s-t Separator, the above problem is NP-hard. Moreover, we prove the problem to be W[1]-hard when parameterized by k or by ℓ. Finally, we prove SS$st$S to admit no kernel of size polynomial in $k + \ell$ unless coNP $\subseteq$ NP$_{/\text{poly}}$.

Theorem 9.4. Small Secluded s-t Separator *is NP-hard and W*[1]*-hard when parameterized by k or by ℓ.*

In the proof of Theorem 9.4, we give a polynomial parameter transformation from the following problem:

Cutting at Most k Vertices with Terminal
Input: An undirected graph $G = (V, E)$, a vertex $s \in V$, and two integers $k \geq 1$, $\ell \geq 0$.
Question: Is there a set $S \subseteq V$ such that $s \in S$, $|S| \leq k$, and $|N_G(S)| \leq \ell$?

Fomin et al. [FGK13] proved Cutting at Most k Vertices with Terminal to be NP-hard and W[1]-hard when parameterized by k or by ℓ.

Construction 9.5. Let $\mathcal{I} = (G = (V, E), s, k, \ell)$ be an instance of Cutting at Most k Vertices with Terminal. We construct an instance $\mathcal{I}' = (G', s', t', k', \ell')$ of SS$st$S equivalent to $\mathcal{I}$ as follows. Let initially be $G' := G$. Add two vertices s' and t' and two edges $\{s', s\}$ and $\{s, t'\}$ to G'. Finally, set $k' := k$ and $\ell' := \ell + 2$. Clearly, the construction can be done in polynomial time. ▲

Proof of Theorem 9.4. We give a polynomial parameter transformation from Cutting at Most k Vertices with Terminal to SS$st$S. Let $\mathcal{I} = (G =$

$(V, E), s, k, \ell)$ be an instance of Cutting at Most k Vertices with Terminal and let $\mathcal{I}' := (G', s', t', k', \ell')$ be the instance of SS$st$S obtained from $\mathcal{I}$ by Construction 9.5. We show that $\mathcal{I}$ is a yes-instance of Cutting at Most k Vertices with Terminal if and only if $\mathcal{I}'$ is a yes-instance of SS$st$S.

($\Rightarrow$) Let $\mathcal{I}$ be a yes-instance and let $S \subseteq V(G)$ be a solution to $\mathcal{I}$, that is, $s \in S$, $|S| \leq k$, and $|N_G(S)| \leq \ell$. We claim that S is also a solution to $\mathcal{I}'$. Since $s \in S$ and s' and t' are both only adjacent to s, S separates s' from t' in G'. Moreover, $|S| \leq k = k'$ and, as $N_{G'}(S) = N_G(S) \cup \{s', t'\}$, we have $|N_{G'}(S)| \leq \ell + 2 = \ell'$. Hence, S' is a solution to $\mathcal{I}'$, and $\mathcal{I}'$ is a yes-instance.

($\Leftarrow$) Let $\mathcal{I}'$ be a yes-instance and let $S' \subseteq V(G') \setminus \{s', t'\}$ be an s'-t' separator in G' with $|S'| \leq k'$ and $|N_{G'}(S')| \leq \ell'$. We claim that S' is also a solution to $\mathcal{I}$. Note that $|S'| \leq k' = k$. Since S' is an s'-t' separator in G' and s' and t' are both adjacent to s, it follows that $s \in S'$ and $s', t' \in N_{G'}(S')$. Thus, we have $s \in S'$ and

$$|N_G(S')| = |N_{G'-\{s',t'\}}(S')| = |N_{G'}(S')| - 2 \leq \ell' - 2 = \ell.$$

Hence, S' is a solution to $\mathcal{I}$ proving $\mathcal{I}$ being a yes-instance.

Note that k' and ℓ' only depend on k and ℓ, respectively. Since Cutting at Most k Vertices with Terminal parameterized by k or by ℓ is W[1]-hard [FGK13], it follows that SS$st$S parameterized by k or by ℓ is W[1]-hard. □

Note that the above reduction seemingly fails when asking for secluded *inclusion-wise* minimal separators. Thus, studying this question could be future work.

We proved SS$st$S to be W[1]-hard when parameterized by k or by ℓ. When parameterized by $k + \ell$, however, SS$st$S is fixed-parameter tractable [Bev+18]. We show that, unless coNP $\subseteq$ NP$_{/\text{poly}}$, SS$st$S admits no kernel of size polynomial in $k + \ell$:

Theorem 9.6. *Unless coNP $\subseteq$ NP$_{/poly}$,* Small Secluded *s-t* Separator *parameterized by $k + \ell$ admits no polynomial kernel.*

We prove Theorem 9.6 using an OR-cross-composition with the following relation on the input instances.

Definition 9.2. An instance $\mathcal{I} = (G, s, t, k, \ell)$ of SS$st$S is *malformed* if $\max\{k, \ell\} > |V(G)|$. Two instances $\mathcal{I} = (G, s, t, k, \ell)$ and $\mathcal{I}' = (G', s', t', k', \ell')$ are $\mathcal{R}$-equivalent if they are both malformed, or $k = k'$ and $\ell = \ell'$.

The following is immediate.

Observation 9.7. *Relation $\mathcal{R}$ from Definition 9.2 is a polynomial equivalence relation for* SS$st$S*.*

The following will form our OR-cross-composition.

Construction 9.8. Let $\mathcal{I}_1, \ldots, \mathcal{I}_p$, with $\mathcal{I}_q = (G_q, s_q, t_q, k, \ell)$ for every $q \in \{1, \ldots, p\}$, be $\mathcal{R}$-equivalent, not malformed instances of SS$st$S. We construct the instance $\mathcal{I} := (G, s_1, t_p, k, \ell)$ of SS$st$S as follows. Initially, let G be the disjoint union of $G_1, \ldots, G_p$, that is $G = G_1 \uplus \ldots \uplus G_p$. Identify each t_q with s_{q+1} for all $q \in \{1, \ldots, p-1\}$. Call the obtained vertex st_q, $q \in \{1, \ldots, p-1\}$. For each st_q, $q \in \{1, \ldots, p-1\}$, add a set W_q of $k + \ell + 1$ vertices, and make each adjacent to st_q. Refer to s_1 also as s and as st_0, and refer to t_p also as t and as st_p. The construction is clearly doable in polynomial time. ▲

Proof of Theorem 9.6. We apply an OR-cross-composition with input problem SS$st$S to SS$st$S parameterized by $k+\ell$. Let $\mathcal{I}_1, \ldots, \mathcal{I}_p$, with $\mathcal{I}_q = (G_q, s_q, t_q, k, \ell)$ for every $q \in \{1, \ldots, p\}$, be $\mathcal{R}$-equivalent, not malformed instances of SS$st$S. Let $\mathcal{I} := (G, s_1, t_p, k, \ell)$ be the instance of SS$st$S obtained from $\mathcal{I}_1, \ldots, \mathcal{I}_p$ by Construction 9.8. We claim that $\mathcal{I}$ is a yes-instance of SS$st$S if and only if there exists $q \in \{1, \ldots, p\}$ such that $\mathcal{I}_q$ is a yes-instance of SS$st$S.

($\Leftarrow$) Let $\mathcal{I}_q$ for some $q \in \{1, \ldots, p\}$ be a yes-instance of SS$st$S. Let $S \subseteq V(G_q) \setminus \{s_q, t_q\}$ be an s_q-t_q separator of size at most k in G_q such that $|N_{G_q}(S)| \leq \ell$. By construction of G, for all $V' \subseteq V(G_r) \setminus \{s_r, t_r\}$, $r \in \{1, \ldots, p\}$, it holds that $N_G(V') = N_{G_r}(V')$. Moreover, since G is obtained by a "serial" composition of $\mathcal{I}_1, \ldots, \mathcal{I}_p$, every s-t path in G contains $s = st_0, st_1, \ldots, st_{p-1}, st_p = t$ in this order. Hence, any vertex set $V' \subseteq V(G_r) \setminus \{s_r, t_r\}$ separating st_{r-1} and st_r in G, $r \in \{1, \ldots, p\}$, also separates s and t in G. Altogether, S is an s-t separator in G of size at most k with $|N_G(S)| = |N_{G_q}(S)| \leq \ell$. Thus, $\mathcal{I}$ is a yes-instance of SS$st$S.

($\Rightarrow$) Let $\mathcal{I}$ be a yes-instance of SS$st$S, and let $S \subseteq V(G) \setminus \{s, t\}$ be a minimal s-t separator (of size at most k) such that $|N_G(S)| \leq \ell$. Observe that $S \cap \{st_1, \ldots, st_{p-1}\} = \emptyset$, since every st_r, $r \in \{1, \ldots, p-1\}$, is adjacent to at least $k + \ell + 1$ vertices. Moreover, since every vertex contained in W_q, $q \in \{1, \ldots, p\}$, is of degree one and hence not participating in any minimal s-t separator in G, no vertex from W_q is contained in S since S is chosen as minimal. We claim that there exists a $q \in \{1, \ldots, p\}$ with $S \subseteq V(G_q) \setminus \{s_q, t_q\}$. Following the argumentation above, since S separates s and t, there is at least one $r \in \{1, \ldots, p\}$ such that S separates st_{r-1} and st_r. Let q be the minimal index such that S separates st_{q-1} and st_q Suppose that there is an

$r \neq q$ such that $S \cap V(G_r) \setminus \{s_r, t_r\} \neq \emptyset$. Since S separates s from st_q, $S' = S \cap (V(G_q) \setminus \{s_q, t_q\})$ is an s-t separator of G of size smaller than S. This contradicts the minimality of S. Hence, $S \subseteq V(G_q) \setminus \{s_q, t_q\}$. Since S separates st_{q-1} and st_q in G, it follows that S separates s_q and t_q in G_q. Together with $|S| \leq k$ and $N_{G_q}(S) = N_G(S)$ implying $|N_G(S)| \leq \ell$, it follows that $\mathcal{I}_q$ is a yes-instance. □

9.3. Feedback Vertex Set with Small Neighborhood

In this section, we study secluded versions of the Feedback Vertex Set (FVS) problem, which asks, given a graph G and an integer k, whether there is a set $F \subseteq V(G)$ with $|F| \leq k$ such that $G - F$ contains no cycle. We prove Secluded Feedback Vertex Set to be NP-complete and to admit a kernel of size polynomial in the size k of the closed neighborhood (Section 9.3.1), and Small Secluded Feedback Vertex Set to be W[1]-hard when parameterized by the size ℓ of the open neighborhood (Section 9.3.2).

9.3.1. Secluded Feedback Vertex Set

We show in this section that the problem below is NP-hard and admits a polynomial kernel.

Secluded Feedback Vertex Set (SFVS)
Input: An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Is there a set $S \subseteq V$ such that $G - S$ is cycle-free and $|N_G[S]| \leq k$?

We first prove the NP-hardness.

Theorem 9.9. Secluded Feedback Vertex Set *is NP-hard.*

The proof is by a reduction from FVS and works by attaching to each vertex in the original graph a large set of new degree-one neighbors.

Proof. We provide a polynomial-time many-one reduction from Feedback Vertex Set. Let $(G = (V, E), k)$ be an instance of Feedback Vertex Set. We construct an equivalent instance $(G' = (V', E'), k')$ of SFVS as follows. Let initially $G' := G$. For each vertex $v \in V$ add a set W_v of n^2 vertices and make each adjacent to v. Let $W := \bigcup_{v \in V} W_v$. Observe that each vertex in W has degree one and thus is never part of a cycle in G'. Further, set $k' := k \cdot (n^2 + n)$. We claim that (G, k) is a yes-instance of FVS if and only if (G', k') is a yes-instance of SFVS.

($\Rightarrow$) Let $S \subseteq V$ be a feedback vertex set in G. Then S forms a feedback vertex set in G'. Moreover, we have k vertices, each having at most $n^2 + n$ neighbors. Thus, $|N_{G'}[S]| \leq k \cdot (n^2 + n) = k'$. It follows that (G', k') is a yes-instance of SFVS.

($\Leftarrow$) Let S be a minimal solution to (G', k'), that is, S is a feedback vertex set in G' such that $|N_{G'}[S]| \leq k'$ and $S \setminus \{v\}$ is not a feedback vertex set in G' for every $v \in S$. By minimality of S, and since no vertex in W appears in any cycle in G', S does not contain any vertex in W. Hence, $S \subseteq V$ and thus $|S| \leq k$ as each vertex $v \in V$ has at least n^2 neighbors from W_v in G'. Since S forms a feedback vertex set in G', S also forms a feedback vertex set in G. It follows that (G, k) is a yes-instance of FVS. □

On the positive side, SFVS admits a kernel of size polynomial in k, and hence remains fixed-parameter tractable when parameterized by k:

Theorem 9.10. SECLUDED FEEDBACK VERTEX SET *admits a kernel with* $O(k^5)$ *vertices.*

In the remainder of this section, we describe the data reduction rules that yield the polynomial kernel. The reduction rules are inspired by the kernelization algorithm for the TREE DELETION SET problem given by Giannopoulou et al. [Gia+16]. On a high-level, our approach consists of the following three steps:

1. Compute a subgraph H that contains all vertices participating in the cycles of the input graph G.

2. Delete vertices outside of H not neighboring any vertex in H (Reduction Rule 9.1), replace long paths of degree-two vertices in H by paths of length three (Reduction Rule 9.2), and shrink for every vertex excluded from any solution the number of neighbors outside of H to at least k (Reduction Rule 9.3).

3. Check whether there are too many cycles only intersecting in one vertex (Reduction Rule 9.4) or in two vertices (Reduction Rule 9.5), where in either case conclude that we are facing a no-instance.

Finally, after the graph is reduced by the above steps, we analyze the size of the reduced graph via decomposing its set of vertices.

For our first step of our approach, we start by introducing the following notation.

Definition 9.3 (2-core [Sei83])**.** A *2-core* of a graph G is a maximum subgraph H of G such that for each $v \in V(H)$ it holds true that $\deg_H(v) \geq 2$.

Note that a 2-core H of a given graph G is unique and can be found in polynomial time [Sei83]. If H is a 2-core of G, then we use

$$\deg_{H|0}(v) := \begin{cases} \deg_H(v), & \text{if } v \in V(H), \\ 0, & \text{if } v \notin V(H). \end{cases}$$

Observation 9.11. *Let G be a graph, H its 2-core, and C a connected component of $G - V(H)$. Then $|N_G(C) \cap V(H)| \leq 1$ and $|N_G(H) \cap V(C)| \leq 1$.*

Proof. We only prove the first statement (the second statement follows analogously). Towards a contradiction, assume that $|N_G(C) \cap V(H)| \geq 2$. Then, there are vertices $x, y \in V(H)$ with $x \neq y$ such that x and y have neighbors $a, b \in V(C)$. If $a = b$, then $G' = G[V(H) \cup \{a\}]$ is a subgraph of G such that $\deg_{G'}(v) \geq 2$ for every $v \in V(G')$, contradicting the choice of H as the 2-core of G. If $a \neq b$, then, since C is connected, there is a path P_C in C connecting a and b. Thus, $G' = G[V(H) \cup V(P_C)]$ is a subgraph of G such that $\deg_{G'}(v) \geq 2$ for every $v \in V(G')$, again contradicting the choice of H as the 2-core of G. $\square$

Note that only the vertices in the 2-core are involved in cycles of G. However, the vertices outside the 2-core can influence the size of the closed neighborhood of the feedback vertex set.

For step two of our approach, we apply the following reduction rules to our input instance with G given its 2-core H. First, we introduce the following notation: we say that a feedback vertex set F in G is

- *secluded* if $|N_G[F]| \leq k$, and
- *minimal* if $F \setminus \{v\}$ is not a secluded feedback vertex set in G for all $v \in F$.

Our first reduction rule concerns vertices not neighboring H.

Reduction Rule 9.1. *If* $\deg_{H|0}(v) = 0$ *for every* $v \in N_G[u]$*, then delete* u*.*

Correctness proof. ($\Rightarrow$) Let F be a minimal secluded feedback vertex set in G. Since $\deg_{H|0}(v) = 0$ for all $v \in N_G[u]$, none of them is involved in a cycle. Hence,

$N_G[u] \cap F = \emptyset$. In particular, it follows from $N_G(u) \cap F = \emptyset$ that $u \notin N_G[F]$. Hence, F is a secluded feedback vertex set in $G - \{u\}$ as well.

($\Leftarrow$) Let F be a minimal secluded feedback vertex set in $G_u \coloneqq G - \{u\}$. We have to show that F is a secluded feedback vertex set in G as well. First observe that since $\deg_{H|0}(v) = 0$ for all $v \in N_G[u]$, H is also the 2-core of G_u. As only vertices in H participate in cycles of G_u and F is chosen as minimal, none of the vertices $N_G(u) \subseteq V(G_u)$ is contained in F. It follows that $|N_G[F]| = |N_{G_u}[F]| \leq k$, and thus F is a secluded feedback vertex set in G as well. □

Note that, if Reduction Rule 9.1 has been exhaustively applied, then $\deg_{H|0}(v) = 0$ implies that v has exactly one neighbor which is contained in the 2-core of the graph.

Next, we replace any long path in H of degree-two vertices by a path of length three.

Reduction Rule 9.2. *If* $(v_0, v_1, \dots, v_\ell, v_{\ell+1})$ *is a path in the input graph such that* $\ell \geq 3$*,* $\deg_{H|0}(v_i) = 2$ *for every* $i \in \{1, \dots, \ell\}$*,* $\deg_{H|0}(v_0) \geq 2$*, and* $\deg_{H|0}(v_{\ell+1}) \geq 2$*, then let* $r = \min\{\deg_G(v_i) \mid i \in \{1, \dots, \ell\}\} - 2$ *and remove vertices* $v_1, \dots, v_\ell$ *and their neighbors not in the 2-core. Then introduce two vertices* u_1 *and* u_2 *with edges* $\{v_0, u_1\}$*,* $\{u_1, u_2\}$*, and* $\{u_2, v_{l+1}\}$*,* r *vertices connected to* u_1*, and* r *vertices connected to* u_2*.*

Correctness proof. ($\Rightarrow$) Let F be a minimal secluded feedback vertex set in G, and let G' be the graph obtained from G by applying Reduction Rule 9.2. We distinguish three cases of how F intersects $\{v_0, \dots, v_{\ell+1}\}$. The case where $F \cap \{v_0, \dots, v_{\ell+1}\} = \emptyset$ is trivial.

Suppose $F \cap \{v_1, \dots, v_\ell\} \neq \emptyset$. Since $\deg_{H|0}(v_i) = 2$ for all $i \in \{1, \dots, \ell\}$, each of the vertices $v_1, \dots, v_\ell$ participates in the same set of cycles of G. Hence, it follows that $F \cap \{v_1, \dots, v_\ell\} = \{v_q\}$ for some $q \in \{1, \dots, \ell\}$. Moreover, the set of cycles where $v_1, \dots, v_\ell$ appear in is a subset of the set of cycles where v_0 appears in and a subset of the set of cycles where $v_{\ell+1}$ appears in. Hence, due to minimality of F we have that $v_q \in F$ implies $v_0 \notin F$ and $v_{\ell+1} \notin F$. Due to the definition of r, the number of neighbors of v_q not in the 2-core is at least r. Then $F' = (F \setminus \{v_q\}) \cup \{u_1\}$ is a secluded feedback vertex set of G' with $|F'| = |F|$ and $|N_G[F]| \geq |N_{G'}(F')|$.

Suppose $F \cap \{v_1, \dots, v_\ell\} = \emptyset$ but $F \cap \{v_0, v_{\ell+1}\} \neq \emptyset$. Then $|F \cap \{v_0, v_{\ell+1}\}| = |N_G[F] \cap \{v_1, v_\ell\}| = |N_{G'}[F] \cap \{u_1, u_2\}|$. It follows that F is a secluded feedback vertex set in G' with $|N_{G'}[F]| = |N_G[F]|$.

($\Leftarrow$) Let F be a minimal secluded feedback vertex set in G'. We distinguish three cases of how F intersects $\{v_0, u_1, u_2, v_{\ell+1}\}$. The case where $F \cap \{v_0, u_1, u_2, v_{\ell+1}\} = \emptyset$ is trivial.

Suppose that $F \cap \{u_1, u_2\} \neq \emptyset$. Since F is minimal, either u_1 or u_2 is contained in F, since both vertices participate in the same set of cycles in G'. Without loss of generality, let $u_1 \in F$. Moreover, $F \cap \{v_0, v_\ell\} = \emptyset$, as otherwise $F \setminus \{u_1\}$ is a smaller secluded feedback vertex set in G', contradicting the minimality of F. By the choice of r, there exists $q \in \{1, \ldots, \ell\}$ such that $\deg_G(v_q) - 2 = r$. Then $F' := (F \setminus \{u_1\}) \cup \{v_q\}$ is a feedback vertex set in G with $|N_G[F']| = |N_{G'}[F]|$.

Suppose that $F \cap \{v_0, v_{\ell+1}\} \neq \emptyset$. Since F is minimal, it follows that $F \cap \{u_1, u_2\} = \emptyset$. Observe that F is also a feedback vertex set in G, as v_0 and $v_{\ell+1}$ participate in every cycle containing any vertex in $\{v_1, \ldots, v_\ell\}$. Since $|F \cap \{v_0, v_{\ell+1}\}| = |N_{G'}[F] \cap \{u_1, u_2\}| = |N_G[F] \cap \{v_1, v_\ell\}|$, it follows that $|N_G[F]| = |N_{G'}[F]|$. Hence, F is a secluded feedback vertex set in G. □

Finally in this second step of our approach, we delete neighbors of high-degree vertices (which are excluded from any solution).

Reduction Rule 9.3. *If there is $v \in V(G)$ with $\deg_G(v) > \max\{k, \deg_{H|0}(v)\}$, then remove one of the neighbors of v being not contained in the 2-core.*

Correctness proof. First observe that, as $\deg_G(v) > k$, vertex v cannot be contained in any secluded feedback vertex set. As additionally $\deg_G(v) > \deg_{H|0}(v)$, we know that there is a vertex $w \in N_G(v) \setminus V(H)$. Since w is not in the 2-core, it is not involved in the cycles of G. Since $\deg_G(v) > k$, removing w from G results in $\deg_{G-\{w\}}(v) \geq k$ and hence, v cannot be contained in any secluded feedback vertex set of $G - \{w\}$. Altogether, G has a feedback vertex set F with $|N_G[F]| \leq k$ if and only if $G - \{w\}$ has a feedback vertex set F' with $|N_{G-\{w\}}[F']| \leq k$. □

For step three of our approach, let $\mathrm{petal}(x)$ for $x \in V(G)$ denote the maximum cardinality of a set of cycles where each cycle contains x and any two cycles are vertex-disjoint except for x. Our first reduction rule in this step is the following.

Reduction Rule 9.4. *If there is a vertex $x \in V(G)$ such that $\mathrm{petal}(x) \geq \lceil \frac{k}{2} \rceil$, then output that (G, k) is a* no*-instance of SFVS.*

Correctness proof. There are at least $\lceil \frac{k}{2} \rceil$ cycles in G, which are vertex-disjoint except for x. Assume that G allows a feedback vertex set F with $|N_G[F]| \leq k$. Clearly, F must contain at least one vertex in each of the cycles. Thus $N_G[F]$

must contain at least three vertices of each cycle. As only x can be shared among these triples, we get $|N_G[F]| \geq 2 \cdot \lceil \frac{k}{2} \rceil + 1 > k$. It follows that G admits no secluded feedback vertex set. □

Our last reduction rule in this third step of our approach deals with the case when there are too many cycles intersecting in exactly two vertices.

Reduction Rule 9.5. *Let x, y be two vertices of G. If there are at least k internally vertex-disjoint paths of length at least two with endpoints x and y in G, then output that (G, k) is a `no`-instance of SFVS.*

Correctness proof. Observe that if neither x nor y belong to a feedback vertex set F of G, then we need at least $k-1$ vertices to hit all the cycles, since otherwise there are at least two distinct paths P_1, P_2 of length at least 2 between x and y with $(V(P_1) \cup V(P_2)) \cap F = \emptyset$ and thus the graph induced by $V(P_1) \cup V(P_2) \cup \{x, y\}$ contains a cycle. Since each of the $k-1$ vertices has at least two vertices in its open neighborhood and only the vertices x and y can be shared among these, the closed neighborhood contains at least $k+1$ vertices. Moreover, the open neighborhood of both x and y contains one vertex from each of the k paths. Hence, their closed neighborhood is of size at least $k+1$ and they cannot be included in the solution. □

We proved that all Reduction Rules 9.1 to 9.5 are correct. Note that Reduction Rules 9.1 to 9.3 and 9.5 can be applied trivially in polynomial time. Reduction Rule 9.4 can be applied exhaustively in polynomial time due to the following.

Proposition 9.12 ([Tho10]). *Let G be a graph and x be a vertex of G. In polynomial time we can either find a set of $\ell + 1$ cycles only intersecting in x (proving that* $\text{petal}(x) \geq \ell + 1$*) or a set of vertices $Z \subseteq V(G) \setminus \{x\}$ of size at most 2ℓ intersecting every cycle containing x.*

An instance (G, k) of SFVS is called *reduced* if none of the Reduction Rules 9.1 to 9.5 can be applied. Following the proof by Giannopoulou et al. [Gia+16], we first give a structural decomposition lemma, then upper-bound the size of components of the decomposition, and finally upper-bound the number of components in the decomposition to obtain the polynomial kernel for SFVS parameterized by k. We start with the following structural decomposition lemma, which identifies the set B.

Lemma 9.13. *There is a polynomial-time algorithm that, given a reduced instance (G, k) of SFVS, either correctly decides that (G, k) is a* `no`*-instance or finds two sets F and M' such that, denoting $B = F \cup M'$, the following holds:*

(i) *F is a feedback vertex set of G.*

(ii) *Each connected component of $G - B$ has at most 2 neighbors in M'.*

(iii) *For every connected component C in $G - B$ and $x \in B$, $|N_G(x) \cap C| \leq 1$, that is, every vertex x of B has at most one neighbor in every connected component C of $G - B$.*

(iv) *$|B| \leq 4k^2 + 2k$.*

Similarly to Giannopoulou et al. [Gia+16], we also make use of the following.

Definition 9.4. For a rooted tree T and vertex set M in $V(T)$ the *lowest common ancestor-closure* (*LCA-closure*) $\operatorname{lcac}(M)$ is obtained by the following process. Initially, set $M' = M$. Then, as long as there are vertices x and y in M' whose lowest common ancestor w is not in M', add w to M'. Finally, output M' as the LCA-closure of M.

Fomin et al. [Fom+12] proved the following properties of an LCA-closure.

Lemma 9.14 ([Fom+12])**.** *Let T be a tree and $M \subseteq V(T)$. If $M' = \operatorname{lcac}(M)$, then $|M'| \leq 2|M|$ and for every connected component C of $T - M'$, $|N_T(C)| \leq 2$.*

We continue with proving our structural decomposition lemma.

Proof of Lemma 9.13. Note that if there is a feedback vertex set of G with closed neighborhood of size at most k, then it is also a feedback vertex set in G of size at most k. Thus, we can apply the 2-approximation algorithm for Feedback Vertex Set on G due to Bafna et al. [BBF99] to find in polynomial time a feedback vertex set F of G. If $|F| > 2k$, then we output that (G, k) is a no-instance of SFVS. Hence, we assume $|F| \leq 2k$ in the following. Since F is a feedback vertex set in G, property (i) is trivially fulfilled. Moreover, $G - F$ is a collection of trees $T_1, \ldots, T_\ell$. We select for each of the trees T_i some root vertex $v_i \in V(T_i)$. It remains to construct the set M' such that $F \cup M'$ fulfills conditions (ii)–(iv).

Recall that the instance (G, k) is reduced. Hence, Reduction Rule 9.4 is not applicable, and thus $\operatorname{petal}(x) < \lceil \frac{k}{2} \rceil$ for every $x \in F$. We apply Proposition 9.12

to each vertex in $v \in F$, obtaining a set $Z_v \subseteq V(G) \setminus \{v\}$ intersecting each cycle containing v with $|Z_v| \le k$. Let

$$\begin{aligned} Z &:= Z_1 \cup \ldots \cup Z_{|F|}, \\ M_i &:= V(T_i) \cap Z \text{ for every } i \in \{1, \ldots, \ell\}, \\ M_i' &:= \operatorname{lcac}(M_i) \text{ for every } i \in \{1, \ldots, \ell\}, \text{ and} \\ M' &:= \bigcup_{i \in \{1,\ldots,\ell\}} M_i'. \end{aligned}$$

Observe that $|Z| \le 2k^2$ and, due to Lemma 9.14, that $|M_i'| \le 2|M_i|$. It follows that

$$|M'| \le \sum_{i \in \{1,\ldots,\ell\}} |M_i'| \le \sum_{i \in \{1,\ldots,\ell\}} 2|M_i| \le 2|Z| \le 4k^2.$$

Moreover, let $B := F \cup M'$ (note that $F \cap M' = \emptyset$). For every connected component C in $G - B$ it holds that $|N_{G-F}(C)| \le 2$ (hence, property (ii) is fulfilled). Altogether, $|B| = |F| + |M'| \le 2k + 4k^2$, yielding property (iv). It remains to show that property (iii) is fulfilled.

Let C be a connected component of $G - B$ and $x \in B$ some vertex. Suppose that x has two neighbors in C. Then $C_x := C \cup \{x\}$ induces a cycle in G as C is connected. If $x \in F$, then this contradicts the set $Z_x \subseteq Z \subseteq M' \cup (F \setminus \{x\})$ hitting every cycle containing x. If $x \in M'$, then this contradicts the set F hitting each cycle in G. Hence, property (iii) is fulfilled. □

Next, we show that if B is as in Lemma 9.13, then the size and the number of the connected components in $G - B$ is polynomially upper-bounded in the size k of the closed neighborhood of the feedback vertex set in question. We first upper-bound the size of each connected component in $G - B$ as follows.

Lemma 9.15. *Let (G, k) and B be as in Lemma 9.13, and let C be a connected component of $G - B$. Then the number $|V(C)|$ of vertices of the connected component C is at most $(12k + 7)(k + 1)$.*

Proof. Le H be the 2-core of G. We distinguish two cases on the size of $C_H := V(C) \cap V(H)$, namely $|C_H| = 0$ on the one hand, and $|C_H| > 0$ on the other hand.

Case 1: $|C_H| = 0$. Observe that C is a connected component in $G - V(H)$. Hence, by Observation 9.11, there is at most one vertex in C adjacent to H. If $x \in V(C)$ is adjacent to H, then no other vertex of C is adjacent to H.

Suppose that $|V(C)| > 1$. Since C is connected, there is a vertex $u \in V(C)$ such that $N_G[u] \subseteq G - V(H)$, contradicting the fact that the instance is reduced regarding Reduction Rule 9.1. Hence, $|V(C)| \leq 1$.

Case 2: $|C_H| > 0$. Recall that (G, k) is reduced. On the one hand, due to Reduction Rule 9.1, we know that every vertex in $C - V(H)$ has a neighbor in C_H. On the other hand, due to Reduction Rule 9.3, each vertex in C_H has at most k neighbors in $C - V(H)$. Hence, it follows that $|V(C)| \leq (k+1) \cdot |C_H|$. It remains to upper-bound the number of vertices in C_H. To this end, we count the number of vertices in $G[C_H]$ having degree 1, 2, and at least 3 in $G[C_H]$ in the following.

Let $D_H^1 \subseteq C_H$ be the set of vertices in $G[C_H]$ having degree exactly one. Since $D_H^1 \subseteq V(H)$, it holds that $\deg_{H|0}(v) \geq 2$ for each $v \in D_H^1$. Since there is exactly one neighbor of v in $G[C_H]$, at least one other neighbor is contained in $V(H) \cap B$. Let B_C denote the vertices of C having at least one neighbor in B. Note that $D_H^1 \subseteq B_C$. Due to Lemma 9.13(ii), C has at most two neighbors in M' (recall $B = F \cup M'$). Moreover, due to Lemma 9.13(iii), each vertex in B has at most one neighbor in C. It follows that $|B_C| \leq |F| + 2 \leq 2k + 2$, and hence $|D_H^1| \leq 2k + 2$.

Let $D_H^{\geq 3} \subseteq C_H$ be the set of vertices in $G[C_H]$ having degree at least three. Since $G[C_H]$ is acyclic (recall that $F \subseteq B$ is a feedback vertex set), it follows that D_H^1 forms the leaves in $G[C_H]$. We know that on trees the number of inner vertices of degree at least three is at most the number of leaves minus one. Hence, $|D_H^{\geq 3}| \leq |D_H^1| - 1 \leq 2k + 1$.

Let $D_H^{-2} := B_C \cup D_H^{\geq 3}$. Observe that $C_H \setminus D_H^{-2}$ only contains vertices having degree exactly two in $G[C_H]$. Moreover, these vertices participate only in paths connecting vertices in D_H^{-2}. Since $|D_H^{-2}| \leq 2k + 2 + 2k + 1 = 4k + 3$, and $G[C_H]$ is acyclic, there are at most $4k + 3 - 1 = 4k + 2$ many of these paths. Moreover, due to Reduction Rule 9.2, these paths contain at most two inner vertices. Hence, $|C_H| \leq |C_H \setminus D_H^{-2}| + |D_H^{-2}| \leq 2 \cdot (4k + 2) + 4k + 3 = 12k + 7$. It follows that $|V(C)| \leq (k+1) \cdot |C_H| \leq (k+1) \cdot (12k + 7)$. $\square$

Having an upper bound on the sizes of the set B and of each connected component in $G - B$, it remains to count the number of connected components in $G - B$. With the next lemma, we give an $O(k^3)$ upper bound on the number of connected components in $G - B$.

Lemma 9.16. *Let (G, k) and B be as in Lemma 9.13. Then the number of connected components in $G - B$ is at most $15k^3 + 8k^2 - k - 1$.*

Proof. We partition the connected components of $G - B$ by the number of their neighbors in B, namely in those that have exactly one neighbor and in those that have at least two neighbors in B. For $x, y \in B$, denote by

$\mathcal{C}_x$ the set of connected components in $G - B$ having vertex x as their only neighbor in B, and by

$\mathcal{C}_{xy}$ the set of connected components having at least x and y as their neighbors in B.

Observe that the set of connected components of $G - B$ is exactly $\bigcup_{x \in B} \mathcal{C}_x \cup \bigcup_{\{x,y\} \subseteq B} \mathcal{C}_{xy}$, and hence the number of connected components of $G - B$ is at most $|\bigcup_{x \in B} \mathcal{C}_x| + |\bigcup_{\{x,y\} \subseteq B} \mathcal{C}_{xy}|$. Further observe that

$$|\bigcup_{x \in B} \mathcal{C}_x| \leq |B|k \leq 4k^3 + 2k^2. \tag{9.1}$$

Hence, it remains to upper-bound the cardinality of $\bigcup_{\{x,y\} \subseteq B} \mathcal{C}_{xy}$. To this end, observe that

$$\bigcup_{\{x,y\} \subseteq B} \mathcal{C}_{xy} = \underbrace{\bigcup_{\{x,y\} \subseteq F} \mathcal{C}_{xy}}_{=:\mathcal{C}^1} \cup \underbrace{\bigcup_{x \in F, y \in M'} \mathcal{C}_{xy}}_{=:\widehat{\mathcal{C}}^2} \cup \underbrace{\bigcup_{\{x,y\} \subseteq M'} \mathcal{C}_{xy}}_{=:\mathcal{C}^3}. \tag{9.2}$$

Notice that the equality is still true if we replace $\widehat{\mathcal{C}}^2$ by $\mathcal{C}^2 := \widehat{\mathcal{C}}^2 \setminus \mathcal{C}^3$, since $\mathcal{C}^3$ appears in the union on the right hand-side. Hence, in the remainder of this proof, we upper-bound the size of the sets $\mathcal{C}^1$, $\mathcal{C}^2$, and $\mathcal{C}^3$. Observe that for $\mathcal{C}^1$, we have

$$|\mathcal{C}^1| \leq \binom{2k}{2}(k+1) = 2k^3 + k^2 - k. \tag{9.3}$$

Next we upper-bound the size of $\mathcal{C}^2$. To this end, let $x \in F$ be some fixed vertex in F. Consider the set S_x of vertices $y \in M'$ such that there are at least two connected components of $G - B$ neighboring both x and y. Observe that for each $y \in S_x$, the set of connected components in $\mathcal{C}^2$ neighboring both x and y is unique, as otherwise there is a connected component in $\mathcal{C}^2$ containing two vertices in M' and hence belonging to $\mathcal{C}^3$, contradicting our definition of $\mathcal{C}^2 := \widehat{\mathcal{C}}^2 \setminus \mathcal{C}^3$. Since for each $y \in S_x$ there are at least two connected components in $\mathcal{C}^2$, they together with x and y form a cycle in G. Hence, due to Reduction Rule 9.4, the number of vertices in S_x is at most $k/2$. On the other hand, since each such component provides a separate path of length at least two between x and y, there are at most k connected components neighboring both x and y for any $y \in S_x$ due to Reduction Rule 9.5. Finally, observe that the number of vertices $y \in M'$ such that there is at most one connected component of $G - B$ neighboring with both x and y is trivially upper-bounded by $|M'| \leq 4k^2$. Altogether, we obtain that (recall that $|F| \leq 2k$)

$$|\mathcal{C}^2| \leq \sum_{x\in F}(4k^2 + (k/2)(k+1)) \leq 2k(4k^2 + (k/2)(k+1)) = 9k^3 + k^2. \tag{9.4}$$

Last, we upper-bound the size of $\mathcal{C}^3 = \bigcup_{\{x,y\}\subseteq M'} \mathcal{C}_{xy}$. Observe that due to Lemma 9.13(ii), for each $x, y \in M'$, each connected component $C \in \mathcal{C}_{xy}$ is only neighboring x and y out of M', that is, $N_G(C) \cap M' = \{x, y\}$. Moreover, since C is connected, x and y are connected via a path within C. It is well-known that if there are at least r paths connecting vertex pairs out of r vertices in a graph, then there is a cycle in the graph. Hence, since F is a feedback vertex set in G, there are at most $|M'| - 1$ connected components in $\mathcal{C}^3$. Thus, we obtain that (recall that $|M'| \leq 4k^2$)

$$|\mathcal{C}^3| \leq |M'| - 1 \leq 4k^2 - 1. \tag{9.5}$$

Altogether, the number of connected components in $G - B$ is at most

$$\begin{aligned}
|\bigcup_{x\in B} \mathcal{C}_x| + |\bigcup_{\{x,y\}\subseteq B} \mathcal{C}_{xy}| &\overset{(9.1),(9.2)}{\leq} 4k^3 + 2k^2 + |\mathcal{C}^1| + |\mathcal{C}^2| + |\mathcal{C}^3| \\
&\overset{(9.3)\text{–}(9.5)}{\leq} 4k^3 + 2k^2 + 2k^3 + k^2 - k + 9k^3 + k^2 \\
&\qquad + 4k^2 - 1 \\
&= 15k^3 + 8k^2 - k - 1. \qquad \square
\end{aligned}$$

Finally, putting everything together, we can prove the main result of this section.

Proof of Theorem 9.10. Let (G', k) be the input instance of SFVS. Compute the 2-core H of G. Apply Reduction Rules 9.1 to 9.5 exhaustively to obtain an equivalent instance (G, k) such that (G, k) is reduced. Next, apply Lemma 9.13 and either report that (G, k) is a `no`-instance or obtain the set $B = F \cup M'$

in G with $|B| \leq 4k^2 + 2k$. Let $\mathcal{C}$ denote the set of connected components in $G - B$. By Lemma 9.16, we know that $|\mathcal{C}| \leq 15k^3 + 8k^2 - k - 1$. Moreover, due to Lemma 9.15, for each $C \in \mathcal{C}$ it holds that $|V(C)| \leq (k+1) \cdot (12k+7)$. It follows that the number of vertices in G is at most

$$\begin{aligned} |B| + |\mathcal{C}| \cdot \max_{C \in \mathcal{C}} |V(C)| &\leq 4k^2 + 2k + (15k^3 + 8k^2 - k - 1) \cdot (k+1) \cdot (12k+7) \\ &\in O(k^5). \end{aligned}$$ □

9.3.2. Small Secluded Feedback Vertex Set

In this section, we prove that the small secluded variant of Feedback Vertex Set is W[1]-hard when parameterized by ℓ.

Small Secluded Feedback Vertex Set (SSFVS)
Input: An undirected graph $G = (V, E)$ and two integers k, ℓ.
Question: Is there a set $S \subseteq V$ such that $G - S$ is cycle-free, $|S| \leq k$, and $|N_G(S)| \leq \ell$?

Theorem 9.17. Small Secluded Feedback Vertex Set *is W[1]-hard with respect to ℓ.*

We provide a parameterized reduction from the Multicolored Independent Set (MIS) problem:

Multicolored Independent Set (MIS)
Input: An undirected k-partite graph $G = (V = V_1 \uplus \ldots \uplus V_k, E)$.
Question: Is there an independent set $X \subseteq V$ of size k in G with $|X \cap V_i| = 1$ for all $i \in \{1, \ldots, k\}$?

Fellows et al. [Fel+09] proved MIS to be W[1]-hard when parameterized by the size k of the independent set. In our proof of Theorem 9.17, we use the following.

Construction 9.18. Let $G = (V = V_1 \uplus \ldots \uplus V_k, E)$ be an instance of MIS with $|V_i| \geq 2$ and no edge $\{v, w\} \in E$ with $v, w \in V_i$. We create an instance (G', k', ℓ) of SSFVS with $k' := |V| - k$ and $\ell := k + 1$ as follows (refer to Figure 9.2 for an illustrative sketch).

Initially, let $G' := G$. For each $i \in \{1, \ldots, k\}$ turn V_i into a clique, that is, add the edge sets $\{\{a, b\} \mid a, b \in V_i, a \neq b\}$. Next, add to G' a vertex u and a set L of $k' + \ell$ vertices. Finally, connect each vertex in $V \cup L$ to u by an edge. ▲

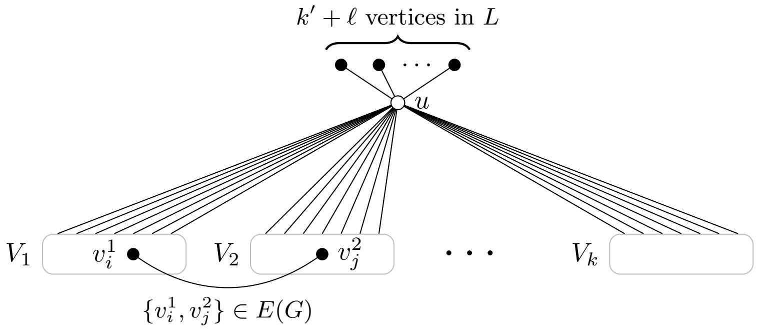


Figure 9.2.: Illustrative sketch of the construction of graph G' on an input graph $G = (V = V_1 \uplus \ldots \uplus V_k, E)$ as used in the proof of Theorem 9.17. The rectangles indicate cliques with vertex sets V_i, $i \in \{1, \ldots, k\}$.

We are set to prove our main result of this section.

Proof of Theorem 9.17. Let $G = (V = V_1 \uplus \ldots \uplus V_k, E)$ be an instance of MIS. We can assume that for each $i \in \{1, \ldots, k\}$ we have $|V_i| \geq 2$ and there is no edge $\{v, w\} \in E$ with $v, w \in V_i$. Let (G', k', ℓ) be the instance of SSFVS obtained from G by Construction 9.18. We prove that (G, k) is a yes-instance of MIS if and only if (G', k', ℓ) is a yes-instance of SSFVS.

($\Rightarrow$) Let (G, k) be a yes-instance of MIS and let $X \subseteq V$ with $|X| = k$ be a multicolored independent set in G. We delete all vertices in $S := V(G') \setminus (X \cup L \cup \{u\})$ from G'. Observe that $|S| = |V| - k = k'$. Moreover, $N_{G'}(S) = k + 1 = \ell$. Since there is no edge between any two vertices in X, $G - S$ forms a star with center u and $k' + \ell + 1 + k$ vertices. Since every star is acyclic, (G', k', ℓ) is a yes-instance of SSFVS.

($\Leftarrow$) Let (G', k', ℓ) be a yes-instance of SSFVS and let $S \subseteq V(G')$ be a solution. Observe that $G'[V_i \cup \{u\}]$ forms a clique of size $|V_i| + 1$ for each $i \in \{1, \ldots, k\}$. Since $N_{G'}[u] \geq k' + \ell + 1$, we have that $u \notin S$. Hence, all but at most one vertex in each V_i must be deleted. Since $k' = |V| - k$ and $|V_i| \geq 2$ for every $i \in \{1, \ldots, k\}$, S contains exactly $|V_i| - 1$ vertices of V_i for each $i \in \{1, \ldots, k\}$. Hence, $|S| = |V| - k$ and $N_{G'}(S) = k + 1 = \ell$. Let $X := V \setminus S$ denote the set of vertices in V not contained in S. Recall that $|X| = k$ and $|X \cap V_i| = 1$ for every $i \in \{1, \ldots, k\}$. Next, suppose there is an edge between two vertices $v, w \in X$. Since $u \notin S$ and u is incident to all vertices in V, the vertices u, v, w form a triangle in G'. This contradicts the fact that S is a solution for (G', k', ℓ), that is, that $G' - S$ is acyclic. It follows that $E(G'[X]) = \emptyset$, that is, no two vertices in X are connected by an edge. Together with $|X| = k$ and $|X \cap V_i| = 1$ for every $i \in \{1, \ldots, k\}$, it follows that X forms a multicolored independent set in G. Thus, (G, k) is a yes-instance of MIS. □

9.4. Concluding Remarks

We studied two well-known graph problems in the secluded setup. It seems that the "secluded" prefix alone still allows for tractability results: Secluded s-t Separator remains polynomial-time solvable, and Secluded Feedback Vertex Set still admits a polynomial kernel, yet with $O(k^5)$ vertices (Theorem 9.10). As to the latter, we wonder whether this can be improved:

Open Problem 13. Does Secluded Feedback Vertex Set admit a kernel with $O(k^c)$ vertices where $c < 5$?

In turn, the "small secluded" prefix seems to make problems harder: Small Secluded s-t Separator is NP-hard and even hard to preprocess regarding $k + \ell$, and Small Secluded Feedback Vertex Set is W[1]-hard when parameterized by ℓ. As to the latter, we left open its parameterized complexity regarding k and $k + \ell$.

Open Problem 14. Is Small Secluded Feedback Vertex Set fixed-parameter tractable when parameterized by k or by $k + \ell$? If so, does it admit a polynomial kernel?

Asking for a secluded dominating set makes little sense since, by definition, the closed neighborhood of a dominating set forms the whole graph. This is not true for a q-dominating set with $q \geq 2$, that is, a set of vertices such that its closed q-neighborhood forms the whole graph. Now, asking for a secluded q-dominating set makes mathematically sense. Indeed, one can even ask for p-secluded q-dominating sets with $p < q$. Here p-secluded means that the size of the closed p-neighborhood should be small (where p can be understood as security measure in safe routing, for instance). These problems admit different complexity classifications depending on whether $p \leq \frac{q}{2}$ [Bev+18] (see Table 9.2 for a summary). We restate one open case:

Open Problem 15. What is the parameterized complexity of Small p-Secluded q-Dominating Set with $p > \frac{q}{2}$ when parameterized by ℓ?

Table 9.2.: Overview of the results [Bev+18] for q-dominating set in the p-secluded setups. FPT and noPK stand for fixed-parameter tractable and no polynomial kernel unless coNP $\subseteq$ NP$_{/\text{poly}}$, respectively.

	$p \sim q$	Parameterized Complexity		
		k	ℓ	$k + \ell$
p-Secluded				
q-Dominating Set	$p \leq \frac{q}{2}$	W[2]-hard	-	-
q-Dominating Set	$p > \frac{q}{2}$	FPT, noPK	-	-
Small p-Secluded				
q-Dominating Set	$p \leq \frac{q}{2}$	$\rightarrow$	$\rightarrow$	W[2]-hard
q-Dominating Set	$p > \frac{q}{2}$	W[2]-hard	*open*	FPT, noPK

Motivated by the results on p-secluded q-dominating sets, future work could be the study of p-Secluded Π and Small p-Secluded Π.

Outlook

Beside the fifteen concrete open problems (see Appendix B), this thesis paves the way for seeking for more fractal-like structures exploitable in cross-compositions, investigating trade-offs between running times and sizes of kernelizations, in particular for polynomial-time solvable problems, and widening the range of applicability of the losing-weight technique. Moreover, it underlines the study of neighborhood-constrained graph optimization problems as fruitful objects for discovering the mentioned trade-offs while employing the losing-weight technique. We elaborate on what is said in some more detail.

We were not able to employ the T-fractal to refute the existence of polynomial kernelization for the Directed Feedback Vertex Set problem. In contrast to Directed Small Cycle Transversal, we need to hit each, not only short cycles. Notably, the directed variant of a T-fractal admits the property that many cycles intersect in few vertices, which then, in turn, might work as instance selector. This is promising, yet possibly not enough for a non-existence proof. We wonder whether some different fractal-like graph is required for a (cross-)composition, if one exists.

The diminisher framework is widely applicable, even for polynomial-time solvable problems. Further applications of the diminisher framework are of interest, for instance for variants of kernelization where different (to polynomial) running times are allowed. Moreover, we wonder about any connection of the diminisher framework and polynomial Turing kernelization.

The losing-weight technique due to Frank and Tardos [FT87] is not only applicable to weighted problems with linear goal functions, but also to unweighted problems and problems with non-linear, but what we call α-linearizable goal functions. Extending the range of applicability of the technique is future work. Moreover, we wonder about a losing-weight technique that applies to polynomial-time solvable problems. To this end, both running time and quality (regarding the obtained weight vector's encoding length) might be improved in order to achieve non-trivial results.

Several of this thesis' contributions are mostly "negative", that is, fall into the field of kernelization lower bounds. This, in turn, also motivates for further developing variants of kernelization. There are several promising variants of kernelization in the literature, for instance "lossy" kernelization [Lok+17], just to name a most recent one. We think that polynomial Turing kernelizations that rely on the adaptive power given to them, while only very few are known, form a promising direction for research. What seems to be missing here are some sort of meta theorems (see, e.g., [Bod+16, EGS18, GSS16]), basic frameworks, or possibly a deeper understanding of this "hidden" adaptive power.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009 (cited on p. 12).

[AD16] Muad Abu-Ata and Feodor F. Dragan. "Metric tree-like structures in real-world networks: an empirical study". In: *Networks* 67.1 (2016), pp. 49–68. DOI: 10.1002/net.21631 (cited on p. 114).

[AF06] Faisal N. Abu-Khzam and Henning Fernau. "Kernels: annotated, proper and induced". In: *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC'06)*. Vol. 4169. Lecture Notes in Computer Science. Springer, 2006, pp. 264–275. DOI: 10.1007/11847250_24 (cited on pp. 2, 9).

[AGV15] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. "Subcubic equivalences between graph centrality problems, APSP and diameter". In: *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*. SIAM, 2015, pp. 1681–1697. DOI: 10.1137/1.9781611973730.112 (cited on p. 90).

[Alo+11] Noga Alon, Gregory Z. Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. "Solving MAX-r-SAT above a tight lower bound". In: *Algorithmica* 61.3 (2011), pp. 638–655. DOI: 10.1007/s00453-010-9428-7 (cited on pp. 4, 5, 10).

[Alt+06] Ernst Althaus, Gruia Călinescu, Ion I. Mandoiu, Sushil K. Prasad, N. Tchervenski, and Alexander Zelikovsky. "Power efficient range assignment for symmetric connectivity in static ad hoc wireless networks". In: *Wireless Networks* 12.3 (2006), pp. 287–299. DOI: 10.1007/s11276-005-5275-x (cited on p. 147).

[AV14] Amir Abboud and Virginia Vassilevska Williams. "Popular conjectures imply strong lower bounds for dynamic problems". In: *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*. IEEE Computer Society, 2014, pp. 434–443. DOI: 10.1109/FOCS.2014.53 (cited on p. 90).

[AVW16] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. "Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs". In: *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*. Society for Industrial and

Applied Mathematics, 2016, pp. 377–391. DOI: 10.1137/1.9781611974331.ch28 (cited on pp. 90, 98, 121).

[AVY18] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. “Matching triangles and basing hardness on an extremely popular conjecture”. In: *SIAM Journal on Computing* 47.3 (2018), pp. 1098–1122. DOI: 10.1137/15M1050987 (cited on pp. 18, 92, 103).

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. “Color-coding”. In: *Journal of the ACM* 42.4 (1995), pp. 844–856. DOI: 10.1145/210332.210337 (cited on pp. 82, 84, 97).

[Bai+10] Georg Baier, Thomas Erlebach, Alexander Hall, Ekkehard Köhler, Petr Kolman, Ondrej Pangrác, Heiko Schilling, and Martin Skutella. “Length-bounded cuts and flows”. In: *ACM Transactions on Algorithms* 7.1 (2010), p. 4. DOI: 10.1145/1868237.1868241 (cited on p. 45).

[Ban+14] Nikhil Bansal, Uriel Feige, Robert Krauthgamer, Konstantin Makarychev, Viswanath Nagarajan, Joseph Naor, and Roy Schwartz. “Min-max graph partitioning and small set expansion”. In: *SIAM Journal on Computing* 43.2 (2014), pp. 872–904. DOI: 10.1137/120873996 (cited on p. 149).

[Bar+98] Reuven Bar-Yehuda, Dan Geiger, Joseph Naor, and Ron M. Roth. “Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference”. In: *SIAM Journal on Computing* 27.4 (1998), pp. 942–959. DOI: 10.1137/S0097539796305109 (cited on p. 177).

[Baz+19] Cristina Bazgan, Till Fluschnik, André Nichterlein, Rolf Niedermeier, and Maximilian Stahlberg. “A more fine-grained complexity analysis of finding the most vital edges for undirected shortest paths”. In: *Networks* 73.1 (2019), pp. 23–37. DOI: 10.1002/net.21832 (cited on pp. x, 45).

[BBF99] V. Bafna, P. Berman, and T. Fujito. “A 2-approximation algorithm for the undirected feedback vertex set problem”. In: *SIAM Journal on Discrete Mathematics* 12.3 (1999), pp. 289–297. DOI: 10.1137/S0895480196305124 (cited on p. 214).

[BCH16] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. “Into the square: on the complexity of some quadratic-time solvable problems”. In: *Electronic Notes in Theoretical Computer Science* 322 (2016), pp. 51–67. DOI: 10.1016/j.entcs.2016.03.005 (cited on pp. 114, 115, 121).

[Ben+17a] Matthias Bentert, René van Bevern, André Nichterlein, and Rolf Niedermeier. “Parameterized algorithms for power-efficient connected symmetric wireless sensor networks”. In: *Proceedings of the 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks*

(ALGOSENSORS'17). Vol. 10718. Lecture Notes in Computer Science. Springer, 2017, pp. 26–40. DOI: 10.1007/978-3-319-72751-6_3 (cited on p. 149).

[Ben+17b] Matthias Bentert, Till Fluschnik, André Nichterlein, and Rolf Niedermeier. "Parameterized aspects of triangle enumeration". In: *Proceedings of the 21st International Symposium on Fundamentals of Computation Theory (FCT'17)*. Vol. 10472. Lecture Notes in Computer Science. Springer, 2017, pp. 96–110. DOI: 10.1007/978-3-662-55751-8_9 (cited on p. x).

[Ben+18] Matthias Bentert, Alexander Dittmann, Leon Kellerhals, André Nichterlein, and Rolf Niedermeier. "An adaptive version of Brandes' algorithm for betweenness centrality". In: *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC'18)*. Vol. 123. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 36:1–36:13. DOI: 10.4230/LIPIcs.ISAAC.2018.36 (cited on p. 129).

[Ben+19] Matthias Bentert, Till Fluschnik, André Nichterlein, and Rolf Niedermeier. "Parameterized aspects of triangle enumeration". In: *Journal of Computer and System Sciences* 103 (2019), pp. 61–77. DOI: 10.1016/j.jcss.2019.02.004 (cited on pp. x, 127).

[Bet+10] Nadja Betzler, Jiong Guo, Christian Komusiewicz, and Rolf Niedermeier. "Average parameterization and partial kernelization for computing medians". In: *Proceedings of the 9th Latin American Symposium on Theoretical Informatics (LATIN'10)*. Vol. 6034. Lecture Notes in Computer Science. Springer, 2010, pp. 60–71. DOI: 10.1007/978-3-642-12200-2_7 (cited on p. 5).

[Bet+11a] Nadja Betzler, René van Bevern, Michael R. Fellows, Christian Komusiewicz, and Rolf Niedermeier. "Parameterized algorithmics for finding connected motifs in biological networks". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8.5 (2011), pp. 1296–1308. DOI: 10.1109/TCBB.2011.19 (cited on p. 84).

[Bet+11b] Nadja Betzler, Jiong Guo, Christian Komusiewicz, and Rolf Niedermeier. "Average parameterization and partial kernelization for computing medians". In: *Journal of Computer and System Sciences* 77.4 (2011), pp. 774–789. DOI: 10.1016/j.jcss.2010.07.005 (cited on pp. 4, 10).

[Bev+17] René van Bevern, Till Fluschnik, George B. Mertzios, Hendrik Molter, Manuel Sorge, and Ondřej Suchý. "Finding secluded places of special interest in graphs". In: *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC'16)*. Vol. 63. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 5:1–5:16. DOI: 10.4230/LIPIcs.IPEC.2016.5 (cited on p. vii).

[Bev+18] René van Bevern, Till Fluschnik, George B. Mertzios, Hendrik Molter, Manuel Sorge, and Ondřej Suchý. “The parameterized complexity of finding secluded solutions to some classical optimization problems on graphs”. In: *Discrete Optimization* 30 (2018), pp. 20–50. DOI: 10.1016/j.disopt.2018.05.002 (cited on pp. vii, 199, 201, 202, 206, 221, 222).

[BFS10] Hans L. Bodlaender, F. V. Fomin, and S. Saurabh. *Open Problems in Parameterized and Exact Computation — WorKer 2010*. Tech. rep. available at http://fpt.wdfiles.com/local--files/open-problems/open-problems.pdf. 2010 (cited on p. 41).

[BFT18] René van Bevern, Till Fluschnik, and Oxana Yu. Tsidulko. “Parameterized algorithms and data reduction for safe convoy routing”. In: *Proceedings of the 18th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS’18)*. Vol. 65. OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Aug. 29, 2018, 10:1–10:19. DOI: 10.4230/OASIcs.ATMOS.2018.10 (cited on p. viii).

[BFT19] René van Bevern, Till Fluschnik, and Oxana Yu. Tsidulko. “On $(1+\varepsilon)$-approximate data reduction for the rural postman problem”. In: *Proceedings of the 18th International Conference on Mathematical Optimization Theory and Operations Research (MOTOR’19)*. Vol. 11548. Lecture Notes in Computer Science. Springer, 2019, pp. 279–294. DOI: 10.1007/978-3-030-22629-9_20 (cited on p. x).

[BFT20] René van Bevern, Till Fluschnik, and Oxana Yu. Tsidulko. “Parameterized algorithms and data reduction for the short secluded s-t-path problem”. In: *Networks* 75.1 (2020), pp. 34–63. DOI: 10.1002/net.21904 (cited on pp. viii, 159, 171, 188, 190).

[BG93] Jonathan F. Buss and Judy Goldsmith. “Nondeterminism within P”. In: *SIAM Journal on Computing* 22.3 (1993), pp. 560–572. DOI: 10.1137/0222038 (cited on pp. 2, 86).

[BGR14] Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. “Two-page book embeddings of 4-planar graphs”. In: *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS’14)*. Vol. 25. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014, pp. 137–148. DOI: 10.4230/LIPIcs.STACS.2014.137 (cited on p. 49).

[Bin+12] Daniel Binkele-Raible, Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. “Kernel(s) for problems with no kernel: on out-trees with many leaves”. In: *ACM Transactions on Algo-*

rithms 8.4 (2012), p. 38. DOI: 10.1145/2344422.2344428 (cited on pp. 4, 11).

[BJ92] Thang Nguyen Bui and Curt Jones. "Finding good approximate vertex and edge partitions is NP-hard". In: *Information Processing Letters* 42.3 (1992), pp. 153–159. DOI: 10.1016/0020-0190(92)90140-Q (cited on p. 201).

[BJK11] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Cross-composition: a new technique for kernelization lower bounds". In: *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS'11)*. Vol. 9. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011, pp. 165–176. DOI: 10.4230/LIPIcs.STACS.2011.165 (cited on p. 4).

[BJK14] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. "Kernelization lower bounds by cross-composition". In: *SIAM Journal on Discrete Mathematics* 28.1 (2014), pp. 277–305. DOI: 10.1137/120880240 (cited on pp. 8, 21).

[BKM01] Gunnar Brinkmann, Jack H. Koolen, and V. Moulton. "On the hyperbolicity of chordal graphs". In: *Annals of Combinatorics* 5.1 (2001), pp. 61–69. DOI: 10.1007/s00026-001-8007-7 (cited on p. 116).

[BKS95] Amotz Bar-Noy, Samir Khuller, and Baruch Schieber. *The Complexity of Finding Most Vital Arcs and Nodes*. Tech. rep. College Park, MD, USA, 1995 (cited on pp. 49, 50, 52).

[BMS15] Ahmad Biniaz, Anil Maheshwari, and Michiel H. M. Smid. "On the hardness of full Steiner tree problems". In: *Journal of Discrete Algorithms* 34 (2015), pp. 118–127. DOI: 10.1016/j.jda.2015.05.013 (cited on p. 78).

[BNN15] Cristina Bazgan, André Nichterlein, and Rolf Niedermeier. "A refined complexity analysis of finding the most vital edges for undirected shortest paths". In: *Proceedings of the 9th International Conference on Algorithms and Complexity (CIAC'15)*. Vol. 9079. Lecture Notes in Computer Science. Springer, 2015, pp. 47–60. DOI: 10.1007/978-3-319-18173-8_3 (cited on p. 45).

[Bod+08a] Hans L. Bodlaender, Erik D. Demaine, Michael R. Fellows, Jiong Guo, Danny Hermelin, Daniel Lokshtanov, Moritz Müller, Venkatesh Raman, Johan van Rooij, and Frances A. Rosamond. *Open Problems in Parameterized and Exact Computation — IWPEC 2008*. Tech. rep. , Utrecht, The Netherlands, 2008 (cited on p. 11).

[Bod+08b] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. "On problems without polynomial kernels (extended abstract)". In: *Proceedings of the 35th International Colloquium on Automata, Lan-*

guages and Programming (ICALP'08). Vol. 5125. Lecture Notes in Computer Science. Springer, 2008, pp. 563–574. DOI: 10.1016/j.jcss.2009.04.001 (cited on p. 4).

[Bod+09] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. "On problems without polynomial kernels". In: *Journal of Computer and System Sciences* 75.8 (2009), pp. 423–434. DOI: 10.1016/j.jcss.2009.04.001 (cited on pp. 3, 5, 7, 8, 10, 21).

[Bod+16] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. "(Meta) Kernelization". In: *Journal of the ACM* 63.5 (2016), 44:1–44:69. DOI: 10.1145/2973749 (cited on pp. 60, 61, 224).

[Bod98] Hans L. Bodlaender. "A partial k-arboretum of graphs with bounded treewidth". In: *Theoretical Computer Science* 209.1-2 (1998), pp. 1–45. DOI: 10.1016/S0304-3975(97)00228-4 (cited on p. 25).

[Bor+15] Michele Borassi, David Coudert, Pierluigi Crescenzi, and Andrea Marino. "On computing the hyperbolicity of real-world graphs". In: *Proceedings of 23rd Annual European Symposium on Algorithms (ESA'15)*. Vol. 9294. Lecture Notes in Computer Science. Springer, 2015, pp. 215–226. DOI: 10.1007/978-3-662-48350-3_19 (cited on p. 114).

[Bri+18] Markus Brill, Till Fluschnik, Vincent Froese, Brijnesh J. Jain, Rolf Niedermeier, and David Schultz. "Exact mean computation in dynamic time warping spaces". In: *Proceedings of the 2018 SIAM International Conference on Data Mining (SDM'18)*. Society for Industrial and Applied Mathematics, 2018, pp. 540–548. DOI: 10.1137/1.9781611975321.61 (cited on p. x).

[Bri+19] Markus Brill, Till Fluschnik, Vincent Froese, Brijnesh J. Jain, Rolf Niedermeier, and David Schultz. "Exact mean computation in dynamic time warping spaces". In: *Data Mining and Knowledge Discovery* 33.1 (2019), pp. 252–291. DOI: 10.1007/s10618-018-0604-8 (cited on p. x).

[Bri14] Karl Bringmann. "Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails". In: *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*. 2014, pp. 661–670. DOI: 10.1109/FOCS.2014.76 (cited on p. 90).

[Bru+06] Maurizio Bruglieri, Matthias Ehrgott, Horst W. Hamacher, and Francesco Maffioli. "An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints". In: *Discrete Applied Mathematics*. Proceedings of the 2nd Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2003) 154.9 (2006), pp. 1344–1357. DOI: 10.1016/j.dam.2005.05.036 (cited on p. 201).

[BTY09] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. “Kernel bounds for disjoint cycles and disjoint paths”. In: *Proceedings of 17th Annual European Symposium on Algorithms (ESA'09)*. Vol. 5757. Lecture Notes in Computer Science. Springer, 2009, pp. 635–646. DOI: 10.1007/978-3-642-04128-0_57 (cited on p. 4).

[BTY11] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. “Kernel bounds for disjoint cycles and disjoint paths”. In: *Theoretical Computer Science* 412.35 (2011), pp. 4570–4578. DOI: 10.1016/j.tcs.2011.04.039 (cited on p. 9).

[Cai+97] Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. “Advice classes of parameterized tractability”. In: *Annals of Pure and Applied Logic* 84.1 (1997), pp. 119–138. DOI: 10.1016/S0168-0072(95)00020-8 (cited on p. 3).

[Cai08] Leizhen Cai. “Parameterized complexity of cardinality constrained optimization problems”. In: *The Computer Journal* 51.1 (2008), pp. 102–121. DOI: 10.1093/comjnl/bxm086 (cited on p. 201).

[CCC06] Leizhen Cai, Siu Man Chan, and Siu On Chan. “Random separation: A new method for solving fixed-cardinality optimization problems”. In: *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC'06)*. Vol. 4169. Lecture Notes in Computer Science. Springer, 2006, pp. 239–250. DOI: 10.1007/11847250_22 (cited on p. 201).

[CCL15] Nathann Cohen, David Coudert, and Aurélien Lancin. “On computing the Gromov hyperbolicity”. In: *ACM Journal of Experimental Algorithmics* 20 (2015), 1.6:1–1.6:18. DOI: 10.1145/2780652 (cited on pp. 114, 116, 117).

[CD14] David Coudert and Guillaume Ducoffe. “Recognition of C_4-free and 1/2-hyperbolic graphs”. In: *SIAM Journal on Discrete Mathematics* 28.3 (2014), pp. 1601–1617. DOI: 10.1137/140954787 (cited on p. 114).

[CFM09] Yijia Chen, Jörg Flum, and Moritz Müller. “Lower bounds for kernelizations and other preprocessing procedures”. In: *Proceedings of the 5th Conference on Computability in Europe (CiE'09)*. Vol. 5635. Lecture Notes in Computer Science. Springer, 2009, pp. 118–128. DOI: 10.1007/978-3-642-03073-4_13 (cited on p. 4).

[CFM11] Yijia Chen, Jörg Flum, and Moritz Müller. “Lower bounds for kernelizations and other preprocessing procedures”. In: *Theory of Computing Systems* 48.4 (2011), pp. 803–839. DOI: 10.1007/s00224-010-9270-y (cited on pp. 5, 65, 67–70, 73, 83, 87, 97).

[Che+06] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. “Strong computational lower bounds via parameterized complexity”. In: *Journal of*

Computer and System Sciences 72.8 (2006), pp. 1346–1367. DOI: 10.1016/j.jcss.2006.04.007 (cited on p. 18).

[Che+17] Shiri Chechik, Matthew P. Johnson, Merav Parter, and David Peleg. "Secluded connectivity problems". In: *Algorithmica* 79.3 (2017), pp. 708–741. DOI: 10.1007/s00453-016-0222-z (cited on pp. 141, 160, 199, 200).

[Chv+75] J Chvátalová, AK Dewdney, NE Gibbs, and RR Korfhage. *The Bandwidth Problem for Graphs—A Collection of Recent Results*. Tech. rep. CS 7502. Department of Computer Science, Southern Methodist University, 1975 (cited on p. 77).

[CKJ01] Jianer Chen, Iyad A. Kanj, and Weijia Jia. "Vertex cover: further observations and further improvements". In: *Journal of Algorithms* 41.2 (2001), pp. 280–301. DOI: 10.1006/jagm.2001.1186 (cited on p. 3).

[CN85] Norishige Chiba and Takao Nishizeki. "Arboricity and subgraph listing algorithms". In: *SIAM Journal on Computing* 14.1 (1985), pp. 210–223. DOI: 10.1137/0214017 (cited on p. 110).

[Coh+17] Nathann Cohen, David Coudert, Guillaume Ducoffe, and Aurélien Lancin. "Applying clique-decomposition for computing Gromov hyperbolicity". In: *Theoretical Computer Science* 690 (2017), pp. 114–139. DOI: 10.1016/j.tcs.2017.06.001 (cited on pp. 114, 118).

[CPP16] Marek Cygan, Marcin Pilipczuk, and Michal Pilipczuk. "Known algorithms for edge clique cover are probably optimal". In: *SIAM Journal on Computing* 45.1 (2016), pp. 67–83. DOI: 10.1137/130947076 (cited on p. 68).

[CPS04] Andrea E. F. Clementi, Paolo Penna, and Riccardo Silvestri. "On the power assignment problem in radio networks". In: *Mobile Networks and Applications (MONET)* 9.2 (2004), pp. 125–140. DOI: 10.1023/B:MONE.0000013624.32948.87 (cited on p. 147).

[Cyg+12] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. "Kernelization hardness of connectivity problems in d-degenerate graphs". In: *Discrete Applied Mathematics* 160.15 (2012), pp. 2131–2141. DOI: 10.1016/j.dam.2012.05.016 (cited on p. 84).

[Cyg+14] Marek Cygan, Daniel Lokshtanov, Fedor Fomin, Bart M. P. Jansen, Lukasz Kowalik, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Open problems of the FPT School 2014, 17-22 August 2014, Będlewo, Poland*. Tech. rep. Available at http://fptschool.mimuw.edu.pl/opl.pdf. 2014 (cited on pp. 41, 44, 146).

[Cyg+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameter-*

ized Algorithms. Springer, 2015. DOI: 10.1007/978-3-319-21275-3 (cited on pp. 12, 178).

[DF13] Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. DOI: 10.1007/978-1-4471-5559-1 (cited on pp. 12, 109).

[DF95a] Rodney G. Downey and Michael R. Fellows. “Fixed-parameter tractability and completeness I: basic results”. In: *SIAM Journal on Computing* 24.4 (1995), pp. 873–921. DOI: 10.1137/S0097539792228228 (cited on p. 16).

[DF95b] Rodney G. Downey and Michael R. Fellows. “Parameterized computational feasibility”. In: *Feasible Mathematics II. Progress in Computer Science and Applied Logic*. Vol. 13. Birkhäuser Boston, 1995, pp. 219–244. DOI: 10.1007/978-1-4612-2566-9_7 (cited on p. 1).

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999 (cited on pp. 74, 86).

[DFS97] Rodney G. Downey, Michael R. Fellows, and Ulrike Stege. “Parameterized complexity: a framework for systematically confronting computational intractability”. In: *Proceedings of a DIMACS Workshop on Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*. Vol. 49. Discrete Mathematics and Theoretical Computer Science (DIMACS). DIMACS/AMS, 1997, pp. 49–100. DOI: 10.1090/dimacs/049/04 (cited on pp. 2, 3, 9).

[Die10] Reinhard Diestel. *Graph Theory*. 4th. Vol. 173. Graduate Texts in Mathematics. Springer, 2010 (cited on pp. 12, 29, 75).

[DK18] Pavel Dvorák and Dusan Knop. “Parameterized complexity of length-bounded cuts and multicuts”. In: *Algorithmica* 80.12 (2018), pp. 3597–3617. DOI: 10.1007/s00453-018-0408-7 (cited on p. 45).

[DLS14] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. “Kernelization lower bounds through colors and IDs”. In: *ACM Transactions on Algorithms* 11.2 (2014), 13:1–13:20. DOI: 10.1145/2650261 (cited on pp. 9, 21).

[DM10] Holger Dell and Dieter van Melkebeek. “Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses”. In: *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC’10)*. ACM, 2010, pp. 251–260. DOI: 10.1145/1806689.1806725 (cited on p. 4).

[Dow+03] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto, and F. Rosamond. “Cutting up is hard to do: the parameterized complexity of k-Cut and related problems”. In: *Electronic Notes in Theoretical Computer Science* 78 (2003), pp. 209–222. DOI: 10.1016/S1571-0661(04)81014-4 (cited on p. 201).

[DPS02] Josep Díaz, Jordi Petit, and Maria J. Serna. “A survey of graph layout problems”. In: *ACM Computing Surveys* 34.3 (2002), pp. 313–356. DOI: 10.1145/568522.568523 (cited on pp. 75–77).

[Dru12] Andrew Drucker. “New limits to classical and quantum instance compression”. In: *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS’12)*. IEEE Computer Society, 2012, pp. 609–618. DOI: 10.1109/FOCS.2012.71 (cited on p. 4).

[Dru15] Andrew Drucker. “New limits to classical and quantum instance compression”. In: *SIAM Journal on Computing* 44.5 (2015), pp. 1443–1479. DOI: 10.1137/130927115 (cited on p. 8).

[Duf65] Richard J Duffin. “Topology of series-parallel networks”. In: *Journal of Mathematical Analysis and Applications* 10.2 (1965), pp. 303–318. DOI: https://doi.org/10.1016/0022-247X(65)90125-3 (cited on p. 36).

[DW71] S. E. Dreyfus and R. A. Wagner. “The Steiner problem in graphs”. In: *Networks* 1.3 (1971), pp. 195–207. DOI: 10.1002/net.3230010302 (cited on p. 79).

[EGS18] Eduard Eiben, Robert Ganian, and Stefan Szeider. “Meta-kernelization using well-structured modulators”. In: *Discrete Applied Mathematics* 248 (2018), pp. 153–167. DOI: 10.1016/j.dam.2017.09.018 (cited on p. 224).

[Eis+19] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. “An algorithmic theory of integer programming”. In: *CoRR* abs/1904.01361 (2019). arXiv: 1904.01361 (cited on p. 157).

[Est+05] Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, and Frances A. Rosamond. “FPT is P-time extremal structure I”. In: *Algorithms and Complexity in Durham 2005 - Proceedings of the First ACiD Workshop, 8-10 July 2005, Durham, UK*. Vol. 4. Texts in Algorithmics. King’s College, London, 2005, pp. 1–41 (cited on pp. 5, 11).

[Ets+17] Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. “Polynomial kernels for weighted problems”. In: *Journal of Computer and System Sciences* 84.Supplement C (2017), pp. 1–10. DOI: 10.1016/j.jcss.2016.06.004 (cited on pp. 142, 144, 146, 156, 157).

[Fel+08] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. “Graph layout problems parameterized by vertex cover”. In: *Proceedings of the 19th International Symposium on Algorithms and Computation (ISAAC’08)*. Ed. by Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga. Vol. 5369. Lecture Notes in Computer Science.

Springer, 2008, pp. 294–305. DOI: 10.1007/978-3-540-92182-0_28 (cited on p. 144).

[Fel+09] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. "On the parameterized complexity of multiple-interval graph problems". In: *Theoretical Computer Science* 410.1 (2009), pp. 53–61. DOI: 10.1016/j.tcs.2008.09.065 (cited on pp. 178, 219).

[Fel+12a] Michael R. Fellows, Jiong Guo, Dániel Marx, and Saket Saurabh. "Data Reduction and Problem Kernels (Dagstuhl Seminar 12241)". In: *Dagstuhl Reports* 2.6 (2012), pp. 26–50. DOI: 10.4230/DagRep.2.6.26 (cited on p. 41).

[Fel+12b] Michael R. Fellows, Ariel Kulik, Frances A. Rosamond, and Hadas Shachnai. "Parameterized approximation via fidelity preserving transformations". In: *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP'12).* Vol. 7391. Lecture Notes in Computer Science. Springer, 2012, pp. 351–362. DOI: 10.1007/978-3-642-31594-7_30 (cited on pp. 5, 10).

[Fel+18] Michael R. Fellows, Lars Jaffke, Aliz Izabella Király, Frances A. Rosamond, and Mathias Weller. "What is known about vertex cover kernelization?" In: *Adventures Between Lower Bounds and Higher Altitudes - Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday.* Vol. 11011. Lecture Notes in Computer Science. Springer, 2018, pp. 330–356. DOI: 10.1007/978-3-319-98355-4_19 (cited on p. 2).

[Fer+09] Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Daniel Raible, Saket Saurabh, and Yngve Villanger. "Kernel(s) for problems with no kernel: on out-trees with many leaves". In: *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS'09).* Vol. 3. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009, pp. 421–432. DOI: 10.4230/LIPIcs.STACS.2009.1843 (cited on p. 5).

[Fer+18] Henning Fernau, Till Fluschnik, Danny Hermelin, Andreas Krebs, Hendrik Molter, and Rolf Niedermeier. "Diminishable parameterized problems and strict polynomial kernelization". In: *Proceedings of the 14th Conference on Computability in Europe (CiE'18).* Vol. 10936. Lecture Notes in Computer Science. Springer, 2018, pp. 161–171. DOI: 10.1007/978-3-319-94418-0_17 (cited on pp. vii, 87).

[Fer+20] Henning Fernau, Till Fluschnik, Danny Hermelin, Andreas Krebs, Hendrik Molter, and Rolf Niedermeier. "Diminishable parameterized problems and strict polynomial kernelization". In: *Computability* 9.1 (2020), pp. 1–24. DOI: 10.3233/COM-180220 (cited on pp. vii, 67, 89).

[Fer17] Henning Fernau. "Extremal kernelization: a commemorative paper". In: *Proceedings of the 28th International Workshop on Combinatorial Algorithms*

(IWOCA'17). Vol. 10765. Lecture Notes in Computer Science. Springer, 2017, pp. 24–36. DOI: 10.1007/978-3-319-78825-8_3 (cited on p. 86).

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. DOI: 10.1007/3-540-29953-X (cited on p. 12).

[FGK13] F. Fomin, P. Golovach, and J. Korhonen. "On the parameterized complexity of cutting a few vertices from a graph". In: *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science (MFCS'13)*. Vol. 8087. Lecture Notes in Computer Science. Springer, 2013, pp. 421–432. DOI: 10.1007/978-3-642-40313-2_38 (cited on pp. 200, 201, 205, 206).

[FIV15] Hervé Fournier, Anas Ismail, and Antoine Vigneron. "Computing the Gromov hyperbolicity of a discrete metric space". In: *Information Processing Letters* 115.6-8 (2015), pp. 576–579. DOI: 10.1016/j.ipl.2015.02.002 (cited on p. 114).

[Flu+15] Till Fluschnik, Stefan Kratsch, Rolf Niedermeier, and Manuel Sorge. "The parameterized complexity of the minimum shared edges problem". In: *Proceedings of the 35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS'15)*. Vol. 45. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, pp. 448–462. DOI: 10.4230/LIPIcs.FSTTCS.2015.448 (cited on p. x).

[Flu+16] Till Fluschnik, Danny Hermelin, André Nichterlein, and Rolf Niedermeier. "Fractals for kernelization lower bounds, with an application to length-bounded cut problems". In: *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*. Vol. 55. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 25:1–25:14. DOI: 10.4230/LIPIcs.ICALP.2016.25 (cited on p. vii).

[Flu+17a] Till Fluschnik, Meike Hatzel, Steffen Härtlein, Hendrik Molter, and Henning Seidler. "The minimum shared edges problem on grid-like graphs". In: *Proceedings of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'17)*. Vol. 10520. Springer, 2017, pp. 249–262. DOI: 10.1007/978-3-319-68705-6_19 (cited on pp. x, 21).

[Flu+17b] Till Fluschnik, Christian Komusiewicz, George B. Mertzios, André Nichterlein, Rolf Niedermeier, and Nimrod Talmon. "When can graph hyperbolicity be computed in linear time?" In: *Proceedings of the 15th International Symposium on Algorithms and Data Structures (WADS'17)*. Vol. 10389. Lecture Notes in Computer Science. Springer, 2017, pp. 397–408. DOI: 10.1007/978-3-319-62127-2_34 (cited on pp. vii, 90).

[Flu+18a] Till Fluschnik, Danny Hermelin, André Nichterlein, and Rolf Niedermeier. "Fractals for kernelization lower bounds". In: *SIAM Journal on Discrete*

Mathematics 32.1 (2018), pp. 656–681. DOI: 10.1137/16M1088740 (cited on pp. vii, 23, 43).

[Flu+18b] Till Fluschnik, Hendrik Molter, Rolf Niedermeier, and Philipp Zschoche. "Temporal graph classes: a view through temporal separators". In: *Proceedings of the 44th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'11)*. Vol. 11159. Lecture Notes in Computer Science. Springer, 2018, pp. 216–227. DOI: 10.1007/978-3-030-00256-5_18 (cited on p. x).

[Flu+19a] Till Fluschnik, Christian Komusiewicz, George B. Mertzios, André Nichterlein, Rolf Niedermeier, and Nimrod Talmon. "When can graph hyperbolicity be computed in linear time?" In: *Algorithmica* 81.5 (2019), pp. 2016–2045. DOI: 10.1007/s00453-018-0522-6 (cited on pp. vii, 113, 138).

[Flu+19b] Till Fluschnik, Stefan Kratsch, Rolf Niedermeier, and Manuel Sorge. "The parameterized complexity of the minimum shared edges problem". In: *Journal of Computer and System Sciences* 106 (2019), pp. 23–48. DOI: 10.1016/j.jcss.2018.12.002 (cited on pp. x, 21).

[Flu+19c] Till Fluschnik, Rolf Niedermeier, Valentin Rohm, and Philipp Zschoche. "Multistage vertex cover". In: *Proceedings of the 14th International Symposium on Parameterized and Exact Computation (IPEC'19)*. Vol. 148. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 14:1–14:14. DOI: 10.4230/LIPIcs.IPEC.2019.14 (cited on p. x).

[Flu+19d] Till Fluschnik, Piotr Skowron, Mervin Triphaus, and Kai Wilker. "Fair knapsack". In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*. AAAI Press, 2019, pp. 1941–1948. DOI: 10.1609/aaai.v33i01.33011941 (cited on p. x).

[Flu+20] Till Fluschnik, Hendrik Molter, Rolf Niedermeier, Malte Renken, and Philipp Zschoche. "Temporal graph classes: a view through temporal separators". In: *Theoretical Computer Science* 806 (2020), pp. 197–218. DOI: 10.1016/j.tcs.2019.03.031 (cited on p. x).

[FMN18] Till Fluschnik, George B. Mertzios, and André Nichterlein. "Kernelization lower bounds for finding constant-size subgraphs". In: *Proceedings of the 14th Conference on Computability in Europe (CiE'18)*. Vol. 10936. Lecture Notes in Computer Science. Springer, 2018, pp. 183–193. DOI: 10.1007/978-3-319-94418-0_19 (cited on pp. vii, 89).

[FMS17] Till Fluschnik, Marco Morik, and Manuel Sorge. "The complexity of routing with few collisions". In: *Proceedings of the 21st International Symposium on Fundamentals of Computation Theory (FCT'17)*. Ed. by Ralf Klasing and Marc Zeitoun. Vol. 10472. Lecture Notes in Computer Science. Springer, 2017, pp. 257–270. DOI: 10.1007/978-3-662-55751-8_21 (cited on p. x).

[FMS19] Till Fluschnik, Marco Morik, and Manuel Sorge. “The complexity of routing with collision avoidance”. In: *Journal of Computer and System Sciences* 102 (2019), pp. 69–86. DOI: 10.1016/j.jcss.2019.01.001 (cited on p. x).

[Fom+12] F. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh. “Planar F-deletion: approximation, kernelization and optimal FPT algorithms”. In: *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS’12)*. IEEE Computer Society, 2012, pp. 470–479. DOI: 10.1109/FOCS.2012.62 (cited on p. 214).

[Fom+17a] Fedor V. Fomin, Petr A. Golovach, Nikolay Karpov, and Alexander S. Kulikov. “Parameterized complexity of secluded connectivity problems”. In: *Theory of Computing Systems* 61.3 (2017), pp. 795–819. DOI: 10.1007/s00224-016-9717-x (cited on pp. 141, 160, 161).

[Fom+17b] Fedor V. Fomin, Daniel Lokshtanov, Michal Pilipczuk, Saket Saurabh, and Marcin Wrochna. “Fully polynomial-time parameterized computations for graphs and matrices of low treewidth”. In: *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’17)*. SIAM, 2017, pp. 1419–1432. DOI: 10.1137/1.9781611974782.92 (cited on p. 90).

[Fom+18] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, Michal Pilipczuk, and Marcin Wrochna. “Fully polynomial-time parameterized computations for graphs and matrices of low treewidth”. In: *ACM Transactions on Algorithms* 14.3 (2018), 34:1–34:45. DOI: 10.1145/3186898 (cited on p. 90).

[Fom+19] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. DOI: 10.1017/9781107415157 (cited on p. 2).

[FR07] Henning Fernau and Daniel Raible. “Alliances in graphs: a complexity-theoretic study”. In: *Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM’07), Volume II*. Institute of Computer Science AS CR, Prague, 2007, pp. 61–70 (cited on p. 86).

[FS08] Lance Fortnow and Rahul Santhanam. “Infeasibility of instance compression and succinct PCPs for NP”. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC’08)*. ACM, 2008, pp. 133–142. DOI: 10.1145/1374376.1374398 (cited on p. 4).

[FS11] Lance Fortnow and Rahul Santhanam. “Infeasibility of instance compression and succinct PCPs for NP”. In: *Journal of Computer and System Sciences* 77.1 (2011), pp. 91–106. DOI: 10.1016/j.jcss.2010.06.007 (cited on pp. 3, 7, 8).

[FS14] Fedor V. Fomin and Saket Saurabh. “Kernelization methods for fixed-parameter tractability”. In: *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2014, pp. 260–282 (cited on p. 2).

[FS16] Till Fluschnik and Manuel Sorge. “The minimum shared edges problem on planar graphs”. Available on arXiv:1602.01385. 2016 (cited on p. x).

[FT87] András Frank and Éva Tardos. “An application of simultaneous diophantine approximation in combinatorial optimization”. In: *Combinatorica* 7.1 (1987), pp. 49–65. DOI: 10.1007/BF02579200 (cited on pp. ix, 6, 141–145, 156, 157, 223).

[Gae04] Marco Gaertler. “Clustering”. In: *Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]*. Ed. by Ulrik Brandes and Thomas Erlebach. Vol. 3418. Lecture Notes in Computer Science. Springer, 2004, pp. 178–215. DOI: 10.1007/978-3-540-31955-9_8 (cited on p. 199).

[Gia+16] Archontia C. Giannopoulou, Daniel Lokshtanov, Saket Saurabh, and Ondřej Suchý. “Tree deletion set has a polynomial kernel (but no $\mathrm{OPT}^{O(1)}$ approximation)”. In: *SIAM Journal on Discrete Mathematics* 30.3 (2016), pp. 1371–1384. DOI: 10.1137/15M1038876 (cited on pp. 209, 213, 214).

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979 (cited on p. 12).

[GJS76] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. “Some simplified NP-complete graph problems”. In: *Theoretical Computer Science* 1.3 (1976), pp. 237–267. DOI: 10.1016/0304-3975(76)90059-1 (cited on p. 53).

[GL14] Venkatesan Guruswami and Euiwoong Lee. “Inapproximability of feedback vertex set for bounded length cycles”. In: *Electronic Colloquium on Computational Complexity (ECCC)* 21 (2014), p. 6 (cited on pp. 59, 60).

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. “The ellipsoid method and its consequences in combinatorial optimization”. In: *Combinatorica* 1.2 (1981), pp. 169–197. DOI: 10.1007/BF02579273 (cited on pp. 143, 144).

[GMN17] Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. “Polynomial fixed-parameter algorithms: a case study for longest path on interval graphs”. In: *Theoretical Computer Science* 689 (2017), pp. 67–95. DOI: 10.1016/j.tcs.2017.05.017 (cited on pp. 90, 115).

[GN07] Jiong Guo and Rolf Niedermeier. “Invitation to data reduction and problem kernelization”. In: *ACM SIGACT News* 38.1 (2007), pp. 31–45. DOI: 10.1145/1233481.1233493 (cited on pp. 2, 10, 11, 68).

[GO95] Anka Gajentaan and Mark H. Overmars. “On a class of $O(n^2)$ problems in computational geometry”. In: *Computational Geometry: Theory and Applications* 5 (1995), pp. 165–185. DOI: 10.1016/j.comgeo.2011.11.006 (cited on p. 18).

[Gol+17] Petr A. Golovach, Pinar Heggernes, Paloma T. Lima, and Pedro Montealegre. “Finding connected secluded subgraphs”. In: *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC’17)*. Vol. 89. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 18:1–18:13. DOI: 10.4230/LIPIcs.IPEC.2017.18 (cited on p. 160).

[Gro87] Mikhael Gromov. “Hyperbolic groups”. In: *Essays in Group Theory, MSRI Publ., vol. 8*. Springer New York, 1987, pp. 75–263 (cited on p. 113).

[GSS16] Robert Ganian, Friedrich Slivovsky, and Stefan Szeider. “Meta-kernelization with structural parameters”. In: *Journal of Computer and System Sciences* 82.2 (2016), pp. 333–346. DOI: 10.1016/j.jcss.2015.08.003 (cited on p. 224).

[GT11] Petr A. Golovach and Dimitrios M. Thilikos. “Paths of bounded length and their cuts: parameterized complexity and algorithms”. In: *Discrete Optimization* 8.1 (2011), pp. 72–86. DOI: 10.1016/j.disopt.2010.09.009 (cited on pp. 5, 21, 44, 45, 48, 53).

[Hea85] Lenwood Scott Heath. “Algorithms for Embedding Graphs in Books”. PhD thesis. University of North Carolina at Chapel Hill, USA, 1985 (cited on p. 49).

[Her+13] Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. “A completeness theory for polynomial (Turing) kernelization”. In: *Proceedings of the 8th International Symposium on Parameterized and Exact Computation (IPEC’13)*. Vol. 8246. Lecture Notes in Computer Science. Springer, 2013, pp. 202–215. DOI: 10.1007/978-3-319-03898-8_18 (cited on p. 4).

[Her+15] Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. “A completeness theory for polynomial (Turing) kernelization”. In: *Algorithmica* 71.3 (2015), pp. 702–730. DOI: 10.1007/s00453-014-9910-8 (cited on pp. 11, 80, 82, 94, 190, 191, 194).

[HKS15] F. Hüffner, C. Komusiewicz, and M. Sorge. “Finding highly connected subgraphs”. In: *Proceedings of the 41st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM’15)*. Vol. 8939.

Lecture Notes in Computer Science. Springer, 2015, pp. 254–265. DOI: 10.1007/978-3-662-46078-8_21 (cited on pp. 199, 201).

[HP10] Michel Habib and Christophe Paul. "A survey of the algorithmic aspects of modular decomposition". In: *Computer Science Review* 4.1 (2010), pp. 41–59. DOI: 10.1016/j.cosrev.2010.01.001 (cited on p. 120).

[HPV98] Michel Habib, Christophe Paul, and Laurent Viennoti. "A synthesis on partition refinement: a useful routine for strings, graphs, boolean matrices and automata". In: *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS'98)*. Vol. 1373. Springer, 1998, pp. 25–38. DOI: 10.1007/BFb0028546 (cited on p. 188).

[HS17] Ronald de Haan and Stefan Szeider. "Parameterized complexity classes beyond para-NP". In: *Journal of Computer and System Sciences* 87 (2017), pp. 16–57. DOI: 10.1016/j.jcss.2017.02.002 (cited on p. 16).

[HS19] Ronald de Haan and Stefan Szeider. "A compendium of parameterized problems at higher levels of the polynomial hierarchy". In: *Algorithms* 12.9 (2019), p. 188. DOI: 10.3390/a12090188 (cited on p. 17).

[Hua+13] Chao-Wen Huang, Chia-Wei Lee, Huang-Ming Gao, and Sun-Yuan Hsieh. "The internal Steiner tree problem: Hardness and approximations". In: *Journal of Complexity* 29.1 (2013), pp. 27–43. DOI: 10.1016/j.jco.2012.08.005 (cited on p. 87).

[Hüf+09] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. "Isolation concepts for clique enumeration: comparison and computational experiments". In: *Theoretical Computer Science* 410.52 (2009), pp. 5384–5397. DOI: 10.1016/j.tcs.2009.05.008 (cited on pp. 199, 201).

[HW12] Danny Hermelin and Xi Wu. "Weak compositions and their applications to polynomial lower bounds for kernelization". In: *Proceedings of the 23th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*. Society for Industrial and Applied Mathematics, 2012, pp. 104–113. DOI: 10.1137/1.9781611973099.9 (cited on p. 4).

[IIO05] H. Ito, K. Iwama, and T. Osumi. "Linear-time enumeration of isolated cliques". In: *Proceedings of 13th Annual European Symposium on Algorithms (ESA'05)*. Vol. 3669. Lecture Notes in Computer Science. Springer, 2005, pp. 119–130. DOI: 10.1007/11561071_13 (cited on pp. 199, 201).

[IP01] Russell Impagliazzo and Ramamohan Paturi. "On the complexity of k-SAT". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 367–375. DOI: 10.1006/jcss.2000.1727 (cited on p. 18).

[IPS82] Alon Itai, Yehoshua Perl, and Yossi Shiloach. "The complexity of finding maximum disjoint paths with length constraints". In: *Networks* 12.3 (1982), pp. 277–286. DOI: 10.1002/net.3230120306 (cited on pp. 43, 45).

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which problems have strongly exponential complexity?" In: *Journal of Computer and System Sciences* 63.4 (2001), pp. 512–530. DOI: 10.1006/jcss.2001.1774 (cited on p. 18).

[Iwa17] Yoichi Iwata. "Linear-time kernelization for feedback vertex set". In: *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP'17)*. Vol. 80. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 68:1–68:14. DOI: 10.4230/LIPIcs.ICALP.2017.68 (cited on p. 2).

[Jan17] Bart M. P. Jansen. "Turing kernelization for finding long paths and cycles in restricted graph classes". In: *Journal of Computer and System Sciences* 85 (2017), pp. 18–37. DOI: 10.1016/j.jcss.2016.10.008 (cited on p. 11).

[Kha79] Leonid G Khachiyan. "A polynomial algorithm in linear programming". In: *Doklady Academii Nauk SSSR*. Vol. 244. 1979, pp. 1093–1096 (cited on p. 143).

[Kha80] L.G. Khachiyan. "Polynomial algorithms in linear programming". In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72. DOI: https://doi.org/10.1016/0041-5553(80)90061-0 (cited on p. 143).

[KHH04] P. Kristiansen, S. M. Hedetniemi, and S. T. Hedetniemi. "Alliances in graphs". In: *Journal of Combinatorial Mathematics and Combinatorial Computing* 48 (2004), pp. 157–177 (cited on p. 86).

[Kin92] Nancy G. Kinnersley. "The vertex separation number of a graph equals its path-width". In: *Information Processing Letters* 42.6 (1992), pp. 345–350. DOI: 10.1016/0020-0190(92)90234-M (cited on pp. 15, 76).

[KL82] Richard M. Karp and Richard Lipton. "Turing machines that take advice". In: *L'Enseignement Mathématique* 28.2 (1982), pp. 191–209 (cited on p. 17).

[KM02] Jack H. Koolen and Vincent Moulton. "Hyperbolic bridged graphs". In: *European Journal of Combinatorics* 23.6 (2002), pp. 683–699. DOI: 10.1006/eujc.2002.0591 (cited on p. 129).

[Kno17] Dušan Knop. "Structural properties of graphs and efficient algorithms: Problems Between Parameters". PhD thesis. Univerzita Karlova, Matematicko-fyzikální fakulta, 2017 (cited on p. 45).

[Kom+09] C. Komusiewicz, F. Hüffner, H. Moser, and R. Niedermeier. "Isolation concepts for efficiently enumerating dense subgraphs". In: *Theoretical*

Computer Science 410.38-40 (2009), pp. 3640–3654. DOI: 10.1016/j.tcs.2009.04.021 (cited on pp. 199, 201).

[Kra14] Stefan Kratsch. "Recent developments in kernelization: a survey". In: *Bulletin of the EATCS* 113 (2014), pp. 58–97 (cited on pp. 2, 11, 68).

[KS15] Christian Komusiewicz and Manuel Sorge. "An algorithmic framework for fixed-cardinality optimization in sparse graphs applied to dense subgraph problems". In: *Discrete Applied Mathematics* 193 (2015), pp. 145–161. DOI: 10.1016/j.dam.2015.04.029 (cited on p. 201).

[KT06] Jon M. Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley, 2006 (cited on p. 205).

[KW12] Stefan Kratsch and Magnus Wahlström. "Compression via matroids: a randomized polynomial kernel for odd cycle transversal". In: *Proceedings of the 23th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*. Ed. by Yuval Rabani. SIAM, 2012, pp. 94–103. DOI: 10.1137/1.9781611973099.8 (cited on p. 5).

[Len83] Hendrik W. Lenstra. "Integer programming with a fixed number of variables". In: *Mathematics of Operations Research* 8 (1983), pp. 538–548. DOI: 10.1287/moor.8.4.538 (cited on p. 131).

[LF20] Max-Jonathan Luckow and Till Fluschnik. "On the computational complexity of length- and neighborhood-constrained path problems". In: *Information Processing Letters* 156 (2020), p. 105913. DOI: 10.1016/j.ipl.2019.105913 (cited on pp. vii, 159, 160, 171, 188, 190, 197).

[Lin+17] Mugang Lin, Qilong Feng, Jianer Chen, and Wenjun Li. "Partition on trees with supply and demand: kernelization and algorithms". In: *Theoretical Computer Science* 657 (2017), pp. 11–19. DOI: 10.1016/j.tcs.2016.06.044 (cited on p. 10).

[LLL82] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. "Factoring polynomials with rational coefficients". In: *Mathematische Annalen* 261.4 (1982), pp. 515–534 (cited on pp. 144, 157).

[LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. "Lower bounds based on the exponential time hypothesis". In: *Bulletin of the EATCS* 105 (2011), pp. 41–72 (cited on p. 97).

[LMS12] Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. "Kernelization - preprocessing with a guarantee". In: *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*. Vol. 7370. Lecture Notes in Computer Science. Springer, 2012, pp. 129–161. DOI: 10.1007/978-3-642-30891-8_10 (cited on pp. 2, 68).

[LMS13] Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. “Imbalance is fixed parameter tractable”. In: *Information Processing Letters* 113.19-21 (2013), pp. 714–718. DOI: 10.1016/j.ipl.2013.06.010 (cited on p. 41).

[Lok+17] Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. “Lossy kernelization”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC’17)*. ACM, 2017, pp. 224–237. DOI: 10.1145/3055399.3055456 (cited on pp. 5, 224).

[Lok09] Daniel Lokshtanov. “New methods in parameterized algorithms and complexity”. PhD thesis. University of Bergen, Norway, 2009 (cited on pp. 5, 11).

[Luc17] Max-Jonathan Luckow. “Paths under Neighborhood Constraints—Algorithms and Complexity”. Bachelor’s Thesis. Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, 2017 (cited on p. 160).

[LX02] Guo-Hui Lin and Guoliang Xue. “On the terminal Steiner tree problem”. In: *Information Processing Letters* 84.2 (2002), pp. 103–107. DOI: 10.1016/S0020-0190(02)00227-2 (cited on p. 78).

[Mar06] D. Marx. “Parameterized graph separation problems”. In: *Theoretical Computer Science* 351.3 (2006), pp. 394–406. DOI: 10.1016/j.tcs.2005.10.007 (cited on pp. 200, 201).

[MB83] David W. Matula and Leland L. Beck. “Smallest-last ordering and clustering and graph coloring algorithms”. In: *Journal of the ACM* 30.3 (1983), pp. 417–427. DOI: 10.1145/2402.322385 (cited on p. 107).

[Men27] Karl Menger. “Über reguläre Baumkurven”. In: *Mathematische Annalen* 96.1 (1927), pp. 572–582 (cited on p. 27).

[MMG89] Kavindra Malik, Ashok K. Mittal, and Santosh K. Gupta. “The k most vital arcs in the shortest path problem”. In: *Operations Research Letters* 8.4 (1989), pp. 223–227. DOI: 10.1016/0167-6377(89)90065-5 (cited on p. 45).

[MNN17] George B. Mertzios, André Nichterlein, and Rolf Niedermeier. “The power of linear-time data reduction for maximum matching”. In: *Proceedings of the 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS’17)*. Vol. 83. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 46:1–46:14. DOI: 10.4230/LIPIcs.MFCS.2017.46 (cited on pp. 90, 115).

[Moh01] Bojan Mohar. “Face covers and the genus problem for apex graphs”. In: *Journal of Combinatorial Theory, Series B* 82.1 (2001), pp. 102–117. DOI: 10.1006/jctb.2000.2026 (cited on p. 49).

[MP14] Dieter Mitsche and Pawel Pralat. “On the hyperbolicity of random graphs”. In: *The Electronic Journal of Combinatorics* 21.2 (2014), P2.39 (cited on pp. 116, 117).

[MV15] Dániel Marx and László A. Végh. “Fixed-parameter algorithms for minimum-cost edge-connectivity augmentation”. In: *ACM Transactions on Algorithms* 11.4 (2015), 27:1–27:24. DOI: 10.1145/2700210 (cited on pp. 142, 144, 146, 156, 157).

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006 (cited on p. 12).

[OST18] Kahina Ouazine, Hachem Slimani, and Abdelkamel Tari. “Alliances in graphs: parameters, properties and applications—a survey”. In: *AKCE International Journal of Graphs and Combinatorics* 15.2 (2018), pp. 115–154. DOI: https://doi.org/10.1016/j.akcej.2017.05.002 (cited on p. 86).

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994 (cited on p. 12).

[PS16] Feng Pan and Aaron Schild. “Interdiction problems on planar graphs”. In: *Discrete Applied Mathematics* 198 (2016), pp. 215–231. DOI: 10.1016/j.dam.2015.05.036 (cited on p. 45).

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982 (cited on p. 119).

[RS10] Prasad Raghavendra and David Steurer. “Graph expansion and the unique games conjecture”. In: *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC’10)*. ACM, 2010, pp. 755–764. DOI: 10.1145/1806689.1806792 (cited on p. 149).

[SBL87] Anneke A. Schoone, Hans L. Bodlaender, and Jan van Leeuwen. “Diameter increase caused by edge deletion”. In: *Journal of Graph Theory* 11.3 (1987), pp. 409–427. DOI: 10.1002/jgt.3190110315 (cited on p. 53).

[Sch+12] Alexander Schäfer, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. “Parameterized computational complexity of finding small-diameter subgraphs”. In: *Optimization Letters* 6.5 (2012), pp. 883–891. DOI: 10.1007/s11590-011-0311-5 (cited on p. 4).

[Sei83] S. Seidman. “Network structure and minimum degree”. In: *Social Networks* 5.3 (1983), pp. 269–287. DOI: 10.1016/0378-8733(83)90028-X (cited on p. 210).

[Sto76] Larry J. Stockmeyer. “The polynomial-time hierarchy”. In: *Theoretical Computer Science* 3.1 (1976), pp. 1–22. DOI: 10.1016/0304-3975(76)90061-X (cited on p. 17).

[SW18] Manuel Sorge and Mathias Weller. *The graph parameter hierarchy.* 2018 (cited on p. 15).

[Tho10] S. Thomassé. "A $4k^2$ kernel for feedback vertex set". In: *ACM Transactions on Algorithms* 6.2 (2010). DOI: 10.1145/1721837.1721848 (cited on p. 213).

[Vas+15] Virginia Vassilevska Williams, Joshua R. Wang, Ryan Williams, and Huacheng Yu. "Finding four-node subgraphs in triangle time". In: *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15).* SIAM, 2015, pp. 1671–1680. DOI: 10.1137/1.9781611973730.111 (cited on p. 138).

[VW18] Virginia Vassilevska Williams and R. Ryan Williams. "Subcubic equivalences between path, matrix, and triangle problems". In: *Journal of the ACM* 65.5 (2018), 27:1–27:38. DOI: 10.1145/3186893 (cited on pp. 18, 92, 99, 100, 103, 104).

[WBT19] Jouke Witteveen, Ralph Bottesch, and Leen Torenvliet. "A hierarchy of polynomial kernels". In: *Proceedings of the 45th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'19).* Vol. 11376. Lecture Notes in Computer Science. Springer, 2019, pp. 504–518. DOI: 10.1007/978-3-030-10801-4_39 (cited on p. 11).

[Wel13] Mathias Weller. "Institut für Softwaretechnik und Theoretische Informatik, TU Berlin". PhD thesis. Berlin Institute of Technology, 2013 (cited on p. 17).

[Wes00] Douglas B. West. *Introduction to Graph Theory.* 2nd ed. Prentice Hall, 2000 (cited on p. 12).

[WY14] Ryan Williams and Huacheng Yu. "Finding orthogonal vectors in discrete structures". In: *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14).* SIAM, 2014, pp. 1867–1877. DOI: 10.1137/1.9781611973402.135 (cited on p. 121).

[XK17] Mingyu Xiao and Shaowei Kou. "Kernelization and parameterized algorithms for 3-path vertex cover". In: *Proceedings of the 14th Annual Conference on Theory and Applications of Models of Computation (TAMC 2017).* Vol. 10185. Lecture Notes in Computer Science. Springer, 2017, pp. 654–668. DOI: 10.1007/978-3-319-55911-7_47 (cited on p. 10).

[XZ11] Ge Xia and Yong Zhang. "On the small cycle transversal of planar graphs". In: *Theoretical Computer Science* 412.29 (2011), pp. 3501–3509. DOI: 10.1016/j.tcs.2011.02.040 (cited on pp. 44, 60, 61).

[XZ12] Ge Xia and Yong Zhang. "Kernelization for cycle transversal problems". In: *Discrete Applied Mathematics* 160.7-8 (2012), pp. 1224–1231. DOI: 10.1016/j.dam.2011.12.024 (cited on p. 60).

[Yan78] Mihalis Yannakakis. “Node- and edge-deletion NP-complete problems”. In: *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC’78)*. ACM, 1978, pp. 253–264. DOI: 10.1145/800133.804355 (cited on p. 59).

[Yap83] Chee-Keng Yap. “Some consequences of non-uniform conditions on uniform classes”. In: *Theoretical Computer Science* 26 (1983), pp. 287–300. DOI: 10.1016/0304-3975(83)90020-8 (cited on pp. 8, 17).

[Zsc+18] Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. “The complexity of finding small separators in temporal graphs”. In: *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS’18)*. Vol. 117. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 45:1–45:17. DOI: 10.4230/LIPIcs.MFCS.2018.45 (cited on pp. x, 61).

[Zsc+20] Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. “The complexity of finding small separators in temporal graphs”. In: *Journal of Computer and System Sciences* 107 (2020), pp. 72–92. DOI: 10.1016/j.jcss.2019.07.006 (cited on p. x).

[Zsc17] Philipp Zschoche. “On Finding Separators in Temporal Graphs”. MA thesis. Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, 2017 (cited on p. 40).

Appendices

Appendix A.

Problem Zoo

3SUM
Input: Numbers $x_1, \ldots, x_n \in \mathbb{Z}$ with $|z| \in O(n^3)$.
Question: Are there distinct $i, j, k \in \{1, \ldots, n\}$ such that $x_i + x_j + x_k = 0$?

All Pairs Shortest Paths (APSP)
Input: An undirected graph $G = (V, E)$ with $O(\log(|V|))$-bit edge weights.
Task: Compute the lengths $\text{dist}_G(v, w)$ of a shortest v-w path in G for each pair of vertices $v, w \in V$.

Biclique
Input: An undirected bipartite graph $G = (V = A \uplus B, E)$ and an integer k.
Question: Is there a vertex set $X \subseteq V$ of G such that $|X \cap A| = |X \cap B| = k$ and each vertex in $X \cap A$ is adjacent to each vertex in $X \cap B$?

Clique
Input: An undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Is there a vertex set $X \subseteq V$ of G such that $|X| \geq k$ and for all distinct $v, w \in X$ there is $\{v, w\} \in E$?

CNF-Sat
Input: A Boolean formula ϕ in conjunctive normal form (CNF).
Question: Is ϕ satisfiable?

Colorful Graph Motif
Input: An undirected graph $G = (V, E)$, an integer k, and a vertex coloring function $\text{col} : V \to \{1, \ldots, k\}$.
Question: Is there a vertex set $X \subseteq V$ such that $G[X]$ is connected and X contains exactly one vertex of each color?

Connected Vertex Cover
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Is there a vertex set $X \subseteq V$ in G with $|X| \leq k$ and each edge of G is incident to at least one vertex in X and $G[X]$ is connected?

Cutting k Vertices
Input: An undirected graph $G = (V, E)$ and two integers $k \geq 1$ and $\ell \geq 0$.
Question: Is there a non-empty set $S \subseteq V$ such that $|S| = k$ and $|N_G(S)| \leq \ell$?

Cutting at Most k Vertices with Terminal
Input: An undirected graph $G = (V, E)$, a vertex $s \in V$, and two integers $k \geq 1$, $\ell \geq 0$.
Question: Is there a set $S \subseteq V$ such that $s \in S$, $|S| \leq k$, and $|N_G(S)| \leq \ell$?

Directed Path
Input: A directed graph $G = (V, E)$ and an integer k.
Question: Is there a directed path P of length at least k in G?

Directed Small Cycle Transversal (DSCT)
Input: A directed graph $G = (V, E)$, two integers $k, \ell \geq 0$.
Question: Is there a subset $F \subseteq E$ of cardinality at most k such that there is no induced directed cycle of length at most ℓ in $G - F$?

Defensive Alliance
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Is there a vertex set $X \subseteq V$ in G with $|X| \leq k$ such that for each vertex $x \in X$ it holds true that $|N_G[x] \cap X] \geq |N_G[x] \setminus X|$?

Dominating Set
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Is there a vertex set $X \subseteq V$ of G such that $|X| \leq k$ and $N_G[X] = V$?

Edge Clique Cover
Input: An undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Can all edges E of G be covered by k subgraphs of G each forming a clique?

Graph Motif
Input: An undirected graph $G = (V, E)$, an integer k, a vertex coloring function $\text{col} : V \to \{1, \ldots, k\}$, and a multiset M of elements in $\{1, \ldots, k\}$.
Question: Is there a $|M|$-vertex set $X \subseteq V$ such that $G[X]$ is connected and the set of colors of the vertices in X match exactly M?

Hamiltonian Path
Input: An undirected graph $G = (V, E)$.
Question: Is there a path in G that visits each vertex exactly once?

Hitting Set
Input: Given a universe U, a family $\mathcal{F} \subseteq 2^U$ of subsets of U, and an integer k.
Question: Is there a subset $U' \subseteq U$ such that $|U'| \leq k$ and $F \cap U' \neq \emptyset$ for all $F \in \mathcal{F}$?

Independent Set
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Is there a vertex set $X \subseteq V$ of G such that $|X| \geq k$ and $G[X]$ is edge-free?

Internal Steiner Tree
Input: An undirected graph $G = (V = N \uplus T, E)$ and an integer k.
Question: Is there a subgraph H of G such that H is a tree with T being part of its internal vertices?

k-CNF-Sat
Input: A Boolean formula ϕ in conjunctive normal form (CNF) with at most k literals in each clause.
Question: Is ϕ satisfiable?

Length-Bounded Edge-Cut (LBEC)
Input: An undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, and two integers $k, \ell \geq 0$.
Question: Is there a subset $F \subseteq E$ of cardinality at most k such that $\text{dist}_{G-F}(s, t) \geq \ell$?

Longest Path
Input: An undirected graph $G = (V, E)$ and an integer k.
Question: Is there a path P of length at least k in G?

Long Unsecluded Path (LUP)
Input: An undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, and two integers $k \geq 2$ and $\ell \geq 0$.
Question: Is there an s-t path P in G such that $|V(P)| \geq k$ and $|N_G(V(P))| \geq \ell$?

Minimum Diameter Arc Deletion (MDAD)
Input: A strongly connected directed graph $G = (V, E)$, and two integers $k, \ell \geq 0$.
Question: Is there is a subset $F \subseteq E$ of cardinality at most k such that $G - F$ is strongly connected and $\operatorname{diam}(G - F) \geq \ell$?

Minimum Diameter Edge Deletion (MDED)
Input: A connected, undirected graph $G = (V, E)$, and two integers $k, \ell \geq 0$.
Question: Is there a subset $F \subseteq E$ of cardinality at most k such that $G - F$ is connected and $\operatorname{diam}(G - F) \geq \ell$?

Min-Power Symmetric Connectivity (MiPoSyCo)
Input: A connected undirected graph $G = (V, E)$ and edge weights $w\colon E \to \mathbb{N}$.
Task: Find a connected spanning subgraph $T = (V, F)$ of G that minimizes $\sum_{v \in V} \max_{\{u,v\} \in F} w(\{u, v\})$.

Multicolored Clique (MCC)
Input: An undirected k-partite graph $G = (V = V_1 \uplus \ldots \uplus V_k, E)$.
Question: Is there an vertex set $X \subseteq V$ of size k in G with $|X \cap V_i| = 1$ for all $i \in \{1, \ldots, k\}$ and all vertices in X are pairwise adjacent in G?

Multicolored Independent Set (MIS)
Input: An undirected k-partite graph $G = (V = V_1 \uplus \ldots \uplus V_k, E)$.
Question: Is there an independent set $X \subseteq V$ of size k in G with $|X \cap V_i| = 1$ for all $i \in \{1, \ldots, k\}$?

Multicolored Path (MCP)
Input: An undirected graph $G = (V, E)$ and a vertex coloring $\operatorname{col}\colon V \to \{1, \ldots, k\}$.
Question: Is there a simple path P in G that contains exactly one vertex of each color?

Multi-Component Annotated Π (MCA-Π)
Input: An undirected graph $G = (V, E)$, a vertex subset $D \subseteq V$, and an integer k.
Question: Is there a vertex set S in a connected component $G' = (V', E')$ of G such that $(D \cap V') \subseteq S \subseteq V'$, $|S \setminus (D \cap V')| \leq k$, and S fulfills property Π in G'?

MCA-Defensive Alliance
Input: An undirected graph $G = (V, E)$, a vertex subset $D \subseteq V$, and an integer k.
Question: Is there a vertex set S in a connected component $G' = (V', E')$ of G such that $(D \cap V') \subseteq S \subseteq V'$, $|S \setminus (D \cap V')| \leq k$, and S is a defensive alliance in G'?

MCA-Vertex Cover
Input: An undirected graph $G = (V, E)$, a vertex subset $D \subseteq V$, and an integer k.
Question: Is there a vertex set S in a connected component $G' = (V', E')$ of G such that $(D \cap V') \subseteq S \subseteq V'$, $|S \setminus (D \cap V')| \leq k$, and S is a vertex cover of G'?

Negative Weight Triangle (NWT)
Input: An undirected graph G with edge weights $w\colon E(G) \to \mathbb{Z}$.
Question: Is there a triangle T in G with $\sum_{e \in E(T)} w(e) < 0$?

Orthogonal Vectors
Input: Two sets $\vec{A}$ and $\vec{B}$ each containing n binary vectors of length $\ell = O(\log n)$.
Question: Are there two vectors $\vec{a} \in \vec{A}$ and $\vec{b} \in \vec{B}$ such that $\vec{a}$ and $\vec{b}$ are orthogonal, that is, such that there is no position i for which $\vec{a}[i] = \vec{b}[i] = 1$?

Planar Length-Bounded Edge-Cut (Planar-LBEC)
Input: An undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, and two integers $k, \ell \geq 0$, where G admits a planar embedding with s and t being incident to the outer face.
Question: Is there a subset $F \subseteq E$ of cardinality at most k such that $\text{dist}_{G-F}(s,t) \geq \ell$?

Rooted Path
Input: An undirected graph $G = (V, E)$, a vertex $r \in V$, and an integer $k \in \mathbb{N}$.
Question: Is there a path P with endpoint r of length at least k in G?

Secluded Feedback Vertex Set (SFVS)
Input: An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
Question: Is there a set $S \subseteq V$ such that $G - S$ is cycle-free and $|N_G[S]| \leq k$?

Secluded s-t Separator (S$st$S)
Input: An undirected graph $G = (V, E)$, two distinct vertices $s, t \in V$, and an integer $k \geq 0$.
Question: Is there an s-t separator $S \subseteq V \setminus \{s, t\}$ such that $|N_G[S]| \leq k$?

Secluded Π
Input: An undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$.
Question: Is there a vertex subset $S \subseteq V$ such that S satisfies $\Pi(G, S)$ and $|N_G[S]| \leq k$?

Secluded Path
Input: An undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, vertex-weights $w \colon V \to \mathbb{N}$, and two integers $k, C \in \mathbb{N}$.
Question: Is there an s-t path P such that the size of the closed neighborhood $|N_G[V(P)]| \leq k$ and the weight of the closed neighborhood $w(N_G[V(P)]) \leq C$?

Set Cover
Input: A universe U, a family of sets $\mathcal{F} \subseteq 2^U$, and an integer k.
Question: Is there a subset $\mathcal{C} \subseteq \mathcal{F}$ such that $|\mathcal{C}| \leq k$ and $U = \bigcup_{C \in \mathcal{C}} C$?

Short Secluded Path (SSP)
Input: An undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, and two integers $k \geq 2$ and $\ell \geq 0$.
Question: Is there an s-t path P in G such that $|V(P)| \leq k$ and $|N_G(V(P))| \leq \ell$?

Small Secluded Feedback Vertex Set (SSFVS)
Input: An undirected graph $G = (V, E)$ and two integers k, ℓ.
Question: Is there a set $S \subseteq V$ such that $G - S$ is cycle-free, $|S| \leq k$, and $|N_G(S)| \leq \ell$?

Small Secluded s-t Separator (SS$st$S)
Input: An undirected graph $G = (V, E)$, two distinct vertices $s, t \in V$, and two integers $k \geq 0$, $\ell \geq 0$.
Question: Is there an s-t separator $S \subseteq V \setminus \{s, t\}$ such that $|S| \leq k$ and $|N_G(S)| \leq \ell$?

Small Secluded Π
Input: An undirected graph $G = (V, E)$ and two integers $k \geq 1, \ell \geq 0$.
Question: Is there a vertex subset $S \subseteq V$ such that S satisfies $\Pi(G, S)$, $|S| \leq k$, and $|N_G(S)| \leq \ell$?

Small Set Expansion (SSE)
Input: An undirected graph G with edge weights $w : E(G) \to \mathbb{Q}_+$.
Question: Find a non-empty subset $S \subseteq V(G)$ of size at most $|S| \leq n/2$ that minimizes

$$\frac{1}{|S|} \sum_{e \in (S, V(G) \setminus S)} w(e),$$

where $(S, V(G) \setminus S)$ denotes the set of all edges with exactly one endpoint in S.

Short Unsecluded Path (SUP)

Input: An undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, and two integers $k \geq 2$ and $\ell \geq 0$.

Question: Is there an s-t path P in G such that $|V(P)| \leq k$ and $|N_G(V(P))| \geq \ell$?

Terminal Steiner Tree (TST)

Input: An undirected graph $G = (V = N \uplus T, E)$ and an integer k.

Question: Is there a subgraph H of G such that H is a tree with T being its set of leaves?

Triangle Collection (TC)

Input: An undirected graph G with surjective coloring $\text{col} : V(G) \to \{1, \dots, f\}$.

Question: Does there for all color-triples $C \in \binom{\{1,\dots,f\}}{3}$ exist a triangle with vertex set $T = \{x, y, z\}$ in G such that $\text{col}(T) = C$?

Vertex Cover

Input: An undirected graph $G = (V, E)$ and an integer k.

Question: Is there a vertex set $X \subseteq V$ in G with $|X| \leq k$ and each edge in E is incident to at least one vertex in X?

Vertex-Weighted Short Secluded Path (VW-SSP)

Input: An undirected graph $G = (V, E)$ with two distinct vertices $s, t \in V$, two integers $k \geq 2$ and $\ell \geq 0$, and vertex weights $\kappa : V \to \mathbb{N}$, $\lambda : V \to \mathbb{N}_0$, and $\eta : V \to \mathbb{N}_0$.

Question: Is there an s-t-path P with $\sum_{v \in V(P)} \kappa(v) \leq k$ and $\sum_{v \in V(P)} \eta(v) + \sum_{v \in N(V(P))} \lambda(v) \leq \ell$ in G?

Appendix B. Open Problem List

Open Problem 1. Is there a planar vertex-deletion variant of T-fractals suitable for the fractalism technique?

Open Problem 2. Does the vertex-deletion variant of Length-Bounded Edge-Cut parameterized by $k+\ell$ admit a polynomial kernel on planar graphs?

Open Problem 3. Does Small Cycle Transversal parameterized by $k+\ell$ admit a polynomial kernel in general undirected graphs?

Open Problem 4. Is Internal Steiner Tree parameterized by $k+|T|$ diminishable?

Open Problem 5. Is Longest Path parameterized by the solution size k diminishable?

Open Problem 6. Is Connected Vertex Cover parameterized by k or Hitting Set parameterized by n diminishable?

Open Problem 7. Assuming the ETH to hold, does Clique parameterized by the cutwidth cw admit a strong diminisher?

Open Problem 8. Is there a proper (n^{α}, d^{β})-kernel with $3 \leq \alpha \cdot \beta < 5$ for Negative Weight Triangle or Triangle Collection each parameterized by the degeneracy d?

Open Problem 9. Does Hyperbolicity admit a problem kernel computable in quadratic time of size polynomial (or subexponential) in the vertex cover number?

Open Problem 10. Is Hyperbolicity solvable in truly subcubic time?

Open Problem 12. What is the parameterized complexity of Short Unsecluded Path parameterized by ℓ and of Long Unsecluded Path parameterized by $k+\ell$?

Open Problem 13. Does Secluded Feedback Vertex Set admit a kernel with $O(k^c)$ vertices where $c < 5$?

Open Problem 14. Is Small Secluded Feedback Vertex Set fixed-parameter tractable when parameterized by k or by $k+\ell$? If so, does it admit a polynomial kernel?

Open Problem 15. What is the parameterized complexity of Small p-Secluded q-Dominating Set with $p > \frac{q}{2}$ when parameterized by ℓ?

Index