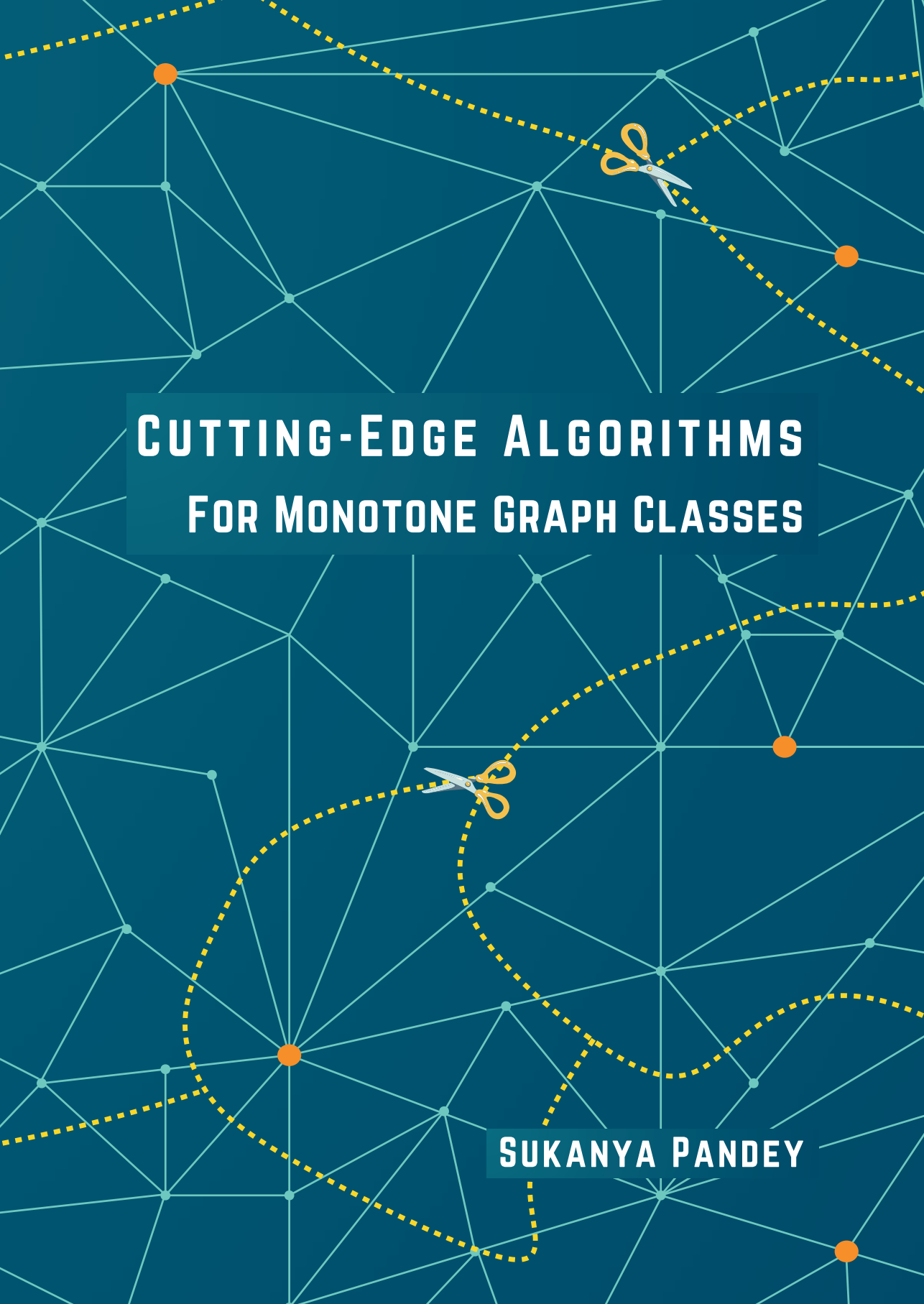

Cutting-Edge Algorithms
For Monotone Graph Classes

Sukanya Pandey

Cutting-Edge Algorithms for Monotone Graph Classes

Sukanya Pandey

ISBN: 978-94-6510-268-9

Cutting-Edge Algorithms for Monotone Graph Classes

Algoritmen voor Monotone Graafklassen op het Scherpst van de Snede

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit Utrecht
op gezag van de
rector magnificus, prof. dr. H.R.B.M. Kummeling,
ingevolge het besluit van het College voor Promoties
in het openbaar te verdedigen op

maandag 21 oktober 2024 des middags te 12.15 uur

door

Sukanya Pandey

geboren op 29 augustus 1994
te Patna, Bihar, India

Promotor:
Prof. dr. H.L. Bodlaender

Copromotor:
Dr. E.J. van Leeuwen

Beoordelingscommissie:
Prof. dr. L.S. Chandran
Dr. C.E. Groenland
Prof. dr. D. Hermelin
Prof. dr. M.J. van Kreveld
Prof. dr. G. Schäfer

Acknowledgement

Five years ago, I embarked on a journey that would ultimately culminate into the creation of this thesis. Like any eager traveler, I set out with starry eyes, filled with anticipation for the excitement and challenges that lay ahead. But as with any long journey, what I did not foresee were the stretches of unremarkable and tedious moments. The true test, I realized, was not just in overcoming challenges, but in maintaining the enthusiasm to keep moving forward. This became far easier thanks to the delightful and supportive companions I was fortunate enough to have by my side. My heart is full of gratitude for everyone who has been a part of my life over these past five years, contributing to my well-being in countless ways.

My deepest thanks go to my advisor, Erik Jan van Leeuwen. He was my first friend in this foreign land that I now call home. During the challenging years of the pandemic, Erik Jan's concern for not only my productivity but also my emotional well-being brought me great comfort. He has always made himself available whenever I needed guidance and support, and for that, I am incredibly grateful. Any creative pursuit is marked by early failures and misguided ideas, and to succeed, one must feel encouraged to explore without fear of judgment. Erik Jan created a space where I could speak my mind and be my authentic self. I cherish all that I have learned from him about science and research, and I will always hold dear the memories of our fun-filled banters.

I am grateful to the members of the reading committee for their diligent efforts in critically evaluating this thesis. Their insightful remarks greatly enhanced its presentation.

The research presented in this thesis was carried out within the Algorithms and Complexity group at Utrecht University. I extend my heartfelt thanks to the group and in particular, Hans Bodlaender, my promotor and the group leader, for fostering a spirit of collaboration within the group. Hans' trust in the abilities of those he leads makes him an exceptional leader. He has always been approachable, generously sharing his wisdom in complex situations through his rich repertoire of stories.

I am especially grateful to my friends and colleagues, Bob Krekelberg, Isja Mannens, Jelle Oostveen, Jeremy Kirn, Krisztina Szilagyi, Lucas Meijer, and

Thekla Hamm. Working from the office was filled with joy and laughter because of them. I cherish the memories of our shared adventures – whether it was tackling Escape-room challenges, enjoying board game nights, exploring new restaurants, or simply being there for each other as friends.

A portion of this research was conducted in collaboration with my coauthors from Durham University. I am grateful to Matthew Johnson, Barnaby Martin, Daniël Paulusma, and Siani Smith for welcoming me to Durham and making our work together such a joyful experience. I also extend my thanks to Daniël Paulusma for inviting me to Dagstuhl for the workshop on Vertex Partioning. It was an invaluable experience to interact with the brilliant researchers working on structural graph theory.

I am eternally grateful to my friends Ranjana and Amit for their friendship. I deeply appreciate how they welcomed me into their home whenever I felt homesick and helped me stay connected to my roots. My heartfelt thanks also go to Abhigya, whose company makes time fly by. I have cherished every moment of our endless conversations and our biking adventures through charming Dutch towns.

I would finally like to express my utmost gratitude to my family. I am priviledged to have their unwavering support through every challenge in life.

Contents

I

Foundation

1

Introduction

1.1. Prelude

On a rainy evening in Utrecht, you decide to host ten of your friends to play board games. You ask each of them for a list of board games that they would like to play. Fortunately, the group collectively owns all the games on the list. However, you realize that each of the games on the list can be played with strictly fewer than ten players. You make peace with splitting the group in two, each group playing a different selection of games. *How do you decide which of your friends should be in the same group?*

After fiddling with the list of game preferences for some time, you realize that it is not so easy to ensure that everyone gets to play at least one game of their choice, while also ensuring that not too many of your friends are cross with you because they do not get to play some game on their list. Luckily, you are aware of the dictum, "When in doubt, draw a graph"! So you draw a graph with each of your friends and yourself as nodes of the graph. Whenever two people like the same game, you connect their nodes by a line. You start to notice that some pairs of your friends' lists of games have more than one game in common. Certainly, friends whose lists have the most in common should be in the same group. So, you decide to write the number of common games next to the line connecting them. All that is left to do is identify the set of lines in the graph that results in the smallest sum of numbers written next to them.

At first, you try to find these lines by trial and error. However, after the third try, you lose patience and start looking for an easier approach. While you are hopelessly staring at the graph, you hear a familiar voice say, "Lose hope

not, you must! Stumbled upon the age-old graph problem, you have: finding the minimum-cut, hm". You look around, startled, only to see a diminutive figure with large pointed ears float away into the distance. You are not one to ignore the advice of a floating figure, so you look up how to find a minimum-cut in a graph and manage to divide your friends into two groups that play together. Each group is happy with their selection of games and while it pours heavily outside, the warmth of friendly banter and laughter fills your home.

Finding the minimum cut is arguably one of the oldest and most famous problems in graph theory. It is a natural dual to the problem of finding the maximum flow in a network. The first ever algorithm to solve the problem was given by Ford and Fulkerson [86] as early as 1956. While their algorithm was *good*, it fueled the quest to find more efficient algorithms [74, 68], with the fastest algorithm, at the time of writing this thesis, being the one due to Chen *et al.* [47]. Several practical problems can be modeled as problems on graphs. Be it finding the shortest path from point A to point B, or finding the shortest tour visiting every major city of a country, or creating a railway network connecting a set of cities in a cost-effective way, it translates naturally to an algorithmic problem on graphs. Decades of research have focused on finding more efficient algorithms for a multitude of problems on graphs.

Generally speaking, the faster an algorithm, the "more efficient" it is. *How do we quantify "fastness"?* The *"fastness"* of an algorithm, known as its *run-time*, is the maximum number of computations that an algorithm performs on any given input. It is measured as a function of the size of its input. Typically, computer scientists define the size of any input to be the number of bits used to describe it in binary encoding. The run-time of an algorithm is expressed using the big-Oh notation, written as $O(\cdot)$. For functions f, g, we say that $f(x) = O(g(x))$, if there exists some natural number x_0 and some real constant c, such that for all $x \geq x_0$, $f(x) \leq c \cdot g(x)$. We can think of big-Oh as the asymptotic order of growth of a function.

How fast is fast enough? The consensus in the community of computer scientists is that algorithms with a polynomial run-time, that is, run-time of the form $O(n^c)$, where n is the size of the input and c is some constant, are "efficient". We call such algorithms *polynomial-time algorithms*. On the other hand, algorithms with an exponential run-time, that is, run-time of the form $O(c^n)$, are considered "inefficient". The terms "efficient" and "inefficient" must be taken with a grain of salt for two reasons. First, in practice, some exponential-time algorithms can perform better than polynomial-time algorithms if the size of the input is small enough. Secondly, the run-time is defined for the *worst-case* input. When we say that an algorithm runs in time $O(2^n)$, we mean that there exists at least one input of size n for which the algorithm must

perform $O(2^n)$ computations.

In general, the goal of an algorithmist is to design polynomial-time algorithms, if possible. All the aforementioned algorithms to find the minimum cut in a network are polynomial-time algorithms. However, there are problems for which generations of algorithmists have failed to design any. For example, an extension of the problem of finding the minimum cut is that of finding the minimum k-cut where k is part of the input. In this problem, instead of splitting the graph into two parts, you are asked to split it into k parts by removing a subset of edges of weight at most s, for some positive integer s. The latter is not known to have any polynomial-time algorithm. However, if you were to give a standard computer a solution to the problem, it could easily verify whether your solution is correct in polynomial time. Already at the start of 1970s, we knew of hundreds of problems that exhibited a similar behavior. The book of Garey and Johnson [93] compiles a couple of hundreds of them in its appendix.

Before we delve into the discussion about categories of problems, we need to define what we mean by a problem. A problem is a question to be answered. Inherently, it consists of free variables and parameters whose values are unspecified. An *instance* of the problem has a specific input, where the values of these variables and parameters are fixed. A *decision problem* is a problem that can be answered as YES or NO. It is of the form:

Input: A, B, C
Question: *Does there exist... ?/ Is it true that... ?*, etc.

An instance of a decision problem that can be answered as YES is called a YES-instance; otherwise, it is called a NO-instance. Another class of problems, called *optmization problems*, are those for which the answer is the minimum/maximum value of some quantity of interest.

Any problem that can be solved by a polynomial-time algorithm belongs in the class P. Decision problems for which there exists an algorithm, which given a proof of a YES-instance can verify it in polynomial-time, constitute the complexity class NP. Problems that are at least as hard any problem in NP are called NP-*hard* and the problems in NP that are NP-hard are called NP-*complete*. It is widely believed that $\mathsf{P} \neq \mathsf{NP}$. However, a proof of this belief has eluded generations of computer scientists.

1.2. Dealing with NP-hardness

What do algorithmists do when they encounter NP-hard problems? The inability to design polynomial-time algorithms for these problems is no reason

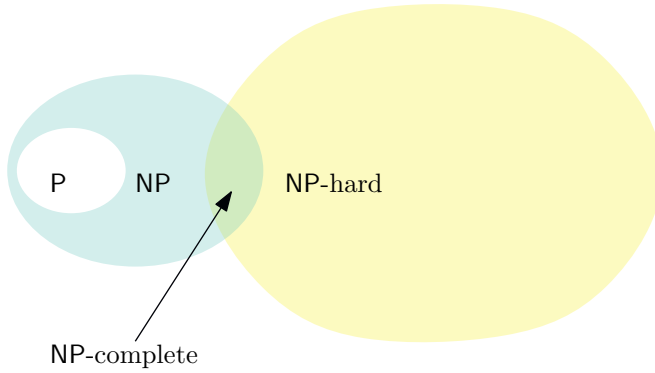

Figure 1.1: The figure shows how different complexity classes relate to each other.

to get disheartened. In this section, we shall see four broad approaches to cope with NP-hardness. First, one can compromise the *optimality* of the solution to design polynomial-time algorithms that find solutions reasonably close to the optimum. Second, for sufficiently small inputs, exponential-time algorithms perform reasonably well. Therefore, it is worthwhile to try to design "efficient" exponential-time algorithms. Third, in the same spirit as the previous approach, one can strive to design algorithms that are exponential-time in some small "*parameter*" of the input while being polynomial-time in the input size. Finally, one can isolate classes of input to the problem where one can hope to solve the problem in polynomial time. In what follows, we discuss each of these approaches in some detail with respect to NP-hard problems on graphs.

Approximation algorithms Finding an optimal solution is not *always* a wise pursuit. Sometimes in practice, the difference between a suboptimal solution and an optimal solution is negligible. Approximation algorithms are those that find an *"approximate"* solution to an NP-hard optimization problem, with a provable guarantee that the "approximate" solution is not too far from the optimum. These algorithms classically run in polynomial-time and strive to get a solution as close to the optimum as possible. In a wide variety problems, the solution found by an approximation algorithm is within a constant multiplicative factor of the optimum. This multiplicative factor is called the *approximation ratio*. Just like the notion of NP-hardness, there exist several notions of *inapproximability*. An example of a problem that is notoriously hard to approximate is the Maximum Independent Set, where given a graph G with n vertices, the goal is to find the size of the largest subset of vertices that does not have an edge between any pair of vertices. Unless P = NP, Maximum Independent Set cannot be approximated within $n^{1-\varepsilon}$ factor of the optimum, for any $\varepsilon > 0$. We refer the reader to the book

of Vazirani [185] or Williamson and Shmoys [189] for more information on approximation algorithms.

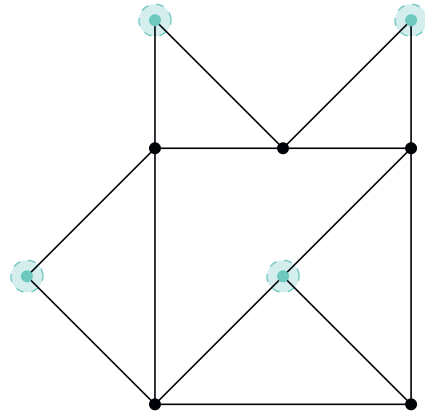

Figure 1.2: The figure shows a graph on 9 vertices. The vertices drawn in turquoise constitute a maximum independent set.

Exact Exponential algorithms Every NP-complete problem can be solved by enumerating all possible solutions. However, this can be excruciatingly slow. While designing exact algorithms for NP-complete problems, we make peace with the run-time being exponential while striving to make the base of the exponent as small as possible. For example, consider again the problem Maximum Independent Set. The *brute-force* approach to solve the problem would be to enumerate all the subsets of vertices in $O(2^n)$ time and for each subset, check if an edge exists in G for any pair of vertices in the subset. The first improvement on the brute-force run-time was by Tarjan and Trojanowski [179] in 1977. Their algorithm ran in time $O(2^{n/3})$ by using clever *branching* strategies. Besides *branching*, several techniques are known to design reasonably fast exponential algorithms. For example, Held and Karp [112] used *dynamic programming across subsets* to solve the well-known problem, Travelling Salesperson in $O(2^n)$ time instead of $O(n!) \sim n^{O(n)}$ time that brute-force would take. Algebraic tools like Inclusion-Exclusion and Measure and Conquer also show great promise while designing exponential algorithms [126, 19, 141, 83, 84, 183].

On one hand we see the development of several strategies to design algorithms that perform better than brute-force, while on the other some problems keep evading all attempts at breaking the $O(2^n)$ barrier. The *Exponential-Time Hypothesis* (ETH), states that for a certain decision problem, known as 3-SAT, there exists no algorithm that can decide it in time $O(2^{o(n)})$ for any $\varepsilon > 0$. Another rather strong conjecture, called the *Strong Exponential*

Time Hypothesis (SETH), states that the decision problem called CNF-SAT cannot be solved by any algorithm running in time $O((2-\delta)^n)$ for any $\delta > 0$. While the particular decision problems are not relevant to this discussion, if you believe any of these conjectures, then you believe that for some decision problems brute-force is inevitable. We refer the reader to the book of Fomin and Kratsch [82] for an in-depth study of exponential algorithms.

Parameterized algorithms The class of NP-hard problems is vast and diverse. While the P versus NP-hard dichotomy may help us distinguish between problems that allow polynomial-time algorithms and those that do not, any two NP-hard problems can vary greatly in their complexity. Consider two close relatives of the Maximum Independent Set, namely, Clique and Vertex Cover. In the problem Clique, given a graph G with n vertices, one is asked if there exists a subset of vertices of size at least k such that there is an edge between every pair of vertices in the subset. Such a subset is called a *clique* of G. In Vertex Cover one is asked to find a subset of vertices of G of size at most k such that every edge of G has an endpoint in the subset. Such a subset is called a *vertex cover*. Both Clique and Vertex Cover are NP-hard [125]. While Vertex Cover can be solved in time $O(2^k \cdot n^2)$ [36], Clique can not be solved by any algorithm that runs in time $n^{(1-\varepsilon)k}$, for any fixed $\varepsilon > 0$ [38], unless ETH fails. This elucidates the need for a more precise analysis of the complexity of NP-hard problems.

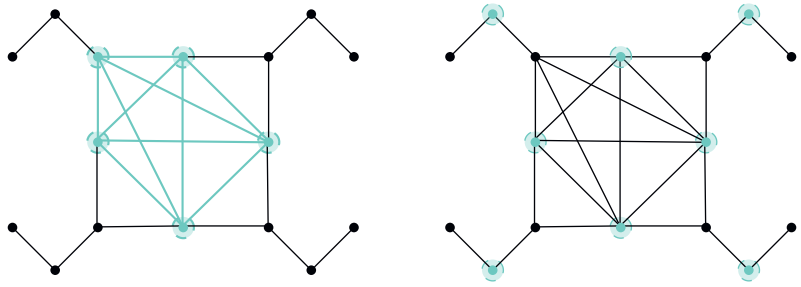

Figure 1.3: The figure shows a graph on 16 vertices. In graph A, we show the maximum clique in turquoise. In the graph B, the vertices in the minimum vertex cover are drawn as turquoise discs.

Downey and Fellows [70, 71] introduced the paradigm of parameterized complexity in 1992. The fundamental goal of parameterized analysis is to identify the properties of the problem that make it intractable. In that endeavor, we study the complexity of any problem not only with respect to the size of its input but also other measures that may be the source of its complexity. A parameterized problem takes the following form:

Input:	A, B, C
Parameter:	k
Question:	*Does there exist...?/ What is the maximum size of...?*, etc.

The parameter k is some relevant secondary measure that captures a property of the input instance. It could be a wide variety of things, for example, the size of the subgraph sought after, some measure of structural complexity of the input, etc.

Parameterized problems that can be solved in time $n^{f(k)}$ belong to the class XP, read as *slicewise polynomial*. Problems that can be solved by an algorithm that runs in time $f(k) \cdot n^c$, for some constant c, belong to the class FPT, and are *fixed-parameter tractable*. We also refer to the algorithms with such running times as FPT algorithms. Clearly, $\mathsf{FPT} \subseteq \mathsf{XP}$. In fact, in their monograph [72] Downey and Fellows presented a proof of the fact that $\mathsf{FPT} \subset \mathsf{XP}$. They also defined a hierarchy of complexity classes, called W-hierarchy. According to this hierarchy:

$$\mathsf{FPT} \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[P] \subseteq \mathsf{XP}$$

It is conjectured that each inclusion in the above hierarchy is strict, i.e., $\mathsf{FPT} \neq W[1] \neq W[2] \neq \ldots W[P] \neq \mathsf{XP}$. We refer the readers to the monograph of Downey and Fellows [72] and the book of Cygan *et al.* [58] for a detailed and comprehensive description of the topic of parameterized algorithms.

Restriction to special graph classes It turns out that several problems that are NP-hard on general graphs, can be solved in polynomial time when the input is restricted to have a particular structure. For example, the problem Maximum Clique, as we know, is NP-hard [125] and even $W[1]$-hard [38] on general graphs. However, it can be solved in polynomial time when the input is restricted to the class of planar graphs. The algorithm makes use of Kuratowski's theorem [137], which implies that any planar graph cannot have a subgraph that is a clique of size five or more. Consequently, if k is the size of the maximum clique in the given planar graph, the problem boils down to deciding if $k = 1, 2, 3$ or 4. The simple approach would be to enumerate all subsets of vertices of size 4 and check if any of them forms a clique. If yes, the size of the maximum clique must be 4 as it cannot be any larger. If not, then enumerate all subsets of size 3 and check if any of them forms a clique. If yes, output 3, else output 2 (if the planar graph has an edge). One can easily extrapolate from this that on any class of graphs that cannot have a subgraph

that is a clique of size at least ℓ, for some constant ℓ, Maximum Clique can be solved in polynomial time. Therefore, it is reasonable to conclude that absence of large cliques in the input makes the problem tractable.

The above discussion points to the fact that the hardness of any problem depends on the structure of the input. Particularly, the exclusion of "problematic" structures from the input can make the problem tractable. This observation has fueled decades of research on the complexity of problems on special classes of graphs.

A graph class $\mathcal{G}$ is *hereditary* if for every graph in the class, all of its induced subgraphs (subgraphs obtained by deletion of vertices) also belong to the class. It is folklore that hereditary classes of graphs can be characterized by a set of minimal graphs denoted by $\text{Forb}(\mathcal{G})$, such that if $F \in \text{Forb}(\mathcal{G})$ then $F \not\subseteq_i G$ for any $G \in \mathcal{G}$. Similarly, a graph class is closed under taking minors if for every graph G in the class, the graph obtained by deleting its vertices or edges, or contracting its edges also belongs to the class. It was shown by Robertson and Seymour that any minor-closed graph class can be characterized by a finite set of forbidden minors [170]. The class of graphs that exclude a graph H as an induced subgraph are called H-free graphs and those that exclude H as a minor are called H-minor-free graphs.

Several notoriously intractable problems can be solved in polynomial-time on H-free and H-minor-free graphs. For example, Maximum Independent Set, can be solved in polynomial-time on P_5-free graphs (graphs excluding a path on 5 vertices as an induced subgraph) [142] and $K_{1,3}$-free (claw-free) graphs [154]. For more information on graph classes, we refer the reader to the book of Golumbic [104] and the more recent one by Brandstädt *et al.* [33].

1.3. Outline of the Thesis

In this thesis, we continue the research on well-known NP-hard problems by looking at them from two lenses, namely, parameterized complexity and restriction of input to classes of graphs forbidding certain subgraphs. The thesis comprises three parts.

Part I. Foundations

In this part we introduce the fundamental notions that we implicitly use throughout the thesis. While **Chapter 1** serves as a gentle exposure to relevant concepts in algorithmics and complexity, **Chapter 2** gives a formal definition of the terminologies and notations used across further chapters.

Part II. Multiway Cut

This part focuses on the problem Minimum Multiway Cut. This problem is a natural generalization of the problem of finding the minimum (s,t)-cut.

In **Chapter 3**, we present a parameterized subexponential-time algorithm to solve Minimum Multiway Cut when the input is restricted to plane graphs with terminal face-cover number at most k. While it was known that Minimum Multiway Cut can be solved in subexponential time parameterized by the number of terminals, not having an upper bound on the number of terminals poses severe challenges. The algorithm is founded on an in-depth analysis of the structure of the graph induced by the edges of minimum multiway cut in the dual of the input plane graph. Assuming ETH, the running time of this algorithm is tight. This chapter contains an extended version of joint work with Erik Jan van Leeuwen [161].

In **Chapter 4**, we prove that Minimum Multiway Cut remains NP-hard even on unweighted planar subcubic graphs. While the gadgets created are inspired by the original proof of Dahlhaus *et al.* [60], the real achievement is reducing the maximum degree of the hard instance from 11 in [60] to 3. The proof relies on several astute observations about the interactions of various novel gadgets. This chapter is based on joint work with Johnson *et al.* [122].

Part III. Complexity Framework for Forbidden Subgraphs

In this part, we develop an algorithmic meta-classification for problems that satisfy three properties. The properties are:

C1. The problem is polynomial-time solvable on graphs of bounded treewidth.

C2. The problem is NP-hard on subcubic graphs.

C3. The problem is NP-hard on edge subdivisions of subcubic graphs.

We show that a huge compendium of problems satisfy these three properties.

In **Chapter 5**, we prove the statement of our meta-classification. We then compare our framework with the existing frameworks for minor-freeness and topological-minor-freeness. We also discuss the limitations of our framework. This chapter contains results from joint work with Johnson *et al.* [120].

Finally, in **Chapter 6**, we consider problems that *almost* fit our framework, i.e. they satisfy condition C1 but not C2. We prove that Independent Feedback Vertex Set is one such problem that is tractable on subcubic graphs. Some problems known to be tractable on subcubic graphs are Feedback Vertex Set, Matching Cut, and k-Coloring. For all these problems we obtain complexity dichotomies on classes of graphs forbidding a finite set of

subgraphs. The results presented in this chapter were published in joint work with Johnson*et al.* [121].

1.4. List of Publications

The results presented in this thesis are based on the following publications.

- Sukanya Pandey and Erik Jan van Leeuwen. **Planar Multiway Cut with Terminals on Few Faces**. In Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, pages 2032-2063. SIAM, 2022.
- Matthew Johnson, Barnaby Martin, Sukanya Pandey, Daniël Paulusma, Siani Smith, and Erik Jan van Leeuwen. **Edge Multiway Cut and Node Multiway Cut are Hard for Planar Subcubic Graphs**. In Proceedings of 19th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2024, volume 294 of LIPIcs, pages 29:1-29:17, Schloss Dagstuhl–Leibniz Zentrum für Informatik, 2024.
- Matthew Johnson, Barnaby Martin, Jelle Oostveen, Sukanya Pandey, Daniël Paulusma, Siani Smith, and Erik Jan van Leeuwen. **Complexity Framework for Forbidden Subgraphs I. The Framework**. arXiv:2211.12887v5 [math.CO] (Under review.)
- Matthew Johnson, Barnaby Martin, Sukanya Pandey, Daniël Paulusma, Siani Smith, and Erik Jan van Leeuwen. **Complexity Framework for Forbidden Subgraphs III: When Problems Are Tractable on Subcubic Graphs**. In Proceedings of 48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, volume 272 of LIPIcs, pages 57:1-57:15, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023.

2

Definitions

In this chapter, we formally define the mathematical notations and graph-theoretic terminologies that we shall use throughout this thesis.

2.1. Mathematical Notations

The set of natural numbers $\mathbb{N}$ is the set $\{0, 1, 2, \ldots\}$. We denote by $\mathbb{R}^+$ and $\mathbb{Q}^+$ the sets of positive real and rational numbers, respectively. For a natural number n, we denote by $[n]$ the set of numbers $\{0, 1, 2, \ldots n\}$. For natural numbers x and y, $[x, y]$ is the set $\{i : x \leq i \leq y\}$ and (x, y) denotes the set $\{i : x < i < y\}$.

We use *weight functions* in the input for several problems in this thesis. A weight function is of the form $\omega : S \longrightarrow U$, where S is part of the input and $U = \mathbb{N}$. For a set $X \subseteq S$, its weight is defined as $w(X) = \sum_{x \in X} w(x)$.

Asymptotic functions The running time of an algorithm is a function, $T(n)$, that measures the number of computations performed by the algorithm on an input of size n in the worst-case. The domain of this function is the set of natural numbers $\mathbb{N}$. Running times are described by asymptotic functions.

For a function $g(n)$, we denote by $O(g(n))$ the class of functions:

$$O(g(n)) = \Big\{ f(n) : \exists\ c, n_0 \in \mathbb{R}^+ \text{ such that } f(n) \leq c \cdot g(n)\ \forall n \geq n_0 \Big\}.$$

If a function $f(n) \in O(g(n))$, we write $f(n) = O(g(n))$ to imply that $g(n)$ is an asymptotic upper bound for $f(n)$. The function $O(\cdot)$, however, does not

denote a tight bound. The following notion of asymptotic upper bound is stronger.

$$o(g(n)) = \Big\{ f(n) : \quad \forall c > 0, c \in \mathbb{R}^+, \ \exists n_0 \in \mathbb{R}^+, n_0 > 0, \\ \text{such that } f(n) \leq c \cdot g(n), \ \forall n \geq n_0 \Big\}.$$

Equivalently, $o(\cdot)$ is defined as:

$$o(g(n)) = \Big\{ f(n) : \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \Big\}.$$

For more information on asymptotic functions, we refer the reader to the book of Cormen*et al.* [55] which contains a comprehensive coverage of the topic.

2.2. Parameterized Complexity

We follow the notations and definitions in the books of Cygan*et al.* [58] and Downey and Fellows [72].

Basics A *parameterized problem* is a language $\mathcal{L} \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed, finite language. For any instance $(x, k) \in \mathcal{L}$, k is called the parameter.

A parameterized problem $\mathcal{L}$ is called *fixed-parameter tractable* if there exists an algorithm $\mathcal{A}$ (called *fixed parameter algorithm*), a function $f : \mathbb{N} \to \mathbb{N}$, and a positive constant c, such that given $(x, k) \in \Sigma^* \times \mathbb{N}$, $\mathcal{A}$ correctly decides if $(x, k) \in \mathcal{L}$ in time bounded by $f(k) \cdot |(x, k)|^c$. The complexity class containing all fixed-parameter tractable algorithms is called FPT. Any problem that is FPT is known to have a *kernelization algorithm*. A *kernelization algorithm* or a *kernel* is defined as an algorithm $\mathcal{A}$, which given an instance (I, k) of a parameterized problem $\mathcal{L}$, run in polynomial-time and returns an equivalent instance (I', k') such that there exists a computable function $g(\cdot)$ for which $|I'| + k' \leq g(k)$. If for some problem, $g(\cdot)$ is a polynomial, then the problem is said to have a *polynomial-kernel*.

A problem is *slicewise polynomial* if there exists an algorithm $\mathcal{A}$, functions $f, g : \mathbb{N} \to \mathbb{N}$ such that for any $(x, k) \in \Sigma^* \times \mathbb{N}$, $\mathcal{A}$ correctly decides if $(x, k) \in \mathcal{L}$ in time bounded by $f(k) \cdot |(x, k)|^{g(k)}$. The complexity class containing all slicewise polynomial problems is called XP.

We sometimes use the notation $O^*(\cdot)$ to describe the running time of FPT or XP algorithms, if the polynomial factor is not particularly interesting. We say that a function $f(n, k) = O^*(g(k))$ if $f(n, k)$ is upper bounded by $g(k) \cdot n^{O(1)}$.

Exponential Time Hypothesis A propositional formula ϕ, on Boolean variables $x_1, x_2, \ldots x_n$, is said to be in *Conjunctive-Normal Form*(CNF) if it is a conjunction of disjunctions, i.e., $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, where each $C_i = x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k}$. The problem CNF-SAT is defined as: given a CNF formula ϕ on n Boolean variables containing m clauses, does there exist an assignment of true/false values to each variable such that ϕ evaluates to true? If we restrict the number of variables in each clause to be 3, the problem is referred to as 3-SAT.

The *Exponential Time Hypothesis* states that there cannot exist any algorithm solving 3-SAT in time $2^{o(n)}$.

Let q-SAT be the problem derived from CNF-SAT by restricting the number of variables in each clause to be at most q. Then the *Strong Exponential Time Hypothesis* states that as q tends to infinity, there cannot exist any algorithm that solves q-SAT in time $O^*((2-\varepsilon)^n)$, for any $\varepsilon > 0$.

2.3. Graphs

Basic terminology A *graph* is a pair $G(V, E)$, such that $E \subseteq \binom{V}{2}$. Unless otherwise specified, a graph is *undirected*. The elements of V are called the *vertices* of the graph and the elements of E its *edges*. For vertices $u, v \in V$, an edge between u and v is denoted by uv. We also denote the cardinality of the vertex and edge sets as $|V| = n$ and $|E| = m$. We also use the notations $V(G)$ or $E(G)$ to denote the vertex and edge sets of the graph G respectively.

A graph is *edge-weighted* if there exists a function $\omega : E \longrightarrow \mathbb{Q}^+$ such that $\omega(e) \in \mathbb{Q}^+$, for all $e \in E$. A graph is *node-weighted* if such a function has the domain V.

A *subgraph* $G'(V', E')$ of G, written as $G' \subseteq G$, is a graph such that $V' \subseteq V$ and $E' \subseteq E$. If $G' \subseteq G$ and $G' \neq G$, then G' is a *proper subgraph* of G. G' is an *induced subgraph* of G, written as $G' \subseteq_i G$, if for all $x, y \in V'$, if $xy \in E$ then $xy \in E'$.

A *walk* in a graph is a sequence of vertices $v_1, v_2, \ldots v_\ell$ such that for all $1 \leq i < \ell$, $v_i v_{i+1} \in E$. If $v_0 = v_\ell$, the walk is *closed*. A *path* is a non-empty graph P of the form $V(P) = \{x_0, x_1, \ldots, x_k\}$ and $E(P) = \{x_0x_1, \ldots, x_{k-1}x_k\}$, where all x_i are distinct. The graph formed by adding the edge x_0x_k to the path P is called a *cycle*. The number of edges in a path (cycle) is its *length*. We use the notation $P[x_i, x_j]$ to denote the subpath of P between vertices x_i and x_j.

A graph is *connected* if there is a path between any two vertices in a graph, and *disconnected* otherwise.

A connected undirected graph without cycles is called a *tree*. An undirected

graph without cycles is called a *forest*. For a tree N, we use $N[x, y]$ to denote the unique path in N between the vertices x and y.

The *girth* of a graph G that is not a forest is the length of a shortest cycle in G.

The *contraction* of an edge $e = (u, v)$ of G is the operation of identifying u and v, while removing any loops or parallel edges that arise. We use the notation G/e. This extends to sets $F \subseteq E(G)$ of edges, for which we can use the notation G/F. If v had degree 2 and its two neighbors in G are non-adjacent, then the operation is called the *(vertex) dissolution* of v, and v is said to have been *dissolved*.

A graph H is a *minor* of graph G if H can be obtained from G after a series of vertex and edge deletions and edge contractions. H is a *topological minor* of G if it can be obtained from G after a series of vertex and edge deletions and vertex dissolutions.

A *matching* in a graph $G(V, E)$ is a subset of E such that no two edges in the subset share any endpoint.

Connectivity A *cut vertex* of a connected graph is a vertex whose removal yields a disconnected graph. A *biconnected graph* is a graph without cut vertices. A *biconnected component* or *block* of a graph is a maximal subgraph that is biconnected.

A *bridge* of a connected graph is an edge whose removal yields a disconnected graph. A *bridgeless graph* is a graph that has no bridges. A *bridgeless component* or *bridge block* of a graph is a maximal subgraph that is bridgeless. A bridge block that consists of more than one edge is called a *nontrivial* bridge block; the other bridge blocks consist of just a single edge, which is a bridge in the graph. Bridge blocks are incident on each other at cut vertices of the graph.

Given two disjoint vertex subsets $X, Y \subseteq V$, an (X, Y)*-cut* is a set of edges whose removal leaves no path between any vertex of X and any vertex of Y. If such a cut also induces a matching in the graph, it is called a *matching cut*.

Given a subset $T \subseteq V$, a *(edge) multiway cut* of T is a set of edges whose removal leaves no path between any pair of distinct vertices in T.

Given a subset $T \subseteq V$ (called terminals), a *Steiner tree* on T is a minimally connected subgraph H of G such that there is a path in H between any two terminals in T.

Graph Classes A graph G is H*-subgraph-free* if H is not a subgraph of G, H*-free* if H is not an induced subgraph of G, H*-topological-minor-free* if H is not a topological-minor of G, and H*-minor-free* if H is not a minor of G.

For a set $\mathcal{H}$ of graphs, a graph G is $\mathcal{H}$*-subgraph-free* if G is H-subgraph-free for every $H \in \mathcal{H}$. If $\mathcal{H} = \{H_1, \ldots, H_p\}$ for some integer $p \geq 1$, we also say that G is $(H_1, \ldots, H_p)$*-subgraph-free*. We also define the analogous notions of being

$\mathcal{H}$-free, $\mathcal{H}$-topological-minor-free and $\mathcal{H}$-minor-free.

A class of graphs is *hereditary* if it is closed under deletion of vertices and *monotone* if it is closed under the deletion of edges.

Decompositions and Width Measures A *tree decomposition* of a graph $G(V, E)$ is a pair $(T, \mathcal{X})$ where T is a tree and $\mathcal{X}$ is a collection of subsets of V called *bags* such that the following holds:

- A vertex $i \in T$ is a *node* and corresponds to exactly one bag $X_i \in \mathcal{X}$.
- The tree T has the following properties.
  1. For each edge $vw \in E$, there is at least one node of T that contains both v and w.
  2. For any vertex $u \in V$, the nodes of T corresponding to the bags of $\mathcal{X}$ that contain u induce a connected subgraph of T.

The *width* of $(T, \mathcal{X})$ is one less than the size of the largest bag in $\mathcal{X}$. The *treewidth* of G is the minimum width of its tree decompositions. If we require T to be a path, then we obtain the notions *path decomposition* and *pathwidth*.

A graph parameter p *dominates* a parameter q if there is a function f such that $p(G) \leq f(q(G))$ for every graph G. If p dominates q, but q does not dominate p, then p is *more powerful* than q. If p dominates q and vice versa, then we say that p and q are *equivalent*. Note that every graph of pathwidth at most c has treewidth at most c. However, the class of trees has treewidth 1, but unbounded pathwidth (see [67]). Hence, treewidth is more powerful than pathwidth.

For further information on graphs, we refer the reader to the books of Diestel [67] or Harary [110].

II

Multiway Cut

Overview

A graph with weighted edges, and a subset of its vertices called *terminals*, is given. How would you pairwise disconnect the terminals by removing a subset of the edges of the graph of minimum possible weight? Widely known as the Minimum (Edge) Multiway Cut problem, this question is a natural generalization of the popular minimum (s,t)-cut problem. A variant of the problem was first introduced by T.C. Hu in 1969 [115]. Formally, we can define the Minimum Edge Multiway Cut problem as follows:

Minimum Edge Multiway Cut	
Instance:	A graph $G(V,E)$, weight function $\omega : E(G) \to \mathbb{Q}^+$, $T \subseteq V(G)$
Question:	What is the minimum possible weight of an edge multiway cut of (G,T)?

The subset of edges (vertices) which when deleted from the input graph pairwise disconnects all the vertices of T is called an *edge (node) multiway cut* (also known as multiterminal cut). When all the edge weights are equal (equivalently, the edges are not weighted), the goal is to find an edge multiway cut of minimum cardinality.

In the decision variant of this problem, the goal is to decide if for a given integer s there is an edge multiway cut of size at most s. We formally define the decision problem as:

Edge Multiway Cut	
Instance:	A graph $G(V,E)$, a set of terminals $T \subseteq V$, and an integer s.
Question:	Does (G,T) have an edge multiway cut $S \subseteq E$ of size at most s?

We shall refer to the aforementioned problems as Multiway Cut when it is clear from the context which variant of the problem we are referring to.

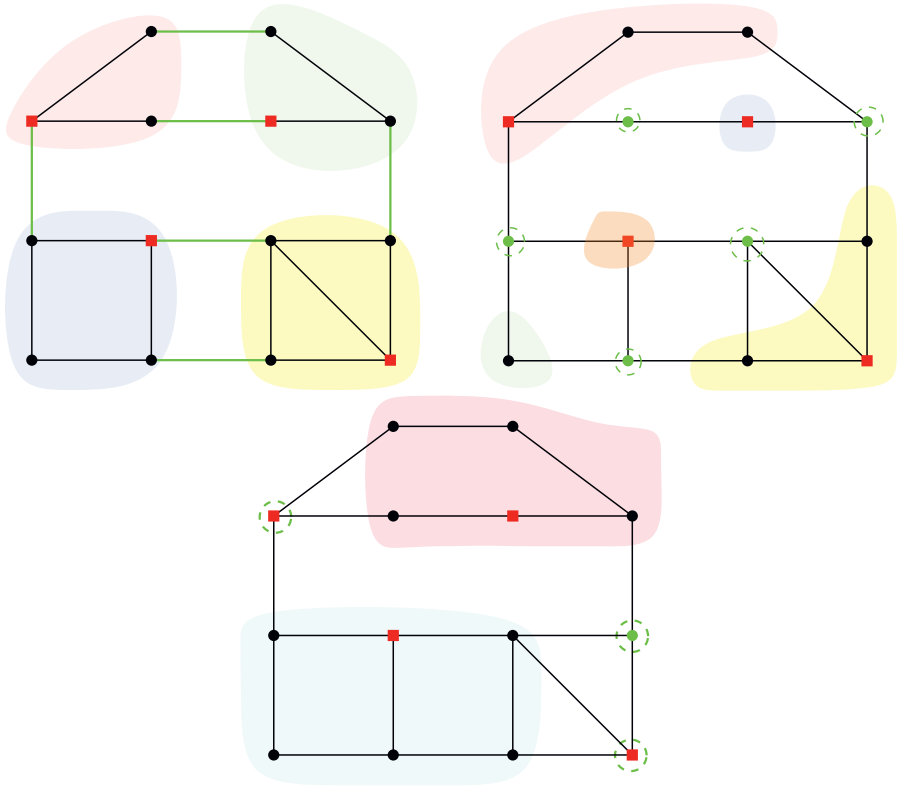

Figure 2.1: The three different types of multiway cuts that we consider in this part. In all figures, the red square nodes form the terminal set T. In the top left figure, the green lines form an edge multiway cut. In the top right, the green encircled vertices form a node multiway cut not containing a vertex of T. In the bottom figure, the green encircled vertices form a node multiway cut that contains two vertices of T. The coloured parts depict the components formed after removing the edges/vertices of the multiway cut.

The study of the complexity of Edge Multiway Cut was pioneered by Dahlhaus *et al.* in their seminal paper [60]. The authors showed that Edge Multiway Cut is NP-hard on arbitrary graphs for any fixed $t \geq 3$, where t is the number of terminals. They also showed that Minimum Multiway Cut is Max-SNP-hard for all fixed $t \geq 3$ even if the edges are unweighted, while giving an $O(tnm \log(n^2/m))$-time approximation algorithm, which given an arbitrary graph and an arbitrary t finds a solution within a $2 - 2/t$ ratio of the optimum. Following these hardness results, began a twofold quest: to find approximation algorithms with a better approximation ratio [40, 124] and to find exact algorithms that were more efficient than any brute-force

approach [148, 41, 191, 129, 57, 46, 108, 49].

Until 2006, not much was known with regard to the parameterized complexity of Multiway Cut. Marx [148] got the ball rolling with his result that Node Multiway Cut, where one must delete vertices instead of edges, is FPT parameterized by the size of the solution s. It was only two years after Garg *et al.* [98] pioneered the study of Node Multiway Cut showing that Node Multiway Cut is NP-hard. Formally, Node Multiway Cut can be defined as:

Node Multiway Cut	
Instance:	A graph $G(V, E)$, a set of terminals $T \subseteq V$ and an integer s.
Question:	Does (G, T) have a node multiway cut $S \subseteq V \setminus T$ of size at most s?

Marx's algorithm ran in time $O^*(4^{s^3})$. Since then, the run time has been considerably improved [191, 41, 46, 108, 57], with the current best being $O^*(1.84^s)$ for Edge Multiway Cut [41], and $O^*(2^s)$ for Node Multiway Cut [57]. Starting with the seminal work of Marx [148], significant research effort was also put in understanding the parameterized complexity of Edge Multiway Cut for the parameter s [191, 41, 46, 108, 57]. It is a notoriously hard problem to determine if Multiway Cut has a polynomial kernel. The best known kernel on general graphs has size $2^{O(s)}$, which follows from the FPTalgorithm of Marx [148]. Kratsch and Wahlström [133] presented a kernel that has $O(s^{t+1})$ vertices for Node Multiway Cut if $|T| \leq t$. Their result implies a polynomial kernel also for Edge Multiway Cut. However, when $|T|$ is unbounded, the question whether a polynomial kernel exists for Multiway Cut is far from resolved. Recently, Wahlström [187] made progress towards answering this question by demonstrating a kernel of size $k^{O(\log^3 k)}$. On directed graphs, Chitnis *et al.* [49] showed that Node Multiway Cut and Edge Multiway Cut are equivalent and admit an FPT algorithm running in time $O^*(2^{2^{O(s)}})$.

Restriction to planar graphs Given the hardness of the problem on arbitrary graphs, even for small constant values of t, it was but natural to look for specific graph classes where the problem might be more tractable with respect to the number of terminals t. In fact, Dahlhaus *et al.* [60] already considered the restriction to planar graphs. When t is part of the input, Edge Multiway Cut still is NP-hard, on edge-weighted planar subcubic graphs. When the edges are unweighted, Edge Multiway Cut is NP-hard on planar graphs of maximum degree 11. In contrast, they also showed that

Edge Multiway Cut could be solved in polynomial time for any constant t. For $t = 3$, their algorithm runs in time $O(n^3 \log\ n)$ and for $t \geq 4$ in time $O((4t)^t \cdot n^{2t-1} \log\ n)$. In the parameterized complexity parlance, their result implied that Edge Multiway Cut belongs to the class XP. The obvious next step was to investigate if it was also FPT. In 2012, however, Marx [149] showed that assuming ETH holds, the problem does not admit any algorithm running in time $f(t) \cdot n^{o(\sqrt{t})}$ and showed that it is $W[1]$-hard.

In a companion paper, Klein and Marx [129] gave a subexponential algorithm for Planar Multiway Cut with a run-time of $2^{O(t)} \cdot n^{O(\sqrt{t})}$. Later, Colin de Verdière [63] showed that this result extends to surfaces of any fixed genus. He showed this even holds for the more general Edge Multicut problem, where given a set of terminal pairs, we ask for the smallest set of edges, which when removed, disconnects all the given terminal pairs. This yielded an algorithm which solves the problem in time $(g+t)^{O(g+t)} n^{O(\sqrt{g^2+gt})}$, where g is the genus of the surface. This bound is almost tight assuming ETH holds, as was recently shown by Cohen-Added *et al.* [54].

Besides the above, some other algorithms for Planar Multiway Cut are noteworthy. In particular, Hartvigsen [111] gave an $O(t4^t n^{2t-4} \log n)$ time algorithm, which improved on the original exact algorithm by Dahlhaus *et al.* [60]. The simple algorithm by Yeh [194], unfortunately, seems incorrect [48]. The mentioned algorithm of Klein and Marx [129] is faster than all of them. Benz also considered the generalization to Edge Multicut, first with terminals only on the outer face [14] and for the general case [13]. Unfortunately, the latter general result seems to have several flaws [63]. The algorithm by Colin de Verdière [63] for this problem that we already mentioned is also faster and more general. Finally, for planar graphs and parameter s, Pilipczuk *et al.* [164] showed that even a polynomial kernel in s exists for Edge Multiway Cut, leading to an algorithm with running time $O^*(2^{O(\sqrt{s} \log\ s)})$. This kernel was recently extended to Node Multiway Cut by Jansen *et al.* [117].

Restriction to other hereditary graph classes There have been surprisingly few studies investigating the complexity of Multiway Cut on other hereditary graphs classes. Until recently, even the classical complexity of the problem was unknown on well-known hereditary graphs classes like chordal or interval graphs. Misra *et al.* [155] showed that on chordal graphs, Node Multiway Cut admits a polynomial kernel parameterized by the solution size. Later, Bergougnoux *et al.* [15] showed that Node Multiway Cut is FPT parameterized by the rankwidth, and XP parameterized by the mimwidth of the input graph. Their result implies a polynomial time algorithm for the problem on interval graphs, permutation graphs, bi-interval graphs, circular arc and circular permutation graphs, convex graphs, and k-polygon and dilworth-k graphs

for fixed k. Recently, Galby *et al.* improved both the aforementioned results on chordal graphs by showing that Node Multiway Cut is polynomial-time solvable on the class of chordal graphs [90].

Approximation algorithms We already mentioned several constant-factor approximations for general graphs [60, 40, 124] and planar graphs [44]. Bateni *et al.* [12] presented a PTAS for Edge Multiway Cut on planar graphs, which combined the technique of brick-decomposition from [32], the clustering technique from [11], and a technique to find short cycles enclosing prescribed amounts of weight from [163]. The more general Minimum Edge Multicut problem is known to be APX-hard, even on trees [97]. However, Cohen-Addad *et al.* [53] recently gave a $(1+\varepsilon)$-approximation scheme for Minimum Edge Multicut on graphs embedded on surfaces, with a fixed number of terminals. Their approximation scheme runs in time $(g+t)^{O((g+t)^3)} \cdot (1/\varepsilon)^{O(g+t)} \cdot n \log n$.

Outline In this part, we shall present two results on Multiway Cut. In Chapter 3, we investigate the complexity of Edge Multiway Cut on planar graphs, parameterized by the *terminal face cover number*. We present an algorithm that runs in subexponential time in this parameter, and is tight assuming ETH. In Chapter 4, we show that both Edge Multiway Cut and Node Multiway Cut are NP-hard on planar subcubic graphs. Prior to our work, hardness was known only for planar graphs of maximum degree 11. Given that both the versions of Multiway Cut are polynomial time solvable on graphs of maximum degree 2, we obtain a complexity dichotomy in terms of the maximum degree of the input graph.

3

Planar Multiway Cut With Terminals On A Few Faces

We consider the Edge Multiway Cut problem on planar graphs. When the number of terminals is t, it is known that this can be solved in $n^{O(\sqrt{t})}$ time [Klein, Marx, ICALP 2012] and not in $n^{o(\sqrt{t})}$ time under the Exponential Time Hypothesis [Marx, ICALP 2012]. A generalization of this parameter is the number k of faces of the planar graph that jointly cover all terminals. For the related Steiner Tree problem, an $n^{O(\sqrt{k})}$ time algorithm was recently shown [Kisfaludi-Bak *et al.*, SODA 2019]. By a completely different approach, in this chapter, we prove that Edge Multiway Cut can be solved in $2^{O(k^2 \log k)} \cdot n^{O(\sqrt{k})}$ time as well.

3.1. Introduction

Our focus, in this chapter, is the problem Planar Multiway Cut. Formally, we define the problem as:

Planar Multiway Cut	
Instance:	Plane graph $G(V, E)$, $T \subseteq V$, $\omega : E \longrightarrow \mathbb{Q}^+$
Question:	What is the minimum possible weight of the cut that pairwise separates the vertices in T?

In the overview of Part-I, we discussed in detail the enormous body of research surrounding Edge Multiway Cut and its variants. In particular, we discussed previous work that looked into the parameterized complexity of the problem with respect to parameters like the number of terminals and the size of the edge (node) multiway cut. On planar graphs, Dahlhaus *et al.* showed the problem is polynomial time solvable if the number of terminals is fixed, but NP-hard when the number of terminals is part of the input [60]. Their result implies that Planar Multiway Cut is in the class XP. Klein and Marx improved the complexity to $2^{O(t)} \cdot n^{O(\sqrt{t})}$ [129], following which Marx showed that assuming ETH, this is the best one can do [149].

Parameterization by terminal face cover number Due to the intractability of Planar Multiway Cut when the number of terminals is part of the input, it is worthwhile to look for other parameters that might generalize t. In particular, we could look at imposing restrictions on the location of terminals in the input plane-embedded graph. An extensively explored such restriction is when all the terminals are present on a small number of faces of the planar graph. This restriction on the input graph was studied by Ford and Fulkerson in their classic paper [86]. The minimum number of faces of the input planar graph that cover all the terminals was termed the *terminal face cover number* $\gamma(G)$ by Krauthgamer *et al.* [135]. The terminal face cover number is of broad interest as a parameter and has been studied with respect to several cut and flow problems [135, 134, 44, 151], shortest path problems [89, 45], finding non-crossing walks [76], and the minimum Steiner tree problem [118]. In particular, Planar Steiner Tree has an algorithm running in time $2^{O(\gamma(G)\log\gamma(G))}n^{O(\sqrt{\gamma(G)})}$.

The case when $\gamma(G) = 1$ is known as an Okamura-Seymour graph. It was shown by Chen and Wu [44] that when $\gamma(G) = 1$ and G is biconnected, the minimum multiway cut in G forms a minimum Steiner tree in its planar dual[1].

[1]They consider an augmented planar dual, which differs from the standard dual graph

Consequently, one can find a minimum Steiner tree in the dual graph using the algorithm by Erickson *et al.* [79] (see also Bern [16]) for the case when all the terminals lie on the outer face boundary of the graph. They also gave an approximation algorithm for the case when $\gamma(G) > 1$, which runs in time $\max\{O(n^2 \log n \log \gamma(G)), O(\gamma(G)^2 n^{1.5} \log^2 n + tn)\}$ and finds a solution within a ratio of $2 - 2/t$ of the optimum multiway cut. However, for the case when $\gamma(G) > 1$, no exact algorithm was known.

Main theorem We resolve the complexity of Planar Multiway Cut parameterized by the terminal face cover number, hereafter referred to as k. Given an edge-weighted planar graph $G(V, E)$ with a fixed embedding, a set of terminals $T \subseteq V$, and a collection of faces $\mathcal{F}$ that cover all the terminals in T, our goal is to find a minimum weight cut of G that pairwise separates the terminals in T from each other. Note that $|\mathcal{F}| = k$. We show the following:

Theorem 3.1.1. *Planar Multiway Cut can be solved in time* $2^{O(k^2 \log k)} n^{O(\sqrt{k})}$. *Unless ETH fails, there can be no algorithm that solves Planar Multiway Cut and runs in time* $n^{o(\sqrt{k})}$.

Our main contribution is the algorithm. Since $k \leq t$, the lower bound immediately follows from Marx's result [149].

We note that our algorithm requires that the set $\mathcal{F}$ is known. Fortunately, such a set of size k can be computed in time $2^{O(k)} n^{O(1)}$ through the algorithm of Bienstock and Monma [17]. Hence, we can run their algorithm before our own, and this does not affect the running time of the final algorithm.

We also note that while the running time of our algorithm is reminiscent of the algorithm by Kisfaludi-Bak *et al.* [118] for Planar Steiner Tree parameterized by the terminal face cover number, our algorithm is and needs to be substantially more involved. The intuition of their algorithm is to show that a minimum Steiner tree for a set of terminals that can be covered by k faces has bounded treewidth. Then they can 'trace' this tree by a recursive algorithm that guesses the vertices of the solution in a separator implied by the tree decomposition. While one can show that the dual of a minimum multiway cut is a subgraph of the planar dual of bounded treewidth, tracing a solution through such a tree decomposition is not straightforward, and we need significant assistance from tools from topology, particularly homotopy, which were not needed for Planar Steiner Tree.

Our work leans on a proper specification of the homotopy of the dual of the solution. This approach was first applied explicitly to Edge Multicut (and by extension to Edge Multiway Cut) by Colin de Verdière [63] and is also

in that the outer face is represented by several vertices, one per interval of the boundary vertices between two consecutive terminals (see also Section 3.4.2.)

evident in the work of Klein and Marx [129]. In this sense, our algorithm has more in common with the approaches of Klein and Marx [129] and Colin de Verdière [63], which extensively rely on topological arguments, and particularly homotopy. The understanding of homotopy and its uses in cut and flow problems on graphs on surfaces was built through a sequence of works, see e.g. [42, 43, 78, 77, 76]. While our algorithm is similar in spirit to those of Colin de Verdière [63] and Klein and Marx [129], significant effort is needed to generalize from the parameter number of terminals employed in those works to the parameter number of faces covering terminals. This not only affects the structural results that employ homotopy, but also makes the final algorithms more involved.

Intuitively, we describe the high-level topology that must be respected by the dual of some optimum solution. Part of this topology is a planar graph, of which we can find a sphere-cut decomposition of width $O(\sqrt{k})$. We then apply a dynamic programming routine on this decomposition to find the optimum solution. While this dynamic programming routine is sufficient to find the high-level structure, the number of terminals is too large for this to effectively deal with the local structure of the solution. We then merge the popular algorithm by Dreyfus-Wagner [73] for finding minimum Steiner trees, which runs in polynomial time in cases relevant for us [79, 16], with the algorithm of Frank-Schrijver [88] to find shortest paths homotopic to a given prescription. We argue that this prescription can be efficiently guessed. This leads to a neat dynamic program used to find the local structure, and then finally, an optimum solution.

We provide a more detailed overview of our algorithm in Section 3.2.

3.2. Bird's-eye View of the Algorithm

Since our approach is reminiscent of the one by Klein and Marx [129] as well as Colin de Verdière [63] used for the weaker parameter t, the number of terminals, we start by giving a short overview of their work.

3.2.1. Previous Approaches

The starting observation is that cuts in planar graphs correspond to cycles in the dual of the graph [167]. Hence, it makes sense that almost the entire algorithms and structural descriptions of Klein and Marx and Colin de Verdière, as well as ours, work with the planar dual. If the number t of terminals is bounded, this quickly leads to the intuition that the planar dual of a multiway cut should be a planar graph with t faces, one for each terminal [60]. By dissolving vertices of degree 2 of this planar graph and applying Euler's formula, one obtains

a planar graph S with $O(t)$ faces, vertices, and edges. One can then guess what S looks like by exhaustive enumeration and guess the vertices of the dual corresponding to the vertices of S. Then it remains to expand the edges of the graph back into shortest paths.

However, these paths are not just any shortest paths. Instead, they must contort themselves between the terminals in a particular way, such that they perform their job in separating the terminals. This is where topological arguments come in. The layout of these paths is described using the sequence of crossings with a Steiner tree on the terminals. The main thrust of the work of Klein and Marx and Colin de Verdière is to argue that in some optimal solution these crossing sequences are short, in the sense that their length depends only on t. One can then guess the crossing sequences of each of the paths in the optimum and find a shortest path following a particular crossing sequence in polynomial time using an algorithm by Frank and Schrijver's algorithm [88, Section 5].

The above leads to an $f(t) \cdot n^{O(t)}$ time algorithm. To improve to an $f(t)n^{O(\sqrt{t})}$ time algorithm, it suffices to observe that since S is planar, it has treewidth $O(\sqrt{t})$, as follows from the planar separator theorem [140]. One can then replace the guessing of the vertices of the dual corresponding to vertices of S by a dynamic program on the tree decomposition.

3.2.2. Warm-up: An $n^{O(k)}$-time Algorithm

We design our approach along the same lines, in that we show that the topology of the dual graph of the minimum multiway cut is constrained and then enumerate all the possible topologies. For a certain topology, we find the shortest solution respecting the topology. However, since we must deal with a huge number of terminals, possibly $O(n)$ many of them, this is not straightforward. Indeed, we need stronger structural properties of the solution to limit the number of diverse topologies.

Structure: Global and Local From a global perspective, however, the structure of an optimal solution looks very similar. If we think of a terminal face as a single terminal, then as in the previous works, we see that (some part of) the dual of the solution must separate the different terminal faces. We call this part the *skeleton* of the dual of the solution[2]. The paths between its

[2]Our notion of skeleton should not be confused with the notion of skeleton as defined by Cohen-Addad *et al.* [53]. In particular, our notion of a (shrunken) skeleton is very reminiscent of the multicut dual of Colin de Verdière [63], provides a high-level view of the solution, and has no weight restrictions. The skeleton of [53] instead can be seen as 'orthogonal' to the solution; by controlling its weight, the skeleton can be portalized (as in Arora [10]) and the solution can be approximated by combining Steiner trees of a particular homotopy between portals. This is different from our use and definition of a skeleton.

branching points are called *bones*. By dissolving dual vertices of degree 2 in the skeleton, we obtain the *shrunken skeleton*; its edges are called *shrunken bones*. The shrunken skeleton has k faces and $O(k)$ vertices and edges by Euler's formula. It follows that the shrunken skeleton of the optimum can be guessed by exhaustive enumeration and has treewidth $O(\sqrt{k})$, a great starting point. This is discussed in more detail in Section 3.5.

Now consider the local parts of the solution, namely the part of the dual of the solution inside each of the k faces of the skeleton. Chen and Wu [44] proved that when there is a single terminal face that is a simple cycle, then the dual of a minimum multiway cut is a minimum Steiner tree in the dual graph. To be more precise, this holds in the augmented dual graph. This graph splits the dual vertex of s of the face corresponding to the simple cycle into r parts, where r is the number of terminals of the face, such that all dual edges of each maximal subpath of the cycle without terminals belong to the same part of s. The solution is then a minimum Steiner tree on the augmented dual vertices. We generalize this argument to prove that inside each face of the skeleton, the solution is a forest of minimum Steiner trees on the augmented terminals, defined with respect to the terminal face inside the face of the skeleton. We call these *nerves*. All nerves attach to the boundary of the face of the skeleton. Crucially, the augmented terminals belonging to each of these nerves form an *interval* of the set of augmented terminals. Then, by applying the Dreyfus-Wagner algorithm [73], any nerve can be found in polynomial time [79, 16], if we know the interval. See Section 3.5.

Algorithm: Global and Local As a warm-up, we now discuss how to find an algorithm with running time $f(k)n^{O(k)}$. First, guess the shrunken skeleton of the optimum solution by exhaustive enumeration. Then, for each shrunken bone of this shrunken skeleton, we note that it needs to expand to separate two terminal faces, say $F_\alpha, F_\beta \in \mathcal{F}$, and some terminals on each of them. For both the terminal faces, we guess the intervals of augmented terminals I_α and I_β covered by nerves that attach to the corresponding bone. Since intervals are between two augmented terminals, there are $n^{O(k)}$ intervals, and we can guess the optimal intervals by exhaustive enumeration in the same time.

We now apply a dynamic program to find the bone. The crux here is to find the paths between the attachment points of different nerves. While each of those paths individually again has a short crossing sequence, as can be argued by adapting the approach of Klein and Marx [129] and Colin de Verdière [63], there are too many of these paths to efficiently guess all these crossing sequences. Instead, we argue that we can group these crossing sequences while keeping them small, and that we do not need to guess the crossing sequences for all groups.

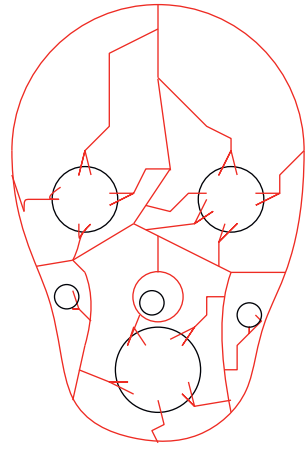

Figure 3.1: This figure illustrates the broken bones of the minimum multiway cut dual. In the figure, the terminal faces are drawn as black circles. In red, we depict the partial solution, which we term the broken bones. We fix these broken bones by splints to get C^+.

We start by discussing the grouping. We group consecutive nerves towards the same terminal face. We then prove that the union of the paths between the nerves in such a group has a bounded crossing sequence (see Section 3.6). Then we can guess the optimal crossing sequence by exhaustive enumeration. Assuming we know the starting and ending nerve of the group, we can then employ a dynamic program to find all nerves in between, while ensuring that the path between the attachment points of the nerves follows the guessed crossing sequence. This will take care of all (augmented) terminals between the intervals.

Next, we argue that we only need few groups. We say that two groups of nerves *alternate* if they are towards different terminal faces. If there are two alternating groups of nerves on the left part of the bone, and two alternating groups of nerves on the right part of the bone, then we can observe that these nerves effectively cordon off part of the plane. Inside this region, we can show that all terminals effectively lie on a cycle. This enables us to use the ideas of Chen and Wu [44] again to find an optimal solution.

To conclude our algorithm, we note that we need at most four groups. If there are indeed four, then there is a part between them for which we have a polynomial time algorithm. For the groups themselves, we know the paths between their attachment points has a crossing sequence of small size, and we can guess the optimal crossing sequences by exhaustive enumeration. For each group, we use a dynamic program that runs in polynomial time. By applying Frank and Schrijver's algorithm [88, Section 5], we can find shortest paths that

follow a particular crossing sequence in polynomial time. See Section 3.7.3 for details.

The total running time of this algorithm is indeed $f(k) \cdot n^{O(k)}$. The latter term originates from the guessing of the terminals corresponding to the branching points of the skeleton, the guessing of the intervals, and the guessing of the nerves at the bookends of the groups. For each shrunken bone, we call this information a *broken bone* (we effectively guess its parts) and the solution that fixes it a *splint*.

3.2.3. Towards the Final Algorithm

We now develop the ideas for the final algorithm. To avoid guessing the broken bones globally, we aim to apply the fact that the shrunken skeleton has bounded treewidth. Using a tree decomposition directly is cumbersome and not very intuitive. Instead, we use sphere-cut decomposition.

A sphere-cut decomposition is a branch decomposition, which can be thought of as a set of separators of the graph organized in a tree-like fashion. The tree structure enables dynamic programming in the usual manner. The crux is that the vertices of each separator induce a noose in the planar graph. A *noose* in this sense is a (closed) curve in the plane that intersects every face of the planar graph at most once and intersects the drawing only in the vertices of the separator. The tree-structure of the decomposition is organized in such a way that each separator has two child-separators and the symmetric difference of the corresponding two nooses is the noose of the parent-separator. A formal definition is in Section 3.3.1.

We now apply a dynamic program where we just maintain the broken bones for the shrunken bones of the faces intersected by a noose of the decomposition. In fact, this still is too much information, and instead we maintain only a relevant part of those broken bones, namely the first nerve that we encounter in either direction on the shrunken bones of each face that is intersected by the noose. We argue that this yields sufficient information to know all broken bones and obtain an optimum solution. See Section 3.8.2 for details.

It is known that there is a sphere-cut decomposition of a planar multigraph on k vertices where all nooses (and separators) have $O(\sqrt{k})$ vertices [69, 150, 165]. This will lead to the $n^{O(\sqrt{k})}$ running time. This decomposition requires that the planar graph is connected, has no bridges, nor has self-loops. Unfortunately, our shrunken skeleton does not satisfy any of those demands out of the box. The issue of self-loops is quickly handled by not dissolving all vertices of degree 2 when obtaining a shrunken skeleton from a skeleton, but only those that do not lead to a self-loop.

Next, we ensure connectivity of the shrunken skeleton. To this end, we con-

sider a connected component of the shrunken skeleton that is 'innermost' in the embedding of the shrunken skeleton. We can guess which of the terminal faces are enclosed by this connected component. However, for one of those terminal faces, some terminal might not be enclosed by the connected component. We cannot guess this terminal (as there are n choices), even though it is necessary to know the terminal to correctly guess the structure of the optimum solution. To circumvent this issue, we argue that we can pick a single terminal of this face as a representative of this 'exposed' terminal. Thus we completely avoid knowing which terminal is exposed. This enables a $2^{O(k)}n^{O(1)}$ time subroutine to guess the components of the dual of an optimum solution and to reduce to the case where such duals are connected. We then argue that this implies that the shrunken skeleton is connected as well. See Section 3.4.2 for details.

Finally, we consider bridges of the shrunken skeleton. It seems hard to avoid them completely. Instead, we design another dynamic program (see Section 3.8.1). We use a bridge block tree of the skeleton to guide this dynamic program. Recall that a bridge block tree is a tree representation of the bridge blocks (bridgeless components) of a graph and the cut vertices between those bridge blocks, generalizing the more familiar block cut tree. To ensure this bridge block tree is suitable for the dynamic program, we need a modified version of a bridge block tree that organizes itself according to the embedding of the shrunken skeleton. To this end, we develop an embedding-aware bridge block tree (see Section 3.3.2), which may be of independent interest.

In conclusion, our final algorithm is as follows. First, we perform a subroutine to ensure that the dual of any minimum multiway cut is connected. Then we guess the structure of the dual of such a solution, namely its shrunken skeleton, how the nerves of each bone are grouped, and what the crossing sequences are of each path between nerves of the same group and between the groups. We call this a *topology*.[3] We guess the optimal topology by exhaustive enumeration. Consider its shrunken skeleton and define a dynamic program on its bridge blocks to combine partial solutions for each bridge block. For each bridge block, we show that it has a sphere-cut decomposition with nooses of bounded size, which enables us to find a partial solution using a dynamic program. The guessing of the topology and the dynamic programs combined lead to an algorithm running in time $2^{O(k^2 \log k)}n^{O(\sqrt{k})}$.

[3]The word topology has many well-known meanings, including a branch of mathematics. We use the term here in a cartographic sense, as an abstract map of the solution. This is in line with previous uses in the literature, e.g. [63].

3.3. Preliminaries

Topology and Planar Graphs A Jordan *arc* in the plane is an injective continuous map of $[0,1]$ to $\mathbb{R}^2$. A Jordan *curve* in the plane is an injective continuous map of $\mathbb{S}^1$ to $\mathbb{R}^2$. If one of the points on this curve is special, we may also call this a closed arc on this point. Consider a set Z of Jordan (possibly closed) arcs in the plane. Let $P(Z)$ denote the union of the set of points of each arc of Z. Observe that $\mathbb{R}^2 \setminus P(Z)$ is an open set. A *region* of Z is a maximal subset X of $\mathbb{R}^2 \setminus P(Z)$ that is *arc-connected*; that is, there is a Jordan arc in X (meaning all its points belong to X) between any pair of points in X. If $\mathbb{R}^2 \setminus P(Z)$ has more than one region, then Z is *separating*. Let X be a region of a separating set Z. The *boundary* ∂X of a region X is the set of all points $p \in \mathbb{R}^2$ such that every open disk around p contains both a point of X and of $\mathbb{R}^2 \setminus X$. The *complement* of a region X is the union of $Y \cup \partial Y$ for each region of $\mathbb{R}^2 \setminus (X \cup \partial X)$. Note that the complement of a region is not necessarily a region itself, but possibly a union of regions (and their boundaries), particularly if X has holes.

We say that a region X *encloses* a set Y if $Y \subseteq X \cup \partial X$ and *strictly encloses* Y if $Y \subseteq X$.

For the definition of planar graphs, we follow Diestel [67]. A graph $G(V,E)$ is *plane* if V corresponds to a set of points (vertex points) in the plane and E corresponds to a set of arcs in the plane (edge arcs) between the points corresponding to its endpoints, such that the interior of each edge arc contains no vertex point and no point of any other edge arcs. We call the vertex points and edge arcs an *embedding* of G. If G admits an embedding, we call the graph a *planar graph*. A *face* of a plane graph is any region of the set of edge arcs. Exactly one face is *unbounded*, also called the *outer face*, whereas all other faces are *bounded*.

Two plane graphs are *equivalent* if they are isomorphic and the circular order of the edges around each vertex is the same in both embeddings. In particular, this means that boundaries of the faces of the embeddings have the same edge sets and there is a bijection between the faces of both embeddings.

Let $G(V,E,F)$ and $G^*(V^*,E^*,F^*)$ be two plane graphs, where V,E and F (V^*,E^* and F^*) denote the set of vertices, edges, and faces of G (G^*). G^* is called a *plane dual* of G, if there exist bijections

$$\begin{array}{ccc} f^* : V \to F^* & e^* : E \to E^* & v^* : F \to V^* \\ v \to f^*(v) & e \to e^*(e) & f \to v^*(f) \end{array}$$

satisfying the following conditions:

(a.) $v^*(f) \in f, \forall f \in F$

(b.) e and e^* intersect in exactly one point, which lies in the interior of both

e and e^*.

(c.) $v \in f^*(v), \forall v \in V$

We note the following basic properties, which follow from Diestel [67]. We will often use them without explicitly referring to this proposition.

Proposition 3.3.1. *If G is connected, then the edges of G bounding each face form a closed walk (also known as a face walk). If G is bridgeless, then for any edge, the faces on both sides are distinct.*

Let O be a set of points in the plane, called *obstacles*. Consider two Jordan arcs a, b between the same pair of points, such that neither arc contains a point of O. Then these arcs are *homotopic* if and only if there is a continuous deformation between a and b that does not cross a point of O. In the plane, this means that the region induced by the union of a and b does not contain any point of O.

3.3.1. Sphere-cut Decompositions

A main component of our algorithm is a dynamic program over a planar graph of bounded treewidth. However, using a normal tree decomposition is rather cumbersome in this case, and it turns out to be much easier to instead use a sphere-cut decomposition, a branch decomposition especially suited for planar graphs. We define all necessary notions and state the relevant theorem below.

A *branch decomposition* of a graph $G(V, E)$ is a pair (R, η) of a ternary tree R and a bijection η between the leaves of R and the edges in E. For an edge e of R, we define the *middle set* $\mathbf{mid}(e)$ to be the set of vertices in V for which an incident edge is mapped by η to a leaf in the one component of $R - e$ and an incident edge is mapped by η to a leaf in the other component. The *width* of the branch decomposition is defined as the maximum size of the middle set of any edge of R. The *branchwidth* of G is then the minimum width of any branch decomposition of G.

Let G be a (planar) graph with a fixed embedding on the sphere. Then a *noose* $\vec{\gamma}$ (with respect to G) is a closed, directed curve in the sphere that meets the embedding of G only in its vertices and that traverses each face at most once. The *length* of the noose is equal to the number of vertices of G traversed by it. If we enumerate the vertices of the noose, we implicitly assume that this enumeration follows the order of appearance on the noose, that is, following its direction. Note that a noose cuts the sphere into two regions, each homeomorphic to an open disk. The region bounded by and to the right when following the noose with its direction is denoted by $\mathbf{enc}(\vec{\gamma})$ and the other by $\mathbf{exc}(\vec{\gamma})$.

A sphere-cut decomposition of a graph G with a fixed embedding on the sphere is a triple (R, η, δ) consisting of a branch decomposition (R, η) and a mapping δ from an ordered pair of adjacent vertices x, y of R to nooses (with respect to G) on the sphere, such that

- $\delta(x, y)$ is the same noose as $\delta(y, x)$ but with the direction reversed. Note that then it holds $\mathbf{enc}(\delta(x, y)) = \mathbf{exc}(\delta(y, x))$;
- $\delta(x, y)$ meets the embedding of G exactly in the vertices of the middle set $\mathbf{mid}(x, y)$; moreover, $\mathbf{enc}(x, y)$ contains all the embeddings of all edges of the one component of $R - xy$ and $\mathbf{exc}(x, y)$ contains the embeddings of all other edges.

As noted by Dorn *et al.* [69] and Pilipczuk *et al.* [165], we may assume that a sphere-cut decomposition is *faithful*. That is, for every internal vertex x of R with adjacent vertices y_1, y_2, y_3, we may assume that $\mathbf{enc}(x, y_1)$ is equal to the disjoint union of $\mathbf{enc}(y_2, x)$, $\mathbf{enc}(y_3, x)$, and $(\delta(y_2, x) \cap \delta(y_3, x)) \setminus \delta(x, y_1)$. We also note that $\delta(y_2, x) \cap \delta(y_3, x) \cap \delta(x, y_1)$ consists of two points, each of which may (or may not) coincide with a vertex of G.

As described by Dorn *et al.* [69], we can root any sphere-cut decomposition (R, η, δ) by subdividing an arbitrary edge e of R. Let u be the newly created vertex and e', e'' be the newly created edges. Set $\mathbf{mid}(e') = \mathbf{mid}(e'') = \mathbf{mid}(e)$. Add a new vertex r, connect it to u, and set $\mathbf{mid}(ru) = \emptyset$. We then direct the tree R towards the root r. In the remainder, we assume our sphere-cut decompositions are rooted.

The following result was observed by Dorn *et al.* [69] and follows from [107, 174] (see also Marx and Pilipczuk [150] and Pilipczuk *et al.* [165]).

Theorem 3.3.2. *Every n-vertex connected, bridgeless multigraph without self-loops but with a fixed embedding on the sphere has a faithful sphere-cut decomposition of width $\sqrt{4.5n}$. Moreover, such a sphere-cut decomposition can be found in $O(n^3)$ time.*

3.3.2. Embedding-Aware Bridge Block Trees

An important aspect of our algorithm will be to deal with bridge blocks of a planar graph, as a sphere-cut decomposition can not. We define the following notion, which lends itself in a nice way to the dynamic programming algorithm we develop towards the end of the paper.

We can define a *bridge block tree* of a graph H as follows. Consider the graph F that has a node for every bridge block (i.e. bridgeless component or a bridge), called the BB-nodes of F, and for every endpoint of a bridge, called the C-nodes of F. There is an edge between a BB-node corresponding to a

bridge block B and a C-node that corresponds to a cut vertex v if $v \in V(B)$. It is immediate that this graph is a tree if H is connected and a forest otherwise. A bridge block that has only one edge is called a *trivial* bridge block, otherwise it is *non-trivial*. For simplicity, we call two bridge blocks *neighboring* if they share a cut vertex. We can observe that any nontrivial bridge block neighbors only trivial bridge blocks.

Theorem 3.3.3 (Tarjan [178])**.** *A bridge block tree of a graph can be computed in linear time.*

If H is plane and connected, then we use an extension of this definition. An *embedding-aware bridge block tree* (or *eabb* tree) $L = L(H)$ of H is formed from the bridge block tree F as follows. Root F at a BB-node $\ell = \ell(F)$ that corresponds to a bridge block that has an edge bordering the outer face of H. We now perform two operations on the BB-nodes.

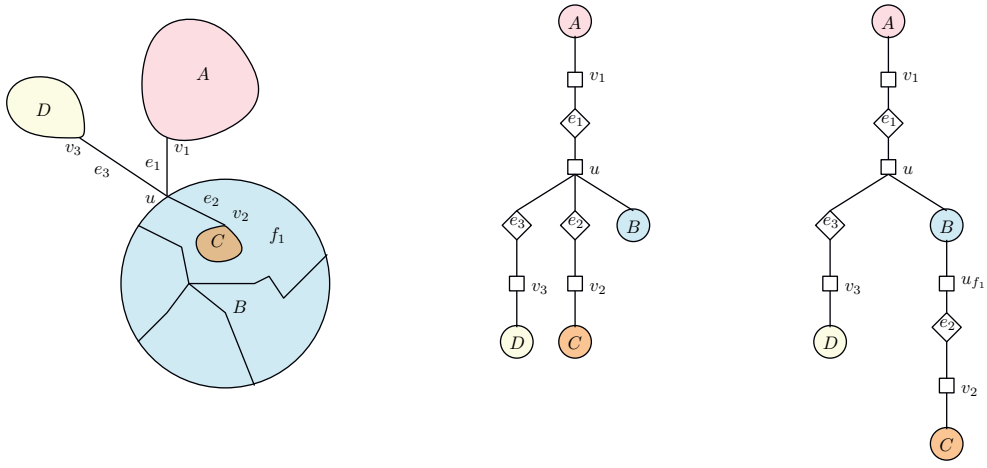

Figure 3.2: The leftmost image is that of a graph with bridge blocks A, B, C and D. The bridges are denoted by e_1, e_2 and e_3, whereas the cut vertices are u, v_1, v_2, and v_3. The figure in the middle shows a bridge-block tree corresponding to the graph. The rightmost figure shows the embedding aware bridge block tree.

First, for any nontrivial bridge block B whose corresponding BB-node b has a C-node parent p in F corresponding to a cut vertex v, note that all other BB-node children c' of p correspond to trivial bridge blocks. For each bounded face f of B, create a new C-node child c_1^f of b corresponding to v and f, and for any child $c' \neq b$ of p corresponding to a bridge edge that is contained in f, make the subtree of F rooted at c' a child of c_1^f. We only add c_1^f if there are any such children c'.

Second, for any nontrivial bridge block B whose corresponding BB-node p has a C-node child c in F corresponding to a cut vertex v, note that all BB-node children c' of c correspond to trivial bridge blocks. For each bounded face f of B, create a new C-node child $c_2^{v,f}$ of p corresponding to v and f, and for any child c' of c corresponding to a bridge edge that is contained in f, make the subtree of F rooted at c' a child of $c_2^{v,f}$. We only add $c_2^{v,f}$ if there are any such children c'.

Perform these two operations on all nontrivial bridge blocks. Observe that the operations essentially apply to the children of C-nodes corresponding to cut vertices contained in a nontrivial bridge block. As nontrivial bridge blocks are not neighboring, the sets of cut vertices contained in nontrivial bridge blocks are pairwise disjoint. Hence, these operations do not interfere with each other and can be performed independently.

Then, finally, order the children of a C-node p in the tree according to the order in which their edges appear around the corresponding cut vertex. The resulting tree is $L(H)$.

Lemma 3.3.4. *An embedding-aware bridge block tree of a plane, connected graph H can be computed in polynomial time.*

Proof. First, we invoke Theorem 3.3.3 to obtain a bridge block tree of H. Using a Doubly-Connected Edge List (DCEL) of the embedding, we can then find the information necessary to make it embedding-aware in polynomial time. □

We now make several observations about $L(H)$. Let B and B' be distinct bridge blocks of a plane graph H. Since H is plane (and ignoring possible intersections of the embedding on the cut vertex), B is enclosed by a bounded face of B', or vice versa, or B and B' are both in each other's outer face. We now define a strict partial order $\prec_P$ on the bridge blocks of H, where $B \prec_P B'$ if B is embedded in a bounded face of B'.

Lemma 3.3.5. *Let B and B' be two bridge blocks of a connected plane graph H. If $B \prec_P B'$, then the node b corresponding to B is a descendant of the node b' corresponding to B' in $L(H)$.*

Proof. Observe that B' needs to be a nontrivial bridge block. Let F be the bridge block tree of H, rooted at $\ell(F)$. Let p denote the C-node parent of b' in F, or $p = \ell(F)$ if $b' = \ell(F)$. We claim that b is a descendant of p in F. Indeed, suppose that b is not a descendant of p. Consider any path in G from a vertex in B to a vertex bordering the outer face. Since $B \prec_P B'$, any such path cannot avoid a vertex of B'. Hence, b is a descendant of p in F. Moreover, any such path must enter B' at the same cut vertex v, which can correspond to $p \neq \ell(F)$ or a C-node that is a child of b'. We only consider

the case when this cut vertex corresponds to the C-node $p \neq \ell(F)$; the other case is similar. Let P be a path from a vertex in B through the cut vertex v corresponding to $p \neq \ell(F)$ to a vertex bordering the outer face. Then the edge of P preceding v must be a bridge in H, and thus a trivial bridge block B''. Let b'' be the corresponding node of F. The first operation ensures that b'' becomes a descendant of b', and thus b becomes a descendant of b' in $L(H)$, as claimed. □

Lemma 3.3.6. *Let H be a connected plane graph. Let c and c' be two C-nodes in $L(H)$ corresponding to the same cut vertex v of H. Then on the path between c and c' in $L(H)$, all other C-nodes correspond to v.*

Proof. This is immediate from the construction of $L(H)$. Indeed, the C-node corresponding to a cut vertex v is only replicated in c_1 or c_2^v for a particular nontrivial bridge block B. As nontrivial bridge blocks are not neighboring, the sets of cut vertices contained in nontrivial bridge blocks are pairwise disjoint. Hence, there can be at most one such nontrivial bridge block B that is responsible for replicating the C-node corresponding to v. From the construction, the property set forth in the lemma holds. □

3.4. Basic Properties and Connectivity of the Planar Multiway Cut Dual

Let $G(V, E)$ be a connected simple undirected planar graph on n vertices and m edges with a fixed embedding. Let $T \subseteq V$ be the set of terminals. The set of faces covering all the terminals in T is denoted by $\mathcal{F} = \{F_\alpha : 1 \leq \alpha \leq k\}$. We call these the *terminal faces* of G. The edges of E are weighted. By removing edges of weight 0 or less and then scaling the weights of the remaining edges, we can obtain an equivalent instance with weights specified by the function $\omega : E \to [1, \ldots, W]$ for some integer W. Note that during this transformation, possibly, the set of terminal faces changes, but there will still be at most k of them. Moreover, the graph might become disconnected, but we can solve the instance associated with each connected component independently. Hence, by abuse of notation, we may assume that our instance is still defined by G, T, k, $\mathcal{F}$, and ω as defined previously.

In the remainder, we use edge weight ∞ to indicate undeletable edges. Instead of ∞, one could use $mW + 1$, but using ∞ simplifies later notation. Note that the initial instance has no undeletable edges, and thus has a finite-weight solution. In future transformations and reductions, we shall always maintain the property that a finite-weight solution exists. By abuse of notation, we still use $\omega : E \to [1, \ldots, W] \cup \{\infty\}$ to denote the weights.

Arbitrarily assign each terminal $t \in T$ to a face of $\mathcal{F}$ that has t on its boundary. For each face $F_\alpha \in \mathcal{F}$, let $T_\alpha \subseteq T$ be the set of terminals on the boundary of F_α that are assigned to F_α. Let $p_\alpha = |T_\alpha|$. We may assume that $p_\alpha > 0$ for each terminal face, or we could reduce the set of terminal faces. Observe that the sets T_α form a partition of T. Note that $\mathcal{F}$ can be partitioned into $\mathcal{F}_1 = \{F_\alpha \mid p_\alpha = 1\}$ (called the *singular faces*) and $\mathcal{F}_2 = \mathcal{F} \setminus \mathcal{F}_1$ (called the *plural faces*); possibly, one of these sets is empty.

For a terminal face F_α, we order the terminals in T_α as follows. Note that G is connected and thus the boundary of F_α forms a closed walk. Pick an arbitrary starting vertex on this closed walk. Now traverse the walk in clockwise direction and add a terminal to the ordering at the first moment it is encountered. Index the terminals in T_α as ${}^\alpha t_1, \ldots, {}^\alpha t_{p_\alpha}$ according to this ordering.

We prove the following transformation is possible, which is reminiscent of Chen and Wu [44, Lemma 8].

Lemma 3.4.1. *In polynomial time, one can transform the instance into an equivalent instance where G is bridgeless, the faces of $\mathcal{F}$ are vertex disjoint, and all vertices of each face in $\mathcal{F}$ are terminals.*

Proof. First, we make G bridgeless. For each edge $e = (u, v)$ of G that is a bridge, remove e and add new vertices w_1, w_2 and edges $(u, w_1), (w_1, v)$ and $(u, w_2), (w_2, v)$. Set the weight of each of those edges to be equal to $\omega(e)/2$. To show equivalence, we note that in any solution for the original instance that contains e, e can be replaced by say (u, w_1) and (u, w_2) to obtain a solution of the new instance of the same weight. Conversely, any minimal solution to the new instance that contains either (u, w_1) or (w_1, v) (but never both) if and only if it contains either (u, w_2) or (w_2, v) (but never both). If it contains either of these edges, they can be replaced by e to obtain a solution of the original instance of the same weight.

By scaling and abuse of notation, we still denote the instance by (G, T, ω) where $\omega : E \to [1, \ldots, W] \cup \{\infty\}$.

For each terminal face $F_\alpha \in \mathcal{F}$, add an edge of weight 1 from ${}^\alpha t_i$ to ${}^\alpha t_{i+1}$ (indices modulo p_α) for each $1 \leq i \leq p_\alpha$. If an edge e from ${}^\alpha t_i$ to ${}^\alpha t_{i+1}$ already existed, first subdivide e to obtain edges e_1 and e_2, and set the weight of e_1 and e_2 to $\omega(e)$; then add the new edge. Call the resulting graph G' and the resulting weight function ω'. Because G is connected and thus the boundary of F_α forms a closed walk, the new edges can be embedded inside the corresponding terminal faces and thus G' is planar. The embedding of G can be extended to G' in a natural way. In particular, a new face F'_α is created for each face F_α whose boundary consists of the vertices of T_α and the newly

created edges. Note that G' and the set $\{F'_\alpha \mid F_\alpha \in \mathcal{F}\}$ has all the properties set forth in the lemma statement.

To show equivalence, we observe that (G, T, ω) has a solution of weight K if and only if (G', T, ω') has a solution of weight $K + \sum_{F_\alpha : p_\alpha > 1} p_\alpha$. Indeed, it is necessary in (G', T, ω') to remove all newly added edges for faces with $p_\alpha > 1$. The remaining graph is G; note that the subdivisions that were potentially performed do not affect anything. □

We call an instance *transformed* if it has the properties as set forth in Lemma 3.4.1. We note that the transformation might make G no longer simple, but have parallel edges or self-loops. In the remainder, we assume that the instance is transformed.

3.4.1. Dual, Cuts, and Connectivity Properties

Let G^* be the dual of G. By definition, G^* has an embedding in the plane such that each vertex of G^* is embedded in the corresponding face and each dual edge crosses the corresponding primal edge exactly once and no other edges. For practical purposes, any time we consider a set C^* of dual edges, we also denote by C^* the subgraph of the dual induced by the edges in C^*. Then C^* is again a planar graph with an embedding where each edge of C^* is embedded as it is in the embedding of G^*. We denote by C the set of edges in G corresponding to the dual edges in C^*.

The following was observed by Dahlhaus *et al.* [60], based on the original observation of Reif [167].

Proposition 3.4.2. *Let C be a (minimum) multiway cut of (G, T, ω) and let C^* be the set of corresponding dual edges. Then each face of C^* encloses at most (exactly) one terminal.*

We will often use this fact without explicitly referring to it. We also require the following structural properties of C^*.

Lemma 3.4.3. *Let C be any inclusion-wise minimal multiway cut of (G, T, ω) and let C^* be the set of corresponding dual edges. C^* is bridgeless.*

Proof. Let e^* be a bridge in C^*. Then there is a face of C^* for which e^* appears on the boundary twice. Removing e^* from C^* does not change the set of vertices of G enclosed by the face. Hence, $C - \{e\}$ is a feasible solution for (G, T, ω). This contradicts that C is inclusion-wise minimal. □

Lemma 3.4.4. *Let C be any inclusion-wise minimal multiway cut of (G, T, ω) and let C^* be the set of corresponding dual edges. Then for any dual vertex v_α corresponding to a plural terminal face $F_\alpha \in \mathcal{F}$, all dual edges incident on v_α are in C^*.*

Proof. Since the instance is transformed, any edge of F_α is between a pair of distinct terminals, and thus must be in C. □

Lemma 3.4.5. *Let C be any multiway cut of (G, T, ω) and C^* the set of corresponding dual edges. Then no terminal face of $\mathcal{F}$ corresponds to a cut-vertex of C^*.*

Proof. Suppose that v_α is a cut vertex of C^* that corresponds to the terminal face $F_\alpha \in \mathcal{F}$. Let $\mathcal{B}^*$ be the set of maximal biconnected components of C^* that intersect exactly in v_α. Since v_α is a cut vertex, $|\mathcal{B}^*| \geq 2$. For any maximal biconnected component $B^* \in \mathcal{B}^*$, there is a simple cycle X_{B^*} of G^* that determines the outer face of B^*, because B^* is biconnected. We call this the *bounding cycle* of B^*. Note that all of B^* is enclosed by X_{B^*}. The planarity of G^* ensures that no two bounding cycles of biconnected components in $\mathcal{B}^*$ can cross, and in fact, they intersect exactly in v_α.

Choose a biconnected component $B^* \in \mathcal{B}^*$ for which its bounding cycle encloses the smallest region in the plane among all biconnected components in $\mathcal{B}^*$. Since the bounding cycles of the biconnected components of $\mathcal{B}^* \setminus \{B^*\}$ do not cross the bounding cycle of B^* nor can they be enclosed by it (by definition of B^*), there is a biconnected component $\tilde{B}^*$ of $\mathcal{B}^* \setminus \{B^*\}$ in the outer face of B^*.

Consider the two edges of X_{B^*} that are incident on v_α. Let e and e' be the edges of G dual to these edges. Now, let t and t' be the endpoints of e and e' that are not enclosed by X_{B^*}. Observe that t and t' are distinct terminals of T_α by the existence of $\tilde{B}^*$; indeed, $\tilde{B}^*$ is in the outer face of B^* and encloses at least one terminal of T_α as the instance is transformed.

We claim that t and t' are in the same face of C^*, contradicting that C is a multiway cut. Since no two bounding cycles of blocks of $\mathcal{B}^*$ can cross each other, if there were a face of C^* enclosing t but not t', its boundary would contain at least one edge of X_{B^*}, namely the one dual to e. This however, contradicts that X_{B^*} is a bounding cycle.

□

3.4.2. Reduction to Connected Duals

We argue that by spending $2^{O(k)}$ time, we can reduce to the case where we may assume that the graph induced by the set of edges dual to the edges of any minimum multiway cut is connected. Our approach extends the work of Klein and Marx [129, Lemma 3.1] to plural faces while simplifying the requirements for later algorithms; we discuss the differences in more detail at the end.

Intuitively, we want to focus on connected components of C^* that are 'innermost' in the embedding: none of its bounded faces encloses another

connected component of C^*. We aim to solve a simplified instance for such a connected component separately (by an algorithm we describe in subsequent sections) and then solve the remaining instance recursively. To make the formalities work, we need a more expansive definition of 'innermost component'.

Internal Sets

We formalize the intuition of an 'innermost component' of a plane graph with the following notion.

Definition 3.4.6. *Let H be any plane graph. Let J be the union of a subset of the biconnected components of H such that there is a single face f of $H - J$ that encloses J and there is a single face f' of J that encloses $H - J$. We call J an* internal set*, f its* enclosing region*, and f' its* exclosing region*. The intersection of the enclosing region and exclosing region is called the* middle region*.*

Refer to Figure 3.3 for an illustration of the above definitions applied to $H = C^*$, where C^* is the set of dual edges corresponding to any multiway cut of (G, T, ω), and $J = D^*$, a set of biconnected components of C^*.

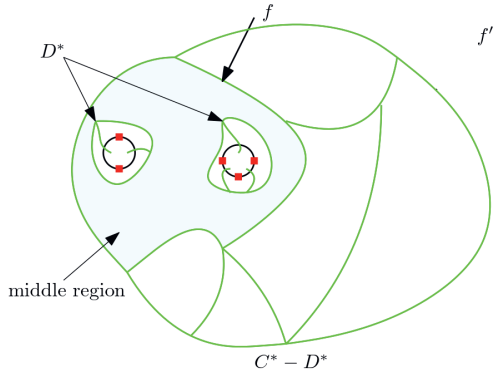

Figure 3.3: C^* is depicted in green. The region of the plane bounded by the face f is the enclosing region of D^*. The plane minus the region bounded by the faces of D^*, is the exclosing region of D^*. The shaded region represents the middle region.

Observe that neither the enclosing region nor the exclosing region is necessarily homeomorphic to a disk, nor are they necessarily equal. Also, the middle region is a face of H. In particular, at least one of the two regions is the outer face of J or $H - J$. Crucially, if J is not connected, then any connected component of H for which none of its bounded faces encloses another connected component of H is an internal set. This corresponds to our earlier intuition

that 'innermost' connected components are internal sets (but the definition of internal sets is much more general).

Remark 3.4.7. *By abuse of terminology, H itself can be viewed as an internal set, with the complement of the outer face of H as its enclosing region and the outer face as its exclosing and middle region. In particular, when $H = C^*$, where C^* is the set of dual edges corresponding to any multiway cut of (G, T, ω), the highly relevant Lemma 3.4.10 below holds when using C^* as the internal set.*

We now prove a useful property of internal sets.

Proposition 3.4.8. *Let J be an internal set of a plane graph H. Then $H - J$ is also an internal set.*

Proof. Since J is the union of a set of biconnected components, so is its complement $H - J$. The enclosing region of $H - J$ is the exclosing region of J and the exclosing region of $H - J$ is the enclosing region of J. □

Internal Sets and Multiway Cuts

We now prove a useful property of internal sets of a minimal multiway cut.

Definition 3.4.9. *Let C be any multiway cut of (G, T, ω) and C^* the set of corresponding dual edges. Let D^* be an internal set of C^*. We say that a point in the plane is* covered *by an internal set D^* if it is enclosed by the complement of the exclosing region of D^*.*

In our intuition of internal sets being 'innermost' connected components, a covered point is enclosed by a bounded face of the component.

Lemma 3.4.10. *Let C be any inclusion-wise minimal multiway cut of (G, T, ω) and C^* the set of corresponding dual edges. Let D^* be an internal set of C^* and D the set of corresponding edges in G. Let $F_\alpha \in \mathcal{F}$ be any terminal face for which a terminal of T_α is covered by D^*. Then $C - D$ does not contain any edge of F_α and at least $\max\{1, p_\alpha - 1\}$ terminals of T_α are covered by D^*. Moreover, there is at most one terminal face F_α for which this number is $p_\alpha - 1$.*

Proof. When $p_\alpha = 1$, trivially all terminals of T_α are covered by D^*. Moreover, C does not contain the edge of F_α, because the corresponding dual edge is a bridge in G^* and C^* is bridgeless by Lemma 3.4.3.

So assume that $p_\alpha > 1$. Let v_α denote the dual vertex corresponding to F_α. Following Lemma 3.4.4, C^* contains all (at least two) dual edges incident on v_α. As a terminal of T_α is covered by D^*, there is an edge of D^* incident on

v_α. By Lemma 3.4.5, v_α is not a cut vertex of C^* and thus the definition of an internal set implies that D^* contains v_α and all its incident dual edges. Hence, $C^* - D^*$ does not contain a dual edge incident on v_α and thus, $C - D$ does not contain an edge of F_α.

Now let t and t' be two distinct terminals of T_α not covered by D^*. Then both t and t' are in the exclosing region D^*. As no dual edges incident on v_α belong to $C^* - D^*$ and D^* is a union of biconnected components, t and t' are in the enclosing region of D^*. Hence, they are in the same face of C^*, which contradicts that C is a multiway cut. Hence, at least $p_\alpha - 1$ terminals of T_α are covered by D^*.

A similar contradiction holds for any two distinct terminals on distinct terminal faces which are not covered by D^*. Hence, there is at most one terminal face of which a terminal is covered by D^* and a terminal is not fully covered by D^*. □

The intuition of our approach is now as follows. If a terminal of T_α for a terminal face $F_\alpha \in \mathcal{F}$ is covered by D^*, then it follows from Lemma 3.4.10 that almost all terminals of T_α are covered by D^*. We then say that this component *covers* this terminal face. Then we could guess (by exhaustive enumeration) in 2^k time the subset $\mathcal{F}_{D^*}$ of terminal faces covered by D^* and obtain our simplified instance.

However, there is an important exception to this intuition, namely the unique terminal t in the middle region of D^*. This terminal t might not be on a face of $\mathcal{F}_{D^*}$, or worse, is on a face of $\mathcal{F}_{D^*}$ (and thus not covered by D^*; the possible existence of such a terminal is hinted at by Lemma 3.4.10). Knowing this terminal is important to the algorithm, but guessing (by exhaustive enumeration) this terminal could require $O(n)$ guesses. Since C^* has at most k terminal faces, this would lead to an undesirable $O(n^k)$ running time overall. We show that knowing the precise terminal t is actually unnecessary. By some small modifications to the weights, we argue that we can pick an arbitrary terminal of T_α that functions as a representative for t, avoiding the $O(n)$ guesses altogether.

Lemma 3.4.11. *Let C be any inclusion-wise minimal multiway cut of (G, T, ω) and C^* the set of corresponding dual edges. Let D^* be an internal set. Let T' be the set of terminals covered by D^* and let t be a terminal in the middle region of D^*. Let $F_\alpha \in \mathcal{F}$ be the unique terminal face that contains t. If F_α is a plural face and $T' \cap T_\alpha = \emptyset$ (none of the terminals of T_α are covered by D^*), let ω' be obtained from ω by setting the weight of each edge of F_α to ∞ and let t' be any terminal of F_α. Otherwise, let $t' = t$ and $\omega' = \omega$. Then:*

a. *for any finite-weight solution B of $(G, T' \cup \{t'\}, \omega')$, $B \cup (C - D)$ is a feasible solution for (G, T, ω);*

b. *D is an optimum solution for $(G, T' \cup \{t'\}, \omega')$, and thus $(G, T' \cup \{t'\}, \omega')$ has a finite-weight solution;*

c. *for any optimum solution B for $(G, T' \cup \{t'\}, \omega')$, B^* is an internal set of $B^* \cup (C^* - D^*)$.*

Proof. a. Let B be a finite-weight solution of $(G, T' \cup \{t'\}, \omega')$. We claim that $B \cup (C - D)$ is a feasible solution for (G, T, ω). Let B^* be the set of corresponding dual edges to B. Since B has finite weight, the construction of ω' and the definition of t' ensures that either $t = t'$ or the face of B^* that encloses t' also encloses T_α. Hence, in either case there is a face of B^* that encloses t and no other terminal of T'.

Consider the enclosing region f of D^*. Every terminal of $T' \cup \{t\}$ is enclosed by f. Then the intersections of each face of B^* with f yields a set of regions in the plane that each contain at most one terminal of $T' \cup \{t\}$. Since each terminal of $T - T' - \{t\}$ is contained in a face of $C^* - D^*$ enclosed by the complement of f, it follows that each face of $B^* \cup (C^* - D^*)$ contains at most one terminal (note that any edge of B^* not enclosed by f only partitions the faces of $C^* - D^*$ and is not relevant to the feasibility of $B \cup (C - D)$).

b. We now prove that D is an optimum solution for $(G, T' \cup \{t'\}, \omega')$. We first argue that D is a feasible solution for $(G, T' \cup \{t'\}, \omega')$ of finite weight. If $p_\alpha = 1$, then F_α consists of a single loop and the optimality of C ensures that C (and thus D) does not contain this loop. D has finite weight by construction. Moreover, $t' = t$ is in the exclosing region of D^* by definition, and thus, D is a feasible solution.

Otherwise, if $p_\alpha > 1$, consider two cases. If D^* covers a terminal of T_α, then $\omega' = \omega$ and D trivially has finite weight. Moreover, $t' = t$ is in the exclosing region of D^* by definition, D is a feasible solution. If none of the terminals of T_α are covered by D^*, then all terminals of T_α are in the exclosing region of D^*. Then D^* does not contain the dual vertex v_α (and therefore, no edge incident on v_α). Thus, D^* has finite weight. As T_α is in the exclosing region of D^*, so is t'. Since all terminals of T' are not in the exclosing region of D^*, D is a feasible solution.

To complete the argument, let B be an optimum solution for $(G, T' \cup \{t'\}, \omega')$ of weight strictly smaller than D. By the feasibility of D, B must have finite weight. Moreover, by the preceding, $B \cup (C - D)$ is a

feasible solution for (G, T, ω). Then it has smaller weight (with respect to ω) than C, a contradiction. Hence, D is an optimum solution for $(G, T' \cup \{t'\}, \omega)$.

c. Next, suppose that B is an optimum solution for $(G, T' \cup \{t'\}, \omega')$. We prove that B^* is an internal set with respect to A^*, where A^* is the set of dual edges corresponding to $A = B \cup (C - D)$. Note that A is an optimum solution for (G, T, ω) by the preceding. We now argue that B^* is enclosed by the enclosing region f of D^* and only possibly shares vertices with f. Suppose that B^* is not enclosed by the enclosing region f of D^* or shares edges with the boundary of f. As mentioned earlier in the proof, dual edges of B^* that are not enclosed by f are irrelevant to the feasibility of A. On the other hand, shared dual edges are effectively counted twice in A. Hence, let X^* be the set of dual edges in B^* that are either not enclosed by f or shared with the boundary of f. Let X be the corresponding set of edges in G. Note that $X^* \neq \emptyset$ by assumption and thus $\omega(X) > 0$. We have just argued that $(B - X) \cup (C - D)$ is still feasible. Moreover, $\omega(B) = \omega(D)$ and thus $\omega(B - X) < \omega(D)$. Then $\omega((B - X) \cup (C - D)) < \omega(D \cup (C - D)) = \omega(C)$, a contradiction to the optimality of C. Hence, B^* is enclosed by f and possibly shares only vertices with f. Hence, we can define f as the enclosing region of B^* and there is a single face of B^* that encloses $A^* - B^*$ that is the exclosing region.

   Finally, suppose that for this optimum solution B for $(G, T' \cup \{t'\}, \omega')$, B^* is not a union of biconnected components of A^*. Since B^* itself is trivially a union of biconnected components, this means that the intersection of the enclosing region and exclosing region of B^* with respect to A^* is in fact a union of at least two regions, and not a single middle region. Each of these regions is a face of A^*. Since at most one of these regions (faces) can contain a terminal by Lemma 3.4.10, A is not an optimum solution, a contradiction.

□

Algorithm

We are now ready to develop the algorithm. Let $\mathcal{A}$ be any algorithm for Planar Multiway Cut that always outputs a feasible solution, but is only guaranteed to find an optimum solution if for all optimum solutions C to the instance it holds that C^* is connected. We show in later sections that we can find such an algorithm $\mathcal{A}$ with the claimed running time bounds. Using it as a black box for now, we can give a recursive algorithm for Planar Multiway Cut.

Let (G, T, ω) be a transformed instance of Planar Multiway Cut with $\mathcal{F}$ the set of terminal faces. Recall that $\mathcal{F}$ can be partitioned into the set $\mathcal{F}_1$ of singular faces and the set $\mathcal{F}_2$ of plural faces; possibly, one of these sets is empty.

In the first phase of the algorithm, consider two new types of sub-instances. For the first type, if $\mathcal{F}_2 \neq \emptyset$, consider each $F_\beta \in \mathcal{F}_2$ and each set $\tilde{T}$ that is the union of the sets $\{T_\alpha \mid F_\alpha \in \tilde{\mathcal{F}}\}$ for a subset $\tilde{\mathcal{F}}$ of $\mathcal{F} \setminus \{F_\beta\}$, and solve recursively on the transformed version of the sub-instance (G, T'', ω'') where $T'' = \tilde{T} \cup \{{}^\beta t_1\}$ and ω'' is equal to ω, except it is set to ∞ for all edges of F_β. For the second type, consider each set T'' that is the union of the sets $\{T_\alpha \mid F_\alpha \in \tilde{\mathcal{F}}\}$ for a strict subset $\tilde{\mathcal{F}}$ of $\mathcal{F}$, and solve recursively on the transformed version of the sub-instance (G, T'', ω'') where $\omega'' = \omega$.

Let B be the solution that is recursively found for any such sub-instance (G, T'', ω'') and let B^* denote the set of corresponding dual edges. In the second phase of the algorithm, let T' be the set of terminals in the outer face of B^*. If there is a terminal face F_α for which $\emptyset \subset T' \cap T_\alpha \subset T_\alpha$, then consider the sub-instance $(G, (T' \setminus T_\alpha) \cup \{{}^\alpha t_1\}, \omega')$, where ω' is obtained from ω'' by setting the weight of each edge of F_α to ∞. Otherwise, consider the sub-instance (G, T', ω). Let Z_B be the solution that is recursively found for such a sub-instance. Among all such combinations $B \cup Z_B$ that are feasible solutions and $\mathcal{A}(G, T, \omega)$, return the feasible solution of minimum weight.

This finishes the description of the algorithm. We prove the following result.

Lemma 3.4.12. *Let $\mathcal{A}$ be any algorithm for* Planar Multiway Cut *that always outputs a feasible solution, but is only guaranteed to find an optimum solution if for all optimum solutions C to the instance it holds that C^* is connected. Let $T(n, k)$ be a monotone function that describes the running time of $\mathcal{A}$ on instances with n vertices and k terminal faces. Then* Planar Multiway Cut *can be solved in $O(2^{O(k)}T(n, k)\text{poly}(n))$ time.*

Proof. We prove that the above algorithm solves a given instance of Planar Multiway Cut in the claimed running time. Note that the described algorithm always returns a feasible solution, because $\mathcal{A}$ returns a feasible solution. To prove that the algorithm returns an optimum solution, let C^* be an optimum solution to (G, T, ω) with the maximum number of connected components. If C^* is connected, then $\mathcal{A}$ will find an optimum solution. Otherwise, let D^* be a connected component of C^* for which none of its bounded faces encloses another connected component of C^*. Then D^* is an internal set.

We observe that the instance constructed by Lemma 3.4.11 with respect to D^* and C^* is one of the sub-instances considered in the first phase of the algorithm. Indeed, we note that D^* is merely a connected component of C^*

(in particular, not equal to C^*) and by Lemma 3.4.10, there is at most one terminal t of the terminal faces covered by D^* such that t is not covered by D^*. Hence, the set of terminal faces covered by D^* is a strict subset of $\mathcal{F}$. If D^* does not cover all terminals of each terminal face covered by D^*, then t exists and is the terminal in the middle region of D^*. The instance of Lemma 3.4.11 simply consists of the terminals on the subset of terminal faces covered by D^*. As we argued, this is a strict subset of $\mathcal{F}$, and thus this instance is considered as a sub-instance in the second type of instances of phase one of the algorithm. If D^* does cover all terminals of each terminal face covered by D^*, then the terminal t in the middle region of D^* belongs to a terminal face not covered by D^*. If this is a singular face, then there is another terminal face not covered by D^* (in the part of the exclosing region of D^* that is not the middle region), and thus the instance of Lemma 3.4.11 is considered in the second type of instances of phase one of the algorithm. If this is a plural face, then the instance of Lemma 3.4.11 is considered in the first type of instances of phase one of the algorithm.

Consider the optimum solution B found by the algorithm for the instance constructed by Lemma 3.4.11 with respect to D^* and C^*. Let $A = B \cup (C - D)$. Let B^* and A^* denote the sets of corresponding dual edges. By Lemma 3.4.11, $\omega(B) = \omega(D)$, A is an optimum solution to (G, T, ω), and B^* is an internal set of A^*. Hence, $C^* - D^*$ is an internal set of A^* by Proposition 3.4.8. Repeating the same arguments as in the previous paragraph, we see that the instance constructed by Lemma 3.4.11 with respect to $C^* - D^*$ and A^* is considered in phase two of the algorithm.

Consider the optimum solution Z_B found by the algorithm for the instance constructed by Lemma 3.4.11 with respect to $C^* - D^*$ and A^*. By Lemma 3.4.11, $\omega(Z) = \omega(C - D)$ and thus $Z \cup B$ is an optimum solution for (G, T, ω). Hence, the algorithm will return an optimum solution. Here we note that if the instance is of the first type, then $F_\alpha = F_\beta$ in the second phase by the fact that B^* is an internal set.

Finally, to see the running time, note that each sub-instance is formed by a set of terminals $T_1 \cup T_2$, where T_1 is the union of the terminals of a subset of the terminal faces of $\mathcal{F}$ and T_2 is the set of first terminals of a subset $\mathcal{F}'$ of the plural faces of $\mathcal{F}$, such that these subsets of terminal faces are disjoint. The weight of all edges of the terminal faces in $\mathcal{F}'$ is set to ∞ and then the instance is transformed. It follows that there are at most 3^k distinct sub-instances ever considered by the algorithm. We also observe that for each sub-instance, compared to the input instance, the number of terminal faces decreases or else a plural face becomes singular. Hence, instead of the above top-down recursive algorithm, we can apply a bottom-up dynamic programming algorithm. Here

we look at the instances in order of increasing total number of terminal faces plus number of plural faces. To fill each table entry, we need to consider at most $(k+1)2^k$ table entries in the first phase and exactly one in the second phase. For each combination, we need polynomial time for feasibility check and to run $\mathcal{A}$. Hence, the total running time is as claimed. □

Klein and Marx [129, Lemma 3.1] also considered biconnected components, but individually. Then in their sub-instances, for a subset X of terminals, they not only need to pairwise separate the terminals of X, but also need to separate X from $T \setminus X$. We use the planarity of the solution to effectively argue that the latter condition is not necessary. Hence, our sub-instances are 'normal' instances of Planar Multiway Cut, without further constraints. To this end, it is required that the definition of an internal set is slightly more involved. Regardless, we need substantially more effort to deal with plural faces.

Following Lemma 3.4.12, we need to develop the required algorithm $\mathcal{A}$. Hence, in the remainder, we assume that the dual of any optimum solution to our instance of Planar Multiway Cut is connected.

3.5. Augmented Planar Multiway Cut Dual: Skeleton and Nerves

An important challenge for our algorithm is to find the part of an optimum solution that cuts the many terminal pairs incident on a single terminal face (of course, in concert with cutting terminal pairs on other faces). In the case that $k = 1$, Chen and Wu [44] proposed the notion of an augmented dual in order to reduce to an instance of Planar Steiner Tree wherein all the terminals lie on a single face. This, in turn, can be solved in polynomial time [79, 16]. Such a simple reduction does not work in our case, but we do borrow and extend the notion of an augmented dual.

Definition 3.5.1. *The* augmented dual *graph G^+ of G with respect to $\mathcal{F}$ is constructed as follows. Starting with the planar dual of G, for each face $F_\alpha \in \mathcal{F}$, replace the corresponding dual vertex v_α by a set of vertices $T^+_\alpha = \{{}^\alpha t^+_1, {}^\alpha t^+_2, \ldots, {}^\alpha t^+_{p_\alpha}\}$ (called the* augmented terminals*), each being an end point of an edge incident on v_α. For $1 \leq j \leq p_\alpha$, each ${}^\alpha t^+_j$ and its incident edge represents the edge $({}^\alpha t_{j-1}, {}^\alpha t_j)$ on the boundary of F_α. The weight function on the edges of G^+ is denoted by ω^+; all the edges in the augmented dual graph have the same weight as their corresponding edge in G under ω.*

Figure 3.4 illustrates the structure of C^+.

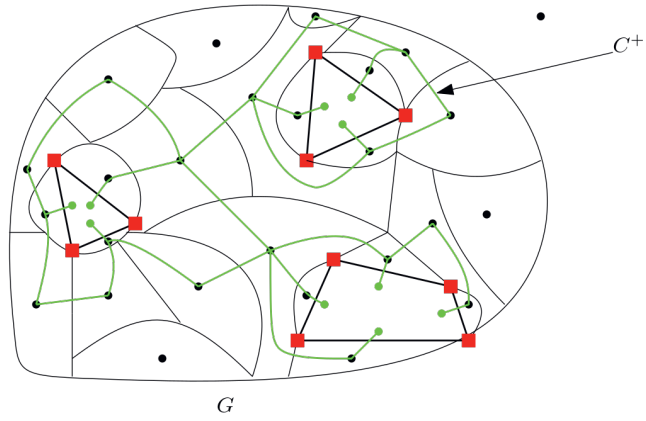

Figure 3.4: The figure shows the graph G in black. The red blocks represent terminals, while the green dots depict the augmented vertices. The black dots represent the dual vertices for each non-terminal face in the augmented dual. The green curves are the edges of C^+.

Observe that there is still a one-to-one correspondence between edges of G and (augmented dual) edges of G^+, as there is between edges of G and (dual) edges of G^*. We also note that an alternative construction of G^+ would be to subdivide each dual edge of G^* incident on the dual vertex v_α corresponding to a terminal face $F_\alpha \in \mathcal{F}$ and subsequently removing v_α. For practical purposes, any time we consider a set C^+ of augmented dual edges, we also denote by C^+ the subgraph of the dual induced by the edges in C^+.

Lemma 3.5.2. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then C^+ is a connected planar graph with k faces, one per terminal face in $\mathcal{F}$.*

Proof. Let C^* be the set of corresponding dual edges. By assumption and Proposition 3.4.2, we know that C^* is a connected subgraph of the dual, each face of which encloses exactly one terminal. First we show that C^+ is connected. Let $v_\alpha \in V(C^*)$ be the dual vertex representing the terminal face F_α. Recall that any dual vertex of a terminal face is not a cut vertex in C^*, by Lemma 3.4.5. Therefore, after splitting v_α into augmented terminals, every pair of augmented terminals remains connected in C^+. Suppose that there exists a path P in C^* from vertex a^* to b^*, such that P passes through v_α. Let x^* and y^* be the vertices on P adjacent to v_α. Note that in C^+, x^* and y^* are respectively the sole neighbors of two of the augmented terminals of F_α. Without loss of generality, let these augmented terminals be ${}^\alpha t_j^+$ and ${}^\alpha t_k^+$. By the preceding argument, there exists a path from ${}^\alpha t_j^+$ and ${}^\alpha t_k^+$ in C^+. To go from a^* to b^*,

we can follow the path from a^* to ${}^{\alpha}t_j^+$, then go along the path from ${}^{\alpha}t_j^+$ to ${}^{\alpha}t_k^+$ and finally from ${}^{\alpha}t_k^+$ to b^*. Since a^* and b^* were arbitrary, this holds for any path passing through the dual vertex of a terminal face. Since no modification was made to any other path of C^*, this proves that C^+ is connected.

Next, we show that C^+ contains exactly k faces, each enclosing a terminal face of $\mathcal{F}$. For all F_α such that $F_\alpha \in \mathcal{F}$, observe that its dual vertex v_α lies at the intersection of face boundaries enclosing each terminal on the boundary of F_α. None of these face boundaries intersect a dual vertex corresponding to any face F_β , for $\beta \neq \alpha$, or else C^* would enclose more than one terminal in that face, contradicting its feasibility. After splitting v_α into augmented terminals, by the preceding paragraph, we get a connected graph. In particular, each pair of consecutive augmented terminals is connected in C^+ by a path. The union of these paths bounds a region in the plane enclosing the face F_α. Therefore, C^+ has exactly k faces. □

3.5.1. Skeleton

Considering Lemma 3.5.2, we note in particular that C^+ has k faces. However, each of these k faces can be highly complex. By 'zooming out', we can show that C^+ in fact has a very nice global structure with k faces that are much easier to describe. To this end, we prove the following.

We define the action of *dissolving* a vertex v of degree 2 with *distinct* neighbors u and w, as removing v from the graph and adding an edge from u to w. If we drop the constraint that u, w are distinct, we speak of *strongly dissolving*. We explicitly mention that while dissolving a vertex we retain any parallel edges that may arise. Moreover, no self-loops may arise while dissolving a vertex, while this is possible when strictly dissolving a vertex.

Definition 3.5.3. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then the set of augmented dual edges that remain after exhaustively removing all the edges incident on a vertex of degree 1 of C^+, induces a graph called the* skeleton *of C^+, denoted by S^+. The set of augmented dual edges that remain after dissolving all vertices of degree 2 from S^+ induces a graph called the* shrunken skeleton *of C^+.*

Lemma 3.5.4. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then the shrunken skeleton of C^+ is a connected planar multi-graph without self-loops and with k faces, such that each of its faces strictly encloses exactly one terminal face of $\mathcal{F}$.*

Proof. Following Lemma 3.5.2, C^+ has k faces. Recall that for any bridge of a planar graph, the faces on both sides of the bridge are the same. Hence,

removing a bridge does not decrease the number of faces. It follows immediately that the skeleton S^+ has k faces, one per terminal face in $\mathcal{F}$. Since each augmented terminal has degree one, its incident edge is a bridge of G^+. Hence, each face of S^+ strictly encloses exactly one terminal face of $\mathcal{F}$. Since C^+ is connected by Lemma 3.5.2, by construction, S^+ is also connected. Note that, while dissolving vertices of degree 2, we can maintain essentially the same embedding as of S^+. Hence, the shrunken skeleton has the same properties.

Finally, we consider the possibility of self-loops. Since self-loops cannot arise by dissolving vertices of degree 2 or removing edges incident on a vertex of degree 1, this means that C^+ and thus even C^* must have a self-loop. This self-loop encloses a terminal by the minimality of C. Then the edge of G corresponding to this self-loop is a bridge of G. However, G is bridgeless by Lemma 3.4.1. □

It is crucial to note that while the skeleton is connected, it is not necessarily bridgeless. This has important algorithmic consequences discussed later.

Definition 3.5.5. *Let $F_\alpha \in \mathcal{F}$ and let f_α be the corresponding face of the skeleton of C^+. We call the set of augmented dual edges of f_α the* spine *of F_α with respect to C^+.*

Note that a spine is not necessarily isomorphic to a cycle, but only to a closed walk, because the skeleton is not necessarily bridgeless.

Definition 3.5.6. *Let $F_\alpha \in \mathcal{F}$. An* enclosing cut *is a cut that separates T_α from $T \setminus T_\alpha$ and also separates each terminal in T_α from every other terminal.*

Remark 3.5.7. *Chen and Wu [44] use the term* island cut *to denote the $(T_\alpha, T \setminus T_\alpha)$-cut for each $F_\alpha \in \mathcal{F}$. Since they find a 2-approximate solution instead of an exact one, it suffices to separate the graphs into islands and then find a multiway cut for each island.*

Corollary 3.5.8. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Consider a face f_α of the skeleton of C^+ and let $F_\alpha \in \mathcal{F}$ be the single terminal face strictly enclosed by it. Then the set of edges of G corresponding to the spine of F_α is a $(T_\alpha, T \setminus T_\alpha)$-cut and the set of edges of G corresponding to the augmented dual edges of C^+ enclosed by f_α is an enclosing cut.*

Proof. This is immediate from the fact that, by Lemma 3.5.4, each face of the skeleton strictly encloses exactly one terminal face of $\mathcal{F}$. □

The following is proved as in Dahlhaus *et al.* [60, Lemma 2.2] with some modifications.

Lemma 3.5.9. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then the shrunken skeleton of C^+ has at most $4k$ vertices and at most $7k$ edges.*

Proof. Consider the shrunken skeleton of C^+. It has no vertices of degree 1 and no self-loops, is connected, and has k faces by Lemma 3.5.4. However, it might have parallel edges and vertices of degree 2.

We now propose a charging scheme to obtain a simple connected planar graph without vertices of degree 1 and 2. To this end, we perform three types of operations: we dissolve vertices of degree 2, remove vertices of degree 1, and remove edges incident on two distinct faces. Note that these operations preserve the properties that the graph is connected and has no self-loops. We charge the number of vertices and edges removed during these operations to the faces of the graph. Any time we use the charge of a face, the operations we perform also destroy the face. Then, we will be able to relate the number of vertices and edges removed to the number of faces of the shrunken skeleton.

Initially, we give each face a charge of $(4, 7)$: these charges are to account for removed vertices and edges respectively. During the process, we move some of the charge onto certain vertices. As an invariant, called the charge invariant, we maintain that

- every face has charge $(4, 7)$;
- every vertex of degree 2 has charge $(2, 3)$, except possibly if its incident edges are parallel;
- every vertex of degree 1 as well as its neighbor has charge $(2, 3)$.

Note that the charge invariant initially holds, as the shrunken skeleton has no vertex of degree 1 and any vertex of degree 2 must be incident on parallel edges (or it would have been dissolved).

We now describe the scheme. Consider a path P of maximum length for which all internal vertices have degree 2, and such that P has length at least 2, has an endpoint of degree 1, or is a single edge parallel to another edge. Let q denote the length of P and let u, v denote its ends. We perform different operations depending on the structure of P:

- there is an edge e between u and v that is not on P. Note that this includes the case when P is a single edge parallel to another edge.

  We observe that e and P form a cycle, and thus e borders two distinct faces. Let f denote one of these faces. Remove e and charge this removal to f. Assign the remaining charge $(4, 6)$ of f to u, v such that they both hold charge $(2, 3)$.

Note that the charge invariant is still satisfied. Indeed, f has disappeared after the removal of e, so every face still has charge $(4, 7)$. The degree of u and v has dropped, but they were assigned charge $(2, 3)$. If their degree becomes 1, note that u, v are still adjacent through P or are incident on an internal vertex of P. These internal vertices must have degree 2 (already prior to the operation) and are not incident on parallel edges by the fact that u, v are distinct, and thus have charge $(2, 3)$.

- u, v both have degree at least 2.

  Dissolve every internal vertex of P. Since they have degree 2 and u, v are distinct, this is well-defined. The available charge of the internal vertices of P is $(2(q-1), 3(q-1))$. Since $q-1$ vertices and $q-1$ edges are removed, this is sufficient.

  The charge invariant is still satisfied, as the degree and charge of u and v is unchanged.

- u has degree 1.

  Note that v must have degree at least 3 or degree 1 by the maximality of P. Iteratively remove every vertex of degree 1 on the path, except v. The available charge from the removed vertices is $(2q, 3q)$. Since q vertices and q edges are removed, this is sufficient. If $q = 1$, then v has charge $(2, 3)$ by the charge invariant and still has degree at least 2 or degree 0 by the definition of P, and thus the charge invariant holds again. If $q \geq 2$, then a charge of at least $(2, 3)$ has remained, which we assign to v. As before, v still has degree at least 2 or degree 0, and the charge invariant holds.

We perform these operations until no such path P exists. Note that the first operation does not remove P; however, after removing all the edges not on P between u and v, it will end up being removed. The remaining graph is a simple connected planar graph without vertices of degree 1 and 2. Moreover, by the charging scheme, the number of vertices and edges removed during the scheme is at most four (respectively, seven) times the number of faces that were removed.

Let v, e, and f be the number of vertices, edges, and faces of the remainder of the shrunken skeleton. By Euler's formula, $v - e + f = 2$. The remainder has no vertices of degree 1 and 2; hence, $e \geq 3v/2$ and thus $v - 3v/2 + f \geq 2$. Then $v \leq 2(f-2) \leq 2f$. We also recall that any simple connected planar graph satisfies $e \leq 3v - 6$ by Euler's formula, and thus $e \leq 6f$. Combined with the charging scheme, the bound on the number of vertices and edges follows. □

3.5.2. Single Face

Per Corollary 3.5.8, for each face $F_\alpha \in \mathcal{F}$ and corresponding face f_α of the skeleton, the spine of F_α is an island cut and the set of augmented dual edges of C^+ enclosed by f_α is an enclosing cut. We show that the enclosing cut has a very specific structure.

Intervals

We need the following definitions.

Definition 3.5.10. *Let $F_\alpha \in \mathcal{F}$. An* interval *of augmented terminals of T^+_α is the set $\{{}^\alpha t^+_i \mid i \in I\}$, where $I \subseteq \{1, \ldots, p_\alpha\}$ is any set such that either $|I| = p_\alpha$ or for each $i \in \{1, \ldots, p_\alpha\}$ except exactly one, it holds that there is a $j \in I$ where $j = i + 1 \mod p_\alpha$. We say that the interval is* between i and j *$(1 \leq i, j \leq p_\alpha)$ if $i \leq j$ and $I = \{i, \ldots, j\}$, or $i > j$ and $I = \{i, \ldots, p_\alpha, 1, \ldots, j\}$.*

It is important to note that we treat intervals as ordered sets of terminals. In particular, the intervals $\{{}^\alpha t^+_1, \ldots, {}^\alpha t^+_{p_\alpha}\}$ and $\{{}^\alpha t^+_2, \ldots, {}^\alpha t^+_{p_\alpha}, {}^\alpha t^+_1\}$ are distinct.

This definition invites several additional definitions that will prove useful in later parts of the paper. Two intervals I, I' of the same terminal face F_α are *consecutive* if they are disjoint and there is a third (possibly empty) interval I'', disjoint from I and I', such that $I \cup I' \cup I'' = T^+_\alpha$.

Given two consecutive intervals $I = \{{}^\alpha t^+_1, \ldots, {}^\alpha t^+_a\}$ and $I' = \{{}^\alpha t^+_{a+1}, \ldots, {}^\alpha t^+_b\}$ of the same terminal face F_α, a terminal $t \in T_\alpha$ is *in-between* I and I' if $t = {}^\alpha t_a$. Note that there is exactly one terminal inbetween two consecutive intervals, unless $I \cup I' = T^+_\alpha$, in which case there are exactly two. A terminal $t \in T_\alpha$ is *between* $I = \{{}^\alpha t^+_1, \ldots, {}^\alpha t^+_a\}$ if $t \in \{t_1, \ldots, t_{a-1}\}$.

A *subinterval* of an interval $I = \{{}^\alpha t^+_1, \ldots, {}^\alpha t^+_a\}$ is any subset $\{{}^\alpha t^+_b, \ldots, {}^\alpha t^+_c\}$ of I such that $1 \leq b \leq c \leq a$ or is the empty interval. A *prefix* of an interval $I = \{{}^\alpha t^+_1, \ldots, {}^\alpha t^+_a\}$ is any subinterval of I containing ${}^\alpha t^+_1$ or is the empty interval. A *suffix* of an interval $I = \{{}^\alpha t^+_1, \ldots, {}^\alpha t^+_a\}$ is any subinterval of I containing ${}^\alpha t^+_a$ or is the empty interval.

(Note that we have started the intervals at ${}^\alpha t^+_1$ only for simplicity and to avoid modulo-calculus, but the definitions extend in the obvious manner.)

Enclosing Cut on your Nerves

Using the notion of intervals, we can describe the enclosing cut. To this end, we rely on the following result.

Theorem 3.5.11 (Chen and Wu [44, Lemma 3])**.** *Let (G, T, ω) be an instance of Edge Multiway Cut where G is a biconnected plane graph such that all terminals of T lie on the outer face. Then a set of edges is a minimal multiway cut for this instance if and only if the corresponding augmented dual edges form a minimal Steiner tree in the augmented dual on the augmented terminals.*

In this result, the biconnectivity of G ensures that the augmented dual G^+ is connected. For the conclusion of the above theorem to hold, it suffices that G^+ is connected.

Corollary 3.5.12. *Let (G, T, ω) be an instance of Edge Multiway Cut where G is a plane graph such that all terminals of T lie on the outer face and G^+ is connected. Then a set of edges is a minimal multiway cut for this instance if and only if the corresponding augmented dual edges form a minimal Steiner tree in G^+ on the augmented terminals.*

We can also obtain the following corollary.

Corollary 3.5.13. *Let (G, T, ω) be an instance of Edge Multiway Cut where G is a plane graph such that the outer face is bounded by a cycle and all terminals of T lie on this cycle. Then a set of edges is a minimal multiway cut for this instance if and only if the corresponding augmented dual edges form a minimal Steiner tree in the augmented dual on the augmented terminals.*

Proof. We observe that in G^*, the vertex representing the outer face is not a cut vertex of the solution by Lemma 3.4.5. Hence, G^+ is connected and the result follows from Corollary 3.5.12. □

It is important to note that the mentioned cycle might be two parallel arcs or a self-loop.

These results suggest that Steiner trees in the augmented dual are important for a multiway cut. This is indeed the case, although the situation is substantially more involved when $|\mathcal{F}| > 1$.

Lemma 3.5.14. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Consider a face f_α of the skeleton S^+ of C^+ and let $F_\alpha \in \mathcal{F}$ be the single terminal face strictly enclosed by it. Let Y^+ be the spine of F_α and let X^+ be the set of remaining augmented dual edges enclosed by f_α. Then X^+ is a forest of minimum Steiner trees with the terminals of each minimum Steiner tree lying in some interval of T^+_α, such that the union of all intervals is T^+_α, and each minimum Steiner tree having a single vertex on Y^+ that is incident on a single edge of that tree.*

Proof. By Corollary 3.5.8, $G \setminus Y$ does not contain any path connecting the terminals of T_α to terminals of $T \setminus T_\alpha$. Therefore, the region of G^+ bounded by Y^+ and the boundary of F_α does not contain any terminal of $T \setminus T_\alpha$. The edges in X thus correspond to a multiway cut of T_α in $G \setminus Y$. Moreover, this multiway cut is minimal in $G \setminus Y$: any edge that could be removed from X while it remains a multiway cut in $G \setminus Y$ can also be removed from C while it remains a multiway cut in G, which would contradict the minimality of C.

Now consider $G \setminus Y$. Note that the spine does not contain any augmented terminals, as those have degree 1 in C^+ and thus are not part of the skeleton by definition. Hence, Y does not contain any edges of F_α and therefore, F_α persists in $G \setminus Y$. Let Q be the component of $G \setminus Y$ that contains F_α. Since the instance is preprocessed by adding an edge between every pair of consecutive vertices on any terminal face of the input graph G, F_α is a simple cycle in Q. Moreover, X is a minimal multiway cut for T_α in Q, as argued above. Hence, by Corollary 3.5.13, X^+ is a minimal Steiner tree in the augmented dual of Q on the terminal set T_α^+. It is in fact of minimum weight; otherwise, we could replace X^+ by a minimum-weight Steiner tree on T_α^+ and obtain a minimum multiway cut for G of smaller weight using Corollary 3.5.13 and the fact that Y forms an island cut.

Now we consider what X^+ looks like in the augmented dual of G. We note that all edges of the augmented dual of $G \setminus Y$ also appear in the augmented dual of G. Hence, we can think of X^+ as a set of edges in G^+. Let X be the corresponding set of edges of G. Deleting the set of edges in Y from G is equivalent to contracting the edges of Y^+ in the augmented dual. Let y^+ be the vertex formed by contracting all the edges in Y^+. Since C^+ is connected by Lemma 3.5.2, it follows that at least one edge of X^+ is incident on y^+.

Deleting y^+ from X^+ creates $\deg(y^+)$ many connected components of X^+. We treat each of these connected components as subsets of edges, which includes the unique edge incident on y^+. We claim that each of these components is a minimum Steiner tree in G^+ containing some interval of T_α^+ in its vertex set. Clearly, each of the components is a Steiner tree on y^+ and some subset of T_α^+. We first show that the augmented terminals contained in each component form an interval of T_α^+. Due to the planarity of G^+, no two tree edges cross each other. Therefore, if the terminals of any two connected components were to cross each other on T_α^+, then they must intersect in some vertex. However, since each connected component is maximally connected, we reach a contradiction. Hence, the terminals form an interval.

Due to the minimality of X^+ in G^+/Y^+, with respect to the terminal set $T_\alpha^+ \cup \{y^+\}$, we claim that each of the components is a minimum Steiner tree on G^+/Y^+ containing its respective interval of T_α^+. Suppose that one of these trees is not a minimum Steiner tree in G^+. Then, there must exist in G^+ a Steiner tree W^+ of lower weight on the same terminal set, such that it either contains some edges of Y^+ or crosses Y^+. In the latter case, there is a subpath of this Steiner tree that intersects Y^+ in at least two vertices, leading to the creation of a cycle enclosing no terminal of T_α. That would contradict the minimality of X^+. In the former scenario, suppose that we replace the concerned component of X^+ with W^+. We claim that all the

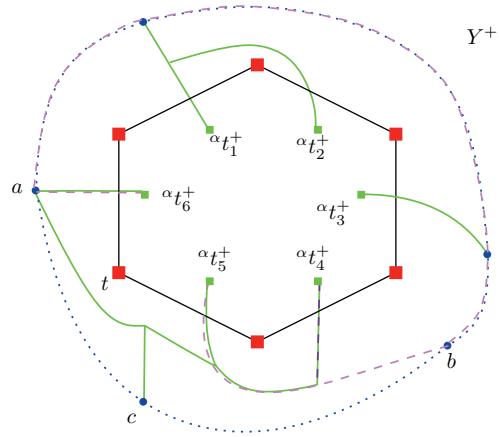

Figure 3.5: The boundary of the face F_α is shown in black. The terminals of T_α lying on the boundary are drawn as red boxes. The spine Y^+ of the minimum enclosing cut dual is drawn in blue dotted lines. The green trees are the minimum Steiner trees of $X^+ \setminus \{y^+\}$ in G^+/Y^+. The violet dashed lines represent the minimum Steiner tree in G^+ on the terminal set $\{^\alpha t_4^+, ^\alpha t_5^+, ^\alpha t_6^+\}$, in which the path between $^\alpha t_5^+$ and $^\alpha t_6^+$ contains the subpath of Y^+ between the vertices a and b that does not enclose t.

previously enclosed terminals by the component of X^+, remain enclosed after the replacement. If for some terminal $t \in T_\alpha$ in the interval, W^+ does not contain a subpath enclosing it, then the path between the augmented vertices flanking it contains edges of Y^+. This path in W^+ crosses all the paths from t to $T_\alpha \setminus \{t\}$, and the segment of Y^+ between the points of intersection crosses all the paths from t to terminals in $T \setminus T_\alpha$. Therefore, the region bounded by W^+ and Y^+ that contains t cannot contain any other terminal of T_α, and t remains enclosed. The replacement would, thus, yield an enclosing cut of T_α, strictly smaller than $X^+ \cup Y^+$. This, again, contradicts the minimality of X^+. Figure 3.5 illustrates the argument.

Finally, suppose that some minimum Steiner tree intersects the spine in more than one vertex. Then $X^+ \cup Y^+$ would contain a cycle that encloses no terminals, which contradicts Proposition 3.4.2.

Therefore, X^+ is a forest of minimum Steiner trees on intervals of T_α^+ with the terminals of each minimum Steiner tree lying in some interval of T_α^+, such that the union of all intervals is T_α^+, and each minimum Steiner tree having a single vertex on Y^+ that is incident on a single edge of that tree. □

Remark 3.5.15. *For the above lemma and proof, it is important to observe that we can replace X^+ by any other forest of minimum Steiner trees on the same intervals and same vertices on Y^+. Indeed, such a forest would form a minimum Steiner tree in G^+/Y^+. Then we can replace X^+ in the argument*

of the lemma by the new forest.

This remark inspires the following definition.

Definition 3.5.16. *Each minimum Steiner tree in the forest X^+ is called a* nerve*. Each nerve intersects the spine in exactly one vertex, which is incident on a single edge of the nerve. The unique vertex of the spine at which a nerve is rooted is called its* attachment point*.*

A nerve for a plural face $F_\alpha \in \mathcal{F}$ can be specified by a triple (v, i, j) with $1 \leq i, j \leq p$ such that the nerve is a minimum Steiner tree in G^+ with terminal set being the attachment point v and the interval of augmented terminals between i and j, and a single edge incident on v.

For a nerve (v, i, j), we may speak of the interval between i and j as simply the interval of the nerve. In this way, we can extend the definitions of consecutive, between, and in between, originally defined for intervals, to nerves in the straightforward manner.

A Steiner tree, as in Definition 3.5.16 might not necessarily exist, but we have argued in Lemma 3.5.14 that our solution is built only from such Steiner trees.

We now argue that nerves can be computed in polynomial time.

Lemma 3.5.17. *Given vertices v and an interval I of a terminal face F_α between i and j, we can compute in polynomial time a nerve (v, i, j), if it exists.*

Proof. It follows from Bern [16] (or Kisfaludi-Bak *et al.* [118]) that a minimum Steiner tree on a set of terminals with at most two faces can be computed in polynomial time. We first compute a minimum Steiner tree Z^+ in G^+ on v and the augmented terminals in the interval between i and j. Then, for each neighbor u of v, we compute a minimum Steiner tree Z_u^+ in $G^+ \setminus \{v\}$ on u and the augmented terminals between i and j. By the preceding, this can be done in polynomial time. Then we compute $\omega(Z^+)$ with $\omega(Z_u^+) + \omega(v, u)$ for all u. If there is a u for which they are equal, then Z_u^+ combined with the edge (v, u) is a nerve. Moreover, any nerve can be constructed in this way. □

Now note that nerves are not necessarily unique for a triple (v, i, j). From now on, we associate a unique minimum Steiner tree with each triple (if it exists), namely the one computed by Lemma 3.5.17. We call this the *unique nerve* (v, i, j). In the remainder, whenever we talk about a nerve, we mean the unique nerve on the same interval and attachment point. Following Remark 3.5.15, we may assume that minimum-weight multiway cuts are built from a skeleton and (unique) nerves.

Definition 3.5.18. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then C^+ (and by extension, C) is called* nerved *if the edges of C^+ that are not part of the skeleton each belong to a nerve.*

Observe that following Remark 3.5.15 and the discussion above, we can assume that any minimum-weight multiway cut is nerved.

3.6. Bones and Homotopy

We now wish to describe the structure of the paths between the attachment points of nerves and to branching points of the skeleton. To this end, we start with the following definition.

Definition 3.6.1. *Each edge of the shrunken skeleton is called a* shrunken bone *and corresponds to a path of the skeleton, called a* bone*. Any vertex of the shrunken skeleton is called a* branching point*.*

By Lemma 3.5.4, each (shrunken) bone is incident on one or two faces of the (shrunken) skeleton, and thus separates at most two faces of $\mathcal{F}$, called the *separated terminal faces* of the (shrunken) bone. Note that both sides of a (shrunken) bone might be incident on the same face, as the shrunken skeleton can contain bridges.

3.6.1. Orienting Nerves

To further discussions in the remainder of the paper, it helps to define an orientation of the nerves, which specifies where a nerve goes with respect to a bone. Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding edges in the augmented dual. For any augmented dual vertex x, there is a small ball centered on x that only contains x and points of its incident augmented dual edges. Let e and e' be two augmented dual edges incident on x. Then this small ball is split into two parts by the union of e and e'. An augmented dual edge is *west* of x with respect to e and e' if it is contained in the part of the ball clockwise from e' to e and *east* otherwise.

Now consider two augmented dual edges e and e' of the skeleton of C^+ that appear in counterclockwise order on the boundary of a face f_α of the skeleton of C^+ (so e appears after e'). Let x be their shared endpoint. If the boundary of f_α is a walk, then we only consider the first appearance of an augmented dual edge to define the ordering. Let $F_\alpha \in \mathcal{F}$ be the terminal face corresponding to f_α. We then say that a nerve (x, h, i) for F_α with $1 \leq h, i \leq p_\alpha$ *extends west* of x if it holds that the augmented dual edge incident on x that belongs to the

nerve is west of x. If the augmented dual edge incident on x that belongs to the nerve is east of x, then the nerve *extends east*. By definition, a nerve has only a single incident edge, and thus it either extends east or it extends west.

3.6.2. Homotopy and the Optimum Solution

We now want to describe the structure of bones. While it might seem that these are just shortest paths between certain branching and attachment points, this does not account for their role in separating pairs of terminals. In order to specify this role, we need assistance from homotopy.

To facilitate the discussion of homotopy, we first need some definitions. First, we need the notion of crossings between two paths P and Q in a planar graph. Then we say that P *crosses* Q if, after contracting any common edges of P and Q, there are two edges of P and two edges of Q, all distinct and incident on the same vertex, such that a clockwise rotation around the vertex will see the two edges of P and Q alternately.

Now we construct a *cut graph* K for G^*, which is a Steiner tree in G^* with the set $V_{\mathcal{F}}^*$ of dual vertices of the terminal faces in $\mathcal{F}$ as terminal set. Here we follow Frank and Schrijver [88, Proposition 1]. First, compute a shortest path between each pair of dual vertices in $V_{\mathcal{F}}^*$. Let $\mathcal{P}$ denote the resulting set of paths. By a slight modification of the weights for the sake of this computation, we can assume all shortest paths in G^* are unique. Then we can assume that any pair P, Q of shortest paths in $\mathcal{P}$ either crosses at most once, or has a common endpoint and does not cross. Now build an auxiliary complete graph on $V_{\mathcal{F}}^*$ with the lengths of the paths in $\mathcal{P}$ as weights and find a minimum spanning tree in this auxiliary graph. The paths in $\mathcal{P}' \subseteq \mathcal{P}$ corresponding to this spanning tree form the cut graph K.

As explained by Frank and Schrijver [88, Proposition 1], we may assume the paths in $\mathcal{P}'$ do not cross. We call the paths in $\mathcal{P}'$ the *spokes* of the cut graph. Note that K has $k-1$ spokes, each of which can be associated with a unique identifier. We also orient each spoke in an arbitrary direction. Then any oriented path P that crosses a spoke Q can be said to cross Q in a particular direction, depending on whether P goes towards the west or east side of Q.

Definition 3.6.2. *The* crossing sequence *or* homotopy string *of a path P in G^+ is an ordered sequence of the spokes of K crossed by it, along with an indication of the orientation of each crossing. Let P and P' be two paths in G^+. We say that P is* homotopic *to P' if they have the same endpoints as well as the same crossing sequence with respect to K.*

Note that we perform a slight abuse here by considering crossings of a path in G^+ with paths in G^*. However, we will never consider crossings of paths

in G^+ that have an endpoint in an augmented dual terminal, alleviating any possible cause for confusion.

Using this precise definition of homotopy, we now fix a particular solution to the instance. Among all possible nerved minimum multiway cuts C of (G, T, ω), we assume henceforth that C is one of which the skeleton of C^+ has the minimum number of crossings with K. Note that the skeleton contains no augmented dual vertices, as those have degree 1 in G^+.

From now on, we fix this solution as 'the' optimal solution or 'the' optimum. Let C^+ be the set of corresponding augmented dual edges. It is reasonable to assume that such a solution is inclusion-wise minimal, that is, removing any of its edges must make it an infeasible multiway cut.

Our main goal in the next subsections will be to show that (a sufficient part of) the bones of the skeleton of the optimum solution have short homotopy strings and that replacing such parts by parts with the same homotopy string leads to another optimum solution.

3.6.3. Nerve Path

We now consider a subpath of a bone that starts and ends at attachment point of two distinct nerves and contains attachment points only of nerves that extend towards the same direction. In particular, the nerves extend towards the same terminal face. We call such a subpath a *nerve path*. We aim to prove the following crucial property of nerve paths, namely that any nerve path has a homotopy string of bounded length.

For the remainder of this section, let u, v be two endpoints of a nerve path P of C^+. Let f_α, f_β be the faces of the skeleton separated by P and let $F_\alpha, F_\beta \in \mathcal{F}$ be the corresponding terminal faces. If $\alpha = \beta$, then without loss of generality, we assume that all the nerves extend towards the west of P.

Lemma 3.6.3. *Any nerve path crosses any path of K $O(k)$ times.*

Proof. We modify the proof of [63, Lemma 5.2] to prove the claim above. Let K be the graph defined above and γ be a path in K. We treat the dual vertices of the terminal faces as obstacles. Since P is a path in the augmented dual G^+, P does not pass through any terminal of K, in particular, through the endpoints of γ.

Let the nerves attaching to the ends of P be denoted N_1 and N_2. Without loss of generality, we assume that both N_1 and N_2 (and all the other nerves on P, if they exist) extend towards F_β. We push all the crossings of P with γ together to a single point p/p. Then the subpaths of P between consecutive crossings of P with γ form cycles containing p/p, which we call *loops*. The corresponding system of loops is $P^* = P/p$. Then, P^* is a set of pairwise

disjoint loops L intersecting at p/p. The contracted point is referred to as the base point of the loops. A face of L is called a *monogon* if it is homeomorphic to an open disc (contains no terminal face in its interior) and has only one copy of the base point on its boundary. Likewise, a *bigon* is an open disc with two copies of p/p on its boundary. In the bone P, a bigon occurs as a strip bounded by two subpaths of the path γ and two subpaths of P. To show that the number of crossings of P with γ is $O(k)$, we need to show that the degree of p/p is $O(k)$ in P^*.

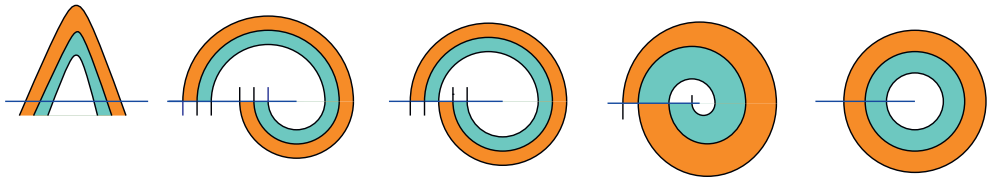

Figure 3.6: The figure shows all possible ways in which a loop in L can be incident on two bigons. The two bigons are shaded in orange and turquoise. The blue line depicts γ.

Figure 3.6 (same as [63, Figure 2]) shows an exhaustive list of configurations of any path crossing an edge of the cut graph K, which lead to the occurrence of a loop incident on two bigons in L. Note that the right-most configuration does not occur in our setting since P is a path.

We claim that no loop in L can be incident on two bigons. Let us assume the contrary. The following claim rules out some configurations giving rise to such a loop.

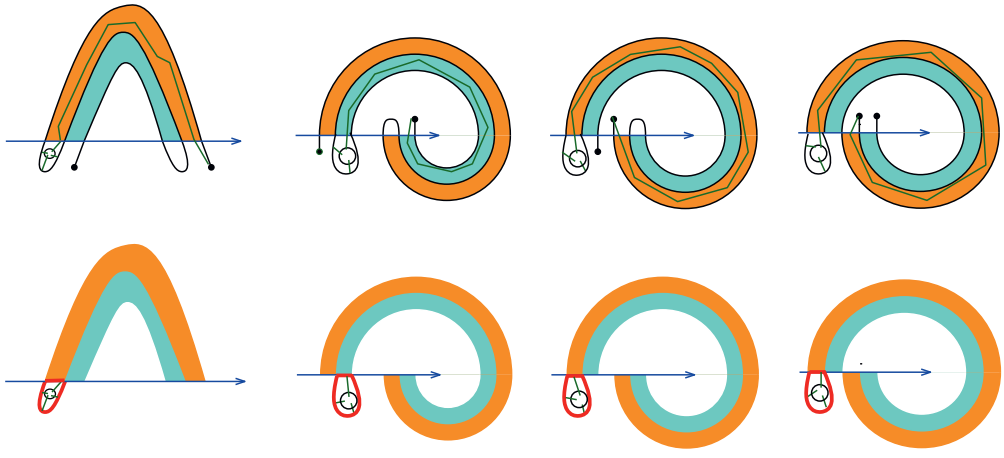

Figure 3.7: The figure shows the configurations described in Claim 3.6.4.

Claim 3.6.4. *Let $\gamma[a, b]$ be a subpath of γ on the boundary of one of the two bigons and $P[a, b]$ a subpath of P that is not incident any of the two bigons. If F_β is contained in the region bounded by $\gamma[a, b] \cup P[a, b]$ and there exists a subpath $P[x, y]$ incident on at least one of the bigons such that $\gamma[a, b] \subseteq \gamma[x, y]$ then we can shortcut P.*

Proof. Any nerve reaching F_β must either cross $\gamma[a, b]$ or have its attachment point on $P[a, b]$. Therefore, we can remove $P[x, y]$ from C^+ and replace it with $\gamma[a, b]$. Since $\gamma[x, y]$ is a shortest path in G^* and therefore G^+, it is at most the length of $P[x, y]$ and since $\gamma[a, b]$ is a subpath of $\gamma[x, y]$, we get a solution of weight at most that of C^+ with strictly fewer crossings. This is a contradiction. ◇

Consider the two bigons depicted as orange and turquoise strips in Figure 3.6. There are three loops that together bound the union of the two bigons, namely, the top-most loop incident on the orange bigon only, the middle loop incident on both the bigons and the bottom-most loop incident on the turquoise bigon only. As neither of the bigons contain any terminal and due to Claim 3.6.4, the terminal face F_β can be contained in the following two regions:

- Region A : the region bounded by the bottom-most loop incident on the turquoise bigon and the subpath of γ between its endpoints.
- Region B: the unbounded region incident on the union of top-most loop incident only on the orange bigon and the subpath of γ between its endpoints.

We consider different configurations of P creating a loop incident on two bigons and in each such configuration, we present an exchange argument to replace P by a shorter path in C^+ that preserves the feasibility of C^+. The replacement is based on the intuition that any nerve that must cross γ to reach F_β can be shortcut to have its attachment point on γ. However, there can be nerves that never cross γ. Then, we must preserve all subpaths of P containing the attachment points of such nerves.

We say that a nerve *traverses* a bigon if any path of the nerve from its attachment point to its leaves crosses each of the two distinct subpaths of γ bounding the bigon an odd number of times.

Case 1: F_β is contained in Region A.
Here, we consider two subcases, namely, whether subpaths of nerves are contained in the orange or the turquoise bigon. Figure 3.8 shows the specific replacement for every configuration possible in this case (modulo symmetric configurations with the same replacement).

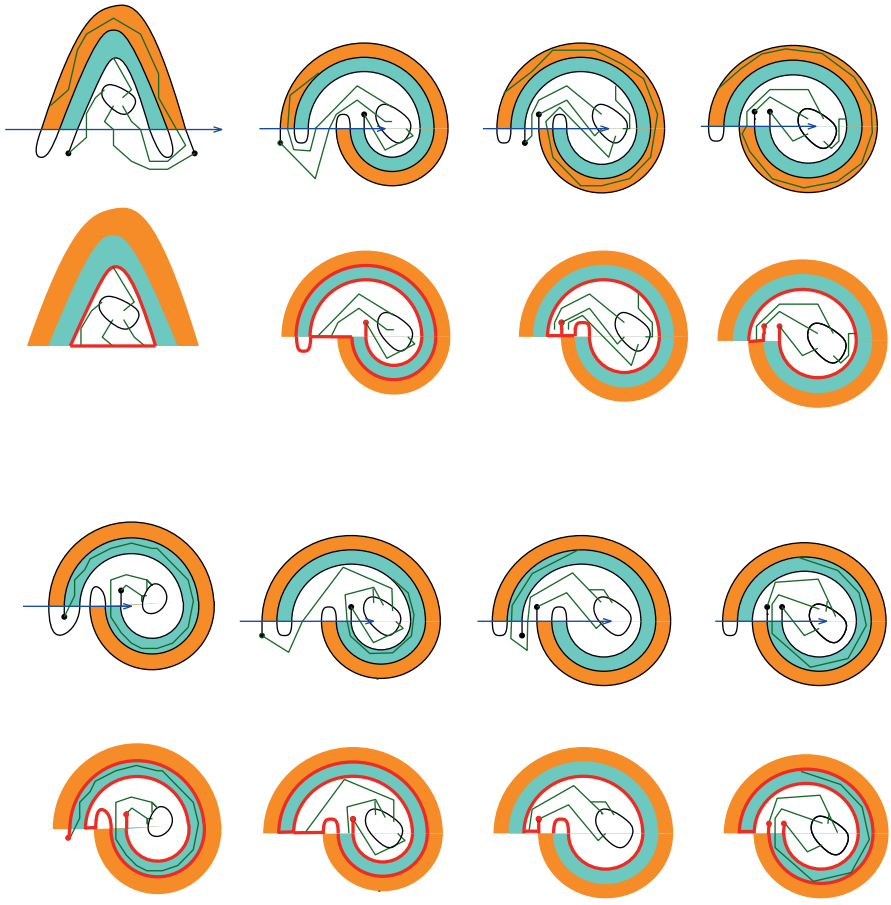

Figure 3.8: Case 1: The terminal face is in the region A. The black curve shows the nerve path P. The directed blue line is the path γ. In dark green lines, we depict the nerves attached to P. The face F_β is drawn as a black splinegon in the outer face of P^*. In the top figure, we consider the case where the nerves extend to the side of the orange bigon whereas in the bottom figure we consider the configurations where the nerves extend to the side of the turquoise bigon.

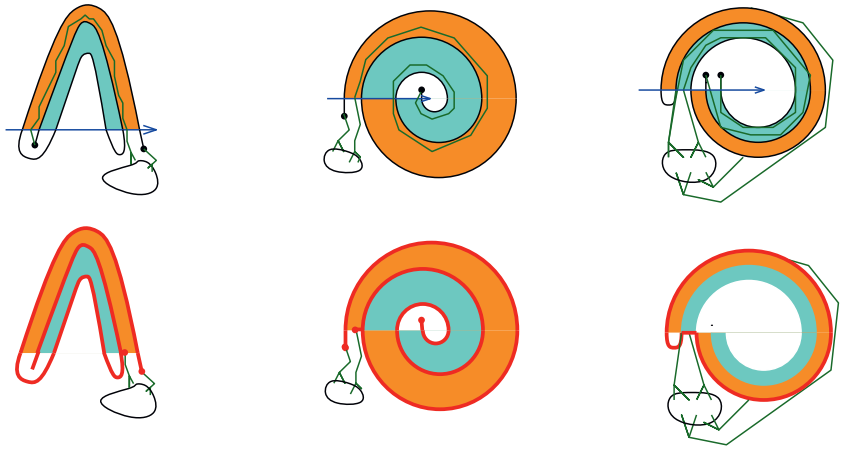

Figure 3.9: Case 2a: In the top row, we show configurations where F_β is contained in the outer face of P^*. Either N_1 or N_2 traverses one of the two bigons. The bottom row shows the replacement as bold red curves.

Case 2: F_β is contained in Region B.
We shall consider two subcases here:
Case 2a. *Either N_1 or N_2 traverses one of the two bigons.*
Suppose that N_1 traverses a bigon. Figure 3.9 depicts this case. Suppose that the leftmost crossing of N_1 with γ is x and the rightmost crossing is y. Then $\gamma[x, y]$ is at most as long as $N_1[x, y]$, allowing us to shortcut the nerve.
Case 2b. *Neither N_1 nor N_2 must traverse any bigon.*
Figure 3.10 depicts this case. The top figure shows the cases where the nerves extend toward the orange bigon whereas the bottom figure depicts the cases where the nerves extend towards the turquoise bigon.

Since in each of the cases we can replace P by a shorter path that has fewer crossings with K, we obtain a contradiction to the minimality of C^+. Therefore, no loop of L can be incident on two bigons.

Next, we deal with the monogons. Some monogons may have nerves attached to them that extend towards the nerve face without ever crossing γ. These are the monogons that we cannot replace by the subpath of γ bounding them on one side. However, in what follows, we argue that only $O(k)$ of these monogons actually exist in C^+.

We orient γ and P arbitrarily. Without loss of generality, assume that the nerves extend to the west of P. Consider the system of loops P^*. Let *good loops* be those that are present to the east (or west) of γ, possibly have nerves attached to them extending to the west (or east) of P, and border a monogon or enclose a bigon that it borders. We can remove the good loops from C^+

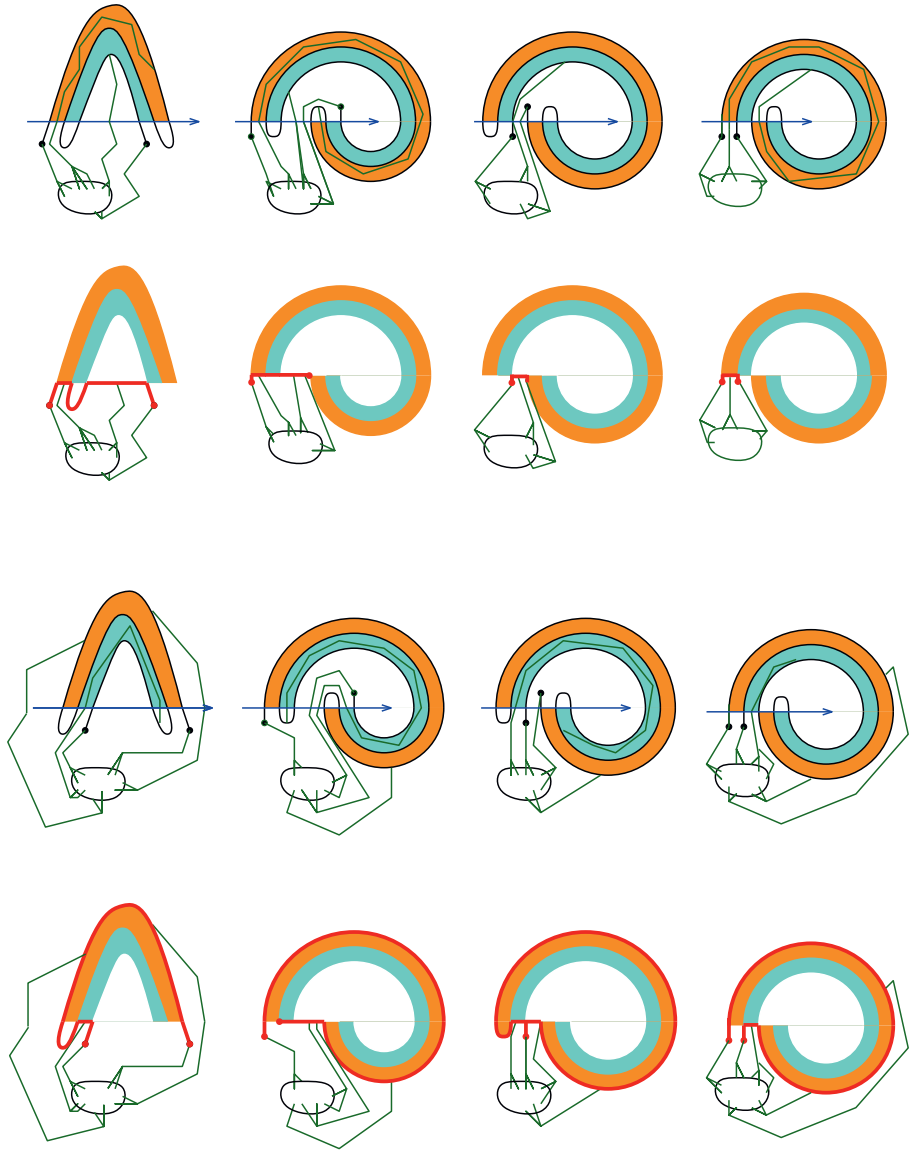

Figure 3.10: Case 2b: The top two rows depict the case when the nerves attached to P extend towards the orange bigon and the bottom two rows depict the case when they extend towards the turquoise bigon.

without contradicting its feasibility by shortcutting (see Figure 3.11). Loops that enclose any terminal face (the end point of a path of K) in their interior are called *obstacle loops*. Now consider loops that form monogons. *Bad monogons* are those that are present on the west (or east) of γ and contain nerves attached to them that extend towards the west (or east) of P. Figure 3.11 shows two bad monogons flanking a good loop. In what follows, we argue that between two obstacle loops, there can be at most four bad monogons.

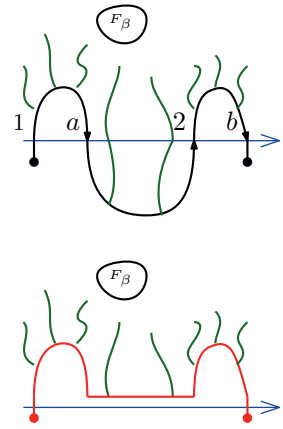

Figure 3.11: The blue line is the path γ and the arrow indicates its orientation. The black curve shows P. The dark green curves are the nerves attached to P. The first and third monogons are bad, and the middle loop is good. The figure at the bottom shows the shortcut of the good loop as a thick red curve.

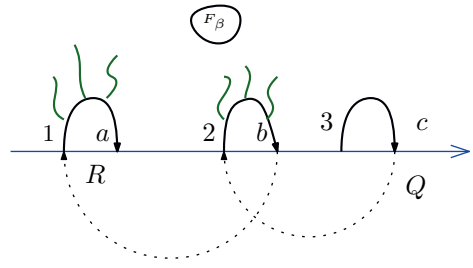

Figure 3.12: We assume that no two of the three consecutive bad monogons appear in the same order as on P.

The *direction* of a bad monogon depends on how its loop is oriented with respect to γ: either the head or the tail of the loop comes first on γ. Two bad monogons are *consecutive* if there exists no other bad monogon between them. Consider any five consecutive bad monogons on P that have no obstacle loop between them. Without loss of generality, we assume that three of them are

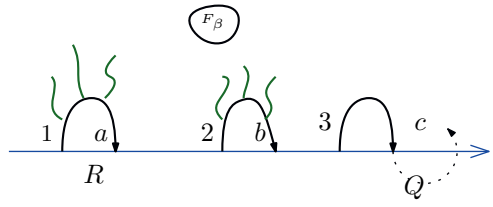

Figure 3.13: Case 1: Q crosses γ for the first time to the right of c.

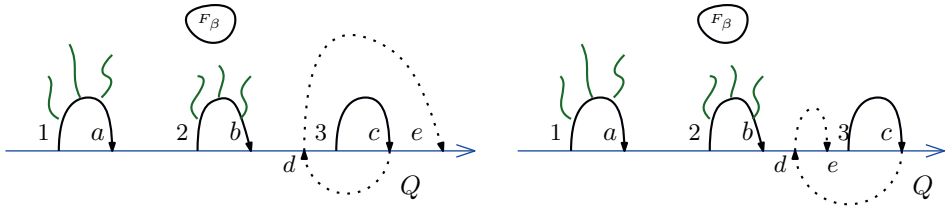

Figure 3.14: Case 2: Q crosses γ between b and 3 for the first time and to the right of the first crossing after that.

bad monogons in the same direction. If any two of these appear in the same order along γ as their order on P, then we can shortcut P as in Figure 3.11, because the subpath of P between them does not contain an obstacle loop and thus contains a good loop. So, we assume that no two of them appear in the same order. We denote the crossing points of these three monogons with γ by $1, a, 2, b, 3, c$ in the order along γ. Let Q be the subpath connecting c to 2 and R be the subpath of P connecting b to 1. Figure 3.12 illustrates the configuration.

We consider the following cases:

Case 1: Q crosses γ for the first time to the right of c
In this case, we can shortcut the loop of Q to the east of γ from c to the crossing. We can do so as this loop is good; indeed, it suffices to observe it is not an obstacle loop, as it is between two of the chosen bad monogons.
Case 2: Q crosses γ between b and 3 and crosses γ a second time to the right of the first crossing
As illustrated in Figure 3.14, we either reach the next monogon of the sequence (as in the right figure) or reach a point e to the right of c, and must cross γ again to reach 2. In the latter case, we follow the subpath of Q that starts at e and restart the case analysis from e onwards. Then, in either of the two scenarios, due to the planarity of C^+, the loop R from b to 1 must cross γ to the right of b to avoid crossing Q. Then, we get a good loop, and can shortcut P.
Case 3: Q crosses γ between b and 3 and again to the left of its

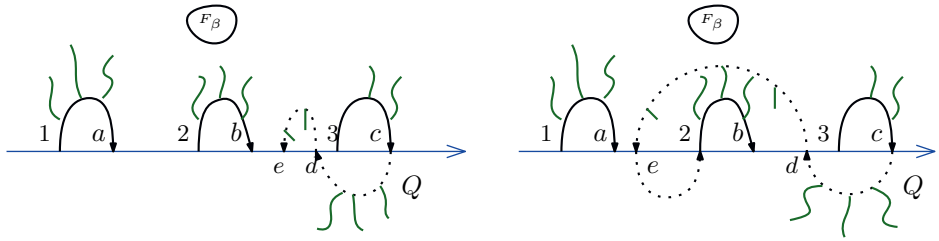

Figure 3.15: Case 3: Q crosses γ for the first time between b and 3 and then to the left of the first crossing.

crossing point
The second crossing could lie in between b and the first crossing of Q, or to the left of 2. In the former case, Q forms a good loop, and we can shortcut it by the piece of γ between the two crossings. In the latter case, there must not be any obstacle in the region enclosed by the loop 2-b and the one formed by Q, or else Q would contain an obstacle loop. Therefore, we get a good loop formed by the loop of Q between the two crossings. This is a contradiction to the loop between 2 and b being a bad monogon, and we can shortcut P.

Case 4: Q crosses γ the first time to the left of 2
Figure 3.16 illustrates all the configurations of Q. In A and C, we get a good loop and we can shortcut P by the piece $\gamma[d, e]$ in A and the piece of γ between the crossing to left of e and the right of d in case C. In configurations B and D, we treat e as the first crossing of Q with γ, and we land in the cases 2 or 3 based on where Q crosses γ after e.

We explicitly mention that these transformations hold good even if $\alpha = \beta$ and b is a bridge of the shrunken skeleton. So, we have argued that there are no good loops in C^+ and at most four consecutive bad monogons between two obstacle loops.

Now, we deal with the remaining bigons. Whenever we encounter a face of L that is a bigon, we remove one of the two loops incident on it, and iterate until no bigons remain. Therefore, in the proof of [42, Lemma 2.1], which essentially counts the number of obstacle loops, we can multiply the total number of crossings by 10 to get an upper bound on the total number of crossings of P with K. The number of obstacle loops remaining in L is at most $6\ell + 2g - 3$, where g is the genus of the surface on which C^+ is embedded and ℓ is the number of obstacles of the surface. We treat the obstacles we place on the dual vertices of the terminal faces as obstacles. Since there are k terminal faces, and therefore k "obstacles", there are k obstacles in all. Therefore $\ell = k$ and $g = 0$. This proves that the number of loops, and thus the degree of p/p is $O(k)$. □

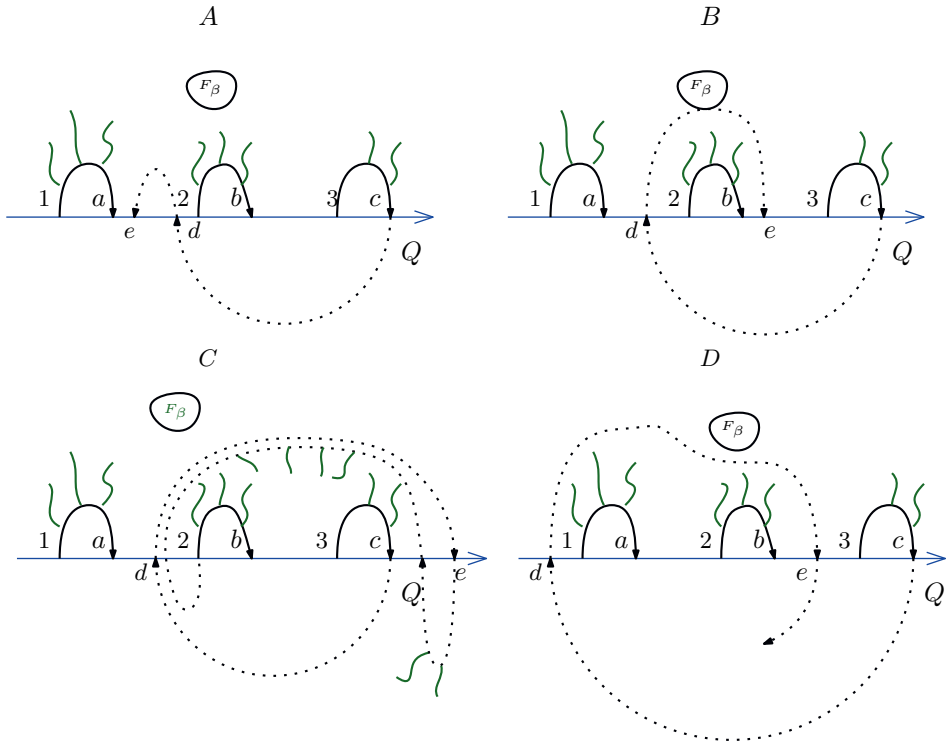

Figure 3.16: Case 4: First crossing of Q is to the left of 2.

3.6.4. Alternating Nerves

Let u, v be two endpoints of a shrunken bone b of the shrunken skeleton of C^+. Then b separates two terminal faces $F_\alpha, F_\beta \in \mathcal{F}$ (possibly $F_\alpha = F_\beta$ if b is a bridge of the shrunken skeleton). We denote by P the path corresponding to b in the skeleton of C^+. While the previous section builds sufficient understanding to build the path between consecutive nerves that extend west towards the same terminal face, this does not help if two consecutive attachment points have nerves towards different terminal faces, or towards the same terminal face, but one extends east and one extends west. To this end, we need more elaborate methods.

Definition 3.6.5. *Let x and y be consecutive attachment points on P (possibly $x = y$) such that x is the attachment point for a nerve that extends west towards F_α and y is the attachment point for a nerve that extends east towards F_β. Then we call these nerves an* alternating pair of nerves.

Let N_u denote the pair of alternating nerves closest to u along with the path between their attachment points. Let N_v be the corresponding pair closer

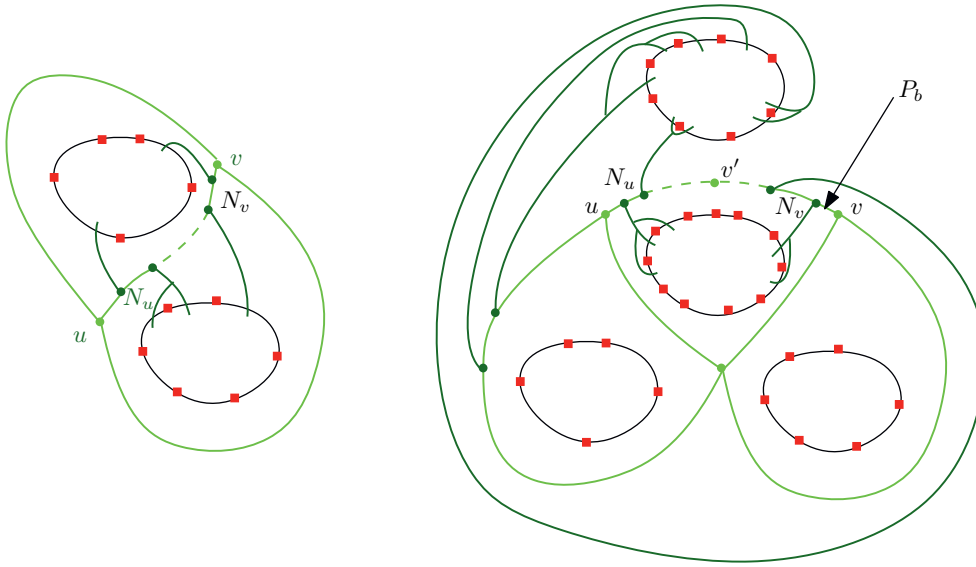

Figure 3.17: The figure shows the two cases when alternating pairs of nerves occur. The left case is handled by Lemma 3.6.7 while the right case is handled by Remark 3.6.6.

to v. Contracting the edges in N_u and N_v to form the vertices u^* and v^* respectively is equivalent to deleting the corresponding edges in G. This forms an isolated, connected region in the plane with pieces of the boundary of F_α and F_β bounding it on two sides. We denote this region by B.

Remark 3.6.6. *We call a pair of alternating nerves* bad *if the subpath of the skeleton between them lies in the region $\mathbb{R}^2 \setminus B$. One such case is shown in the right figure of Figure 3.17. To deal with* bad alternating nerves*, we look at the nerve directly to the left and right of the exposed terminal and add an "artificial" branching point between them. This splits the bone b into two and thus, N_u and N_v no longer form an alternating pair. We also note that each part of b after the split will be found by our algorithm.*

Lemma 3.6.7. *The augmented minimum multiway cut dual C^+ restricted to B comprises a minimum Steiner tree containing the vertices u^*, v^*, and the augmented terminals corresponding to terminals on the pieces of the boundary of F_α and F_β bounding B.*

Proof. The terminals embedded in the region B are not connected to any terminal outside. They lie on the boundary of the outer face of B. Also note that the augmented dual restricted to the region B is connected (because the restriction of C^+ to B is connected) and equal to the augmented dual of the subgraph of G restricted to B. By Corollary 3.5.12, we know that the

minimum multiway cut restricted to B forms a minimum Steiner tree in the augmented dual graph. The terminals of the minimum Steiner tree are the augmented terminals corresponding to the intervals on the segments of F_α and F_β bounding B, along with u^* and v^*. The vertices u^* and v^* serve as augmented vertices representing the intervals on the boundary of B connecting F_α and F_β. □

We now consider the structure of the bone for the parts that are not in the region B, or what they look like when there are no two alternating pairs of nerves. In the previous section, we already discussed what happens to nerve paths. Hence, we only need to consider paths between attachment points of alternating nerves and between an attachment point of a nerve and a branching point. The following result is immediate from [63, Lemma 5.2] with the substitution $g = 0$ and $t = k$.

Lemma 3.6.8. *Each path on a bone of C^+ between the attachment points of a pair of alternating nerves on a bone, as well as the one from a branching point to its closest attachment point, crosses any path of K $O(1)$ times.*

Finally, we argue that replacing certain subpaths of a bone by a homotopically equivalent path still yields an optimal solution. The proof of this lemma is reminiscent of the proof of [63, Lemma 7.2].

Lemma 3.6.9. *Let b be any shrunken bone of the shrunken skeleton of C^+ and let P be the path that forms the corresponding bone. Let x and y be two augmented dual vertices on P_b such that $P_b[x, y]$ contains no attachment points nor branching points, except possibly x or y.*

Then there exists a minimum multiway cut dual which contains a shortest path in the plane homotopic to $P[x, y]$.

Proof. Let P' be a shortest path with the same crossing sequence as $P = P_b[x, y]$. It is sufficient to prove that for any $t_1, t_2 \in T$, P' disconnects all t_1-t_2 paths that are disconnected by P and that are not disconnected by any other part of C^+.

Suppose that R is a t_1-t_2 path that is crossed by P but not by any other part of $C^+ \setminus P$. Let f_1 and f_2 be the faces of C^+ that enclose t_1 and t_2 respectively. Note that P lies in the intersection of the boundaries of f_1 and f_2. Also observe that R is fully contained in the region bounded by the union of f_1, and f_2. Moreover, since t_1 is in f_1 and t_2 is in f_2, R must cross P an odd number of times. Since P and P' have the same endpoints, the union of their drawings in the plane decomposes the plane into a set X of two or more regions. Since P and P' are homotopic and both are disjoint from the boundaries of F_α and F_β, neither t_1 nor t_2 lies in any bounded region of X.

Hence, if R enters a bounded region of X, then R must also exit it, and thus R intersects any bounded region of X an even number of times. However, by the preceding argument, R crosses P an odd number of times. Hence, R must cross P'. □

3.7. Towards a $n^{O(k)}$-time Algorithm

With the structure of the optimum solution in place, we are now ready to develop the algorithm. Our guiding light will be the topology of the optimum solution, which consists of its skeleton and a compact description of its bones and is defined more formally below. As we do not know any optimum solution, we enumerate all possible topologies and argue that we find a feasible, minimum-weight solution for the topology that corresponds to an optimum solution. The algorithm to do this consists of two parts. First, we show that each bridgeless component of the skeleton in the topology can be treated separately. Second, for each bridgeless component of the skeleton, we build a sphere-cut decomposition and perform dynamic programming on the decomposition that finds a solution according to the topology.

3.7.1. Topology

We first define a topology and then show that all topologies can be efficiently enumerated.

Definition 3.7.1. *A* topology *consists of a triple* (S, s, h)*:*

- *a connected plane multigraph* S *(possibly containing parallel edges but no self-loops) with* k *faces. This is the (shrunken)* skeleton *of the topology. We direct the edges of the skeleton in an arbitrary fashion to obtain a* directed skeleton*;*
- *for each face of* S*, a unique corresponding face in* $\mathcal{F}$*;*
- *for each shrunken bone* b*, separating faces* f_α, f_β *of the skeleton (possibly* $\alpha = \beta$*), a structural description, describing:*
  - *a (possibly empty) ordered multisubset* $s(b)$ *of* $\{\alpha, \alpha, \beta, \beta\}$ *such that* $s(b)[i]$ *and* $s(b)[i+1]$ *are not equal for any* $1 \leq i < |s(b)|$*, except possibly for* $i = 2$ *when* $|s(b)| = 4$*;*
  - *for any* $1 \leq i \leq 2|s(b)| + 1$*,* $h(b)[i]$ *is a homotopy string of length* $O(k)$*. When* $|s(b)| = 4$*, then* $h(b)[5]$ *is unspecified.*

If $\alpha = \beta$, then we use α or β to denote east and west respectively in $s(b)$.

The intuition behind the definition of a topology is as follows. The directed skeleton guides the overall structure of the solution that will be found by our algorithm. Ideally, it is equivalent to the shrunken skeleton of an optimum solution. By Lemma 3.5.4, it has k faces and is connected. Moreover, each face of the shrunken skeleton corresponds to a unique face in $\mathcal{F}$. The orientation of the skeleton will prove useful in later definitions and algorithms.

The multiset $s(b)$ describes the direction of maximal sets of consecutive nerves along the bone that have the same direction. By this intuition, it makes sense to force the directions to be alternating, as in the definition. We also note that we do not need to specify more than four such sets, because when we have four sets, we obtain two alternating pairs of nerves and can apply Lemma 3.6.7 to the region between them.

For each set of consecutive nerves as described by $s(b)$, we use $h(b)$ to denote the homotopy of the (up to four) subpaths of the bone to which the nerves attach. We also use $h(b)$ to describe the homotopy of the (up to five) subpaths of the bone between these sets of nerves. Following Lemma 3.6.3, each of those homotopy strings has length $O(k)$. When $|s(b)| = 4$, we do not (need to) specify the homotopy of the middle subpaths of the bone, as we handle this part as in Lemma 3.6.7.

Recall that C is the optimal solution and C^+ its set of corresponding augmented dual edges. We call a topology (S, s, h) *optimal* if the shrunken skeleton S^+ of C^+ is equivalent to (the underlying graph of) S, each face of $\mathcal{F}$ is assigned to the same face in S^+ and S, and for each bone b of the shrunken skeleton, directed in an arbitrary fashion:

- there are $|s(b)|$ nonempty maximal sets of nerves, where nerves in the i-th set are towards $F_{s(b)[i]}$, and their attachment points are consecutive and uninterrupted on b; the nerves preceding a nerve from the i-th set belong to the i-th set or the $i-1$-th set, or if $|s(b)| = 4$ and $i = 3$, can be arbitrary nerves to F_α or F_β, until a nerve of the $i-1$-th set is hit;
- the part of the bone between the nerves at the ends of the i-th set of nerves has homotopy string $h(b)[i]$;
- the part of the bone between the i-th and $i+1$-th sets of nerves, for $0 \leq i \leq |s(b)|$ where we pretend the 0-th nerve is the one end of the bone and the $|s(b)|+1$-th nerve is the other end of the bone, has homotopy string $h(b)[2i+1]$. When $|s(b)| = 4$, we do not have to satisfy this for $i = 2$.

Note that since we fixed C, there is a unique optimal topology, which effectively describes C. We may thus speak of 'the' optimal topology.

Our goal is to enumerate all topologies. Then for each topology, we aim to find a solution that corresponds to this topology. By considering all topologies, we ensure that we consider the optimal topology at some point. Then we argue that we find a multiway cut of minimum weight. Before proceeding with that algorithm, we bound the number of topologies and show how to enumerate them. We require the following auxiliary lemma.

Lemma 3.7.2. *There are $2^{O(k \log k)}$ connected plane multigraphs with $O(k)$ vertices, k faces and without self-loops. Moreover, they can be enumerated in the same time.*

Proof. A trivial bound shows that there are $2^{O(p \log p)}$ labeled simple planar graphs on p vertices (although more precise bounds are known [100]). We can extend this to planar multigraphs by guessing the subset of edges of the simple planar graph that are parallel and partitioning the number of parallel edges over them. Note that a planar multigraph with $O(k)$ vertices and k faces must have $O(k)$ edges. Since there are $2^{O(q)}$ partitions of the integer q, there are $2^{O(k \log k)}$ possible planar multigraphs with $p = O(k)$ vertices and k faces and without self-loops. All such planar graphs can be trivially enumerated in $2^{O(k \log k)}$ time by enumerating all graphs and testing each for planarity [114].

We now consider embeddings. Recall that two embeddings of a planar graph are equivalent if the ordering of the edges around each of the vertices is the same. In our case, there are $2^{O(k \log k)}$ different such orderings. Hence, there are $2^{O(k \log k)}$ plane multigraphs with $O(k)$ vertices and k faces and without self-loops. Since each such planar graph can be enumerated in $2^{O(k \log k)}$ time and embeddings can be enumerated in time linear in their number [37], the bound of $2^{O(k \log k)}$ follows. □

Lemma 3.7.3. *There are $2^{O(k^2 \log k)}$ different topologies. Moreover, they can be enumerated in the same time.*

Proof. By the analysis of Lemma 3.5.9, the skeleton has k faces, $O(k)$ vertices, and $O(k)$ edges. By Lemma 3.7.2, there are $2^{O(k \log k)}$ different skeletons for a topology. For each bone b of a skeleton, there are $O(1)$ choices for $s(b)$. Moreover, each homotopy string has length $O(k)$ and there are at most 8 of them for every bone, meaning length $O(k)$ for each bone and $O(k^2)$ in total. Each entry of a homotopy string has $O(k)$ possible values by the definition of a cut graph. Hence, there are $2^{O(k^2 \log k)}$ topologies.

We note that by Lemma 3.7.2, all skeletons can be enumerated in $2^{O(k \log k)}$ time. It is immediate from the preceding that all topologies can be enumerated in $2^{O(k^2 \log k)}$ time. □

3.7.2. Broken Bones and Splints

We now become more formal about our interpretation of topologies. Let (S, s, h) be a topology. We define the following notion, which states how we interpret s and h for a shrunken bone b.

Definition 3.7.4. *Let (S, s, h) be a topology. Let b be a shrunken bone (edge) of the shrunken skeleton S, separating the faces f_α and f_β, and directed from u to v. Let $s(b) = \{\gamma_1, \ldots, \gamma_{|s(b)|}\}$, where each $\gamma_j \in \{\alpha, \beta\}$. Let F_α and F_β be the terminal faces enclosed by the faces f_α and f_β, respectively. A* broken bone *is a tuple that consists of intervals I_α and I_β of augmented terminals lying on F_α and F_β respectively, augmented dual vertices $x_1^+, \ldots, x_{|s(b)|+1}^+$ and $y_0^+, \ldots, y_{|s(b)|}^+$ that are not augmented dual terminals, and nerves $N_1, \ldots, N_{2|s(b)|}$ towards F_α and F_β, where:*

- *for each $1 \leq i \leq |s(b)|$, nerves N_{2j-1} and N_{2j} extend towards γ_j and have root x_j^+ and y_j^+ respectively. These nerves might be the same, but are non-empty if I_j is non-empty.*
- *the interval of a nerve among $N_1, \ldots, N_{2|s(b)|}$ towards F_α (resp. F_β) is a subinterval of I_α (resp. I_β). Moreover, these subintervals appear in the order indicated by their indices on I_α and I_β (but do not necessarily cover I_α and I_β completely);*
- *if I_α is non-empty, then a prefix of I_α is the interval of a nerve among $N_1, \ldots, N_{2|s(b)|}$. The same holds for a suffix of I_α (possibly, this is the same nerve). The same holds with respect to I_β if it contains at least two augmented terminals.*

A splint, *fixing a given broken bone, is a subset D^+ of the edges of the augmented dual graph G^+ with the following properties:*

- *for each terminal t between I_α (or between I_β), there is a unique bounded face of D^* that encloses t, where D^* is the set of dual edges corresponding to the edges of D^+.*
- *there is a path P_b in D^+ between y_0^+ and $x_{|s(b)|+1}^+$. These two augmented dual vertices are called the* ends *of the splint.*
- *$x_1^+, \ldots, x_{|s(b)|+1}^+$ and $y_0^+, \ldots, y_{|s(b)|}^+$ are in D^+ and appear in the order y_0^+, x_1^+, y_1^+, x_2^+, y_2^+, $\ldots$, $x_{|s(b)|+1}^+$ on P_b.*
- *for each $0 \leq i \leq |s(b)|$ (except $i = 2$ when $|s(b)| = 4$), the (possibly empty) subpath of P_b from y_j^+ to x_{i+1}^+ consists only of vertices that have degree 2 in D^+ and has homotopy string equal to $h(b)[2j + 1]$.*

- *for each* $1 \leq i \leq |s(b)|$*, the only vertices on the subpath of* P_b *between* x_j^+ *and* y_j^+ *that have degree more than 2 in* D^+ *are the attachment points of nerves towards* F_{γ_j}*; these nerves include* N_{2j-1} *and* N_{2j}*. The subpath has homotopy string equal to* $h(b)[2j]$*.*

- *if* $|s(b)| = 4$*, then for the vertices on the subpath of* P_b *between* y_2^+ *and* x_3^+*, the only vertices that have degree more than 2 in* D^+ *are the attachment points of nerves towards* F_α *or* F_β*.*

- *the intervals of all the aforementioned nerves jointly partition* I_α *and* I_β *with the exception that for each* $1 \leq i \leq |s(b)|$*, the nerves* N_{2j-1} *and* N_{2j} *are not necessarily distinct (but are distinct from all other nerves).*

We refer to Figure 3.18 for an illustration of a splint and its broken bone.

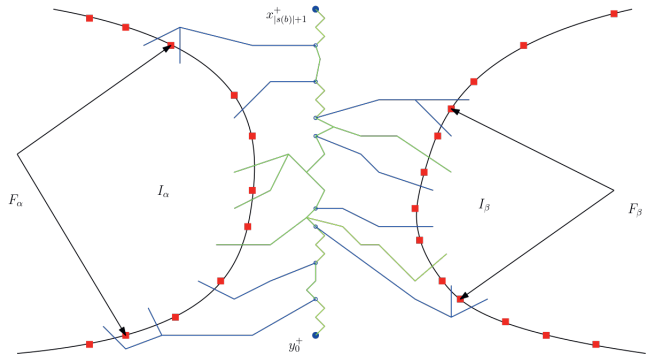

Figure 3.18: In this example, $s(b) = \{\alpha, \beta, \beta, \alpha\}$. The blue lines depict the "guessed" nerves attached to the broken bone. In green, we depict the splint that is used to fix the broken bone. The arrows point to the first and last terminals of the intervals I_α and I_β.

Note that in the definition of a broken bone, it is not strictly necessary to specify $x_1^+, \ldots, x_{|s(b)|}^+$ and $y_1^+, \ldots, y_{|s(b)|}^+$, as their definition is implied by $N_1, \ldots, N_{2|s(b)|}$. Similarly, we do not actually need all the nerves we have specified when $|s(b)| = 4$. We still write this to streamline and simplify the definition (which admittedly is already quite complex).

We now define the notion of an optimal broken bone. We refer back to the definition of an optimal topology to recall the $|s(b)|$ maximal sets of nerves defined there. For the optimal solution C, the optimal topology, and a shrunken bone b, we call a broken bone $x_1^+, \ldots, x_{|s(b)|+1}^+$, $y_0^+, \ldots, y_{|s(b)|}^+$, $N_1, \ldots, N_{2|s(b)|}$, I_α, I_β *optimal* if for the bone of C^+ corresponding to b, its ends are y_0^+ and $x_{|s(b)|+1}^+$ in the direction of the directed skeleton, the i-th set of nerves has the nerves N_{2i-1} and N_{2i} at the ends of the set (where N_{2i-1} and N_{2i} are possibly

equal), and the intervals I_α and I_β correspond to the union of the intervals of all nerves that have their attachment point on the bone.

The latter may lead to a nerve that attaches to a branching point and its corresponding interval to be assigned to two broken bones (for the two bones that meet at the branching point and whose shrunken bones bound the same face of the skeleton). In that case, we break ties arbitrarily, but in such a way that all nerves that attach to this branching point are assigned to the same bone and no terminal face is enclosed by the union of any two nerves assigned to the same bone in this way. This ensures that each augmented terminal is in some interval of some optimal broken bone and moreover, no terminal face and its broken bones 'interrupt' the sequence of nerves of the bone.

Finally, we call a splint *optimal* if it fixes an optimal broken bone and all its nerves are the nerves of C^+ that were defined to belong to the bone. Again, an optimal broken bone and splint are unique with respect to C, so we may speak of 'the' optimal broken bone and splint of a bone.

For a given topology and a particular shrunken bone of the topology, our goal is to enumerate all broken bones for this shrunken bone and find a minimum-weight splint for it. By enumerating all broken bones, we ensure that we consider the optimal broken bone. Then the minimum-weight splint that we find will have the same weight and structure of the optimal splint. We discuss that algorithm in a moment, but first argue that we can enumerate all possible broken bones efficiently.

Proposition 3.7.5. *There are $O(n^{26})$ distinct broken bones fixed by a splint.*

Proof. Using Euler's formula, G^+ has $O(n)$ augmented dual vertices. We note that any interval can be specified by two augmented dual vertices and (thus) any nerve by three augmented dual vertices. Since $|s(b)| \leq 4$, there are at most 8 nerves and two endpoints per broken bone. It follows that there are $O(n^{26})$ distinct broken bones. □

3.7.3. Splinting Algorithm

We now describe an algorithm that, given a topology (S, s, h) and a broken bone $x_1^+, \ldots, x_{|s(b)|+1}^+$, $y_0^+, \ldots, y_{|s(b)|}^+, N_1, \ldots, N_{2|s(b)|}, I_\alpha, I_\beta$ for a bone b of S, finds a splint of minimum weight.

For any $0 \leq i \leq |s(b)|$ (except $i = 2$ when $|s(b)| = 4$), to find the path from y_i^+ to x_{i+1}^+, we invoke the algorithm by Frank and Schrijver [88, Section 5] to find the shortest path with the homotopy string $h(b)[2i + 1]$. These paths do not contain any attachment points of nerves.

The following algorithm computes the weight of the part of a minimum weight splint between the vertices x_i^+ and y_i^+ of the augmented dual, for all

$1 \leq i \leq |s(b)|$. Note that this is a nerve path. For simplicity, we consider only $i = 1$, as the other cases are similar. Without loss of generality, $s(b)[1] = \alpha$. We find the part of the splint starting at x_1^+, with its corresponding nerve N_1, and ending at y_1^+, with its corresponding nerve N_2. Also, we assume that N_1 spans the interval $\{{}^{\alpha}t_{\ell}^+, \ldots, {}^{\alpha}t_{\ell'}^+\}$ and N_2 spans the interval $\{{}^{\alpha}t_{j}^+, \ldots, {}^{\alpha}t_{j'}^+\}$. Then the interval covered by all nerves attaching to this nerve path is $I_1 = \{{}^{\alpha}t_{\ell}, \ldots, {}^{\alpha}t_{j'}\}$.

We compute this part of the splint as follows. By $c[x^+, a, a']$, we denote the weight of the unique nerve on $x^+ \in V(G^+)$ and the interval $\{{}^{\alpha}t_{a}^+, \ldots, {}^{\alpha}t_{a'}^+\}$, which can be computed by Lemma 3.5.17. By $c'[x^+, a, a', e]$ we denote the weight of a partial splint passing through the vertex $x^+ \in V(G^+)$, that encloses every terminal between the augmented terminals $\{{}^{\alpha}t_{\ell}^+, \ldots, {}^{\alpha}t_{a'}^+\}$, contains the unique nerve (x^+, a, a'), and the partial nerve path of the homotopy given by the prefix of $h(b)[2]$ of length e.

The dynamic programming algorithm is given in Algorithm 1 below. We use $d_{h(b)[2](e',e]}(x'^+, x^+)$ to denote the length of a shortest path from x'^+ to x^+ with homotopy string equal to the substring of $h(b)[2]$ between indices e' and e (not including the symbol on index e'). In other words, this substring is equal to the prefix of length e minus the prefix of length e'. This shortest path length can again be computed by the algorithm by Frank and Schrijver [88, Section 5].

We also use $\overset{*}{\min}$ to denote that the minimum is only allowed over certain combinations. In particular, in Line 8, the nerve (x^+, a, a') being considered in combination with nerve $(x'^+, z, a-1)$ and the path between x'^+ and x^+ of homotopy string $h(b)[2](e', e]$ must define a region that only encloses ${}^{\alpha}t_{a-1}$ (the terminal inbetween the two nerves). Moreover, the nerve (x^+, a, a') itself must create a region for every terminal in $\{{}^{\alpha}t_{a}, \ldots, {}^{\alpha}t_{a'-1}\}$. A similar constraint holds in Line 13. Furthermore, if x^+ lies on a path γ of the graph K, then in Line 8, we find the minimum over values of $e' < \gamma$ and enforce the next index on the homotopy string $h(b)[2]$ to be γ.

The same algorithm is used to compute all the other parts of the splints, too. Note that even though the algorithm only computes an optimal value, it can be easily modified to return the optimal solution (nerves and nerve path).

Finally, if $|s(b)| = 4$, then we know from Lemma 3.6.7 that embedded in the region bounded by the nerves attached to y_1^+ and x_2^+, and y_3^+ and x_4^+, along with the paths between them, as well as the segments of F_α and F_β bounding the region on either side is a minimum Steiner tree with its terminals on the boundary of the region. Using the algorithm of Erickson *et al.* [79] (see also Bern [16]), we find the minimum Steiner tree. This finishes the description of the splinting algorithm.

Lemma 3.7.6. *Given a topology (S, s, h), a bone b of S, and the optimal*

$c'[x_1^+, \ell, \ell', 0] = c[x_1^+, \ell, \ell']$;
$c'[x^+, a, a', e] = \infty$ for all $x^+ \neq x_1^+$, $e \neq 0$, $a \neq \ell$, or $a' \neq \ell'$;
for $\ell' < a' < j$ **do**
  for $\ell' < a \leq a'$ **do**
    for $x^+ \in V(G^+)$ **do**
      for $0 \leq e \leq |h(b)[2]|$ **do**

$$c'[x^+, a, a', e] = \overset{*}{\min_{\substack{\ell \leq z < a \\ x'^+ \in V(G^+) \\ 0 \leq e' \leq e}}} \left\{ c'[x'^+, z, a-1, e'] + c[x^+, a, a'] + d_{h(b)[2](e',e]}(x'^+, x^+) \right\}$$

      end
    end
  end
end
return

$$\overset{*}{\min_{\substack{\ell \leq z < j \\ x^+ \in V(G^+) \\ 0 \leq e \leq |h(b)[2]|}}} \left\{ c'[x^+, z, j-1, e] + d_{h(b)[2](e,|h(b)[2]|]}(x^+, y_1^+) + c[y_1^+, j, j'] \right\}$$

Algorithm 1: Splinting Algorithm

broken bone for b, we can fix the broken bone by a minimum-weight splint found through the splinting algorithm. Moreover, the splinting algorithm runs in time $n^{O(1)}$.

Proof. The correctness and optimality of finding the shortest path with the homotopy strings $h(b)[2i+1]$ between y_i^+ and x_{i+1}^+ for $0 \leq i \leq |s(b)|$ (except $i = 2$ when $|s(b)| = 4$) follows from Lemma 3.6.9. Moreover, if $|s(b)| = 4$, the correctness of finding a minimum Steiner tree between the alternating nerves follows from Lemma 3.6.7.

Then, we show that the algorithm finds a feasible splint. This is immediate by the definition of $\overset{*}{\min}$, which ensures that the nerves cover all terminals between their intervals as well as inbetween the nerves.

Next, we argue that the algorithm finds an optimal splint for the optimal broken bone. We again only consider the part of the bone between x_i^+ and y_i^+ for $i = 1$ and $s(b)[1] = \alpha$; the other cases are similar. Let (x^+, a, a') be any nerve that is part of C^+ and where x^+ lies on P_b between x_1^+ and y_1^+. From Lemma 3.5.14 (see also Figure 3.5), it follows that it creates a set of regions that each enclose only a single terminal, each between the interval of the nerve. Consider the prefix of the homotopy string of the subpath of P_b between x_1^+ and x^+ and let it have length e. If $N_1 = (x^+, a, a')$ and (thus) $e = 0$, then c' contains the optimum weight of a partial splint for x^+, a, a', e, namely the weight of N_1. Otherwise, let $(x'^+, z, a - 1)$ be the nerve preceding (x^+, a, a') on P_b and let e' be the length of the homotopy string of the subpath of P_b from x_1^+ to x'^+. Then the subpath of P_b from x'^+ to x^+ has homotopy string $h(b)[2](e', e]$. By Lemma 3.6.9, replacing this subpath by any other path with the same homotopy string, still yields an optimal solution. Hence, the algorithm will consider and allow z, x'^+, e' in the minimization for x^+, a, a', e in Line 8 or 13. Using induction on $x'^+, z, a - 1, e'$, it follows that c' contains the optimum weight of a partial splint for x^+, a, a', e. Moreover, the algorithm returns the value of a minimum-weight splint.

It remains to argue the running time. Finding shortest paths with a specified homotopy using Frank and Schrijver's [88] algorithm takes $n^{O(1)}$ time. Finding all nerves takes polynomial time through Lemma 3.5.17. Finding a minimum Steiner tree in a planar graph, when all the terminals appear on the boundary of a single face, takes another $n^{O(1)}$ time using the algorithm of Erickson *et al.* [79] (see also Bern [16]). Finally, we loop over all $O(n^5 \cdot k^2)$ choices for $x^+, a, a', e, x'^+, z, e'$ to compute c' in Line 8 of the algorithm and $O(n^2 \cdot k)$ choices in Line 13. Therefore, our algorithm runs in $n^{O(1)}$ time. □

Using the algorithm, we can compute a splint for each broken bone of each topology. Since the homotopy of each nerve path and each path between branching points and the starts/ends of nerve paths are maintained, it follows that we can replace each bone and attached nerves of the optimum solution by the minimum-weight splint computed by the algorithm for the corresponding optimal broken bone. In particular, the homotopy strings ensure that the number of crossings of the bones with the cut graph K stays minimum.

Definition 3.7.7. *We call an optimum solution* splinted *if it has the property that each of its bones and attached nerves are splints.*

Remark 3.7.8. *From now on, we assume that the optimum solution is splinted.*

For the sake of intuition, we note that we have now gathered sufficient ideas to prove a $2^{O(k^2 \log k)} n^{O(k)}$ time algorithm for Planar Multiway Cut.

In particular, we can enumerate all topologies in $2^{O(k^2 \log k)}$ time and then enumerate all broken bones for any shrunken bone in $n^{O(k)}$ time total. For the combination of the optimal topology and the optimal broken bones for each shrunken bone, the splinting algorithm then delivers an optimal solution for each shrunken bone, which can be combined to form an optimal solution to the whole (modulo some details). In the next section, we argue how to reduce the $n^{O(k)}$ factor down to $n^{O(\sqrt{k})}$.

3.8. Algorithm Using Sphere-Cut Decomposition

We are now ready to discuss how we go from a topology to a multiway cut. If the topology is optimal, we argue that we find a minimum-weight multiway cut that has the same structure as the optimal multiway cut C. Our aim is to employ Theorem 3.3.2 on the shrunken skeleton to get a small branch decomposition and then apply a dynamic program. However, as already noted a few times, the shrunken skeleton might have bridges and Theorem 3.3.2 cannot be applied directly, but only on the bridge blocks of the shrunken skeleton. Therefore, we first develop a dynamic program that combines solutions of the bridge blocks of the shrunken skeleton, effectively reducing the problem.

3.8.1. Reduction to Bridge Blocks

When we want to combine solutions of different bridge blocks, it is natural to use the bridge block tree in a dynamic program. However, if we do this naively, we immediately run into the issue that in order to compute a solution for a non-trivial bridge block, we need to know the solutions for all bridge blocks contained in its bounded faces. This can be resolved by enforcing an ordering on the computation of the bridge blocks that depends on the embedding. To that end, we proposed the embedding-aware bridge block (eabb) tree in Section 3.3.2. We now show how to use it.

Let (S, s, h) be a topology. Let $L = L(S)$ be the eabb tree for S and let $\mathcal{B}$ denote the set of bridge blocks of S. For a BB-node l of L, let $\mathcal{B}(l)$ denote the bridge block corresponding to l. Extending this notation, for a subtree L' of L, we use $\mathcal{B}(L')$ to denote the set of all bridge blocks corresponding to BB-nodes in L'. For a node l of L, we use L_l to denote the subtree of L rooted at l. In particular, $L_{\ell(L)} = L$, where we recall that $\ell(L)$ is the root of L.

Lemma 3.8.1. *For any node l of L, $\mathcal{B}(L_l)$ is an internal set of S.*

Proof. By Lemma 3.3.5, if there is a bridge block B and a bridge block $B' \in \mathcal{B}(L_l)$ such that $B \prec_P B'$, then $B \in \mathcal{B}(L_l)$. Hence, any bridge block not

in $\mathcal{B}(L_l)$ is in the outer face of each of the blocks in $\mathcal{B}(L_l)$. Hence, there is a single face of $\mathcal{B} - \mathcal{B}(L_l)$ that encloses $\mathcal{B}(L_l)$ and there is a single face of $\mathcal{B}(L_l)$ that encloses $\mathcal{B} - \mathcal{B}(L_l)$. It follows that $\mathcal{B}(L_l)$ is an internal set. □

We now perform a bottom-up dynamic programming with respect to L. The crux here is to understand the relation that a child l has with its parent. This is formed by two parts. The first and easier part is the cut vertex shared by neighboring bridge blocks. The second, more complicated part, is the terminal face F_l in the middle region induced by the internal set $\mathcal{B}(L_l)$. The terminals in T_l are covered jointly by $\mathcal{B}(L_l)$, the blocks induced by siblings of l in L, and by the block induced by l's parent. Using a similar argument as in the proof of Lemma 3.5.14, we can see that each of these is responsible for a single interval of T_l. To help in the computations, we additionally consider the first nerves that cover the prefix and suffix of this interval. We now expand on this intuition of the dynamic program and define the table more formally.

Let l be a node of L. Define $w = w(l)$ as follows: if l is a C-node, then let w be the corresponding cut vertex; if l is a BB-node and l has a parent in L, then this parent is a C-node and we let w be its corresponding cut vertex; otherwise, let w be any vertex of $\mathcal{B}(l)$. Let $F_l \in \mathcal{F}$ denote the unique terminal face in the middle region of $\mathcal{B}(L_l)$.

Definition 3.8.2. *Given a vertex w^+ of the augmented dual, an interval I_l of F_l, and two (possibly empty) nerves N_l^1, N_l^2 towards F_l, a* stretcher *is a set D^+ of augmented dual edges such that:*

(i) *the interval of N_l^1 is a prefix of I_l and the interval of N_l^2 is a suffix of I_l. N_l^1 and N_l^2 are either both empty or both non-empty and cannot be empty if there is at least one terminal between I_l.*

(ii) *there is a (possibly empty) set of nerves in D^+ towards F_l whose augmented terminal sets are intervals that jointly partition I_l. N_l^1 and N_l^2 are among those nerves;*

(iii) *for any terminal $t \in T_l$ between I_l, there is a bounded face of D^* that encloses only t and no other terminals. Here D^* is the set of dual edges corresponding to the augmented dual edges in D^+;*

(iv) *for any terminal $t \in T_\alpha$ of a bounded face f_α in the subgraph of S induced by the bridge blocks of L_l, there is a bounded face of D^* that encloses only t and no other terminals;*

(v) *D^+ contains a splint for each bone in $\mathcal{B}(L_l)$ and N_l^1 and N_l^2 are included in these splints;*

(vi) w^+ *is the vertex of* D^+ *that is an end of all splints for the bones in* $\mathcal{B}(L_l)$ *that are incident on* w.

We call w^+, I_l, N_l^1, *and* N_l^2 *a* binder *of the stretcher and a* binder *for* l. *We say a binder is* valid *if it adheres to* (i).

Proposition 3.8.3. *For each node* l *of* L, *there are* $n^{O(1)}$ *binders. These can be enumerated in the same time.*

Proof. It suffices to observe that a binder has one augmented dual vertex, an interval of a terminal face, and two nerves, each of which can be described by a constant number of vertices of the augmented dual. □

Consider any vertex l of the embedding-aware block-cut tree L of the shrunken skeleton of an optimum solution C^+. Let I_l be the union of intervals of the nerves in C^+ towards F_l that attach to a block in $\mathcal{B}(L_l)$ and let N_l^1 and N_l^2 be the nerves whose intervals are a prefix and suffix of I_l respectively. Following Remark 3.7.8, C^+ is a union of splints for each of the shrunken bones of S. Let w^+ be the vertex of C^+ that is an end of all splints for the bones in $\mathcal{B}(L_l)$ that are incident on $w = w(l)$. Then we call w^+, I_l, N_l^1, and N_l^2 an *optimal binder*. Note that for each optimal binder, a stretcher does exist, which we call an *optimal stretcher*. We may speak of 'the' optimal binder and stretcher, because the minimum-weight solution and topology are uniquely defined.

Then for each binder I_l, N_l^1, N_l^2, w^+, define $A_l[I_l, N_l^1, N_l^2, w^+]$ as a minimum-weight stretcher for this binder. If no such stretcher exists, then we define $A_l[I_l, N_l^1, N_l^2, w^+]$ to be the set of all augmented dual edges. We argue that the table A can be computed in a dynamic programming fashion.

In the next section, we prove the following.

Lemma 3.8.4. *For any BB-node* l *of* L *and the optimal binder* B *for* l, *we can compute a minimum-weight stretcher for* l *when given minimum-weight stretchers for all optimal binders of all children of* l. *Moreover, it can be computed in* $n^{O(\sqrt{|\mathcal{B}(l)|})}$ *time.*

We now describe how to compute a table entry for I_l and w^+ if l is a C-node. Let $l_1, \ldots, l_q$ denote the children of l in L. Note that the children of l are all BB-nodes. We assume that the bridge blocks appear in this order around w in S. Then we compute $A_l[I_l, N_l^1, N_l^2, w^+]$ as the minimum-weight union of $A_{l_j}[I_{l_j}, N_{l_j}^1, N_{l_j}^2, w^+]$ over all families $I_{l_1}, \ldots, I_{l_q}$ of (possibly empty) intervals of F_l that form a partition of I_l and appear in this order on I_l and over all nerves $N_{l_j}^1, N_{l_j}^2$ whose interval is a prefix respectively suffix of I_{l_j}. During this computation, we discard any union that does not form a stretcher. If all

unions are discarded in this way, we set $A_l[I_l, N_l^1, N_l^2, w^+]$ equal to the set of all augmented dual edges.

Lemma 3.8.5. *For any C-node l of L and the optimal binder B for l, we can compute a minimum-weight stretcher when given minimum-weight stretchers for all optimal binders of all children of l. Moreover, the table A will store a stretcher of minimum weight for the optimal binder. Finally, it can be computed in $O(qn^{O(1)})$ time*

Proof. Consider an optimal binder for l and optimal binders for $l_1, \ldots, l_q$. By assumption, A contains a minimum-weight stretcher for the optimal binder for $l_1, \ldots, l_q$. We only need to be concerned with the terminals t inbetween consecutive non-empty intervals I_{l_j} and $I_{l_{j'}}$, where $j < j'$. That is, I_{l_j} and $I_{l_{j'}}$ are non-empty intervals and there is no $j < j'' < j'$ such that $I_{l_{j''}}$ is a non-empty interval. The remainder follows by definition. So consider such a terminal t. Consider the nerves $N^2_{l_j}$ and $N^1_{l_{j'}}$ of the optimal binders for l_j and $l_{j'}$. Now follow the nerve paths of the splints belonging to bones bordering f_j contained in $\mathcal{B}_{l_{j''}}$ for all $j \leq j'' \leq j'$ of the stretcher stored in A. This path has the same homotopy as in the optimal solution and goes between the same vertices as in the optimal solution, namely the roots of $N^2_{l_j}$ and $N^1_{l_{j'}}$. Together with the sides of $N^2_{l_j}$ and $N^1_{l_{j'}}$, which are as in the optimum, this path thus yields a region that encloses t and no other terminals by Lemma 3.6.9. Hence, the algorithm computes a stretcher for the optimal binder.

Finally, it remains to argue that the computed stretcher has minimum weight. We note that the optimal stretcher (for the optimal binder) can be decomposed into optimal stretchers (for the optimal binders) for each of the children $l_1, \ldots, l_q$ by the definition of optimality. Then, it follows by the description of the algorithm that A stores a minimum-weight stretcher for the optimal binder.

Note that a trivial implementation would compute this minimum in $O(n^{O(q)})$ time by enumerating all partitions of I_l and all nerves. However, using a simple dynamic program using partial unions, this can be reduced to $O(qn^{O(1)})$ time. □

3.8.2. Algorithm for a Bridge Block

We now set out to prove Lemma 3.8.4. Consider any BB-node l of L. It is either a single edge or a connected and bridgeless graph without self-loops. Hence, it has a sphere-cut decomposition by Theorem 3.3.2. Assuming that the table A has been computed for all children of l in L (which are C-nodes), we compute the table entry for l. By Lemma 3.8.1, $\mathcal{B}(L_l)$ is an internal set. Let F_l be the terminal face in the middle region of $\mathcal{B}(L_l)$. Consider a binder

I_l, N_l^1, N_l^2, w^+ for l. We assume that the binder is valid. We now wish to compute a stretcher for this binder by a dynamic program over the sphere-cut decomposition.

An important part of the dynamic program is how to incorporate the solutions for the children of l in L. Since we effectively consider $\mathcal{B}(l)$ as a collection of (shrunken) bones, we need to associate a bone with each child to ensure this. We now make this more formal. For each C-node of L that is child l' of l, corresponding to a cut vertex c, consider the middle region $f_{l'}$ of $\mathcal{B}(L_{l'})$ and let b, b' be two of the bones on the boundary of $\mathcal{B}(l)$ that are incident on c. Since l' is a child of l, it follows from the definition of an eabb tree that b and b' are well defined (and possibly $b = b'$). Pick one of b, b' in a consistent manner (say b) and associate l' with this bone. We call l' an *associated child* of the bone b, the cut vertex c, and the middle region $f_{l'}$. Later, when we propose the algorithm, we will discuss how to specify the binders for associated children.

Now let (R, η, δ) be a sphere-cut decomposition of $\mathcal{B}(l)$; refer back to Section 3.3.1 for the definitions. For every pair x, y of adjacent vertices of R such that y is the parent of x, consider the noose $\vec{\gamma} = \delta(x, y)$. Let $\mathcal{F}_{\vec{\gamma}} \subseteq \mathcal{F}$ be the set of terminal faces for which the corresponding face of S is intersected by $\vec{\gamma}$ and let $\mathcal{F}_{\mathbf{enc}(\vec{\gamma})} \subseteq \mathcal{F}$ be the set of terminal faces for which the corresponding face of S is enclosed by $\mathbf{enc}(\vec{\gamma})$. We include in $\mathcal{F}_{\mathbf{enc}(\vec{\gamma})}$ the bridge blocks of any associated children of any bone enclosed by $\mathbf{enc}(\vec{\gamma})$. Note that $\mathcal{F}_{\mathbf{enc}(\vec{\gamma})}$ is possibly empty, but $\mathcal{F}_{\vec{\gamma}}$ never is.

We now describe a bottom-up dynamic programming algorithm that aims to compute a stretcher of minimum weight for the given binder. To develop this algorithm, we first need a notion of what the partial solution is that we compute during the algorithm.

To this end, we need the notion of a partial binder and a partial stretcher. The intuition is that a partial binder is a dynamic programming state for the intersection of a noose $\vec{\gamma}$ with the hypothetical solution prescribed by the topology. For each vertex of the topology intersected by $\vec{\gamma}$, the state stores a corresponding vertex of the augmented dual. For each terminal face F_α in $\mathcal{F}_{\vec{\gamma}}$, we only see part of the hypothetical solution, which is described by an interval of T_α and the (possibly empty) first and last nerves that (possibly together with other nerves) cover the interval. Then a corresponding partial stretcher is the entry stored in the dynamic programming table for the partial binder, which essentially stores a solution for all terminal faces in $\mathcal{F}_{\mathbf{enc}(\vec{\gamma})}$ and a partial solution for all terminals between the mentioned interval.

An important situation occurs when F_l, the face in the middle region of the internal set $\mathcal{B}(L_l)$, is in $\mathcal{F}_{\vec{\gamma}}$. Recall that the binder specifies an interval

of T_l (and certain nerves) that must be covered by the stretcher that we are computing. Hence, in this case, the partial solution needs to satisfy (at least, partially) these demands as well. This constraint is met in part *(I)* of Definition 3.8.6 below.

We now define this intuition more formally. Throughout, if $w_1, \ldots, w_q$ are the q vertices of $\mathcal{B}(l)$ intersected by a noose $\vec{\gamma}$, then we assume that the faces of $\mathcal{F}_{\vec{\gamma}}$ are numbered $F_1, \ldots, F_q$, where F_j is the terminal face corresponding to the face of $\mathcal{B}(l)$ intersected by the part of $\vec{\gamma}$ between w_j and w_{j+1} (or w_1 if $j = q$).

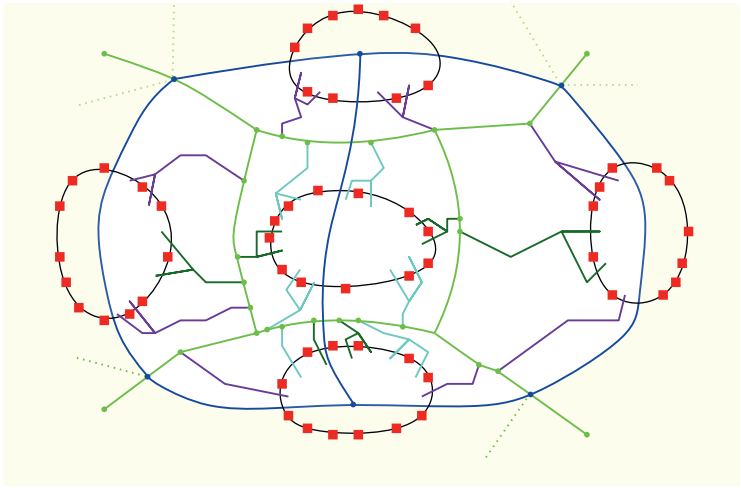

Figure 3.19: The noose $\vec{\gamma}$ is drawn in blue. It intersects the skeleton only in its vertices. The skeleton here is drawn in green. The outermost nerves N_1 and N_2 of each face of the skeleton are shown in purple. These form a part of the partial binder. The outermost nerves that are unique to the partial binders for the child edges (y_1, x) and (y_2, x) of the current edge (x, y) in the sphere-cut decomposition tree R are drawn in turquoise.

Definition 3.8.6. *Given a binder I_l, N_l^1, N_l^2, w^+ for l, the noose $\vec{\gamma} = \delta(x, y)$, augmented dual vertices $w_1^+, \ldots, w_q^+$, where $q = |\mathcal{F}_{\vec{\gamma}}|$, intervals $I_1, \ldots, I_q$ of $F_1, \ldots, F_q$ respectively, and (possibly empty) nerves $N_1^1, N_1^2, \ldots, N_q^1, N_q^2$ towards $F_1, \ldots, F_q$ respectively, a* partial stretcher *for (x, y) is a set D^+ of augmented dual edges such that:*

(I) *the intervals of $N_1^1, \ldots, N_q^1$ are a prefix of $I_1, \ldots, I_q$ (if non-empty) and the intervals of $N_1^2, \ldots, N_q^2$ are a suffix of $I_1, \ldots, I_q$ (if non-empty) respectively. For each $1 \leq i \leq q$, N_i^1 and N_i^2 are either both empty or both non-empty and cannot be empty if I_i has more than one terminal. If*

$F_l \in \mathcal{F}_{\vec{\gamma}}$, *say* $F_l = F_j$, *then:* I_j *is a subinterval of* I_l*; if* I_j *is a prefix of* I_l*, then* $N_j^1 = N_l^1$*; if* I_j *is a suffix of* I_l*, then* $N_j^2 = N_l^2$*;*

(II) *there is a (possibly empty) set of (possibly empty) nerves in* D^+ *towards* $F_1, \ldots, F_q$ *whose corresponding intervals jointly partition* $I_1, \ldots, I_q$*.* $N_1^1, \ldots, N_q^1$ *and* $N_1^2, \ldots, N_q^2$ *are among those nerves;*

(III) *for each terminal* t *between* I_1*, or ..., or* I_q*, there is a bounded face of* D^* *that encloses only* t *and no other terminals of* T*. Here* D^* *is the set of dual edges corresponding to the augmented dual edges in* D^+*;*

(IV) *for any terminal* $t \in T_\alpha$ *of a terminal face* $F_\alpha \in \mathcal{F}_{\mathbf{enc}(\vec{\gamma})}$*, there is a bounded face of* D^* *that encloses only* t *and no other terminals of* T*;*

(V) D^+ *is the union of a set of splints, one for each bone enclosed by* $\mathbf{enc}(\vec{\gamma})$*.* $N_1^1, N_1^2, \ldots, N_q^1, N_q^2$ *are included in these splints;*

(VI) *if* w *is in* $\mathbf{enc}(\vec{\gamma})$*, then* w^+ *is the vertex of* D^+ *that is an end of all splints for the bones in* $\mathbf{enc}(\vec{\gamma})$ *that are incident on* w*;*

(VII) $w_1^+, \ldots, w_q^+$ *are the ends of splints in* D^+ *that correspond to* $w_1, \ldots, w_q$*.*

We call $w_1^+, \ldots, w_q^+$*,* $I_1, \ldots, I_q$*, and* $N_1^1, \ldots, N_q^1, N_1^2, \ldots, N_q^2$ *a* partial binder *of the partial stretcher and of* (x, y)*. We say a partial binder is* valid *if it adheres to Property* (I)*.*

Proposition 3.8.7. *For any binder and noose* $\vec{\gamma} = \delta(x, y)$*, there are* $n^{O(q)}$ *distinct partial binders, where* $q = |\mathcal{F}_{\vec{\gamma}}|$*. These can be enumerated in the same time.*

Proof. It suffices to observe that a partial binder has $O(q)$ augmented dual vertices, intervals of terminal faces, and nerves, each of which can be described by a constant number of vertices of the augmented dual. □

For the dynamic program, we define a function Z that assigns to any partial binder a minimum-weight partial stretcher. If such a partial stretcher does not exist, then Z assigns the set of all augmented dual edges. Below, we show how we compute the table Z and argue that a minimum-weight stretcher is computed. To that end, we perform dynamic programming on the sphere-cut decomposition and describe two cases: a base case on a leaf of the branch decomposition, and an inductive case.

Dynamic Program: Base Case Consider the base case, when x is a leaf. Since $\mathcal{B}(l)$ is bridgeless and connected, $\mathbf{mid}(x, y)$ consists of the endpoints of a single bone $b = (w_1, w_2)$, enclosed by the noose $\vec{\gamma} = \delta(x, y)$. Let w^+, I_l, N_l^1, N_l^2 be a valid binder and w_1^+, w_2^+, I_1, I_2, and $N_1^1, N_2^1, N_1^2, N_2^2$ be a partial binder. We assume that the partial binder is valid, i.e. it satisfies property *(I)* of Definition 3.8.6. We also assume that if w is in $\mathbf{enc}(\vec{\gamma})$, then $w^+ = w_j^+$ when $w = w_j$ for $j \in \{1, 2\}$. Let F_α and F_β be the two terminal faces separated by b.

An important consideration here is how to deal with associated children of b.

The partial binder makes the partial solution for b responsible for covering the terminals between I_1 and I_2, but part of this responsibility can be delegated to the associated children of b. We only need to specify which subintervals of I_1 and I_2 is covered by the associated children by splitting it. We now formalize this intuition.

Suppose l' is an associated child of b. Say it is associated with w_1 and face F_α. Without loss of generality, b follows w_1 in the face ordering. Then we enumerate all possible intervals I_1' and I_1'' of F_α that partition I_1 (with I_1' preceding I_1'') and enumerate all possible (possibly) nerves N_1' and N_1'' towards F_α, such that w_1^+, I_1', N_1^1, and N_1' is a valid binder for l' (called a *split-parameterized binder*) and w_1^+, w_2^+, I_1'', I_2, and $N_1'', N_2^1, N_1^2, N_2^2$ is a valid partial binder (a *split partial binder*). Then we call I_1', I_1'', N_1', and N_1'' a *splitter* for the partial binder with respect to the associated child l'. Tying this to our earlier intuitive understanding, the splitter effectively specifies which part of I_1 is covered by the bone and which part by the associated child. This specification leads to a binder for the associated child (the split-parameterized binder) and a new partial binder for b (the split partial binder). We will later consider every possible splitter and take the best solution we find.

From now on, we assume that the partial binder is split with respect to all associated children of b (we might call it split even if b has no associated children). By abuse of notation, we still use the same variables for it.

We now enumerate all broken bones for b for which $I_\alpha = I_1$, $I_\beta = I_2$, $y_0^+ = w_1^+$, $x_{|s(b)|+1}^+ = w_2^+$, and nerves $N_1, \ldots, N_{2|s(b)|}$ that correspond to N_1^1, N_2^1 and N_1^2, N_2^2. Here we mean by 'correspond' that for the smallest $j \in \{1, \ldots, |s(b)|\}$ for which $s(b)[j] = \alpha$, it holds that $N_{2j-1} = N_1^1$ and for the largest $j \in \{1, \ldots, |s(b)|\}$ for which $s(b)[j] = \alpha$, it holds that $N_{2j} = N_1^2$; a similar condition holds with respect to β. We say that such a broken bone *specializes* the partial binder. Conversely, the partial binder *generalizes* the broken bone. Note that there is a unique partial binder generalizing a broken bone, while this is not true for the converse. Then we use Algorithm 1 to compute a minimum-weight splint for each broken bone that specializes the split partial binder. Finally, set

Z of the partial binder to be equal to a minimum-weight union of the splint found in this manner for the split partial binder and the stretchers stored in A for the split-parameterized binders. This minimum is optimized over all splitters for which the aforementioned union forms a partial stretcher. If no such union yields a partial stretcher, set Z to be equal to the set of all augmented dual edges.

For the optimal topology and the optimal binder for a node l of $L(S)$, a partial binder is *optimal* if C^+ contains a partial stretcher for this partial binder where each of the splints of the partial stretcher (mentioned in part V) are optimal. By Remark 3.7.8, we may assume that any optimum solution indeed consists of splints. We then call this partial stretcher *optimal* as well. It follows that, for a bone b, a split partial binder is *optimal* if it generalizes the optimal broken bone for b. A splitter is *optimal* if the resulting split-parameterized binders and split partial binder are optimal.

Proposition 3.8.8. *Let x be a leaf and b the corresponding bone whose endpoints are in $\mathbf{mid}(x, y)$. Consider any broken bone specializing a valid partial binder B for (x, y) and if w is in $\mathbf{enc}(\vec{\gamma})$, then $w^+ = w_j^+$ when $w = w_j$ for $j \in \{1, 2\}$. If B is split or b has no associated children, then any splint for the broken bone is a partial stretcher for B. If b has associated children, B is optimal, and the splitter is optimal, then the union of a splint for the broken bone and the stretchers stored in A for the optimal split-parameterized binders is a partial stretcher. Finally, for the optimal partial binder, the table entry $Z(B)$ will store a partial stretcher of minimum weight.*

Proof. We verify that all properties of Definition 3.8.6 hold. Property *(I)* holds by definition. Property (II) and (III) follow by the definition of a splint for a broken bone specializing the partial binder. Property (IV) follows from the definition of a splint. Property (V) follows from the definition of a splint for a broken bone specializing the partial binder. Property (VI) follows by assumption and the definition of a splint. Finally, Property (VII) follows by the definition of a splint for a broken bone specializing the partial binder.

If b has associated children, then we augment the argument for Property (III) and (IV). Indeed, Property (III) is satisfied for all terminals except possibly the terminal t inbetween I_1' and I_1'' of the optimal splitter. However, w_1^+, N_1', and the attachment point of N_1'' are as in the optimum and thus the path between w_1^+ and the attachment point of N_1'' has the same homotopy as in the optimum (by the definition of a splint) and goes between the same vertices as the optimum. Together with the flanks of N_1' and N_1'', which are as in the optimum, this path thus yields a region that encloses t and no other terminals by Lemma 3.6.9. Property (IV) follows by the definition of a partial stretcher applied to the associated children.

For the final part, we note that the preceding establishes that for the optimal (original) partial binder and optimal splitter, the algorithm yields a partial stretcher. Note that the optimal broken bone is among the broken bones specializing the optimal partial binder by definition. Then, by the optimality of the splinting algorithm, the resulting splint for the optimal split partial binder is optimal. Then, by the optimality of the table A for each associated child and of the splinting algorithm, the resulting splint and partial stretcher will have minimum weight. □

Dynamic Program: Inductive Case Now consider the inductive case, when x is an internal vertex. Let w^+, I_l, N_l^1, N_l^2 be a valid binder. Let $w_1^+, \ldots, w_q^+, I_1, \ldots, I_q$, and $N_1^1, \ldots, N_q^1, N_1^2, N_q^2$ be a partial binder B for (x, y). We assume that the partial binder satisfies property *(I)* of Definition 3.8.6. We also assume that if w is on $\vec{\gamma}$, then $w^+ = w_j^+$ when $w = w_j$ for $j \in \{1, \ldots, q\}$. Let $v_1^+, \ldots, v_r^+$, $J_1, \ldots, J_r$, and $M_1^1, \ldots, M_r^1, M_1^2, \ldots, M_r^2$ be a valid partial binder B_1 for (y_1, x) and let $u_1^+, \ldots, u_s^+$, $K_1, \ldots, K_s$, and $L_1^1, \ldots, L_s^1, L_1^2, \ldots, L_s^2$ be a valid partial binder B_2 for (y_2, x). Let $\vec{\alpha} = \delta(y_1, x)$ and $\vec{\beta} = \delta(y_2, x)$. Note that the terminal faces in $\mathcal{F}_{\vec{\gamma}}$, $\mathcal{F}_{\vec{\alpha}}$, and $\mathcal{F}_{\vec{\beta}}$ can be categorized into four classes: those that appear in all three (of which there are at most two), those that appear in $\mathcal{F}_{\vec{\gamma}}$ and $\mathcal{F}_{\vec{\alpha}}$ but not $\mathcal{F}_{\vec{\beta}}$, those that appear in $\mathcal{F}_{\vec{\gamma}}$ and $\mathcal{F}_{\vec{\beta}}$, but not in $\mathcal{F}_{\vec{\alpha}}$, and those that appear in $\mathcal{F}_{\vec{\alpha}}$ and $\mathcal{F}_{\vec{\beta}}$, but not in $\mathcal{F}_{\vec{\gamma}}$.

We say that these partial binders *match* if

- for each terminal face $F_j \in \mathcal{F}_{\vec{\gamma}}$ that is in $\mathcal{F}_{\vec{\alpha}}$ (say it is also numbered j in $\mathcal{F}_{\vec{\alpha}}$) but not in $\mathcal{F}_{\vec{\beta}}$, the state for this face is the same in B and B_1. Formally, $I_j = J_j$, $w_j^+ = v_j^+$, $w_{j+1}^+ = v_{j+1}^+$, $N_j^1 = M_j^1$, and $N_j^2 = M_j^2$;
- for each terminal face $F_j \in \mathcal{F}_{\vec{\gamma}}$ that is in $\mathcal{F}_{\vec{\beta}}$ (say it is also numbered j in $\mathcal{F}_{\vec{\beta}}$) but not in $\mathcal{F}_{\vec{\alpha}}$, the state for this face is the same in B and B_1. Formally, $I_j = K_j$, $w_j^+ = u_j^+$, $w_{j+1}^+ = u_{j+1}^+$, $N_j^1 = L_j^1$, and $N_j^2 = L_j^2$;
- for each terminal face $F_j \in \mathcal{F}_{\vec{\gamma}}$ that is in both $\mathcal{F}_{\vec{\alpha}}$ and $\mathcal{F}_{\vec{\beta}}$ (say it is also numbered j in both $\mathcal{F}_{\vec{\alpha}}$ and $\mathcal{F}_{\vec{\beta}}$), the state for this face in B is the 'union' of the states stored in B_1 and B_2. Formally, $I_j = J_j \uplus K_j$, $w_j^+ = v_j^+$, $v_{j+1}^+ = u_j^+$, $w_{j+1}^+ = u_{j+1}^+$, $N_j^1 = M_j^1$, and $N_j^2 = L_j^2$;
- for each terminal face F_j that is in both $\mathcal{F}_{\vec{\alpha}}$ and $\mathcal{F}_{\vec{\beta}}$ (say it is numbered j in both $\mathcal{F}_{\vec{\alpha}}$ and $\mathcal{F}_{\vec{\beta}}$) but not in $\mathcal{F}_{\vec{\gamma}}$, the state for this face in B_1 and B_2 jointly covers all terminals. Formally, $J_j \uplus K_j = T_j^+$, $v_{j+1}^+ = u_j^+$, and $v_j^+ = u_{j+1}^+$.

To simplify the presentation, we did not consider index overflow, e.g., when $j+1 > |\mathcal{F}_{\vec{\gamma}}|$, we should use index 1 instead. The following is immediate from the definition above and that of optimal partial binders.

Proposition 3.8.9. *The triple of optimal partial binders for (x, y), (y_1, x), and (y_2, x) match.*

The algorithm now does the following. For each valid partial binder B for (x, y), we enumerate all matching, valid partial binders B_1, B_2 for (y_1, x) and (y_2, x) respectively and set Z to be equal to the minimum-weight union of the corresponding partial stretchers stored in Z for B_1 and B_2 that form a partial stretcher for B. If no such union forms a partial stretcher for B, then we set Z to be equal to the set of all augmented dual edges.

Proposition 3.8.10. *Consider the triple of optimal partial binders B, B_1, B_2 for (x, y), (y_1, x), and (y_2, x) respectively. The union of any partial stretcher for B_1 and any partial stretcher for B_2 forms a partial stretcher for B. Moreover, the table entry $Z(B)$ will store a partial stretcher of minimum weight.*

Proof. Consider partial stretchers P_1 and P_2 for B_1 and B_2 respectively. It is important to remember the fact that the set of edges (bones) in $\mathbf{enc}(y_1, x)$ and in $\mathbf{enc}(y_2, x)$ are disjoint by the definition of a sphere-cut decomposition. Property (VI) and (VII) now follow using the fact that each vertex in $\mathbf{mid}(x, y)$ is in $\mathbf{mid}(y_1, x)$ or $\mathbf{mid}(y_2, x)$ or matched in the partial binders. It is also clear that $P_1 \cup P_2$ is a union of a set of splints, almost verifying Property (V). Moreover, for each face in $\mathcal{F}_{\mathbf{enc}(\vec{\alpha})}$ or $\mathcal{F}_{\mathbf{enc}(\vec{\beta})}$, Property (IV) is satisfied. We now verify the remaining (parts of the) properties.

By Proposition 3.8.9, the partial binders match. We then consider each of the cases:

- for each terminal face $F_j \in \mathcal{F}_{\vec{\gamma}}$ that is in $\mathcal{F}_{\vec{\alpha}}$ (say it is also numbered j in $\mathcal{F}_{\vec{\alpha}}$) but not in $\mathcal{F}_{\vec{\beta}}$, P_1 contains augmented dual edges to satisfy Property (II), (III), and (V) with respect to F_j;
- for each terminal face $F_j \in \mathcal{F}_{\vec{\gamma}}$ that is in $\mathcal{F}_{\vec{\beta}}$ (say it is also numbered j in $\mathcal{F}_{\vec{\beta}}$) but not in $\mathcal{F}_{\vec{\alpha}}$, P_2 contains augmented dual edges to satisfy Property (II), (III), and (V) with respect to F_j;
- for each terminal face $F_j \in \mathcal{F}_{\vec{\gamma}}$ that is in both $\mathcal{F}_{\vec{\alpha}}$ and $\mathcal{F}_{\vec{\beta}}$ (say it is also numbered j in both $\mathcal{F}_{\vec{\alpha}}$ and $\mathcal{F}_{\vec{\beta}}$), P_1 and P_2 jointly contain augmented dual edges to satisfy Property (II), (III), and (V) with respect to F_j. The only exception is the terminal inbetween J_j and K_j. If such a terminal t indeed exists, then follow the nerve paths stored in the splints of P_1 for

the bones of face f_j in $\mathbf{enc}(\vec{\alpha})$ from the root of nerve M_j^2 (which must be non-empty since t exists) to $v_{j+1}^+ = u_j^+$, and then the nerve paths stored in the splints of P_2 for the bones of face f_j in $\mathbf{enc}(\vec{\beta})$ from $v_{j+1}^+ = u_j^+$ to the root of nerve L_j^1 (which must be non-empty since t exists). This path has the same homotopy as in the optimum, by the optimality of the topology and the splints. It also goes between the same vertices as in the optimum, by the optimality of the partial binders. Together with the sides of M_j^2 and L_j^1, which are as in the optimum, this path thus yields a region that encloses t and no other terminals by Lemma 3.6.9.

- for each terminal face F_j that is in both $\mathcal{F}_{\vec{\alpha}}$ and $\mathcal{F}_{\vec{\beta}}$ (say it is numbered j in both $\mathcal{F}_{\vec{\alpha}}$ and $\mathcal{F}_{\vec{\beta}}$) but not in $\mathcal{F}_{\vec{\gamma}}$, we note it is contained in $\mathcal{F}_{\mathbf{enc}(\vec{\gamma})}$, but not in $\mathcal{F}_{\mathbf{enc}(\vec{\alpha})}$ or $\mathcal{F}_{\mathbf{enc}(\vec{\beta})}$. To verify Property (IV), we note that it holds for all terminals in T_j by Property (III) applied to P_1 and P_2, except for the one or two terminals inbetween J_j and K_j. Let t be such a terminal. Then follow the nerve paths stored in the splints of P_1 for the bones of face f_j in $\mathbf{enc}(\vec{\alpha})$ from the root of nerve M_j^1 (or from L_j^1 if the nerve is empty, or from v_{j+1}^+ if both are empty) to $v_j^+ = u_{j+1}^+$, and then the nerve paths stored in the splints of P_2 for the bones of face f_j in $\mathbf{enc}(\vec{\beta})$ from $v_j^+ = u_{j+1}^+$ to the root of nerve L_j^2 (or to M_j^2 if this nerve is empty or to u_j^+ if both nerves are empty). This path has the same homotopy as in the optimum, by the optimality of the topology and the splints. It also goes between the same vertices as in the optimum, by the optimality of the partial binders. Together with the sides of M_j^1 (or of L_j^1) and L_j^2 (or of M_j^2), which are as in the optimum, this path thus yields a region that encloses t and no other terminals by Lemma 3.6.9.

It follows that the union of P_1 and P_2 is a partial stretcher for B.

For the final part, we note that by the preceding, the table entry is indeed a partial stretcher. It remains to argue that it has minimum weight. It suffices to argue that an optimum partial stretcher for B can be decomposed into (disjoint) partial stretchers for B_1 and B_2; to this end, we note that the optimum partial stretcher for B is decomposed into the optimum partial stretchers for B_1 and B_2 by definition of optimality and matching. Then, it follows by induction and Proposition 3.8.8 that Z stores a minimum-weight partial stretcher for B_1 and B_2, and thus Z will store a minimum-weight partial stretcher for B. □

Finally, we observe that for the root of the sphere-cut decomposition, any partial binder is equal to the given binder and any partial stretcher for the partial binder is a stretcher for the given binder. This is immediate from the

definitions. Thus, we can set A for the given binder as the entry stored in Z for the corresponding (equal) partial binder.

Proof of Lemma 3.8.5. By Theorem 3.3.2, $\mathcal{B}(l)$ has a sphere-cut decomposition of width $O(\sqrt{|\mathcal{B}(l)|})$. We then compute the table Z. By Proposition 3.8.7, we can enumerate all partial binders in $n^{O(\sqrt{|\mathcal{B}(l)|})}$ time. We can immediately see that the dynamic program takes $n^{O(\sqrt{|\mathcal{B}(l)|})}$ time as well. The base case relies on the splinting algorithm, which takes polynomial time by Lemma 3.7.6. As observed previously, for the root of the sphere-cut decomposition, any partial binder is equal to the given binder and any partial stretcher for the partial binder is a stretcher for the given binder. Hence, the correctness of the algorithm follows from Proposition 3.8.8 and Proposition 3.8.10. □

3.9. Proof of Theorem 3.1.1

The following theorem implies the algorithmic part of Theorem 3.1.1. As already mentioned, the lower bound immediately follows from Marx's result [149].

Theorem 3.9.1. *Consider an instance (G, T, ω) of Planar Multiway Cut. We can compute a minimum-weight multiway cut of (G, T) in $2^{O(k^2 \log k)} n^{O(\sqrt{k})}$ time, where k is the number of faces needed to cover all terminals.*

Proof. First, in time $2^{O(k)} n^{O(1)}$ through the algorithm of Bienstock and Monma [17], we compute a set of faces of size k that covers all the terminals. This is the set $\mathcal{F}$. We then transform the instance through the transformations of Section 3.4, and particularly Lemma 3.4.1. We then apply the algorithm of Lemma 3.4.12 to reduce to the case when the dual of any optimum solution is connected.

Now, by Lemma 3.7.3, we can enumerate all topologies in $2^{O(k^2 \log k)}$ time. For each topology (S, s, h), build an embedding-aware bridge block tree $L = L(S)$ in linear time by Lemma 3.3.4. Now we perform the dynamic program of Lemma 3.8.5 and 3.8.4 on $L(S)$. Since the nodes corresponding to the same cut vertex of H effectively form a subtree of L through Lemma 3.3.6, the binders contain the same cut vertex w and w^+ throughout. Then it follows that by induction on the depth of l in L that a minimum-weight stretcher is computed for each optimal binder. Finally, for the root node of L, we consider the minimum-weight stretcher that is found among all binders. One of those will be the optimal binder, and thus we find a stretcher of weight at most that of the optimal stretcher for the optimal binder. Recalling that a topology has $O(k)$ vertices, it follows from Lemma 3.8.5 and 3.8.4 that the running time is $n^{O(\sqrt{k})}$.

Finally, we note here that the running-time our algorithm does not depend on the sum of edge-weights W in a unit cost model of computation. It only linearly depends on $\log W$. □

3.10. Conclusions

We showed in this chapter that Edge Multiway Cut can be solved on planar graphs with terminal face-cover number k, in time $2^{O(k^2 \cdot \log k)} \cdot n^{O(\sqrt{k})}$. While one cannot hope to find an algorithm that solves the problem in $n^{o(\sqrt{k})}$ time unless ETH fails, we would like to point out that there still is room for improvement in the exponent of 2. It is known that Steiner Tree can be solved in time $n^{O(\sqrt{k})}$, on planar graphs with terminal face-cover number k [118]. It remains open to study the complexity of both of these problems with respect to the same parameter in other classes of embedded graphs, for example, bounded genus graphs.

4

Complexity of Multiway Cut on Planar Subcubic Graphs

It is known that the weighted version of Edge Multiway Cut (also known as Multiterminal Cut) is NP-complete on planar graphs of maximum degree 3 [60]. In contrast, for the unweighted version, NP-completeness is only known for planar graphs of maximum degree 11. In fact, the complexity of unweighted Edge Multiway Cut was open for graphs of maximum degree 3 for over twenty years. We prove that the unweighted version is NP-complete even for planar graphs of maximum degree 3. As weighted Edge Multiway Cut is polynomial-time solvable for graphs of maximum degree at most 2, we have now closed the complexity gap. We also prove that (unweighted) Node Multiway Cut (both with and without deletable terminals) is NP-complete for planar graphs of maximum degree 3. By combining our results with known results, we can apply two meta-classifications on graph containment from the literature. These yield full dichotomies for all three problems on $\mathcal{H}$-topological-minor-free graphs and, should $\mathcal{H}$ be finite, on $\mathcal{H}$-subgraph-free graphs as well. Previously, such dichotomies were only implied for $\mathcal{H}$-minor-free graphs.

In this chapter, we consider the *unweighted* edge and node versions of the Multiway Cut problem. As before, let S be the set of edges (or vertices) that we remove from the graph G to pairwise disconnect all the terminals of $T \subseteq V(G)$. We consider two versions of Node Multiway Cut depending on whether S can contain vertices of T or not: Unrestricted Node Multiway Cut, and Restricted Node Multiway Cut.

Our goal in this chapter is to answer the following question:

What is the computational complexity of Edge Multiway Cut *and both versions of* Node Multiway Cut *for planar subcubic graphs?*

Chapter Outline: We describe our motivation behind investigating the above question in Section 4.1. Then we go on to prove our main result, i.e. the NP-completeness of Edge Multiway Cut on planar, subcubic graphs in Section 4.2. Next we discuss the complexity of Node Multiway Cut on subcubic graphs in Section 4.3. Finally, we discuss the consequences of our results and open problems left to investigate in Section 4.4.

4.1. Motivation

Our first reason to study the problem is due to a complexity gap that was left open in the literature for over twenty years. That is, in addition to their NP-completeness result for $|T| = 3$, Dahlhaus et al. [60] also proved that Weighted Edge Multiway Cut is NP-complete on planar subcubic graphs using integral edge weights. Any edge of integer weight j can be replaced by j parallel edges (and vice versa) without changing the problem. Hence, their reduction implies that Edge Multiway Cut is NP-complete on planar graphs of maximum degree at most 11 [60, Theorem 2b]. Dahlhaus *et al.* [60] write that "*The degree bound of* 11 *is not the best possible. Using a slight variant on the construction and considerably more complicated arguments, we believe it can be reduced at least to* 6", but no further arguments were given. Even without the planarity condition and only focussing on the maximum degree bound, the hardness result of [60] is still best known. Given that the problem is polynomial-time solvable if the maximum degree is 2, this means that there is a significant complexity gap that has yet to be addressed.

To the best of our knowledge, there is no explicit hardness result in the literature that proves NP-completeness of either version of Node Multiway Cut on graphs of any fixed degree or on planar graphs.

However, known and straightforward reductions (see e.g. [98, 162]) immediately yield NP-hardness on planar subcubic graphs for Unrestricted Node Multiway Cut (see Theorem 4.1.2), but only on planar graphs of maximum degree 4 for Node Multiway Cut (see Proposition 4.3.1).

Our second reason is the central role planar subcubic graphs play in complexity dichotomies of graph problems restricted to graphs that do not contain any graph from a set $\mathcal{H}$ as a topological minor[1] or subgraph; such graphs are said to be *$\mathcal{H}$-topological-minor-free* or *$\mathcal{H}$-subgraph-free*, respectively. For both the topological minor containment relation [169] and the subgraph relation (see [120]) meta-classifications exist. To apply these meta-classifications, a problem must satisfy certain conditions, in particular being NP-complete for subcubic planar graphs for the topological minor relation, and being NP-complete for subcubic graphs for the subgraph relation. These two conditions are *exactly what is left to prove* for Edge Multiway Cut and both versions of Node Multiway Cut. In contrast, the results of [9, 60, 169] and the aforementioned reductions from [98, 162] imply that all three problems are fully classified for $\mathcal{H}$-minor-free graphs: the problems are polynomial-time solvable if $\mathcal{H}$ contains a planar graph and NP-complete otherwise (see also [120]). Hence, determining the complexity status of our three problems for planar subcubic graphs is a pressing issue.

Our third reason is the rich tradition to investigate the NP-completeness of problems on subcubic graphs and planar subcubic graphs (see e.g. the list in [120]) which continues till this day, as evidenced by recent NP-completeness results for subcubic graphs (e.g. [26, 186]) and planar subcubic graphs (e.g. [30, 196]).

We also note that Edge Multicut, the standard generalization of Edge Multiway Cut, is NP-complete even on subcubic trees [39].

For the above reasons, the fact that the complexity status of our three problems restricted to (planar) subcubic graphs has remained open this long is unexpected.

Our Results: The following three results fully answer our research question.

Theorem 4.1.1. Edge Multiway Cut *is* NP*-complete for planar subcubic graphs.*

Theorem 4.1.2. Unrestricted Node Multiway Cut *is* NP*-complete for planar subcubic graphs.*

Theorem 4.1.3. Node Multiway Cut *is* NP*-complete for planar subcubic graphs.*

[1] A graph G contains a graph H as a *topological minor* if G can be modified into H by a sequence of edge deletions, vertex deletions and vertex dissolutions, where a vertex dissolution is the contraction of an edge incident to a vertex of degree 2 whose (two) neighbors are non-adjacent.

The NP-completeness for Node Multiway Cut for planar subcubic graphs follows from the NP-hardness of Edge Multiway Cut by constructing the line graph of input graph. The hardness for Unrestricted Node Multiway Cut on planar subcubic graphs follows from a straightforward reduction from Vertex Cover.

4.2. The Proof of Theorem 4.1.1

In this section, we show that Edge Multiway Cut is NP-complete on planar subcubic graphs. In spirit, our construction for Edge Multiway Cut is similar to the one by Dahlhaus et al. [60] for graphs of maximum degree 11. For non-terminal vertices of high degree, a local replacement by a (sub)cubic graph is relatively easy. However, for terminal vertices of high degree, a local replacement strategy seems impossible. Hence, the fact that terminals in the construction of Dahlhaus et al. [60] can have degree up to 6 becomes a crucial bottleneck.

To ensure that our constructed graph has maximum degree 3, we therefore need to build different gadgets. We then leverage several deep structural properties of the edge multiway cut in the resulting instance, making for a significantly more involved and technical correctness proof. Crucially, we first prove NP-completeness for a weighted version of the problem on graphs of maximum degree 5, in which the terminals all have degree 3. Then we replace weighted edges and high-degree vertices with appropriate gadgets.

We reduce the problem from a special case of Planar 2P1N-3SAT, which is a restricted version of 3-SAT. Given a CNF-formula Φ with the set of variables X and the set of clauses C, the *incidence graph* of the formula is the graph $G_{X,C}$ which is a bipartite graph with one of the partitions containing a vertex for each variable and the other partition containing a vertex for each clause of Φ. There exists in $G_{X,C}$ an edge between a variable-vertex and a clause-vertex if and only if the variable appears in the clause. We define Planar 2P1N-3SAT as follows.

Planar 2P1N-3SAT	
Instance:	A set $X = \{x_1, \ldots, x_n\}$ of variables and a CNF formula Φ over X and clause set C with each clause containing at most three literals and each variable occurring twice positively and once negatively in Φ such that $G_{X,C}$ is planar.
Question:	Is there an assignment $\mathcal{A} : X \to \{0, 1\}$ that satisfies Φ?

The above problem was shown to be NP-complete in [60]. Let Planar 2P1N-

3SAT-2 be a special case of Planar 2P1N-3SAT in which every variable occurs in at least two clauses of size two. The construction given by Dahlhaus *et al.* [60], generates instances with the property that each variable occurs in at least two clauses having size two. Hence, by their reduction Planar 2P1N-3SAT-2 is NP-complete.

We need two further definitions. Recall that in Weighted Edge Multiway Cut, we are given a function $\omega : E(G) \to \mathbb{Q}^+$ in addition to G, T, k. The goal is to decide if (G, T) admits an edge multiway cut of total weight at most k. If the image of ω is the set X, we denote the corresponding Weighted Edge Multiway Cut problem as X-Edge Multiway Cut. Also note that if an edge/node multiway cut S has the smallest possible size (weight) among all edge/node multiway cuts for the pair (G, T), then S is a *minimum(-weight)* edge/node multiway cut.

We show the reduction in two steps. In the first step, we reduce from Planar 2P1N-3SAT-2 to $\{1, 2, 3, 6\}$-Edge Multiway Cut restricted to planar graphs of maximum degree 5 where the terminals all have degree 3. In the second step, we show how to make the instance unweighted while keeping it planar and making its maximum degree bounded above by 3.

Theorem 4.1.1 (Restated)**.** Edge Multiway Cut *is* NP*-complete for planar subcubic graphs.*

Proof. Clearly, Edge Multiway Cut is in NP. We reduce Edge Multiway Cut from Planar 2P1N-3SAT-2. Let Φ be a given CNF formula with at most three literals in each clause and each variable occurring twice positively and once negatively.

We assume that each clause has size at least 2 and every variable occurs in at least two clauses of size 2. Let $X = \{x_i \mid 1 \leq i \leq n\}$ be the set of variables in Φ and $C = \{c_j \mid 1 \leq j \leq m\}$ be the set of clauses. We assume that the incidence graph $G_{X,C}$ is planar. By the reduction in Dahlhaus et al. [60], Planar 2P1N-3SAT-2 is NP-complete for such instances.

We now describe the graph construction. For each vertex of $G_{X,C}$ corresponding to a clause c_j in C, we create a clause gadget (depending on the size of the clause), as in Figure 4.1. For each vertex of $G_{X,C}$ corresponding to a variable $x_i \in X$, we create a variable gadget, also shown in Figure 4.1. The gadgets have two terminals each (marked as red squares in Figure 4.1), a positive and a negative one. In a variable gadget, the positive terminal is attached to the diamond and the negative one to the hat, by edges of weight 3; refer to Figure 4.1. In a clause gadget, each literal corresponds to a triangle, with these triangles connected in sequence, and the positive and negative terminal are attached to triangles at the start and end of the sequence, again by edges of weight 3.

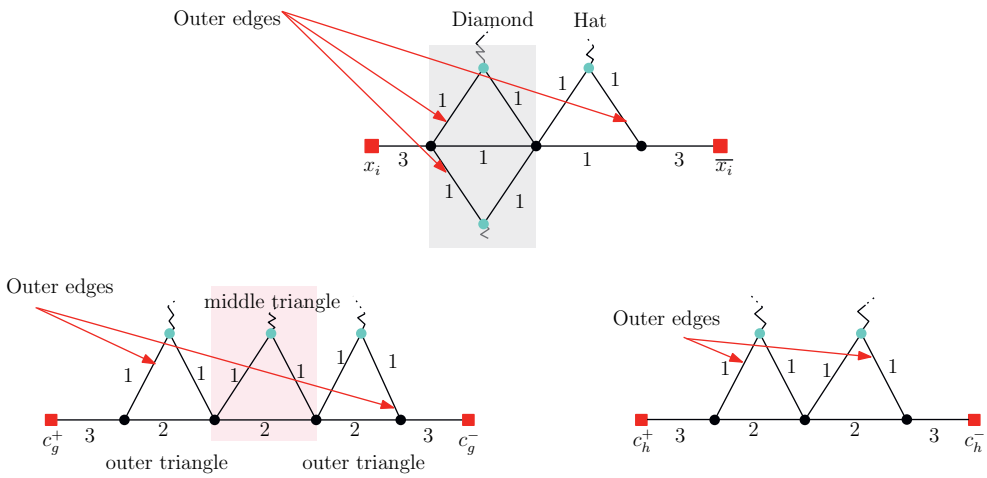

Figure 4.1: The gadgets for the variables (top) as well as those for the clauses (bottom). The bottom-left gadget corresponds to a clause with three literals whereas the bottom-right one corresponds to a clause with two literals. The terminals are depicted as red squares.

Each degree-2 vertex in a gadget (marked blue in Figure 4.1) acts as a connector. For a variable x_i, if $x_i \in c_j$ and $x_i \in c_k$ for clauses c_j, c_k, then we connect the degree-2 vertices of the diamond of x_i to some degree-2 vertex of the gadgets for c_j and c_k, each by an edge of weight 6. If $\overline{x_i} \in c_l$ for clause c_l, then we connect the degree-2 vertex of the hat of x_i and some degree-2 vertex on the gadget for c_l, again by an edge of weight 6. These connecting edges are called *links*. An example of such variable and clause connections is depicted in Figure 4.4. By the assumptions on Φ, we can create the links such that each degree-2 vertex in the variable gadget is incident on exactly one link and corresponds to one occurrence of the variable. Similarly, each degree-2 vertex of a clause gadget is incident on exactly one link.

The graph thus created is denoted by G. We can construct G in such a way that it is planar, because $G_{X,C}$ is planar and has maximum degree 3. Note that G has maximum degree 5. Let T be the set of terminals in the constructed graph G. Note that G has a total of $2n + 2m$ terminals.

We observe that all edges in G have weight at most 6. Non-terminal vertices are incident on edges of total weight at most 8. Crucially, terminals are incident on edges of total weight at most 3.

We introduce some extra notions to describe the constructed graph G. The edges of the two triangles adjacent to a link are called *connector edges*. The edge of such a triangle that is not adjacent to the link is called the *base* of the triangle. The connector edges closest to the terminals are called *outer*

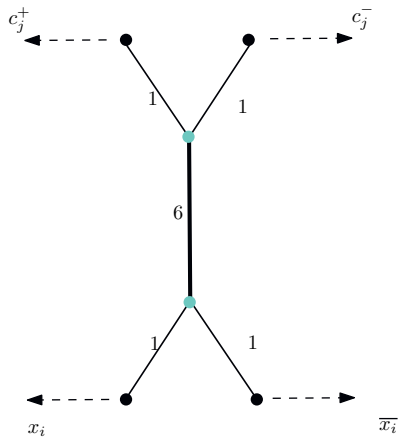

Figure 4.2: The figure shows a link structure formed by the connector edges of a clause-triangle and its corresponding variable-triangle. The two bases that complete the triangles are not drawn.

edges, as indicated in Figure 4.1. The structure formed by the two pairs of connector edges and the link is called the *link structure*; see Figure 4.2. Since each variable occurs twice positively and once negatively in Φ, the constructed graph G has exactly $3n$ link structures.

We now continue the reduction to obtain an unweighted planar subcubic graph. We replace all the edges in G of weight greater than 1 by as many parallel edges between their end-vertices as the weight of the edge. Each of these parallel edges has weight 1. We refer to this graph as G'. Next, for each vertex v in G' of degree greater than 3, we replace v by a large honeycomb (hexagonal grid), as depicted in Figure 4.3, of size 1000×1000 (these numbers are picked for convenience and not optimized). The neighbors of v, of which there are at most eight by the construction of G, are now attached to distinct degree-2 vertices on the boundary of the honeycomb such that the distance along the boundary between any pair of them is 100 cells of the honeycomb. These degree-2 vertices on the boundary are called the *attachment points* of the honeycomb. The edges not belonging to the honeycomb that are incident on these attachment points are called *attaching edges*. In the construction, we ensure that the attaching edges occur in the same cyclical order on the boundary as the edges to the neighbors of v originally occurred around v. Let the resultant graph be $\tilde{G}$.

Note that the degree of any vertex in $\tilde{G}$ is at most 3. For terminals, this was already the case in G'. Note that, therefore, terminals were not replaced by honeycombs to obtain $\tilde{G}$. For non-terminals, this is clear from the construction

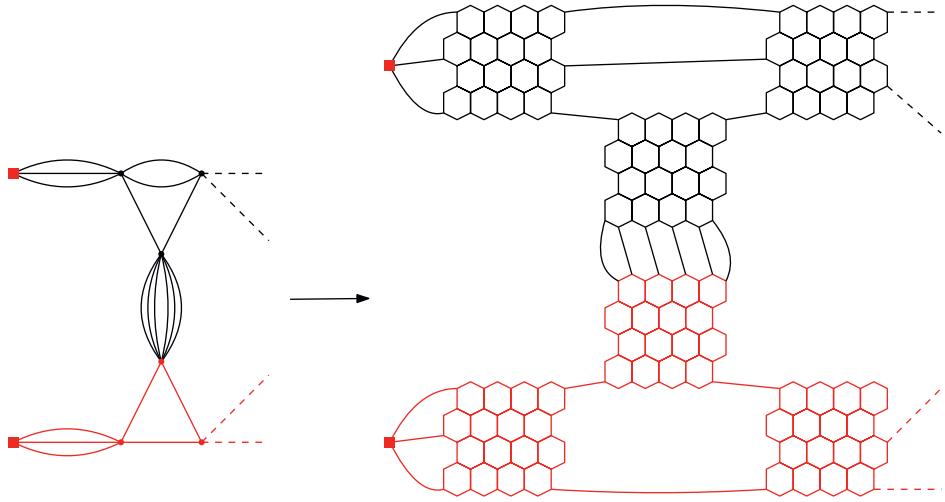

Figure 4.3: Construction of $\tilde{G}$ from G by replacing every edge of weight greater than 1 by as many parallel edges as its weight and then replacing the vertices of degree greater than 3 by a honeycomb of size 1000×1000.

of G' and $\tilde{G}$. Moreover, all the edge weights of $\tilde{G}$ are equal to 1, and thus we can consider it unweighted. Also, all the replacements can be done as to retain a planar embedding of G and hence, $\tilde{G}$ is planar. $\tilde{G}$ has size bounded by a polynomial in $n + m$ and can be constructed in polynomial time. Finally, we set $k = 7n + 2m$.

For the sake of simplicity, we shall first argue that Φ is a YES instance of Planar 2P1N-3SAT-2 if and only if (G, T, k) is a YES instance of $\{1, 2, 3, 6\}$-Edge Multiway Cut. Later, we show that the same holds for $\tilde{G}$ by proving that no edge of the honeycombs is ever present in any minimum edge multiway cut in $\tilde{G}$. We defer the proof of this claim for now.

Suppose that $\mathcal{A}$ is a truth assignment satisfying Φ. Then, we create a set of edges $S \subseteq E(G)$, as follows:

- If a variable is set to "true" by $\mathcal{A}$, then add to S all the three edges of the hat in the corresponding gadget. If a variable is set to "false" by $\mathcal{A}$, then add to S all the five edges of the diamond.
- For each clause, pick a true literal in it and add to S all the three edges of the clause-triangle corresponding to this literal.
- Finally, for each link structure with none of its edges in S yet, add the two connector edges of its clause-triangle to S.

Claim 4.2.1. *S is an edge multiway cut of (G, T) of weight at most $7n + 2m$.*

Proof. For each variable, either the positive literal is true, or the negative one. Hence, either all the three edges of its hat are in S or all the five edges of the diamond. Therefore, all the paths between terminal pairs of the form $x_i - \overline{x_i}$, for all $1 \leq i \leq n$, are disconnected in $G - S$. Consider the link structure in Figure 4.2. By our choice of S, at least one endpoint of each link in $G - S$ is a vertex of degree 1, hence a dead end. Therefore, no path connecting any terminal pair in $G - S$ passes through any link. As all the paths in G between a variable-terminal and a clause-terminal must pass through some link, we know that all terminal pairs of this type are disconnected in $G - S$. Since $\mathcal{A}$ is a satisfying truth assignment of Φ, all the edges of one triangle from every clause gadget are in S. Hence, all the paths between terminal pairs of the form $c_j^+ - c_j^-$, for all $1 \leq j \leq m$, are disconnected in $G - S$. Hence, S is an edge multiway cut.

It remains to show that the weight of S is at most $7n+2m$. Since $\mathcal{A}$ satisfies each clause of Φ at least once, there are exactly m triangle-bases of weight 2 from the clause gadgets in S. Similarly, the variable gadgets contribute exactly n bases to S. Finally, for each of the $3n$ link structures, by the definition of S, either the two connector edges of the variable-triangle are in S or the two connector edges of the clause-triangle. Together, they contribute a weight of $6n$ to the total weight of S. Therefore, S is an edge multiway cut in G of weight at most $7n + 2m$. ◇

Conversely, assume that (G, T, k) is a YES instance of $\{1, 2, 3, 6\}$-EDGE MULTIWAY CUT. Hence, there exists an edge multiway cut of (G, T) of weight at most $7n + 2m$. We shall demonstrate an assignment that satisfies Φ. Before that, we shall discuss some structural properties of a minimum-weight edge multiway cut. In the following arguments, we assume that the clauses under consideration have size three, unless otherwise specified. While making the same arguments for clauses of size two is easier, we prefer to argue about clauses of size three for generality.

Claim 4.2.2 (adapted from [60])**.** *If e is an edge in G incident on a vertex v of degree > 2 such that e has weight greater than or equal to the sum of the other edges incident on v, then there exists a minimum-weight edge multiway cut in G that does not contain e.*

The above claim implies that no such edge e is contained in the solution. To see this, note that an iterative application of the local replacement used in Claim 4.2.2 would cause a conflict in the event that the replacement is cyclical. Suppose that the edges are replaced in the sequence $e \to e_1 \to \ldots \to e_r \to e$. Then the weight of e_1, denoted by $w(e_1)$ must be strictly less than the weight

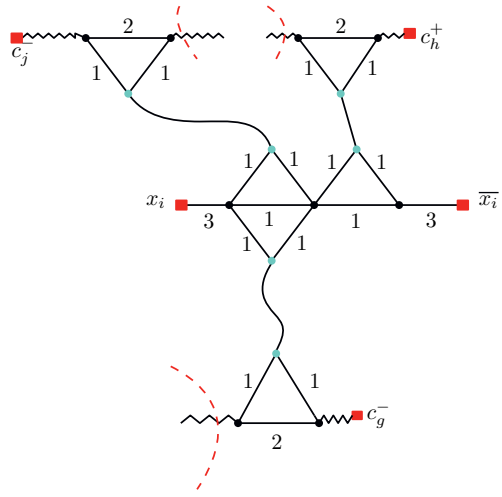

Figure 4.4: Shown in the figure is the variable interface of x_i. The positive literal x_i occurs in the clauses c_j and c_g, whereas $\overline{x_i}$ occurs in c_h. No terminal is reachable from the vertex closest to the red dashed lines in the direction of the paths crossed by it.

of e. Similarly, $w(e_i) < w(e_j)$ for $i < j$. This would mean that $w(e) < w(e)$, which is a contradiction.

Claim 4.2.3 ([60])**.** *If a minimum-weight edge multiway cut contains an edge of a cycle, then it contains at least two edges from that cycle.*

It follows from Claim 4.2.2 and the construction of G that there exists a minimum-weight edge multiway cut for (G, T) that neither contains the edges incident on the terminals nor does it contain the links. Among the minimum-weight edge multiway cuts that satisfy Claim 4.2.2, we shall select one that contains the maximum number of connector edges and from the ones that satisfy both the aforementioned properties, we shall pick one that contains the maximum number of triangle-bases from clause gadgets of size two. Let S be a minimum edge multiway cut that fulfills all these requirements.

We say a link e incident on a gadget *reaches* a terminal t if e is the first edge on a path P from the connector e in the gadget to t and no edge on P is contained in S.

A terminal t is *reachable* by a gadget if one of the links incident on the gadget reaches t. Note that, for any terminal t' in the gadget, if t is reached from some incident link by a path P, then P can be extended to a t'-t path in G using only edges inside the gadget. However, among the edges used by such an extension, at least one must belong to S, or $t = t'$.

Claim 4.2.4. *S contains exactly one base of a triangle from each variable gadget.*

Proof. Clearly, S must contain at least one base from each variable gadget, else by the fact that S contains no edges incident on terminals, a path between the terminals in such a gadget would remain in $G - S$.

Suppose that S contains two bases of some variable gadget, say that of x_i. By Claim 4.2.3, S must also contain at least three connector edges from this variable gadget: at least two connector edges (of the two triangles) of the diamond and at least one connector edge of the hat. We claim that, without loss of generality, at least all the outer connector edges must be in S. If for some triangle the outer connector edge next to terminal t is not in S, then the link incident on this triangle does not reach any terminal $t' \neq t$; otherwise, a t-t' path would remain in $G - S$, a contradiction. Hence, we simultaneously replace all inner connector edges for which the corresponding outer connector edge is not in S by their corresponding outer connector edge. For the resulting set S', the variable terminals of the gadget and their neighbors in G form a connected component of $G - S'$. Since the link incident on a triangle for which the outer connector edge (next to terminal t) was not in S does not reach any terminal $t' \neq t$, S' is feasible. Moreover, it has the same properties we demanded of S. Thus, henceforth, we may assume that all the outer connector edges of the x_i-gadget are in S.

We now distinguish six cases:

Case 1. *No link of the x_i gadget reaches a terminal.*
We can remove one of the two bases from S without connecting any terminal pairs. This is so because in order to disconnect x_i from $\overline{x_i}$, it suffices for S to contain either the base of the diamond along with the two outer connector edges or the base and outer connector edge of the hat. No other terminal pairs are connected via the gadget by the assumption of this case. Hence, we contradict the minimality of S. Figure 4.5 depicts this case.

Case 2. *A link of the x_i-gadget reaches at least two distinct terminals.*
By the definition of reaches, this implies that there is a path in $G - S$ between any two of the reached terminals. This contradicts that S is an edge multiway cut for (G, T). (See Figure 4.6)

Case 3. *Exactly one link e of the x_i-gadget reaches some terminal t.*
We remove from S the base of a triangle that is not attached to e and add the remaining connector edge of the triangle that is attached to e (if it is not already in S). Consequently, although e reaches t, both connector edges incident on e are in S. Since no other link reached any terminals and x_i remains disconnected from $\overline{x_i}$ in $G - S$, we can obtain an edge multiway cut

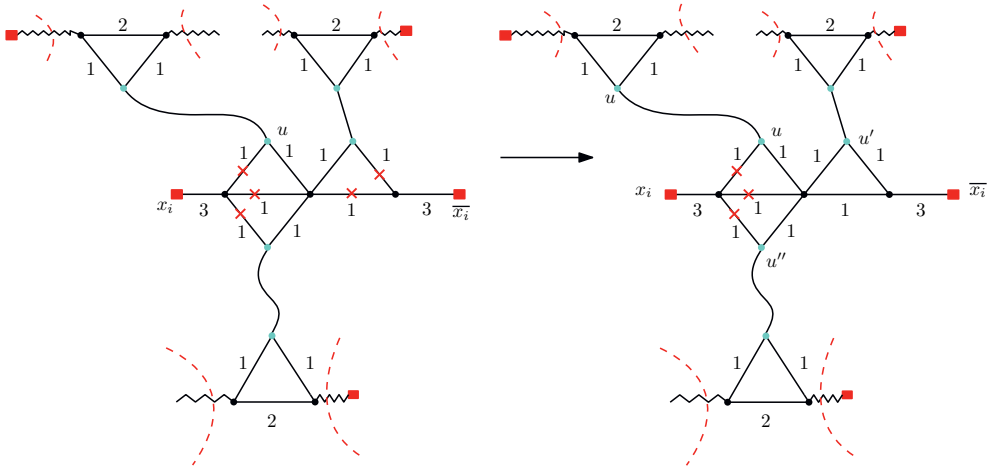

Figure 4.5: Case 1: In the figure on the left, we see the x_i-gadget with the three clause gadgets it is linked to. None of the links can reach any terminal. The red dashed curves indicate that the path is intersected by the multiway cut S. The edges labeled with a red cross are contained in S. In the right figure, we show how S can be modified without compromising its feasibility.

for (G, T) satisfying Claim 4.2.2 that has the same or less weight as S, but has strictly more connector edges than S. This is a contradiction to our choice of S. Figure 4.7 depicts this case.

Case 4. *Exactly two links e, e' of the x_i-gadget reach two distinct terminals t and t', respectively.*
Recall that all three outer connected edges are in S. Now at least one of the inner connector edges of the gadget must be in S, or else t would be connected to t' via this gadget. In particular, both the connector edges of at least one of the two triangles attached to e, e' must be in S. We can remove from S one of the two bases and add instead the remaining connector edge of the other triangle (if it is not already in S). Consequently, although e reaches t and e' reaches t', all connector edges incident on e and e' are in S. Moreover, x_i and $\overline{x_i}$ are not connected to each other in $G - S$, as one base and its corresponding outer connector(s) are still in S. The transformation results in an edge multiway cut for (G, T) satisfying Claim 4.2.2 that has the same or less weight than S, but has strictly more connector edges than S. This is a contradiction to our choice of S. Figure 4.8 shows the situation described above.

Case 5. *All the three links of the x_i-gadget reach distinct terminals t, t', t'', respectively.*
Recall that all three outer connected edges are in S. Now at most one (inner)

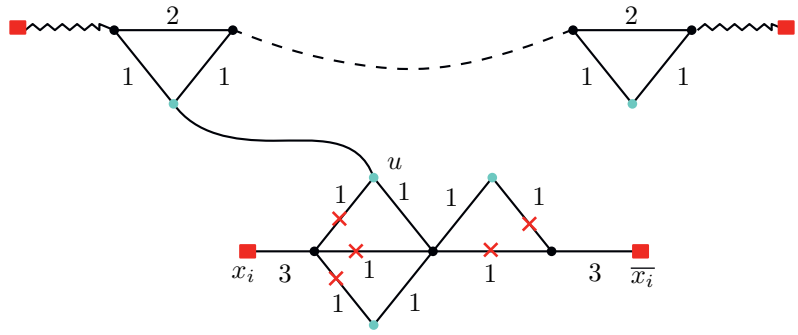

Figure 4.6: Case 2: In the figure we see the x_i-gadget with one of its links reaching two distinct terminals. The dashed curve shows that there exists a path between its endpoints.

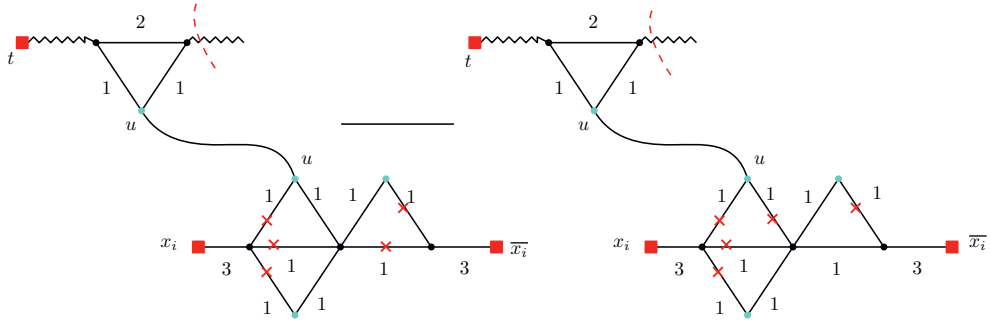

Figure 4.7: Case 3: The left figure shows the situation when exactly one link of the x_i-gadget reaches a terminal. The edges labeled with a red cross are contained in S. The right figure shows the replacement made in this case.

connector edge of the x_i-gadget is not in S, or else at least one pair of terminals among $\{(t, t'), (t', t''), (t'', t)\}$ would remain connected via the gadget. We replace one of the bases in S with this connector edge (if it is not already in S). The resulting edge multiway cut is no heavier. To see that it is also feasible, note that while t, t', t'' are still reached from the links of the gadget, all the connector edges of this gadget are in the edge multiway cut. The terminals x_i and $\overline{x_i}$ are disconnected from each other in $G - S'$ because one triangle-base and its connectors are still in the edge multiway cut. Hence, we obtain an edge multiway cut for (G, T) satisfying Claim 4.2.2 that has the same or less weight than S, but with strictly more connector edges than S, a contradiction to our choice of S. Figure 4.9 depicts this scenario.

Case 6. *At least two links of the x_i-gadget reach exactly one terminal t outside the gadget.*

Recall that every variable occurs in at least two clauses of size two. Hence, t is

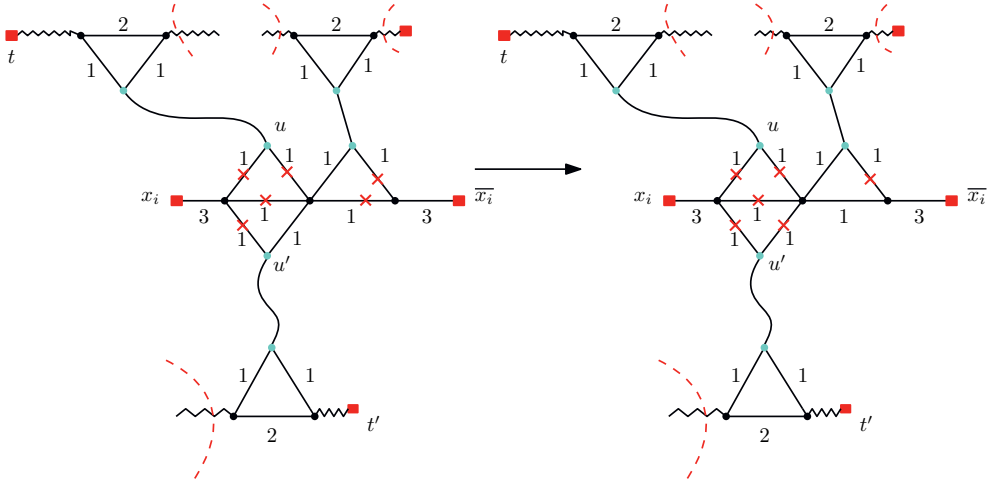

Figure 4.8: Case 4: The left figure shows the situation when exactly two links of the x_i-gadget reach two distinct terminals. The edges labeled with a red cross are contained in S. The right figure depicts the situation after the replacement.

reachable via a link from the x_i-gadget to at least one directly linked clause gadget of a clause of size two. Also recall that S is a minimum-weight edge multiway cut containing the maximum number of bases from clauses of size two.

Suppose that there exists a size-two clause gadget c, directly linked to the x_i-gadget, that does not contain t and via which t is reachable from the x_i-gadget. (See Figure 4.10). That is, some link reaches t via a path P that contains edges of c, but t is not in c. Then S must contain two base-connector pairs from c; else, some terminal of c would not be disconnected from t in $G - S$. Now remove from S the base of one of the two triangles of c and add the remaining two connector edges of c. This does not increase the weight, as the base of the clause-triangle has weight 2 and the connectors have weight 1 each. The only terminal pair that could get connected by the transformation is the pair of terminals on c itself. However, one of the bases is still in the transformed cut. This new cut contradicts our choice of S, as it has strictly more connector edges and satisfies the other conditions.

Suppose t is contained in one of the size-two clause gadgets, c', directly linked to the x_i-gadget. If the link between the x_i-gadget and c' is not one of the links meant in the assumption of this case, then the situation of the previous paragraph holds, and we obtain a contradiction. Thus, t is reachable from the x_i-gadget via both links of c'. Hence, a base-connector pair of the triangle of c' that t is not attached to must be in S. Consider the link of the

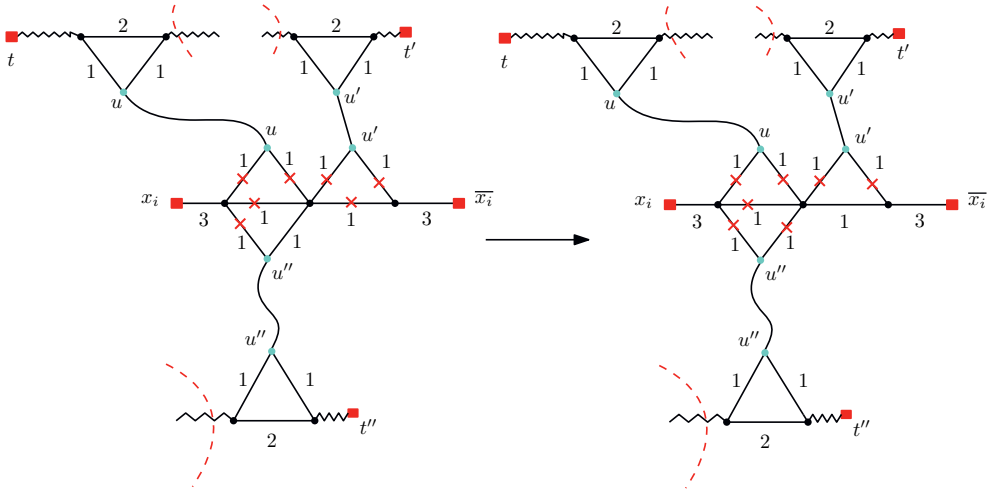

Figure 4.9: Case 5: The left figure shows the situation when all the three links of the x_i-gadget reach three distinct terminals. The edges labeled with a red cross are contained in S. The right figure shows the situation after the replacement.

x_i-gadget that is not attached to c' but reaches t and let P be a corresponding path, starting at this link and going to t. Note that P passes through a clause gadget c'' directly linked to the x_i-gadget. If c'' is a size-two clause gadget, then we obtain a contradiction as before. Hence, c'' corresponds to a size-three clause (as in Figure 4.11). Since P must either enter or leave c'' through one of its outer triangles, a base-connector pair of at least one outer triangle of c'' must be in S, or the attached terminal would reach t in $G - S$, contradicting that S is an edge multiway cut for (G, T). Let Λ be such an outer triangle (see Figure 4.11).

We argue that, without loss of generality, S contains a base-connector pair of the other outer triangle, Δ. Suppose not. Then, in particular, the base of Δ is not in S. If P passes through the link attached to Δ, then one of the endpoints of the base of Δ must be on P. Since the base of Δ is not in S, the terminal t'' next to Δ remains connected to t in $G - S$, a contradiction. Hence, P must either enter or exit c'' via the link attached to its middle triangle μ. Moreover, S must contain a base-connector pair of μ (see Figure 4.11), or t'' would still reach t in $G - S$. We now modify S to obtain a set S'. If both connector edges of Δ are in S, then replace the base of μ by the base of Δ to obtain S'. Then all edges of Δ are in S'. Otherwise, no edge of Δ is in S and thus no terminal is reachable via the link attached to Δ (or it would be connected to t'' in $G - S$). So, we replace the base-connector pair of μ by a base-connector pair of Δ to obtain S'. Then S' is an edge multiway cut for

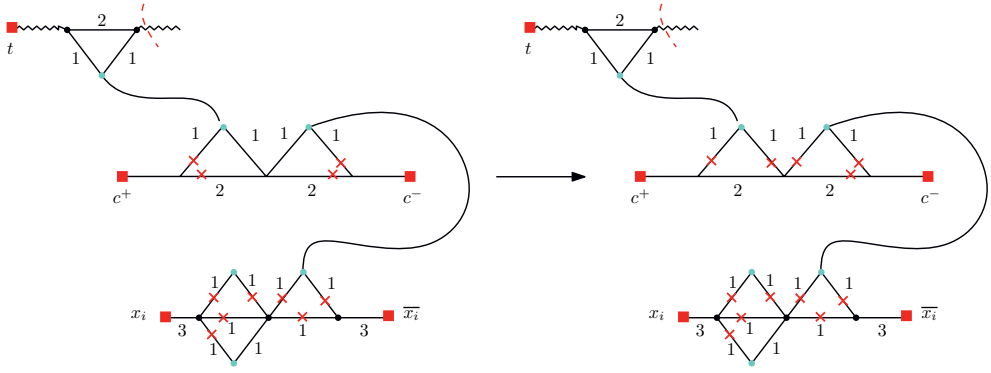

Figure 4.10: Case 6: The figure on the left shows the situation when the x_i-gadget reaches a terminal t via a clause gadget of size two. The dashed curve indicates that there exists a path between its endpoints that is not cut by S. The figure on the right depicts the situation after the replacement.

(G, T) of the same weight at S that has the same properties as S. Hence, we may assume $S = S'$. Then S contains a base-connector pair of Δ.

Now remove from S the base and connector edge of Λ. Then t and t' become connected to each other in $G - S$, but not to any other terminal, or that terminal would already be connected to t in $G - S$. Now add the base and outer connector edge of the triangle in c' that t is attached to. This restores that S is an edge multiway cut for (G, T). Since the edge multiway cut we obtain has the same weight as S, satisfies Claim 4.2.2, has no fewer connectors than S but contains at least one more base of a clause gadget of size two, we contradict our choice of S.

◇

We now focus on the link structures.

Claim 4.2.5. *There cannot exist a link structure in G that contributes less than two edges to S and for which the clause-triangle of the link structure contributes no connector edges to S.*

Proof. Towards a contradiction, suppose that such a link structure exists. Let the clause gadget containing the link structure be c and the variable gadget containing it be x_i. By Claim 4.2.4, we know that there exists a triangle of the x_i-gadget that does not contribute its base to S. Therefore, at least one terminal t of the x_i-gadget is reachable from the clause gadget c. This implies that the clause-triangle of the link structure is the middle triangle of c, or else there would exist a path in $G - S$ between t and the closest clause-terminal on c. Then, since S is feasible, it must contain the base and at least one

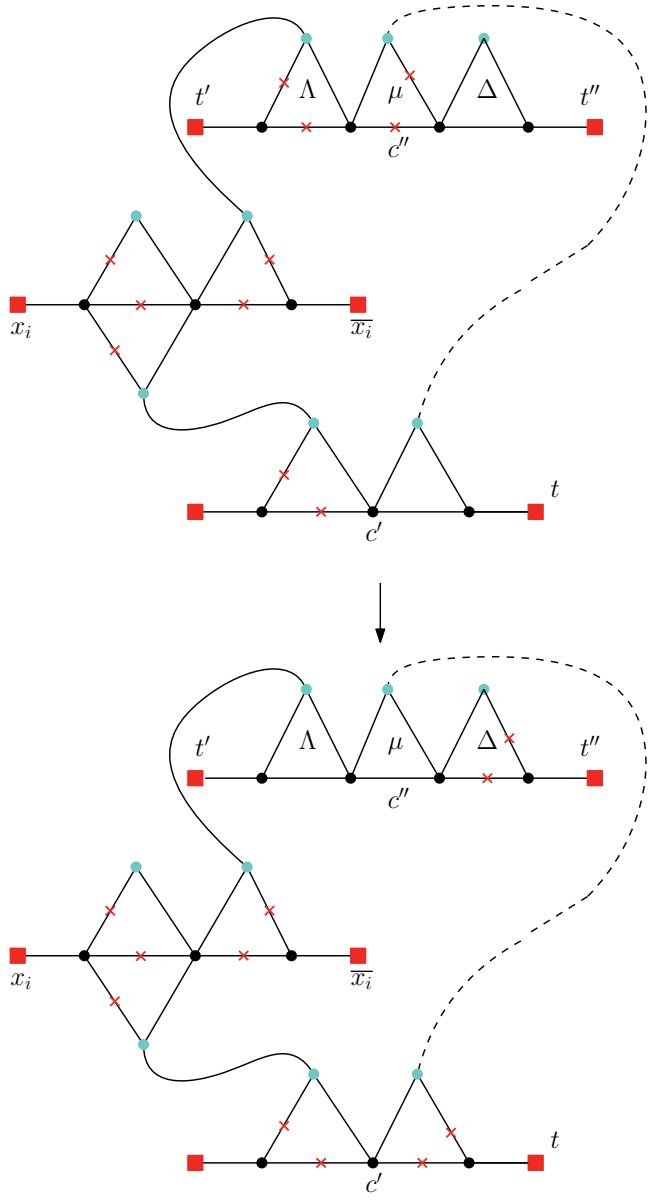

Figure 4.11: Case 6: In the top figure, there is a terminal t reachable via (at least) two links of the x_i-gadget. Moreover, t appears in a clause gadget c' corresponding to a clause of size two that is directly connected to the x_i-gadget. The dashed curve indicates the existence of a path, not cut by S, between its endpoints.

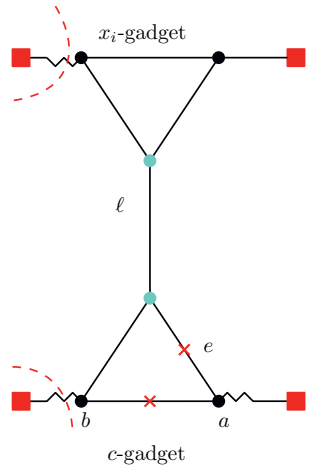

Figure 4.12: A link structure with the variable gadget of x_i at the top and its clause gadget for c at the bottom. The crossed-out edges are assumed to be in the minimum edge multiway cut S. The dashed red lines depict that the terminals cannot be reached from the vertices a or b.

connector edge of each of the two outer triangles of c. Else, at least one of the clause-terminals would be reachable from t in $G - S$.

It must also be the case that both connector edges of each of the outer triangles must be in S or the incident link reaches no terminal $t' \neq t$ is reachable from the incident link; otherwise, t or the incident clause-terminal would be connected to t' in $G - S$. Now, we can remove one of the two bases from S and add the two connector edges of the middle triangle, without compromising the feasibility of the edge multiway cut. Thus, there exists an edge multiway cut of no greater weight than S, satisfying Claim 4.2.2, and containing two more connector edges (those of the clause-triangle of the link structure). This is a contradiction to our choice of S. ◇

Claim 4.2.6. *S contains at least two edges from each link structure.*

Proof. Suppose that there exists a link structure ℓ that contributes less than two edges to S. Suppose that ℓ connects the clause gadget c and the variable gadget x_i. By Claim 4.2.5, we know that the clause-triangle of ℓ must contribute an edge e to S. Therefore, none of the connectors of the variable-triangle attached to ℓ are in S. As a result, the variable-terminal of the x_i-gadget attached to ℓ, say we call it t, is reachable from c via ℓ.

Claim 4.2.3 and the fact that only e is in S, the base of the clause-triangle must also be in S. We do the following replacement: remove from S the

base-connector pair of the clause-triangle and add the base and (possibly two) connectors of the variable-triangle of ℓ, as follows. If the variable-triangle of ℓ is part of a diamond, then we add to S the base and two outer connectors, thereby getting an edge multiway cut of equal weight but strictly more connectors. If the variable-triangle is a hat, then we add to S the base and outer connector of the hat, obtaining an edge multiway cut for (G, T) of strictly smaller weight than S. If we can show that the resultant edge multiway cut is feasible, we obtain a contradiction in either scenario. We claim that such a replacement does not compromise the feasibility of S.

Let a, b be the endpoints of the base of the clause-triangle of ℓ, where a is the endpoint on which e is incident (see Figure 4.12). Note that no terminal other than t should be reachable in $G - S$ from b; else, there would be a path from t to that terminal via ℓ. In particular, the terminal of the clause gadget for c on the side of b can not be reached in $G - S$ from the vertex b. By removing the base-connector pair of the clause-triangle of ℓ, we may expose the clause-terminal on the side of the vertex a (or another terminal outside c) to t. However, by adding the base and (possibly two) connectors closest to t, we disconnect any path between this terminal and t. Since we did not modify the cut in any other way, no new connections would have been made. This shows the feasibility of the resultant edge multiway cut and thus proves our claim.

◇

Claim 4.2.7. *If there exists an edge multiway cut of weight at most $7n + 2m$ for (G, T), then there exists a satisfying truth assignment for Φ.*

Proof. Let S be the edge multiway cut defined before. The immediate consequence of Claims 4.2.4 and 4.2.6 is that the weight of S is at least $n+2\cdot(3n) = 7n$. S must also contain at least one base per clause gadget lest the two terminals on a clause gadget remain connected. Therefore, its weight is at least $7n + 2m$. Since it is an edge multiway cut of weight at most $7n + 2m$, it has exactly one base per clause gadget.

We also claim that for each link structure, if one of the triangles attached to it has its base in S, then the other one cannot: note that if both the triangles had their bases in S, then each of them would also have a connector edge in S by Claim 4.2.3. By Claim 4.2.6 and the assumption that the weight of S is at most $7n + 2m$, the other two connector edges of the link structure are not in S. Since at most one base per variable/clause gadget can be in S, there would be a path between one of the variable-terminals and one of the clause-terminals in the linked gadgets through the link structure, a contradiction to S being an edge multiway cut for (G, T). Figure 4.13 shows one such case.

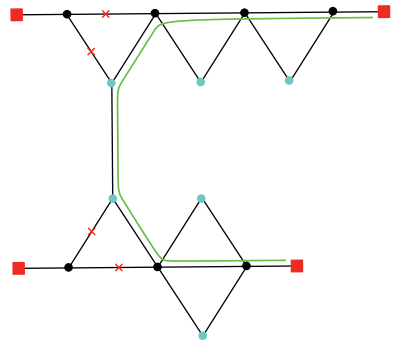

Figure 4.13: The figure shows a link structure with the variable gadget at the bottom and its connected clause gadget at the top. The crossed-out red edges are the ones contained in the minimum edge multiway cut S. The green curve shows the existence of a path between a variable-terminal and a clause-terminal.

We now define the truth assignment $\mathcal{A}$. For each variable-terminal, if the diamond has its base in S, we make it "false", otherwise if the hat has its base in S we make it "true". Each clause gadget has exactly one triangle contributing its base to S. From the above argument, we know that the variable-triangle linked to this clause-triangle must not contribute its base to S. Hence, every clause gadget is attached to one literal triangle such that its base is not in S, and is therefore "true". Hence, every clause is satisfied by the truth assignment $\mathcal{A}$ and Φ is a YES instance of Planar 2P1N-3SAT-2. ◇

The above implies that $\{1,2,3,6\}$-Edge Multiway Cut is NP-complete on planar subcubic graphs. We now proceed to prove that (unweighted) Edge Multiway Cut is NP-complete on planar subcubic graphs. The proof follows from the observation that the honeycombs of $\tilde{G}$ (defined before) do not contribute any edge to any minimum edge multiway cut for $(\tilde{G}, T)$.

Claim 4.2.8. *Any minimum edge multiway cut for $(\tilde{G}, T)$ does not contain any of the honeycomb edges.*

Proof. Let S' be a minimum edge multiway cut for $(\tilde{G}, T)$. Recall that $\tilde{G}$ is planar. Note that for any two vertices s, t, an s-t cut in a planar graph corresponds to a simple (possibly degenerate) cycle in the planar dual [167]. Therefore, the dual of an edge multiway cut comprises several cycles. Let the edges corresponding to S' in the planar dual of $\tilde{G}$ be S^*. In fact, S^* induces a planar graph such that exactly one terminal of T is embedded in the interior of each face of this graph. If any face of the S^* did not contain a terminal, we

could remove the edge in S' dual to one of the edges of this face. This would not connect any terminal pair, and hence contradicts the minimality of S'.

Suppose that S' contains some edges of the honeycomb in $\tilde{G}$ replacing the vertex $v \in V(G')$. We denote the intersection of S' with the edges of this honeycomb by S'_h. Let the set of edges dual to S'_h in be S^*_h. By abuse of notation, we also denote by S^*_h the graph formed by contracting all the edges in $S^* \setminus S^*_h$. Since each face of S^* encloses a terminal, each bounded face of S^*_h must enclose an attachment point of the honeycomb. If not, then we could remove from S' an edge in S'_h dual to some edge of the face of S^*_h not enclosing an attachment point. This does not make any new terminal-to-terminal connections, as the part of the honeycomb enclosed by this face does not contain any path to any of the terminals of T. This would be a contradiction to the minimality of S'.

Next, we observe that no bounded face of S^*_h can enclose more than one attachment point. Suppose that there exists a bounded face in S^*_h that encloses two attachment points. Since the two attachment points are separated by 100 cells of the honeycomb, the length of the face boundary must be at least 50. We could remove all the 50 edges from S' dual to the edges of the face boundary and add all the attaching edges to S', instead. All the terminal-to-terminal paths passing through the honeycomb will remain disconnected after the transformation. Since at most eight attaching edges can be added, we again get a contradiction to the minimality of S'. So, each bounded face of S^*_h must enclose exactly one attachment point.

To enclose the attachment points, each of these faces must cross the boundary of the honeycomb exactly twice. We claim that the faces of S^*_h, enclosing consecutive attachment points on the boundary of the honeycomb, are pairwise edge-disjoint. Suppose that the faces enclosing two consecutive attachment points, a and a', share an edge. Then, they must also share an edge that crosses the boundary of the honeycomb. If they do not, then let e be the last edge of the face enclosing a to cross the boundary and e' be the first edge of the face enclosing a' to cross the boundary of the honeycomb. The edges e and e' along with the other edges not shared between the respective face boundaries bound a region of the plane containing no attachment points, a contradiction!

Therefore, any two faces of S^*_h enclosing consecutive attachment points share an edge which crosses the boundary of the honeycomb. Without loss of generality, let this edge be closer to a. Then, the face enclosing a' must contain at least 50 edges as a and a' are separated by 100 cells of the honeycomb. This implies that S'_h contains at least 50 edges. However, we could remove from it all the 50 edges and add all the (at most eight) attaching edges. This cut is smaller and disconnects all the terminal-terminal paths passing through the

honeycomb. Once again, we contradict the minimality of S'.

Hence, all the faces in S_h^* enclosing attachment points are edge-disjoint. So, there are at least $2 \cdot \deg_{G'}(v)$ edges in S'_h. We could replace this cut by a smaller cut, namely, the edge multiway cut formed by removing the edges in S'_h from S' and adding to it all the attaching edges incident on the attachment points. This cut disconnects all terminal-paths passing through the honeycomb and yet, is smaller than S', a contradiction to its minimality. Hence, S' does not contain any edge of the honeycombs. ◇

By the construction of $\tilde{G}$ and Claims 4.2.1, 4.2.7, and 4.2.8, we conclude that Edge Multiway Cut is NP-complete on planar subcubic graphs. □

4.3. The Proofs of Theorem 4.1.2 and 4.1.3

We observe the hardness of Unrestricted Node Multiway Cut.

Theorem 4.1.2 (Restated)**.** Unrestricted Node Multiway Cut *is* NP*-complete for planar subcubic graphs.*

Proof. It is readily seen that Unrestricted Node Multiway Cut belongs to NP. We now reduce from Vertex Cover on planar subcubic graphs, which is known to be NP-complete [156]. Let G be the graph of an instance of this problem. We keep the same graph, but set $T = V(G)$. Since any two adjacent vertices are now adjacent terminals, any vertex cover in G corresponds to a node multiway cut for (G, T). The result follows. □

As a warm-up, we now observe the following easy result.

Proposition 4.3.1. Node Multiway Cut *is* NP*-complete for planar graphs of maximum degree* 4*.*

Proof. It is readily seen that Node Multiway Cut belongs to NP. We now reduce from Unrestricted Node Multiway Cut on planar subcubic graphs. Let (G, T, k) be an instance of this problem. Let G' be obtained from G by adding a pendant vertex v' per vertex $v \in T$. Let $T' = \{v' \mid v \in T\}$. If (G', T') has a node multiway cut $S \subseteq V(G') \setminus T'$, then S is immediately a node multiway cut for (G, T). Conversely, if (G, T) has a node multiway cut $S \subseteq V(G)$, then S is immediately a node multiway cut for (G', T') with $S \subseteq V(G') \setminus T'$. The result follows. □

To prove that Node Multiway Cut is NP-complete for planar subcubic graphs, we need the following lemma from [120].

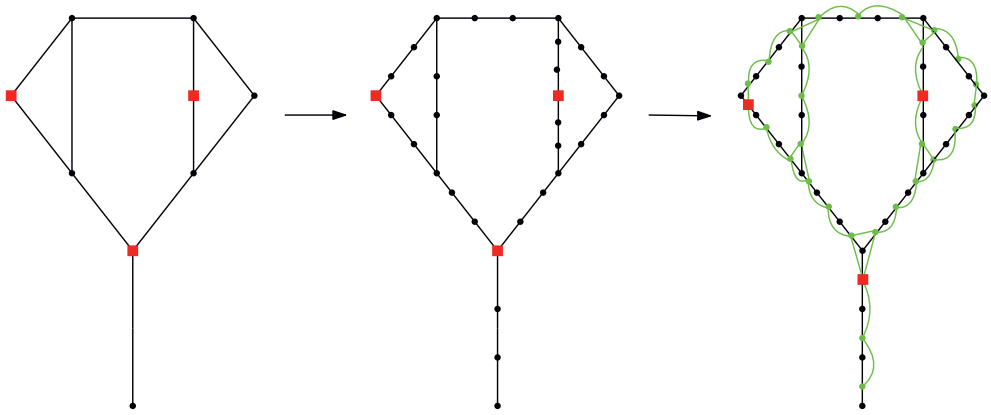

Figure 4.14: The figure shows the construction in Theorem 4.1.3. The leftmost figure is an instance of Edge Multiway Cut on planar subcubic graphs. The figure in between shows a 2-subdivision of the instance. The rightmost figure shows the line graph of the subdivided graphs drawn in green. In each figure, the terminals are shown as red squares.

Lemma 4.3.2. *If* Edge Multiway Cut *is* NP*-complete for a class* $\mathcal{H}$ *of graphs, then it is also* NP*-complete for the class of graphs consisting of the* 1*-subdivisions of the graphs of* $\mathcal{H}$*.*

Proof. Let G belong to $\mathcal{H}$ and T be a set of terminals in G. Let G' be the graph G after subdividing each edge. For each edge e in G, there exist two edges in G'. If an edge of G is in an edge multiway cut for (G, T), then it suffices to replace it by only one of the two edges created from it in G' to disconnect the path e lies on. This yields an edge multiway cut for (G', T) of the same size. Conversely, if an edge of G' is in an edge multiway cut for (G', T), then we replace it by the corresponding original edge of G. This yields an edge multiway cut for (G, T) of the same size. Hence, (G, T) has an edge multiway cut of size at most k if and only if (G', T) has an edge multiway cut of size k. □

We are now ready to prove Theorem 4.1.3.

Theorem 4.1.3 (Restated)**.** Node Multiway Cut *is* NP*-complete for planar subcubic graphs.*

Proof. It is readily seen that Node Multiway Cut belongs to NP. In Theorem 4.1.1, we showed that Edge Multiway Cut is NP-complete on the class of planar subcubic graphs. We will now reduce Node Multiway Cut from

Edge Multiway Cut restricted to the class of planar subcubic graphs. Let G be a planar subcubic graph with a set of terminals T.

From G, we create an instance of Node Multiway Cut by the following operations; here, the *line graph* of a graph $G = (V, E)$ has E as vertex set and for every pair of edges e and f in G, there is an edge between e and f in the line graph of G if and only if e and f share an end-vertex.

- We construct the 2-subdivision of G, which we denote by G'.
- Next, we construct the line graph of G', which we denote by L.
- Finally, we create the terminal set of L as follows: for each terminal t in G', consider the edges incident on it. In the line graph L, these edges must form a clique, K_i for $i \in \{1, 2, 3\} : i = \deg(t)$. In this clique, we pick one vertex and make it a terminal. We denote the terminal set in L by T_L.

Note that L is planar, as G' is planar and every vertex in G' has degree at most 3 [172]. Note also that L is subcubic, as every edge in G' has one end-vertex of degree 2 and the other end-vertex of degree at most 3. Moreover, L and T_L can be constructed in polynomial time.

Claim 4.3.3. *There exists an edge multiway cut of (G, T) of size at most k if and only if there exists a node multiway cut of (L, T_L) of size at most k.*

Proof. We assume that (G, T) has an edge multiway cut S of size at most k. By Lemma 4.3.2, G' also has an edge multiway cut of size at most k. We claim that there exists an edge multiway cut S' of G' of size at most k which does not contain any edge incident on a terminal. Every edge in G' is adjacent to some edge with both its ends having degree two. Therefore, if an edge in the edge multiway cut of G' is incident on a terminal, we can replace it with its adjacent edge, which disconnects all the paths disconnected by the former and does not increase the size of the edge multiway cut. Now, for each edge in S' we add its corresponding vertex in L to a set S_L. Since S' pairwise disconnects the terminals in G', S_L disconnects all the terminal cliques from each other. Therefore, S_L is a node multiway cut of L.

Conversely, let $S'_L \subseteq V(L) \setminus T_L$ be a node multiway cut of (L, T_L) of size at most k. By similar arguments as above, we may assume that S'_L does not contain any vertex from any terminal-clique. We claim that G has an edge multiway cut of size at most k. To that end, we show that G' has an edge multiway cut of size at most k and apply Lemma 4.3.2 to prove the same for G. We add to the edge multiway cut S the edges of G' that correspond to the vertices in S'_L. The size of S is clearly at most k. To see that it is an edge

multiway cut of G', note that pairwise disconnecting the terminal-cliques of L amounts to pairwise disconnecting the set of edges incident on any terminal in G' from its counterparts. This, in turn, pairwise disconnects all the terminals in G'. ◇

By our construction and Claim 4.3.3, Node Multiway Cut is NP-complete on the class of planar subcubic graphs. □

4.4. Conclusions

We proved that Edge Multiway Cut and both versions of Node Multiway Cut are NP-complete for planar subcubic graphs. We immediately have the following dichotomy.

Corollary 4.4.1. *For every $\Delta \geq 1$,* Edge Multiway Cut *and both versions of* Node Multiway Cut *on graphs of maximum degree Δ are polynomial-time solvable if $\Delta \leq 2$, and* NP*-complete if $\Delta \geq 3$.*

From a result of Robertson and Seymour [169], it follows that any problem Π that is NP-hard on subcubic planar graphs but polynomial-time solvable for graphs of bounded treewidth can be fully classified on $\mathcal{H}$-topological minor-free graphs. Namely, Π is polynomial-time solvable if $\mathcal{H}$ contains a subcubic planar graph and NP-hard otherwise. It is known that Edge Multiway Cut and both versions of Node Multiway Cut satisfy the second property [9]. As Theorems 4.1.1–4.1.3 show the first property, we obtain the following dichotomy.

Corollary 4.4.2. *For every set of graphs $\mathcal{H}$,* Edge Multiway Cut *and both versions of* Node Multiway Cut *on $\mathcal{H}$-topological-minor-free graphs are polynomial-time solvable if $\mathcal{H}$ contains a planar subcubic graph, and* NP*-complete otherwise.*

Let the *ℓ-subdivision* of a graph G be the graph obtained from G after replacing each edge uv by a path of length $\ell + 1$ with end-vertices u and v. A problem Π is NP-hard *under edge subdivision of subcubic graphs* if for every integer $j \geq 1$ there is an $\ell \geq j$ such that: if Π is NP-hard for the class $\mathcal{G}$ of subcubic graphs, then Π is NP-hard for the class G^ℓ consisting of the ℓ-subdivisions of the graphs in $\mathcal{G}$. Now say that Π is polynomial-time solvable on graphs of bounded treewidth and NP-hard for subcubic graphs and under edge subdivision of subcubic graphs. The meta-classification from [120] states that for every *finite* set $\mathcal{H}$, Π on $\mathcal{H}$-subgraph-free graphs is polynomial-time solvable if $\mathcal{H}$ contains a graph from $\mathcal{S}$, and NP-hard otherwise. Here, $\mathcal{S}$ is the set consisting of all disjoint unions of zero or more paths and subdivided claws (4-vertex stars

in which edges may be subdivided). Results from [9, 120] show the first two properties. Theorems 4.1.1–4.1.3 show the last property. Thus, we obtain:

Corollary 4.4.3. *For every finite set of graphs $\mathcal{H}$,* Edge Multiway Cut *and both versions of* Node Multiway Cut *on $\mathcal{H}$-subgraph-free graphs are polynomial-time solvable if $\mathcal{H}$ contains a graph from $\mathcal{S}$, and* NP*-complete otherwise.*

The last dichotomy result assumes that $\mathcal{H}$ is a finite set of graphs. We therefore pose the following challenging question: classify the complexity of Edge Multiway Cut and both versions of Node Multiway Cut for $\mathcal{H}$-subgraph-free graphs when $\mathcal{H}$ is infinite. Answering this question requires novel insights into the structure of $\mathcal{H}$-subgraph-free graphs.

III

Forbidden Subgraphs

Overview

Algorithmic meta-theorems are general algorithmic results applying to a whole range of problems, rather than a single problem alone [136]. An *algorithmic meta-theorem* is a statement saying that all problems sharing some property or properties P, restricted to a class of inputs I, can be solved efficiently by a certain form of algorithm. Probably the most famous algorithmic meta-theorem is that of Courcelle [56], which proves that every graph property expressible in monadic second-order logic is decidable in linear time if restricted to graphs of bounded treewidth (see Chapter 2 for a definition of treewidth). Another example is that of Seese [173], which proves that every graph property expressible in first-order logic is decidable in linear time when restricted to graphs of bounded degree. A third example comes from Dawar *et al.* [62], who proved that every first-order definable optimization problem admits a polynomial-time approximation scheme on any class of graphs excluding at least one minor. There is a wealth of further algorithmic meta-theorems (see, for example, [23, 65, 85]), many of which combine structural graph theory (e.g. from graph minors) with logic formulations or other broad problem properties (such as bidimensionality).

An extension of an algorithmic meta-theorem is a so-called *algorithmic meta-classification*. This is a general statement saying that all problems that share some property or properties P admit, over some classes of input restrictions I, a classification according to whether or not they have property S. If the input-restricted class has property S, then this problem is "efficiently solvable"; otherwise it is "computationally hard". Throughout, we let these two notions depend on context; for example, efficiently solvable and computationally hard could mean being solvable in polynomial time and being NP-complete, respectively.

Algorithmic meta-classifications are less common than algorithmic meta-theorems, but let us mention two famous results. Grohe [105] proved that there is a polynomial-time algorithm for finite-domain constraint satisfaction problems whose left-hand input structure is restricted to $\mathcal{C}$ if and only if every graph in $\mathcal{C}$ is homomorphically equivalent to a graph that has bounded treewidth (assuming W[1] $\neq$ FPT). Bulatov [35] and Zhuk [195] proved that for

any finite constraint language Γ over a finite set, CSP(Γ) is either polynomial-time solvable or NP-complete, that is, it cannot belong to any intermediate complexity class (which are known to exist if $\mathsf{P} \neq \mathsf{NP}$, due to Ladner's [138] theorem).

Two well-known meta-classifications apply to the classes of $\mathcal{H}$-minor-free graphs and $\mathcal{H}$-topological-minor-free graphs. For a set $\mathcal{H}$ of graphs, these are the class of graphs G where, starting from G, no graph $H \in \mathcal{H}$ can be obtained by a series of vertex deletions, edge deletions, and edge contractions, respectively a series of vertex deletions, edge deletions, and vertex dissolutions (see Chapter 2 for full definitions). Both are a consequence of a classic result of [169].

Theorem 5.1.4. *Let Π be a problem that is computationally hard on planar graphs, but efficiently solvable for every graph class of bounded treewidth. For any set of graphs $\mathcal{H}$, the problem Π on $\mathcal{H}$-minor-free graphs is efficiently solvable if $\mathcal{H}$ contains a planar graph (or equivalently, if the class of $\mathcal{H}$-minor-free graphs has bounded treewidth) and is computationally hard otherwise.*

Theorem 5.1.5. *Let Π be a problem that is computationally hard on planar subcubic graphs, but efficiently solvable for every graph class of bounded treewidth. For any set of graphs $\mathcal{H}$, the problem Π on $\mathcal{H}$-topological-minor-free graphs is efficiently solvable if $\mathcal{H}$ contains a planar subcubic graph (or equivalently, if the class of $\mathcal{H}$-topological-minor-free graphs has bounded treewidth) and is computationally hard otherwise.*

Later, we will discuss many problems that satisfy the conditions of Theorems 5.1.4 and 5.1.5. We refer, for example, to [87, 158] for a number of problems that satisfy the conditions of Theorem 5.1.5, and thus also of Theorem 5.1.4, and that are NP-complete even for planar subcubic graphs of high girth.

On the other end of the spectrum lie the classes of $\mathcal{H}$-free graphs (or hereditary graph classes). A graph G is $\mathcal{H}$-free if, starting from G, no graph $H \in \mathcal{H}$ can be obtained by a series of vertex deletions. Hereditary graph classes are much more complex in structure than $\mathcal{H}$-minor-free graphs and $\mathcal{H}$-topological-minor-free graphs, and there exist infinite antichains (such as the set of cycles) under the induced subgraph relation. This makes the task of finding algorithmic meta-classifications much harder. In fact, even algorithmic meta-theorems are difficult to obtain for the induced subgraph relation, even for a single forbidden graph H. Indeed, complexity dichotomies for H-free graphs are rare and only known for specific problems (see e.g. [25, 102, 123, 127]).

Despite the above, some attempts have been made to study complexity boundaries, e.g. through the notion of boundary graph classes [5] (see also [7,

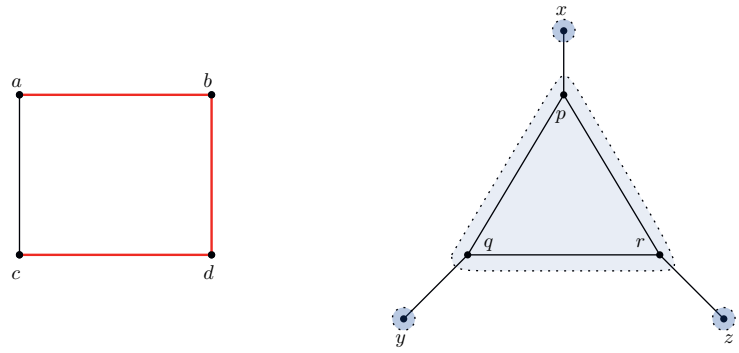

Figure 5.15: The left example shows that the C_4 is not P_4-subgraph-free (the red edges correspond to a P_4 subgraph), but it is P_4-free. The right example shows that the net is not $K_{1,3}$-minor-free (the vertex sets indicated in blue correspond to a $K_{1,3}$ minor), but it is $K_{1,3}$-topological-minor-free.

132, 158]). However, we are far from a broad understanding. For example, after more than forty years of research on Independent Set for H-free graphs starting from the work of Alekseev [5], we currently only know a trichotomy between being polynomial-time solvable, quasi-polynomial-time solvable and NP-complete (see the recent work of Gartland *et al.* [99]). We do not yet know how to obtain the dichotomy between polynomial-time solvable and NP-complete (see [106] for the most recent progress). Many other fundamental problems are still far from being settled for H-free graphs with infinitely many open cases even when H is a connected graph.

Between $\mathcal{H}$-minor-free graphs and $\mathcal{H}$-topological-minor-free graphs on the one side and $\mathcal{H}$-free graphs on the other side, lies the class of *$\mathcal{H}$-subgraph-free graphs*. These are the graphs G where, starting from G, no graph $H \in \mathcal{H}$ can be obtained by a series of vertex or edge deletions. In general, for every set $\mathcal{H}$ of graphs, the following holds (see also Figure 5.15 for some small examples):

$\mathcal{H}$-minor-free graphs $\subseteq$ $\mathcal{H}$-topological-minor-free graphs $\subseteq$ $\mathcal{H}$-subgraph-free graphs $\subseteq$ $\mathcal{H}$-free graphs.

Forbidden subgraphs represent many rich graph classes. To explain this, let P_r, K_r and C_r denote the path, complete graph and cycle on r vertices, respectively, and let $K_{p,q}$ denote the complete bipartite graph whose two partition classes each have size p and q, respectively. It is readily seen that, for example:

- the classes of graphs of maximum degree at most r and $K_{1,r+1}$-subgraph-free graphs coincide;
- the class of graphs with girth larger than g for some integer $g \geq 3$

coincides with the class of $(C_3, \ldots, C_g)$-subgraph-free graphs (and with the class of $(C_3, \ldots, C_g)$-free graphs);

- a class of graphs $\mathcal{G}$ has bounded treedepth if it is a subclass of P_r-subgraph-free graphs for some constant r, and vice versa [159]; and
- for every class $\mathcal{G}$ of degenerate or nowhere dense graphs [160], there exists an integer t such that every $G \in \mathcal{G}$ is $K_{t,t}$-subgraph-free (see [181] for a proof).

Moreover, H-free graphs and H-subgraph-free graphs coincide if and only if $H = K_r$ for some integer $r \geq 1$. This leads to a rich structural landscape.

A substantial body of work has studied the parameterized complexity of graph problems on a restricted set of subgraph-free graph classes (notably through the lens of sparsity, see e.g. [175]). However, $\mathcal{H}$-subgraph-free graphs have been significantly less studied in the context of classical complexity theory than the other classes, despite capturing many natural graph classes. This warrants a more in-depth look at $\mathcal{H}$-subgraph-free graphs.

Adding to this, $\mathcal{H}$-subgraph-free graphs seem to exhibit extreme and unexpected jumps in problem complexity. For example, there exist problems that are PSPACE-complete in general but constant-time solvable for every $\mathcal{H}$-free graph class [147] and thus for every $\mathcal{H}$-subgraph-free graph class, where $\mathcal{H}$ is any (possibly infinite) nonempty set of graphs. Another example is the Clique problem, which is to decide for a given integer k and graph G, if G contains a *clique* (set of pairwise adjacent vertices) of size at least k. The Clique problem is well-known to be NP-hard (see [93]). However, for $\mathcal{H}$-subgraph-free graphs, the situation drastically changes. The reason is that the size of a largest clique is bounded by the number of vertices of a smallest graph in $\mathcal{H}$ and hence, one can just apply brute force to find a largest clique in an $\mathcal{H}$-subgraph-free graph in polynomial time. Hence, the following holds.

Observation 5.1.6. *For every set of graphs $\mathcal{H}$,* Clique *is polynomial-time solvable for $\mathcal{H}$-subgraph-free graphs.*

In contrast to $\mathcal{H}$-free graphs, some work has pointed to more complex dichotomy results being possible. Kamiński [123] gave a complexity dichotomy for Max-Cut restricted to $\mathcal{H}$-subgraph-free graphs, where $\mathcal{H}$ is any finite set of graphs. Twenty years earlier, Alekseev and Korobitsyn [6] did the same for Independent Set, Dominating Set and Long Path; see [103] for a short, alternative proof (similar to the one of [123] for Max-Cut) for the classification for Independent Set for H-subgraph-free graphs. In [102] the computational complexity of List Coloring for $\mathcal{H}$-subgraph-free graphs has been determined for every finite set of graphs $\mathcal{H}$. More recently, Bodlaender *et*

al. [24] determined the computational complexity of Subgraph Isomorphism for H-subgraph-free graphs for all connected graphs H except the case where $H = P_5$, and they reduced all open "disconnected" cases to either $H = P_5$ or $H = 2P_5$. However, even for a classical problem such as Coloring, a complete complexity classification for H-subgraph-free graphs is far from settled [103]. Many more problems have not been studied in this context at all.

Motivated by our apparent lack of understanding of $\mathcal{H}$-subgraph-free graphs, we embark on a deeper investigation of the computational complexity of graph problems restricted to $\mathcal{H}$-subgraph-free graphs. In this way, we will pioneer a new meta-classification of $\mathcal{H}$-subgraph-free graphs, which is only the third meta-classification for graph containment apart from Theorems 5.1.4 and 5.1.5. Besides the aforementioned complexity dichotomies from [6, 103, 123], we will show that many other problems are covered by this meta-classification. In Chapter 5, we will survey and apply known results from the literature and also prove some new results.

Outline In this part, we shall discuss in detail the complexity framework for classes of forbidden subgraphs. We shall define the framework in Chapter 5 and prove a complexity dichotomy for all the problems that lie within the framework. We shall also compare our framework for forbidden subgraphs to its forbidden minors and topological-minors counterparts and discuss some limitations of our framework. Following this, in Chapter 6, we try to classify the problems that flout exactly one condition of our framework, namely, NP-hardness on the class of subcubic graphs. Some examples of such problems are Feedback Vertex Set, Independent Feedback Vertex Set, Coloring, Connected Vertex Cover and Matching Cut. We examine the complexity of these problems on the class of H-subgraph-free graphs, where H is a connected graph.

5

Complexity Framework for Subgraph Free Graphs

For a set of graphs $\mathcal{H}$, a graph G is *$\mathcal{H}$-subgraph-free* if G does not contain any graph from $\mathcal{H}$ as a subgraph. We propose general and easy-to-state conditions on graph problems that explain a large set of results for $\mathcal{H}$-subgraph-free graphs. Namely, a graph problem must be efficiently solvable on graphs of bounded treewidth, NP-hard on subcubic graphs, and NP-hard on subdivided subcubic graphs. Our meta-classification says that if a graph problem Π satisfies all three conditions, then for every finite set $\mathcal{H}$, it is "efficiently solvable" on $\mathcal{H}$-subgraph-free graphs if $\mathcal{H}$ contains a disjoint union of one or more paths and subdivided claws, and Π is "computationally hard" otherwise.

We apply our *meta-classification* on many well-known problems to obtain a dichotomy between polynomial-time solvability and NP-completeness. Apart from capturing numerous explicitly and implicitly known results in the literature, we also prove a number of new results. Moreover, we perform an extensive comparison between the subgraph framework and the existing frameworks for the minor and topological minor relations, and pose several new open problems and research directions.

Before we define the framework, we first recall some terminology. A class of graphs has bounded treewidth if there is a constant c such that every graph in it has treewidth at most c. A graph is subcubic if every vertex has degree at most 3. For an integer $\ell \geq 1$, the *ℓ-subdivision* of an edge $e = uv$ of a graph replaces e by a path of length $\ell + 1$ with endpoints u and v (and ℓ new vertices). The *ℓ-subdivision* of a graph G is the graph obtained from G after ℓ-subdividing each edge (see Figure 5.1 for an example of a 2-subdivision). For a graph class $\mathcal{G}$ and an integer ℓ, let $\mathcal{G}^\ell$ consist of the ℓ-subdivisions of the graphs in $\mathcal{G}$.

A graph problem Π is computationally hard *under edge subdivision of subcubic graphs* if for every integer $j \geq 1$ there is an integer $\ell \geq j$ such that: if Π is computationally hard for the class $\mathcal{G}$ of subcubic graphs, then Π is computationally hard for $\mathcal{G}^\ell$. Commonly, we can prove the condition holds by showing that computational hardness is maintained under ℓ-subdivision for a small integer ℓ (e.g. $\ell = 1, 2, 3, 4$) and then repeatedly apply the ℓ-subdivision operation.

5.1. The Meta-classification for $\mathcal{H}$-subgraph-free graphs

Our framework contains every graph problem Π satisfying the following three conditions:

C1. Π is efficiently solvable for every graph class of bounded treewidth;

C2. Π is computationally hard for the class of subcubic graphs; and

C3. Π is computationally hard under edge subdivision of subcubic graphs.

A problem Π that satisfies conditions C1–C3 is called a *C123-problem*. We defer to Section 5.4.2 the reasons why we cannot simplify condition C3 and, particularly, why being subcubic is important in this condition.

For some $p, q, r \geq 1$, the *subdivided* claw $S_{p,q,r}$ is obtained from the *claw* (the 4-vertex star $K_{1,3}$) after $(p-1)$-, $(q-1)$-, and $(r-1)$-subdividing its three edges respectively. The *disjoint union* $G_1 + G_2$ of two vertex-disjoint graphs G_1 and G_2 is the graph $(V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$. We now define the set $\mathcal{S}$, which is well known in the literature and also plays an important role in this chapter; see the left side of Fig. 5.1 for an example of a graph that belongs to $\mathcal{S}$.

Definition 5.1.1. *The set $\mathcal{S}$ consists of all non-empty disjoint unions of zero or more subdivided claws and paths.*

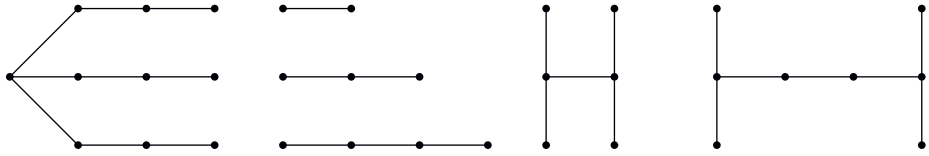

Figure 5.1: Left: An example of a graph in $\mathcal{S}$ (the graph $S_{3,3,3} + P_2 + P_3 + P_4$); also note that $S_{3,3,3}$ is the 2-subdivision of $K_{1,3}$. Right: the graphs $\mathbb{H}_1$ and $\mathbb{H}_3$, where $\mathbb{H}_1$ is the "H"-graph, formed by an edge (the *middle edge*) joining the middle vertices of two P_3s, and $\mathbb{H}_i$ ($i \geq 2$) is obtained from $\mathbb{H}_1$ by $(i-1)$-subdividing the middle edge.

Our main result is the following theorem that can be seen as the "subgraph variant" of Theorems 5.1.4 and 5.1.5. Note that it suggests, just like Theorems 5.1.4 and 5.1.5, that boundedness of treewidth might be the underlying explanation for the polynomial-time solvability.

Theorem 5.1.2. *Let* Π *be a C123-problem. For any* finite *set of graphs* $\mathcal{H}$*, the problem* Π *on* $\mathcal{H}$*-subgraph-free graphs is efficiently solvable if* $\mathcal{H}$ *contains a graph from* $\mathcal{S}$ *(or equivalently, if the class of* $\mathcal{H}$*-subgraph-free graphs has bounded treewidth) and computationally hard otherwise.*

We prove Theorem 5.1.2 as a consequence of a stronger result in Section 5.2. Next, in Section 5.3, we apply our subgraph framework to a wealth of problems as described above. We provide a discussion on limitations of the framework in Section 5.4, and an extensive comparison of the applicability of the three meta-classifications (Theorem 5.1.4, 5.1.5, and 5.1.2) in Section 5.5. Finally, we conclude this chapter with a list of open problems and research directions in Section 6.7.

5.1.1. Impact

The impact of the subgraph framework is three-fold. These impacts follow from the broad overview of the literature provided in this chapter on problems that exhibit zero or more of the properties C1, C2, C3.

First and foremost, we are able to provide a complete dichotomy for many problems on $\mathcal{H}$-subgraph-free graphs by showing they are C123-problems. In this way, we obtain a dichotomy between polynomial-time solvability and NP-completeness for many well-known partitioning, covering and packing problems, network design problems and width parameter problems. The applications of Theorem 5.1.2, as well as a number of applications of Theorem 5.1.4 and 5.1.5, are summarized in Table 5.1. A detailed comparison is deferred to Section 5.5.

The second impact of our framework is that we uncover several open questions in the literature and we subsequently resolve them. In [120] we prove,

graph problem	MF	TMF	SF	plan.	subc. plan.	C1	C2	C3
Pathwidth	✓	✓	✓	[157]	[157]	[22]	[157]	T 5.3.1
Tree-Width	?	?	✓	?	?	[21]	[26]	T 5.3.3
Dominating Set	✓	✓	✓	[93]	[93]	[8]	[93]	[50]
Independent Dominating Set	✓	✓	✓	[52]	[52]	[180]	[52]	[50]
Edge Dominating Set	✓	✓	✓	[193]	[193]	[9]	[193]	[50]
Independent Set	✓	✓	✓	[156]	[156]	[8]	[156]	[166]
Vertex Cover	✓	✓	✓	[156]	[156]	[8]	[156]	[166]
Connected Vertex Cover	✓	✗	✗	[92]	no	[9]	no	triv
Feedback Vertex Set	✓	✗	✗	[176]	no	[9]	no	triv
Independent Feedback Vertex Set	✓	✗	✗	[176]	no	[177]	no	triv
Odd Cycle Transversal	✓	✓	✓	[120]	[120]	[9]	[120]	[120]
Independent Odd Cycle Transv.	✓	✓	✓	[120]	[120]	[9]	[120]	[120]
C_5-Coloring	✓	✓	✗	[144]	[120]	[64]	[91]	no
3-Coloring	✓	✗	✗	[101]	no	[8]	no	triv
Star 3-Coloring	✓	✓	✗	[4]	[28]	[56]	[28]	no
List Coloring	✓	✓	✓	[119]	[120]	[120]	[120]	[120]
P_3-Factor	✓	✓	✓	[190]	[190]	[7]	[7]	[7]
Edge Steiner Tree	✓	✓	✓	[92]	T 5.3.7	[9]	T 5.3.7	T 5.3.7
Node Steiner Tree	✓	✓	✓	[92]	T 5.3.7	[9]	T 5.3.7	T 5.3.7
Steiner Forest	✗	✗	✗	[92]	T 5.3.7	no	T 5.3.7	T 5.3.7
Disjoint Paths	✓	✓	✓	[153]	[153]	[171]	[153]	T 5.3.12
Induced Disjoint Paths	✓	✓	✓	T 5.3.12	T 5.3.12	[171]	T 5.3.12	T 5.3.12
Long Cycle	✓	✓	✓	[94]	[94]	[20]	[94]	T 5.3.13
Long Induced Cycle	✓	✓	✓	[94]	[94]	[116]	T 5.3.13	T 5.3.13
Hamilton Cycle	✓	✓	✗	[94]	[94]	[8]	[96]	no
Long Path	✓	✓	✓	[94]	[94]	[20]	[94]	T 5.3.13
Long Induced Path	✓	✓	✓	T 5.3.13	T 5.3.13	[116]	T 5.3.13	T 5.3.13
Hamilton Path	✓	✓	✗	[94]	[94]	[8]	[96]	no
Max-Cut	✗	✗	✓	no	no	[9]	[192]	[123]
Edge Multiway Cut	✓	✓	✓	[122]	[122]	[66]	[122]	T 5.3.9
Node Multiway Cut	✓	✓	✓	[122]	[122]	[9]	[122]	T 5.3.9
Matching Cut	✓	✗	✗	[31]	no	[31]	no	triv
Perfect Matching Cut	✓	✓	✓	[30]	[30]	[139]	[139]	[81]
Diameter	✗*	✗*	✓	no	no	[1]	[80]	[120]
Radius	✗*	✗*	✓	no	no	[1]	[80]	[120]
Subgraph Isomorphism	✗	✗	✗	[24]	[24]	no	[24]	triv
Clique	✗	✗	✗	no	no	triv	no	triv

Table 5.1: The minor framework (MF), topological minor framework (TMF), and subgraph framework (SF), with the conditions. The problems are mainly chosen to illustrate the wide reach of the frameworks and their differences. If a problem satisfies the conditions of a meta-classification, we indicate this with ✓; if not, with a ✗; and if unknown with a "?". A reference in a column is a reference to where the condition is shown to hold, "triv" means that the condition holds trivially, and "no" means the condition does not hold. A further discussion of the table, explaining the "no"-statements, is in Section 5.5.

as new results, that List Coloring, Odd Cycle Transversal, Independent Odd Cycle Transversal, and C_5-Coloring are NP-complete for planar subcubic graphs, and thus satisfy C2. For the following problems we give explicit proofs to show that they satisfy C3: Pathwidth, Tree-Width, Edge/Node Steiner Tree, and Edge/Node Multiway Cut. Hence, our framework on $\mathcal{H}$-subgraph-free graphs shows the way towards new results.

The third impact of our framework is that it enables a structured investigation into complexity dichotomies for graph problems that do not satisfy some of the conditions, C1, C2 or C3, particularly when only one is not satisfied. We call such problems C23, C13, or C12, respectively. This led to new insights into the complexity of well-studied problems such as Hamilton Cycle [146], Steiner Forest [27], and Feedback Vertex Set as we shall see in Chapter 6. For all these problems, the complexity classifications are different from the one in Theorem 5.1.2. Hence, our framework has the potential to open a new and rich research area.

5.2. The Proof of Theorem 5.1.2

We present a stronger result that will imply Theorem 5.1.2. A graph class closed under edge deletion is also called *monotone* [7, 29, 132]. For a set of graphs $\mathcal{H}$, the class of $\mathcal{H}$-subgraph-free graphs is *finitely defined* if $\mathcal{H}$ is a finite set. We say that a problem Π is C1′D if Π satisfies the following two conditions (see Fig. 5.1 for examples of the subdivided "H"-graphs $\mathbb{H}_i$):

C1′. Π is efficiently solvable for every finitely defined monotone graph class of bounded Pathwidth;

D. For every $i \geq 3$, Π is computationally hard for the class of $(C_3, \ldots, C_i, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_i)$-subgraph-free graphs.

Our first theorem shows that the class of C1′D-problems is a proper superclass of the class of C123-problems.

Theorem 5.2.1. *Every C123-problem is C1′D, but not every C1′D-problem is C123.*

Proof. Let Π be a C123-problem. Then Π satisfies C1 and thus C1′. To show condition D, let $i \geq 3$, and let $\mathcal{G}_i$ be the class of $(C_3, \ldots, C_i, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_i)$-subgraph-free graphs. As Π satisfies C2, Π is computationally hard for the class $\mathcal{G}$ of subcubic graphs, that is, $K_{1,4}$-subgraph-free graphs. As Π satisfies C3, there exists an integer $\ell \geq i+1$, such that Π is computationally hard for $\mathcal{G}^\ell$. We note that $\mathcal{G}^\ell$ is a subclass of $\mathcal{G}_i$. Hence, Π is computationally hard for $\mathcal{G}_i$ and thus satisfies D. We conclude that Π is a C1′D-problem.

To show that the reverse statement does not necessarily hold, we define the following (artificial) example problem. Let $\mathcal{B}$ be the set of all graphs obtained from a cycle after adding a new vertex made adjacent to precisely one vertex of the cycle. Then the problem $\mathcal{B}$-Modified List Coloring takes as input a graph G with a list assignment L and asks whether G simultaneously has a coloring respecting L and has a connected component that is a graph from $\mathcal{B}$.

We now prove that $\mathcal{B}$-Modified List Coloring is not C123 but is C1′D. We distinguish between "being polynomial-time solvable" and "being NP-complete". We first observe that $\mathcal{B}$ satisfies the following four properties:

1. For every integer p, the p-subdivision of any graph in $\mathcal{B}$ is not in $\mathcal{B}$.
2. We can recognize whether a graph belongs to $\mathcal{B}$ in polynomial time.
3. Every graph in $\mathcal{B}$ admits a 3-coloring.
4. For every finite set $\mathcal{H}$ disjoint from $\mathcal{S}$, there is an $\mathcal{H}$-subgraph-free graph in $\mathcal{B}$.

Due to Property 1, $\mathcal{B}$-Modified List Coloring does not satisfy C3. Hence, $\mathcal{B}$-Modified List Coloring is not a C123-problem. We will prove that $\mathcal{B}$-Modified List Coloring is C1′D. As List Coloring is C123 [120], it satisfies C1 and thus C1′. By Property 2, we can check in polynomial time if a graph has a connected component in $\mathcal{B}$. Hence, $\mathcal{B}$-Modified List Coloring satisfies C1′. Below we prove that it also satisfies condition D.

Let $i \geq 3$, and let $\mathcal{G}_i$ be the class of $(C_3, \ldots, C_i, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_i)$-subgraph-free graphs. As List Coloring is C123, it follows from the first statement that it is also C1′D. Hence, List Coloring is NP-complete on $\mathcal{G}_i$. Let (G, L) be an instance of List Coloring where G is a graph from $\mathcal{G}_i$. We note that $\{C_3, \ldots, C_i, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_i)\} \cap \mathcal{S} = \emptyset$. Hence, by Property 4, there is an $\mathcal{H}$-subgraph-free graph $B \in \mathcal{B}$. Let $G' = G + B$. Extend L to a list assignment L' by giving each vertex of B list $\{1, 2, 3\}$. We claim that (G, L) is a yes-instance of List Coloring if and only if (G', L') is a yes-instance of $\mathcal{B}$-Modified List Coloring.

First suppose G has a coloring respecting L. By Property 3, B is 3-colourable. As vertices of B have list $\{1, 2, 3\}$, G' has a coloring respecting L'. As G has $B \in \mathcal{B}$ as a connected component, (G', L') is a yes-instance of $\mathcal{B}$-Modified List Coloring. Now suppose that (G', L') is a yes-instance of $\mathcal{B}$-Modified List Coloring. Then, G' has a coloring respecting L', and thus G has a coloring respecting L. We conclude that $\mathcal{B}$-Modified List Coloring satisfies D and is thus a C1′D-problem. As we already showed that $\mathcal{B}$-Modified List Coloring is not C123, this proves the second statement of the theorem. □

We also need a theorem from Bienstock, Robertson, Seymour and Thomas.

Theorem 5.2.2 ([18]). *For every forest F, all F-minor-free graphs have Pathwidth at most $|V(F)| - 2$.*

We now prove a result, which shows that the conditions C1′ and D are both necessary and sufficient.

Theorem 5.2.3. *Let Π be a problem. Then the following two statements are equivalent:*

(i) *Π is C1′D; and*

(ii) *for any finite set of graphs $\mathcal{H}$, the problem Π on $\mathcal{H}$-subgraph-free graphs is efficiently solvable if $\mathcal{H}$ contains a graph from $\mathcal{S}$ and computationally hard otherwise.*

Proof. First assume that Π is C1′D. Let $\mathcal{H}$ be a finite set of graphs. First suppose that $\mathcal{H}$ contains a graph H from $\mathcal{S}$. Let G be a $\mathcal{H}$-subgraph-free graph. As G is $\mathcal{H}$-subgraph-free, G is H-subgraph-free. It is known (see e.g. [102, 103]) that, for any graph $H' \in \mathcal{S}$, a H'-subgraph-free graph is also H'-minor-free. Hence, G is H-minor-free. So by Theorem 5.2.2, G has constant pathwidth at most $|V(H)| - 2$, meaning we can solve Π efficiently by C1′.

Now suppose that $\mathcal{H}$ contains no graph from $\mathcal{S}$. Let $H \in \mathcal{H}$. As $H \notin \mathcal{S}$, H has a connected component containing a $K_{1,4}$ (or equivalently, a vertex of degree at least 4); or a cycle C_h for some $h \geq 3$; or a graph $\mathbb{H}_i$ for some $i \geq 1$. Hence, the class of H-subgraph-free graphs contains the $K_{1,4}$-subgraph-free graphs; or C_h-subgraph-free graphs for some $h \geq 3$; or $\mathbb{H}_i$-subgraph-free graphs for some $i \geq 1$, each of which contains the $(C_3, \ldots, C_{j(H)}, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_{j(H)})$-subgraph-free graphs, where $j(H) = \max\{h, i\}$. Hence, the class of H-subgraph-free graphs contains the $(C_3, \ldots, C_{j(H)}, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_{j(H)})$-subgraph-free graphs. Consequently, the class of $\mathcal{H}$-subgraph-free graphs contains the $(C_3, \ldots, C_{j^*}, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_{j^*})$-subgraph-free graphs, where $j^* = \max_{H \in \mathcal{H}} j(H)$ (note that j exists, as $\mathcal{H}$ is finite). As Π satisfies D, we find that Π is computationally hard for $(C_3, \ldots, C_{j^*}, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_{j^*})$-subgraph-free graphs, and thus for $\mathcal{H}$-subgraph-free graphs.

Now assume that for any finite set of graphs $\mathcal{H}$, the problem Π on $\mathcal{H}$-subgraph-free graphs is efficiently solvable if $\mathcal{H}$ contains a graph from $\mathcal{S}$ and computationally hard otherwise. We first prove C1′. Let $\mathcal{H}$ be a finite set, such that the class of $\mathcal{H}$-subgraph-free graphs has bounded pathwidth. Recall that the latter holds if and only if $\mathcal{H}$ contains a graph from $\mathcal{S}$ [168]. Hence, Π satisfies C1′.

We now prove that condition D holds. Let $i \geq 3$, and let $\mathcal{G}_i$ be the class of $\mathcal{H}$-subgraph-free graphs, where $\mathcal{H} = \{C_3, \ldots, C_i, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_i\}$. Then $\mathcal{H}$ contains no graph from $\mathcal{S}$. Hence, Π is computationally hard for $\mathcal{H}$-subgraph-free graphs. Consequently, Π satisfies D. □

We are now ready to prove Theorem 5.1.2, which we restate below.

Theorem 5.1.2 (restated). *Let Π be a C123-problem. For any finite set of graphs $\mathcal{H}$, the problem Π on $\mathcal{H}$-subgraph-free graphs is efficiently solvable if $\mathcal{H}$ contains a graph from $\mathcal{S}$ (or equivalently, if the class of $\mathcal{H}$-subgraph-free graphs has bounded treewidth) and computationally hard otherwise.*

Proof. The result follows from combining Theorems 5.2.1 and 5.2.3, and the well-known fact that for a finite set of graphs $\mathcal{H}$, a class of $\mathcal{H}$-subgraph-free graphs has bounded pathwidth if and only if it has bounded treewidth if and only if $\mathcal{H}$ contains a graph from $\mathcal{S}$ [168] (see e.g. [29, 59], for an explanation with respect to the more powerful parameter cliquewidth, and hence, replacing "bounded pathwidth" in C1′ by "bounded treewidth" or "bounded cliquewidth" yields the same equivalence as in Theorem 5.2.3). □

Remark 5.2.4. *We emphasize that we are not aware of any natural C1′D-problem that is not C123. As the conditions C1–C3 are more intuitive, we have therefore chosen to present our subgraph framework in terms of the C1–C3 conditions instead of the C1′-D conditions.*

5.3. Application to NP-Complete Problems

We provide a complete dichotomy between polynomial-time solvability and NP-completeness for many problems on $\mathcal{H}$-subgraph-free graphs by showing they are C123-problems. In Section 5.3.1, we give examples of width parameter problems that are C123. In Section 5.3.2 we show the same for a number of network design problems. In fact, we do a bit more. Namely, we also show that these problems belong to the minor and topological minor frameworks whenever the relevant NP-completeness result applies to subcubic planar graphs, as reflected in Table 5.1. We will not explicitly remark this in the remainder of the section.

5.3.1. Width Parameter Problems

Let Path-Width and Tree-Width be the problems of deciding for a given integer k and graph G, if G has pathwidth, or respectively, treewidth at most k. We observe that it is unclear whether a pathwidth bound is maintained under

subdivision, and thus proving property C3 for Path-Width is non-trivial. We show a more specific result that is sufficient for our purposes. For Treewidth, we can follow a more direct proof.

Theorem 5.3.1. Path-Width *is a C123-problem.*

Proof. Path-Width is linear-time solvable for every graph class of bounded treewidth [22] so satisfies C1. It is NP-complete for 2-subdivisions of planar cubic graphs [157] so satisfies C2. It also satisfies C3, as we will prove the following claim:

Claim 5.3.2. *A graph $G = (V, E)$ that is a 2-subdivision of a graph G'' has pathwidth k if and only if the 1-subdivision G' of G has pathwidth k.*

Proof. First suppose that $G = (V, E)$ that is a 2-subdivision of a graph G'' has pathwidth k. We use the known equivalence of pathwidth to the vertex separation number [128]. We recall the definition. Let L be a bijection between V and $\{1, \ldots, |V|\}$, also called a *layout* of G. Let

$$V_L(i) = \{u \mid L(u) \leq i \text{ and } \exists v \in V : uv \in E \text{ and } L(v) > i\}.$$

Then $\text{vs}_L(G) = \max_{i \in \{1,\ldots,|V|\}}\{|V_L(i)|\}$ and $\text{vs}(G) = \min_L\{\text{vs}_L(G)\}$ is the *vertex separation number* of G.

As shown by Kinnersley [128], G has a layout L such that $\text{vs}_L(G) = k$ since G has pathwidth k. In a 2-subdivision, such as G, any edge uv of the original graph (G'' in this case) gets replaced by edges ua, ab, and bv, where a and b are new vertices specific to the edge uv. In a *standard layout* L for G, $L(a) = L(b) - 1$ and $L(u) < L(a)$ for each such edge uv of G''. By applying the transformation of Ellis, Sudborough and Turner [75, Lemma 2.3] if necessary, we may assume that L is a standard layout and still $\text{vs}_L(G) = k$.

For some edge uv of G'' and its 2-subdivision into ua, ab, bv in G, consider a further subdivision of each of these three edges. Let x, y, z be the newly created vertices respectively. Modify L by placing x directly before a, y between a and b, and z directly after b. Let L' denote the new layout. For simplicity and abusing notation, we use $L'(x) = L(a) - \frac{1}{2}$, $L'(y) = L(a) + \frac{1}{2} = L(b) - \frac{1}{2}$ and $L'(z) = L(b) + \frac{1}{2}$ to denote the positions of x, y and z in the new layout respectively. For any $i < L(a) - \frac{1}{2}$, $V_{L'}(i) = V_L(i)$, because $L(a) > i$ and $L'(x) > i$. Next, we observe that

$$V_{L'}(L'(x)) = V_{L'}(L(a) - \frac{1}{2}) = (V_L(L(a)) \setminus \{a\}) \cup \{x\},$$

because b follows after a in L and now a follows after x in L'. Hence, $V_{L'}(L'(x))$ has the same size as $V_L(L(a))$, so size at most k. Similarly, we can observe

that $V_{L'}(L(a)) = V_L(L(a))$ (note that $L'(u) = L(u) < L(a)$), $V_{L'}(L(a) + \frac{1}{2}) = (V_L(L(a)) \setminus \{a\}) \cup \{y\}$, and $V_{L'}(L(b)) = (V_L(L(a)) \setminus \{a\}) \cup \{b\}$, which all have size at most k. We then observe that $V_{L'}(L(b) + \frac{1}{2})$ is equal to $V_L(L(b))$ with b replaced by z if $b \in V_L(L(b))$. Similarly, for any $i > L(b)$, if $b \in V_L(i)$, we can replace b by z to obtain $V_{L'}(i)$; otherwise, $V_{L'}(i) = V_L(i)$. Note that a is never part of $V_L(i)$ for $i > L(b)$. In all cases, the size remains bounded by k. Hence, $\mathrm{vs}_{L'} \leq k$ and by the aforementioned equivalence between pathwidth and vertex separation number [128], G' has pathwidth at most k.

Now suppose that G' has pathwidth k. As subdivision cannot decrease pathwidth (or consider the converse, contraction cannot increase it), G has pathwidth at most k.

From the above, we conclude that G has pathwidth k if and only if G' has pathwidth k. Hence, the claim, and thus C3, is proven. ◇

This finishes the proof of Theorem 5.3.1. □

Theorem 5.3.3. Tree-Width *is a C123-problem.*

Proof. Tree-Width is linear-time solvable for every graph class of bounded treewidth [21]. Very recently, it was shown that Tree-Width is NP-complete for cubic graphs [26], so the problem satisfies C2. It also satisfies C3, as we will prove the following claim:

Claim 5.3.4. *A graph G has treewidth k if and only if the 1-subdivision of G has treewidth k.*

Proof. Taking a minor of a graph does not increase its treewidth so the treewidth cannot decrease after subdividing an edge. If a graph G has treewidth 1, then G remains a tree after subdividing an edge. Suppose G has treewidth at least 2. Then we can argue similarly as in Kneis*et al.* [130, Lemma 4.2]. Let $(T, \mathcal{X})$ be a tree decomposition of G with width $k \geq 2$. Let G' be the graph obtained by replacing an edge uv with edges ux and xv for a new vertex x. Pick an arbitrary bag B from $\mathcal{X}$ containing u and v. Introduce the bag $\{u, v, x\}$ and make the corresponding node adjacent to the B-node. This yields a tree decomposition of G' of width k, as $k \geq 2$ and the bag we added has size 3. Hence, the claim and thus the theorem is proven. ◇

This concludes the proof of Theorem 5.3.3. □

5.3.2. Network Design Problems

A *(edge) cut* of a graph $G = (V, E)$ is a partition $(S, V \setminus S)$ of V. The *size* of $(S, V \setminus S)$ is the number of edges with one end in S and the other in $V \setminus S$.

The Max-Cut problem is to decide if a graph has a cut of size at least k for some integer k. By combining the next result with Theorem 5.1.2, we recover the classification of [123].

Theorem 5.3.5 ([123])**.** Max-Cut *is a C123-problem.*

Proof. Max-Cut is linear-time solvable for graphs of bounded treewidth [9] and NP-complete for subcubic graphs [192] so satisfies C1 and C2. A cut C of a graph G is *maximum* if G has no cut of greater size. Kamiński [123] proved that a graph $G = (V, E)$ has a maximum cut of size at least c if and only if the 2-subdivision of G has a maximum cut of size at least $c + 2|E|$. This shows C3. □

Let $G = (V, E)$ be a graph. A set $M \subseteq E$ is a *perfect matching* if no two edges in M share an end-vertex and moreover, every vertex of the graph is incident on an edge of M. A set $M \subseteq E$ is an *edge cut* of G if it is possible to partition V into two sets B and R, such that M consists of all the edges with one end-vertex in B and the other one in R. A set $M \subseteq E$ is a *perfect matching cut* of G if M is a perfect matching that is also an edge cut. The Perfect Matching Cut is to decide if a graph has a perfect matching cut. Lucke *et al.* [81] recently showed that Perfect Matching Cut is C123.

Theorem 5.3.6 ([81])**.** Perfect Matching Cut *is a C123-problem.*

Proof. Le and Telle [139] observed that Perfect Matching Cut is polynomial-time solvable for graphs of bounded treewidth. In the same paper [139], they also proved that for every integer $g \geq 3$, it is NP-complete even for subcubic bipartite graphs of girth at least g. Hence, Perfect Matching Cut satisfies C1 and C2. The NP-completeness proof in [139] implicitly showed that to get C3 we may take $k = 4$ (see also [81]).

We note that C2 also follows for Perfect Matching Cut from a recent result of Bonnet, Chakraborty and Duron [30], who proved that Perfect Matching Cut is NP-complete even for 3-connected subcubic planar graphs. □

Given a graph G and a set of terminals $T \subseteq V(G)$, and an integer k, the problems Edge (Node) Steiner Tree are to decide if G has a subtree containing all the terminals of T, such that the subtree has at most k edges (vertices). We give explicit proofs that Node Steiner Tree and Edge Steiner Tree are NP-complete on planar subcubic graphs and that this is maintained under subdivision, leading to these two problems being C123-problems.

Theorem 5.3.7. Edge *and* Node Steiner Tree *are C123-problems.*

Proof. As the two variants are equivalent (on unweighted graphs), we only consider Edge Steiner Tree, which is linear-time solvable for graphs of bounded treewidth [9] so satisfies C1. For showing C2, we reduce from Edge Steiner Tree, which is NP-complete even for grid graphs [92], and thus for planar graphs.

Let (G, T, k) be an instance, where G is a planar graph with $|V(G)| = n$. We build a planar subcubic graph G' where we replace each vertex v in G with a rooted binary tree T_v in which there are n leaf vertices (so the tree contains at most $2n$ vertices and is of depth $\lceil \log n \rceil$). For each edge $e = uv$ of G, add to G' a path e' of length $4n^2$ between some leaf of T_u and a leaf of T_v (ensuring that each leaf is incident with at most one such path and the ordering of the paths is the same as the ordering around u, v to ensure planarity). If v in G is in T, then the root vertex of T_v is a terminal in G' (and these are the only terminals in G' and form the set T'). We note that G is planar subcubic, and we claim that (G, T, k) is a yes-instance if and only if $(G', T', 4n^2 \cdot k + 2n^2)$ is a yes-instance.

First, suppose G has a Steiner tree S with at most k edges. We build a Steiner tree S' in G': if $e = uv$ is in S, then we add to S' a path that comprises e' and paths that join the roots of T_u and T_v to e'. The sum of the lengths of these paths, additional to the $4n^2 \cdot k$, is bounded above by $2 \cdot n \cdot \lceil \log n \rceil \leq 2n^2$.

Now suppose G' has a Steiner tree S' with at most $4n^2 \cdot k + 2n^2$ edges. We build a tree S in G: if $e = uv$ and e' is in S', we add e to S. Then S is a Steiner tree in G. As the length of a path from T_u to T_v is $4n^2$, the sum of the lengths of all such paths in S' is a whole multiple of $4n^2$, so $|E(S)| \leq k$.

Finally, to prove C3, it suffices to show the following claim:

Claim 5.3.8. *A graph G has an edge Steiner tree for terminals T of size at most k if and only if the 1-subdivision of G has an edge Steiner tree for terminals T of size at most $2k$.*

Proof. In order to see this, let G' be the 1-subdivision of G. Let e_1 and e_2 be the two edges obtained from subdividing an edge $e \in E(G)$. Given a Steiner tree S of G with at most k edges, we obtain a Steiner tree of G' with at most $2k$ edges by replacing each edge e of S with e_1 and e_2. Given a Steiner tree S' of G' with at most $2k$ edges, we may assume that for any edge e of G, either none or both of e_1 and e_2 are in S'; if S' contains only one it can safely be discarded. To obtain a Steiner tree of G with at most k edges, include each edge e if both e_1 and e_2 are in S'. ◇

This concludes the proof of Theorem 5.3.7. □

Theorem 5.3.9. Edge *and* Node Multiway Cut *are C123-problems.*

Proof. Edge Multiway Cut is linear-time solvable for graphs of bounded treewidth [66] (also following [9]) and NP-complete for planar subcubic graphs [122] so satisfies C1 and C2. It satisfies C3 as well, as we will prove the following claim:

Claim 5.3.10. *A graph G has an edge multiway cut for a set of terminals T of size at most k if and only if the 1-subdivision of G has an edge multiway cut for T of size at most k.*

Proof. In order to see this, let G' be the 1-subdivision of G. For each edge e in G, there exist two edges in G'. If an edge of G is in an edge multiway cut for G and T, then it suffices to pick only one of the two edges created from it in G' to disconnect the paths e lies on. Vice versa, if an edge e' of G' is in an edge multiway cut for G' and T, then it suffices to pick the unique corresponding edge in G to disconnect the paths e' lies on. ◇

We now turn to Node Multiway Cut, which is linear-time solvable for graph classes of bounded treewidth [9] (it is an extended monadic second-order linear extremum problem) and NP-complete for planar subcubic graphs [122] so satisfies C1 and C2. It satisfies C3, as we will prove the following claim:

Claim 5.3.11. *A graph G has a node multiway cut for a set of terminals T of size at most k if and only if its 1-subdivision has a node multiway cut for T of size at most k.*

Proof. In order to see this, let G' be the 1-subdivision of G. We observe that subdividing any edge of a graph does not create new connections between terminals. Moreover, we can assume that none of the newly introduced vertices of the subdivision are used in some optimal solution for G' and T. ◇

This concludes the proof of Theorem 5.3.9. □

Given a graph G and disjoint vertex pairs $(s_1, t_1), (s_2, t_2), \ldots (s_k, t_k)$, the Disjoint Paths problem is to decide if G has k pairwise vertex-disjoint paths from s_i to t_i for every i. We obtain the Induced Disjoint Paths problem if the paths are required to be mutually induced; a set of paths $P^1, \ldots, P^k$ is *mutually induced* if $P^1, \ldots, P^k$ are pairwise vertex-disjoint and there is no edge between a vertex of some P^i and a vertex of some P^j if $i \neq j$.

Theorem 5.3.12. Disjoint Paths *and* Induced Disjoint Paths *are C123-problems.*

Proof. The Disjoint Paths problem is linear-time solvable for graphs of bounded treewidth [171] and NP-complete for planar subcubic graphs [153] so satisfies C1 and C2. The Induced Disjoint Paths problem is solvable in polynomial time for graphs of bounded mimwidth [116] and thus for bounded treewidth [184], so it satisfies C1. Let G' be the 1-subdivision of a subcubic graph G and let $\mathcal{T}$ be a set of disjoint vertex pairs. Then, $(G, \mathcal{T})$ is a yes-instance of Disjoint Paths if and only if $(G', \mathcal{T})$ is a yes-instance of Disjoint Paths if and only if $(G', \mathcal{T})$ is a yes-instance of Induced Disjoint Paths. Hence, C2 is satisfied for Induced Disjoint Paths as well and C3 is satisfied for both problems. □

The problems Long Path and Long Induced Path are to decide for a given graph G and integer k, whether G contains P_k as a subgraph or induced subgraph, respectively. The Long Cycle and Long Induced Cycle problems are defined similarly. By combining the next result with Theorem 5.1.2, we recover the classification of [123] for Long Path. The classification of Long Cycle was not made explicit in [7], but is implicitly there (combine Proposition 1 of [7] with Lemma 12 of [7]).

Theorem 5.3.13. Long Path, Long Induced Path, Long Cycle *and* Long Induced Cycle *are C123-problems.*

Proof. Bodlaender [20] proved that Long Path and Long Cycle are polynomial-time solvable for graphs of bounded treewidth. Hence, Long Path and Long Cycle satisfy C1. As Hamilton Path (so Long Path with $k = |V(G)|$) and Hamilton Cycle (so Long Cycle with $k = |V(G)|$) are NP-complete for subcubic planar graphs [94], Long Path and Long Cycle satisfy C2.

Let G' be the 1-subdivision of a subcubic graph G. Now the following holds: (G, k) is a yes-instance of Long Path if and only if $(G', 2k)$ is a yes-instance of Long Path if and only if $(G', 2k)$ is a yes-instance of Long Induced Path. Hence, C2 is satisfied for Long Induced Path as well, and C3 is satisfied for both Long Path and Longest Path. Moreover, Long Induced Path satisfies C1; it is even polynomial-time solvable for graphs of bounded mimwidth [116]. We can make the same observations for Long Cycle and Long Induced Cycle. □

5.4. Limitations of our Framework

We give two limitations of our framework.

5.4.1. Forbidding an Infinite Number of Subgraphs

We observe that in Theorems 5.1.4 and 5.1.5, the set of graphs $\mathcal{H}$ is allowed to have infinite size. However, the set of graphs $\mathcal{H}$ in Theorems 5.1.2 and 5.2.3 cannot be allowed to have infinite size. This is because there exist infinite sets $\mathcal{H}$ such that

1. $\mathcal{H}$ contains no graphs from $\mathcal{S}$.
2. All C123-problems are efficiently solvable on $\mathcal{H}$-subgraph-free graphs.

To illustrate this, we give two examples. See, e.g. [123], for another example.

Example 1. Let $\mathcal{H}$ be the set of cycles $\mathcal{C}$. No graph from $\mathcal{C}$ belongs to $\mathcal{S}$. Every $\mathcal{C}$-subgraph-free graph is a forest and thus has treewidth 1. Hence, every C123-problem is efficiently solvable on the class of $\mathcal{C}$-subgraph-free graphs (as it satisfies condition C1).

Example 2. Let $\mathcal{H} = \{\mathbb{H}_1, \mathbb{H}_2, \ldots\}$; see also Fig. 5.1. No graph from $\mathcal{H}$ belongs to $\mathcal{S}$. Every $\mathcal{H}$-subgraph-free graph G is $\mathbb{H}_1$-minor-free. By Theorem 5.2.2, G has pathwidth, and thus treewidth, at most 4. Hence, every C123-problem is efficiently solvable on the class of $\mathcal{H}$-subgraph-free graphs.

5.4.2. Relaxing Condition C3

In C3, we require the class $\mathcal{G}$ to be subcubic. In this way we are able to show in Theorem 5.2.1 that every C123-problem Π satisfies condition D, that is, for every $i \geq 3$, Π is computationally hard for the class of $(C_3, \ldots, C_\ell, K_{1,4}, \mathbb{H}_1, \ldots, \mathbb{H}_\ell)$-subgraph-free graphs.[1] If we allow $\mathcal{G}$ to be any graph class instead of requiring $\mathcal{G}$ to be subcubic, then we can no longer show this, and hence the proof of Theorem 5.1.2 no longer holds in that case. That is, following the same arguments we can only construct a graph class that due to C2, is either $K_{1,4}$-subgraph-free (or equivalently, subcubic) or, due to C3, is $(C_3, \ldots, C_\ell, \mathbb{H}_1, \ldots, \mathbb{H}_\ell)$-subgraph-free. Consequently, in that case, we can only obtain the dichotomy for $\mathcal{H}$-subgraph-free graphs if $|\mathcal{H}| = 1$. This relaxation could potentially lead to a classification of more problems. However, so far, we have not identified any problems that belong to this relaxation but not to our original framework.

We also note that the integers ℓ for which ℓ-subdivision maintains computational hardness is highly problem-specific. For instance, the 1-subdivision of any graph is bipartite and some computationally hard problems, such as Independent Set, become efficiently solvable on bipartite graphs.

[1] The aforementioned papers [87, 158] show a number of problems to be NP-complete for planar subcubic graphs of high girth, whereas we consider subcubic graphs of high girth that, instead of being planar, do not contain any small subdivided "H"-graph as a subgraph.

5.5. Comparison between the Three Frameworks

In this section, we provide an extensive discussion and comparison of the three frameworks mentioned in the overview: Theorem 5.1.4, 5.1.5, and 5.1.2. See also Table 5.1.

5.5.1. Problems That Belong to All Three Frameworks

Apart from Max-Cut and possibly Tree-Width, all C123-problems from Section 5.3 are NP-complete for planar subcubic graphs, and thus also satisfy the conditions of Theorems 5.1.4 and 5.1.5. In the proofs of Section 5.3 we made explicit observations about this. The complexity of Tree-Width is still open for planar graphs and planar subcubic graphs.

5.5.2. Problems That Do Not Belong to Any of the Three Frameworks

Every problem that is NP-complete for graphs of bounded treewidth does not satisfy any of the frameworks. An example is the aforementioned Subgraph Isomorphism problem, which is NP-complete even for input pairs (G_1, G_2) that are linear forests (see, for example, [24] for a proof) and thus have treewidth 1.

As another example, the Steiner Forest problem is to decide for a given integer k, graph G and set of pairs of terminal vertices $S = \{(s_1, t_1), \ldots, (s_p, t_p)\}$, if G has a subforest F with at most k edges, such that s_i and t_i, for every $i \in \{1, \ldots, p\}$, belong to the same connected component of F. It is readily seen that Steiner Forest generalizes Edge Steiner Tree: take all pairs of vertices of T as terminal pairs to obtain an equivalent instance of Steiner Forest. Hence, Steiner Forest is NP-complete on planar subcubic graphs and this is maintained under subdivision, due to Theorem 5.3.7. As Steiner Forest is NP-complete on graphs of treewidth 3 [11], Steiner Forest does not belong to any of the three frameworks. We refer to [27] for a partial complexity classification of Steiner Forest on H-subgraph-free graphs.

As an example on the other extreme end, the Clique problem does not fall under any of the three frameworks for different reasons. As observed in Section 6.1, Clique is polynomial-time solvable for $\mathcal{H}$-subgraph-free graphs for every set of graphs $\mathcal{H}$. Consequently, Clique does not belong to the subgraph framework. Moreover, Clique is polynomial-time solvable for planar graphs, as every clique in a planar graph has size at most 4. Hence, Clique does not belong to the minor and topological minor frameworks either.

5.5.3. Problems That Only Belong to the Minor Framework

We observe that every problem that satisfies the conditions of Theorem 5.1.5 also satisfies the conditions of Theorem 5.1.4. However, there exist problems that satisfy the conditions of Theorem 5.1.4 but not those of Theorems 5.1.5 and 5.1.2. For example, 3-Coloring satisfies C1 (this even holds for its generalization List Coloring [119]). Moreover, 3-Coloring is NP-complete even for 4-regular planar graphs [61]. Hence, 3-Coloring belongs to the minor framework. However, 3-Coloring does not satisfy the conditions of Theorems 5.1.5 and 5.1.2, as 3-Coloring is polynomial-time solvable for subcubic graphs due to Brooks' Theorem [34].

To give some further examples, we can also take the problems Connected Vertex Cover, Feedback Vertex Set and Independent Feedback Vertex Set. It is known that all three problems satisfy C1 [9]. Moreover, Connected Vertex Cover [92] and Feedback Vertex Set [176] are NP-complete for planar graphs of maximum degree at most 4. By taking 1-subdivisions, we find that the same holds for Independent Feedback Vertex Set. However, unlike the related problems Vertex Cover and Odd Cycle Transversal, the three problems do not satisfy the conditions of Theorems 5.1.5 and 5.1.2. This is because Connected Vertex Cover [182], Feedback Vertex Set [182] and Independent Feedback Vertex Set [Chapter 6] are polynomial-time solvable for subcubic graphs. Munaro [158] showed that even Weighted Feedback Vertex Set is polynomial-time solvable for subcubic graphs.

As a final example, we can take the Matching Cut problem. This problem satisfies C1 [31]. Moreover, it is NP-complete for planar graphs of girth 5 [31] but polynomial-time solvable for subcubic graphs [51].

5.5.4. Problems That Only Belong to the Minor and Topological Minor Frameworks

We also know of problems that satisfy the conditions of Theorem 5.1.5, and thus of Theorem 5.1.4, but not those of Theorem 5.1.2. For example, Hamilton Cycle is solvable in polynomial-time for graphs of bounded treewidth [8], so satisfies C1, and it is NP-complete for planar subcubic graphs [96] (even if they are also bipartite and have arbitrarily large girth [158]). Hence, Hamilton Cycle satisfies the conditions of Theorem 5.1.5, and also satisfies C2. However, unlike its generalization Long Cycle, which is C123, Hamilton Cycle does not satisfy C3 [146], so it is not a C123-problem. The same holds for Hamilton Path (which contrasts the C123-property of Long Path).

To give another example, Star 3-Coloring is to decide if a graph G has

a 3-coloring such that the union of every two color classes induces a *star forest* (forest in which each connected component is a star). This problem is known to be NP-complete even for subcubic planar subgraphs of arbitrarily large fixed girth [26], but does not satisfy C3 [146], so is not C123.

To give a final example of a problem that satisfies the conditions of Theorems 5.1.4 and 5.1.5 but not those of Theorem 5.1.2, we can consider the C_5-Coloring problem. This problem is to decide if a given graph allows a homomorphism to C_5. It is known to be NP-complete on both subcubic graphs [91] and planar graphs [144]. In order to show NP-completeness for subcubic planar graphs, one can take the gadget of MacGillivray and Siggers [144] and augment it with a degree reduction gadget[2] However, C_5-Coloring does not satisfy C3 [146], so it not C123.

5.5.5. Problems That Only Belong to the Subgraph Framework

There also exist problems that satisfy the conditions of Theorem 5.1.2, and thus are C123, but that do not satisfy the conditions of Theorems 5.1.4 and 5.1.5. Namely, Max-Cut is polynomial-time solvable for planar graphs [109] (and thus also for planar subcubic graphs). However, we show in Section 5.3 that Max-Cut satisfies the conditions of Theorem 5.1.2, that is, is a C123-problem.

5.6. Conclusions

By giving a meta-classification, we were able to unify a number of known results from the literature, reprove some of them, and give new complexity classifications for a variety of graph problems on classes of graphs characterized by a finite set $\mathcal{H}$ of forbidden subgraphs. Similar frameworks existed (even for infinite sets $\mathcal{H}$) already for the minor and topological minor relations, whereas for the subgraph relation, only some classifications for specific problems existed [6, 102, 123]. We showed that many problems belong to all three frameworks, and also that there exist problems that belong to one framework but not to (some of) the others.

In order to have stronger hardness results for our subgraph framework, we considered the unweighted versions of these problems. However, we note that most of the vertex-weighted and edge-weighted variants of these problems satisfy C1 as well; see [9]. We finish this section by setting out some directions for future work.

[2]The use of this gadget for this purpose was proposed to us by Mark Siggers.

5.6.1. Refining and Extending the Subgraph Framework

We describe three approaches for refining or extending the subgraph framework. First, in the proof of Theorem 5.2.1 we gave an example of a C1'D-problem, namely $\mathcal{B}$-Modified List Coloring, that is not C123. However, this example is rather artificial. To increase our understanding of the conditions C1–C3 of our framework, addressing the following question would be helpful.

Open Problem 5.1. *Do there exist any natural graph C1'D-problems that are not C123-problems?*

As a second approach, we recall from Section 5.4.2 that we cannot relax condition C3 by allowing the class $\mathcal{G}$ to be an arbitrary graph class instead of being subcubic. If we do this nevertheless, we are only able to obtain a dichotomy for $\mathcal{H}$-subgraph-free graphs if $|\mathcal{H}| = 1$. This relaxation could potentially lead to a classification of more problems, and we pose the following open problem.

Open Problem 5.2. *Can we classify more problems for H-subgraph-free graphs by no longer demanding that the class $\mathcal{G}$ in C3 is subcubic?*

So far, we have not identified any problems that belong to the relaxation but not to our original framework.

Recall that the set of forbidden graphs $\mathcal{H}$ is allowed to have infinite size in Theorems 5.1.4 and 5.1.5. For any infinite set of graphs $\mathcal{H}$, a C123-problem on $\mathcal{H}$-subgraph-free graphs is still efficiently solvable if $\mathcal{H}$ contains a graph H from $\mathcal{S}$. However, a C123-problem may no longer be computationally hard for $\mathcal{H}$-subgraph-free graphs if $\mathcal{H}$ has infinite size, as shown in Section 5.4.1 with some examples. Hence, as a third approach for extending the subgraph framework, we propose the following problem. This problem was also posed by Kamiński [123], namely for the C123-problem Max-Cut.

Open Problem 5.3. *Can we obtain dichotomies for C123-problems restricted to $\mathcal{H}$-subgraph-free graphs when $\mathcal{H}$ is allowed to have infinite size?*

In order to solve Open Problem 5.3, we need a better understanding of the treewidth of $\mathcal{H}$-subgraph-free graphs when $\mathcal{H}$ has infinite size. In recent years, such a study has been initiated for the induced subgraph relation; see, for example, [3, 2, 131, 188] for many involved results in this direction.

5.6.2. Finding More Problems Falling under the Three Frameworks

There still exist many natural problems for which it is unknown whether they belong to the minor, topological minor or subgraph framework. For the first two

frameworks, we recall the following open problems, which have been frequently stated as open problems before.

Open Problem 5.4. *Determine the computational complexity of* Tree-Width *for planar graphs and for planar subcubic graphs.*

We now turn to the subgraph framework. We showed that Tree-Width and Pathwidth are C123, but further investigation might reveal more such problems that fit the subgraph framework.

Open Problem 5.5. *Do there exist other width parameters with the property that the problem of computing them is C123?*

We also made a detailed comparison between the minor, topological minor and subgraph frameworks (see Section 5.5). To increase our general understanding of the complexity of graph problems, it would be interesting to find more problems that either belong to all frameworks or just to one or two. In particular, we pose the following question.

Open Problem 5.6. *Does there exist a graph problem that belongs to the minor and subgraph frameworks, but not to the topological minor framework?*

We note that such a problem (if it exists) must be computationally hard for planar graphs and subcubic graphs, but efficiently solvable for subcubic planar graphs.

5.6.3. Dropping One of the Conditions C1, C2, or C3

Another compelling direction is to investigate if we can obtain new complexity dichotomies for computationally hard graph problems that do not satisfy one of the conditions, C1, C2 or C3. We call such problems C23, C13, or C12, respectively.

Some interesting progress has recently been made on such problems (see e.g. [27, 121, 146]). However, we note that in general, obtaining complete classifications is challenging for C12-, C13- and C23-problems. In particular, we need a better understanding of the structure of P_r-subgraph-free graphs and $\mathbb{H}_i$-subgraph-free graphs (recall that $\mathbb{H}_i$ is a subdivided "H"-graph). Recall that a graph is P_r-subgraph-free if and only if it is P_r-(topological)-minor-free. Hence, if a problem is open for the case where $H = P_r$ for one of the frameworks, then it is open for all three of them.

To illustrate the challenges with an example from the literature, consider the aforementioned Subgraph Isomorphism problem. This problem takes as input two graphs G_1 and G_2. Hence, it does not immediately fit in our framework, but one could view it as a C23-problem. The question is whether

G_1 is a subgraph of G_2. Recall that the Subgraph Isomorphism problem is NP-complete even for input pairs (G_1, G_2) that are linear forests and thus even have Pathwidth 1. Yet, even a classification for H-subgraph-free graphs was not straightforward; recall that Bodlaender *et al.* [24] essentially settled the computational complexity of Subgraph Isomorphism for H-subgraph-free graphs except if $H = P_5$ or $H = 2P_5$. These cases are open for the minor and topological minor frameworks as well due to the above observation (which also holds for linear forests).

5.6.4. The Induced Subgraph Relation

We finish this chapter with some remarks on the induced subgraph relation. As mentioned, there exist ongoing and extensive studies on boundary graph classes (cf. [5, 7, 132, 158]) and treewidth classifications (cf. [3, 2, 131, 188]) in the literature. We note that for the induced subgraph relation, it is also useful to check C2 and C3. Namely, let Π be a problem satisfying C2 and C3. For any finite set of graphs $\mathcal{H}$, the problem Π on $\mathcal{H}$-free graphs is computationally hard if $\mathcal{H}$ contains no graph from $\mathcal{S}$. This follows from the same arguments as in the proof of Theorem 5.2.3.[3] Hence, if we aim to classify the computational complexity of problems satisfying C2 and C3 for H-free graphs (which include all C123-problems), then we may assume that $H \in \mathcal{S}$. For many of such problems, such as Independent Set, this already leads to challenging open cases.

As mentioned, we currently do not know even any algorithmic meta-theorem for the induced subgraph relation, not even for a single forbidden graph H. However, a recent result of Lozin and Razgon [143] provides at least an initial starting point. To explain their result, the *line graph* of a graph G has vertex set $E(G)$ and an edge between two vertices e_1 and e_2 if and only if e_1 and e_2 share an end-vertex in G. Let $\mathcal{T}$ be the class of line graphs of $\mathcal{S}$. Lozin and Razgon [143] showed that for any finite set of graphs $\mathcal{H}$, the class of $\mathcal{H}$-free graphs has bounded treewidth if and only if $\mathcal{H}$ contains a complete graph, a complete bipartite graph, a graph from $\mathcal{S}$ and a graph from $\mathcal{T}$. Their characterization leads to the following theorem, which could be viewed as a first meta-classification for the induced subgraph relation.

Theorem 5.6.1. *Let Π be a problem that is NP-complete on every graph class of unbounded treewidth, but polynomial-time solvable for every graph class of bounded treewidth. For every finite set of graphs $\mathcal{H}$, the problem Π on $\mathcal{H}$-free graphs is polynomial-time solvable if $\mathcal{H}$ contains a complete graph, a complete*

[3] The reason is that for any integer k and a sufficiently large integer ℓ, the class of subcubic $(C_3, \ldots, C_\ell, \mathbb{H}_1, \ldots, \mathbb{H}_k)$-free graphs coincides with the class of subcubic $(C_3, \ldots, C_\ell, \mathbb{H}_1, \ldots, \mathbb{H}_k)$-subgraph-free graphs.

bipartite graph, a graph from $\mathcal{S}$ and a graph from $\mathcal{T}$, and it is NP*-complete otherwise.*

Note that by the aforementioned result of Hickingbotham [113], we may replace "treewidth" by "Pathwidth" in Theorem 5.6.1. However, currently, we know of only one problem that satisfies the conditions of Theorem 5.6.1, namely Weighted Edge Steiner Tree [25], where we allow the edges to have weights. As we showed, even Edge Steiner Tree (the unweighted version) is a C123-problem. Even though the conditions of Theorem 5.6.1 are very restrictive, we believe the following open problem is still interesting.

Open Problem 5.7. *Determine which graph problems satisfy the conditions of Theorem 5.6.1.*

6

Problems Tractable on Subcubic Graphs

For any finite set $\mathcal{H} = \{H_1, \ldots, H_p\}$ of graphs, a graph is $\mathcal{H}$-subgraph-free if it does not contain any of $H_1, \ldots, H_p$ as a subgraph. In the previous chapter, meta-classifications have been studied: these show that if graph problems satisfy certain prescribed conditions, their complexity is determined on classes of $\mathcal{H}$-subgraph-free graphs. We focus on problems that can be solved in polynomial-time on classes that have bounded treewidth or maximum degree at most 3 and examine their complexity on H-subgraph-free graph classes where H is a connected graph. With this approach, we obtain comprehensive classifications for (Independent) Feedback Vertex Set, Connected Vertex Cover, Coloring and Matching Cut.

6.1. Introduction

In the previous chapter, we gave a complete classification of the complexity of several problems for $\mathcal{H}$-subgraph-free graphs. In general, such classifications are hard to obtain. There are still many graph problems that are not C123. In [146], results were obtained for problems that satisfy C1 and C2 but not C3. Such problems are called C12-problems and include k-Induced Disjoint Paths, C_5-Coloring, Hamilton Cycle and Star 3-Coloring [146]. In [27], Steiner Forest was investigated as a problem that satisfies C2 and C3 but not C1.

Here, we consider the research question:

How do C13-problems *— that is, problems that satisfy C1 and C3 but not C2 — behave for H-subgraph-free graphs? Can we still classify their computational complexity?*

Let us immediately note some redundancy in the definition of C13-problems: if a problem does not satisfy C2, then C3 is implied. Nevertheless, we retain the terminology to preserve the link to the approach of Chapter 5. To show a problem is a C13 problem there are two requirements: that the problem is efficiently solvable both on classes of bounded treewidth and on classes of subcubic graphs. In fact, the tractable cases for C123 problems rely on that the problems satisfy C1.

A *claw* is the graph $K_{1,3}$. A *subdivided claw* is the graph formed by subdividing one or more edges of the claw. Let $\mathcal{S}$ be the class of graphs that are disjoint union of paths and subdivided claws.

Theorem 6.1.1 (Chapter 5)**.** *Let Π be a problem that satisfies C1. For a finite set $\mathcal{H}$, the problem Π on $\mathcal{H}$-subgraph-free graphs is efficiently solvable if $\mathcal{H}$ contains a graph from $\mathcal{S}$.*

As an important step towards a full dichotomy for C13 problems, we restrict ourselves to considering H-subgraph-free graphs where H is connected. We focus on five well-known NP-complete problems discussed in Section 5.5.3 that are not C123 but C13-problems: Feedback Vertex Set, Independent Feedback Vertex Set, Connected Vertex Cover and Matching Cut. We introduce these problems below. With one exception, we can recognize that they are C13 problems using known results.

For a graph $G = (V, E)$, a set $W \subseteq V$ is a *feedback vertex set* of G if every cycle in G contains a vertex of W. Moreover, W is an *independent feedback vertex set* if W is an independent set. We note that G has a feedback vertex set of size k if and only if the 2-subdivision of G has an independent feedback vertex set of size k. A graph G might contain no independent feedback vertex set: consider, for example, a complete graph on four or more vertices.

(Independent)Feedback Vertex Set	
Instance:	Undirected graph $G(V, E)$, integer k.
Question:	Does G have an (independent) feedback vertex set of size at most k?

A set $W \subseteq V$ is a *connected vertex cover* of G if every edge of E is incident on a vertex of W, and moreover W induces a connected subgraph.

Connected Vertex Cover	
Instance:	Undirected graph $G(V, E)$, integer k.
Question:	Does G have a connected vertex cover of size at most k?

A *k-coloring* of G is a function $c : V \to \{1, \ldots, k\}$ such that for each edge $uv \in E$, $c(u) \neq c(v)$.

Coloring	
Instance:	Undirected graph $G(V, E)$, integer k.
Question:	Does G have a k-coloring?

A *matching cut* of a connected graph is a matching (set of pairwise non-adjacent edges) that is also an edge cut, i.e., its removal creates a disconnected graph.

Matching Cut	
Instance:	Undirected, connected graph $G(V, E)$.
Question:	Does G have a matching cut?

6.1.1. Our Results

Both Feedback Vertex Set [9] and Independent Feedback Vertex Set [177] satisfy C1. Whereas Feedback Vertex Set does have a polynomial-time algorithm on subcubic graphs [182] and thus does not satisfy C2, a polynomial-time algorithm for Independent Feedback Vertex Set on subcubic graphs was not previously known. In Section 6.2, we prove the following result addressing this gap in the literature.

Theorem 6.1.2. *A minimum size independent feedback vertex set of every connected subcubic graph $G \neq K_4$ is also a minimum size feedback vertex set of G. Moreover, it is possible to find a minimum independent feedback vertex set of G in polynomial time.*

The *star* $K_{1,s}$ is the graph that contains a vertex of degree s whose neighbors each have degree 1. A *subdivided star* is obtained from a star by subdividing one or more of its edges.

Definition 6.1.3. *An $S_{w,x,y,z}$ is a graph formed by subdividing the edges of a $K_{1,4}$, $w-1, x-1, y-1$, and $z-1$ times respectively. Each of the subdivided edges is called a* tentacle. *The vertex of degree 4 is the* center.

In Section 6.3, we investigate the structure of H-subgraph-free graphs when H is a subdivided star and use this in Section 6.4 to show a general approach to C13 problems that requires some additional extra properties (that we can solve

the problems on each connected component after possibly removing bridges, and the union of those solutions is the solution for the entire instance). This is sufficient to obtain the following result.

Theorem 6.1.4. *Let q and r be positive integers. The following problems can be solved in polynomial time on $S_{1,1,q,r}$-subgraph-free graphs:* Feedback Vertex Set, Independent Feedback Vertex Set, Connected Vertex Cover, Coloring *and* Matching Cut.

In Section 6.5, we obtain a hardness result.

Theorem 6.1.5. Feedback Vertex Set *and* Independent Feedback Vertex Set *are* NP*-complete on the class of $S_{2,2,2,2}$-subgraph free graphs that have maximum degree* 4.

6.1.2. State-of-the-Art Summaries

We now state complexity classifications for each of the problems. These results, proved in Section 6.6, combine the results above with a number of other results from [51, 81, 93, 95, 103, 145, 158, 166, 176]. None of these papers presented general results for C13 problems. However, we note, for example, that hardness when H contains a cycle follows from past results on classes of bounded girth which were proved separately for each problem, but often using a similar technique. There are other results that just apply to one or two of the problems.

Theorem 6.1.6. *Let H be a connected graph. On H-subgraph-free graphs,* Feedback Vertex Set *and* Independent Feedback Vertex Set *are solvable in polynomial time if $H \in \mathcal{S} \cup \{S_{1,1,q,r} \mid q \geq r \geq 1\}$. They are* NP*-complete if H contains a cycle or more than one vertex of degree at least* 3 *or $H \in \{K_{1,5}, S_{2,2,2,2}\}$.*

Theorem 6.1.7. *Let H be a connected graph. On H-subgraph-free graphs,* Connected Vertex Cover *is solvable in polynomial time if $H \in \mathcal{S} \cup \{S_{1,1,q,r} \mid q \geq r \geq 1\}$. It is* NP*-complete if H contains a cycle or $H = K_{1,5}$.*

The following result refers to trees defined in Figure 6.1.

Theorem 6.1.8. *Let H be a connected graph. On H-subgraph-free graphs,* Coloring *is solvable in polynomial time if $H \in \mathcal{S} \cup \{S_{1,1,q,r} \mid q \geq r \geq 1\}$ or if H is a forest with maximum degree* 4 *and at most seven vertices. It is* NP*-complete if H contains a cycle, or $H \in \{K_{1,5}, S_{2,2,2,2}\}$, or if H contains a subdivision of the tree T_1 as a subgraph, or H contains as a subgraph the tree obtained from T_2 after subdividing some edge p times, $0 \leq p \leq 9$, or H contains one of the trees $S_{2,2,2,2}, T_4, T_5, T_6$ as a subgraph.*

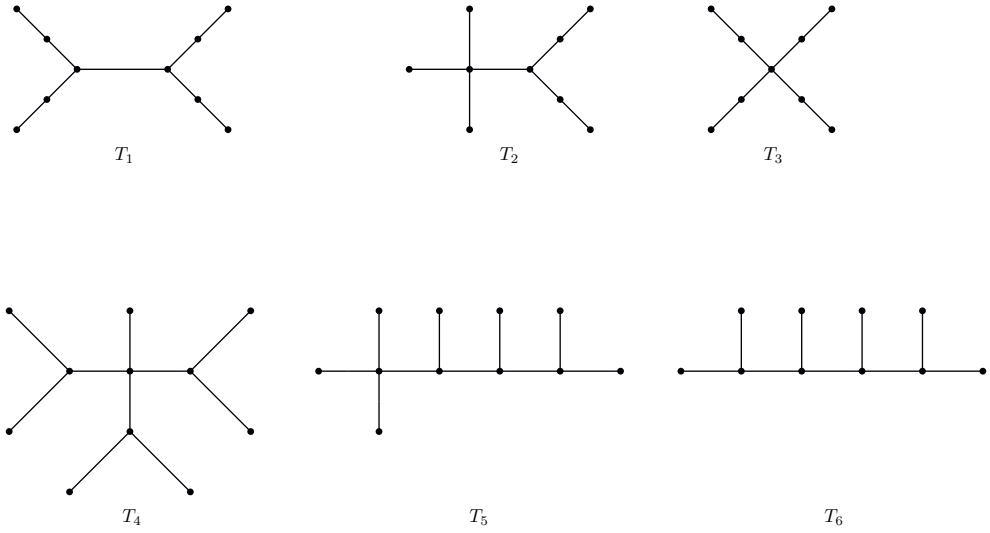

Figure 6.1: Illustration of the trees $T_1, \ldots, T_6$ reproduced from [103]; note that $T_3 = S_{2,2,2,2}$.

Theorem 6.1.9. *Let H be a connected graph. On H-subgraph-free graphs,* Matching Cut *is solvable in polynomial time if $H \in \mathcal{S} \cup \{S_{1,1,q,r} \mid q \geq r \geq 1\}$. It is* NP*-complete if H contains a cycle or $H = K_{1,5}$.*

A graph G contains a graph H as a *subgraph* if H can be obtained from G by vertex deletions and edge deletions; else G is said to be *H-subgraph-free*. If H can be obtained from G using *only* vertex deletions, then H is an *induced* subgraph of G, and if not then G is *H-free*.

For a set of graphs $\mathcal{H} = \{H_1, \ldots, H_p\}$, a graph G is *$\mathcal{H}$-subgraph-free* if G is H-subgraph-free for every $H \in \mathcal{H}$; we also write that G is $(H_1, \ldots, H_p)$-subgraph-free.

There are few studies of complexity classifications of graph problems for H-subgraph-free graphs (compare the greater attention given to problems on H-free graphs). There are results for Independent Set, Dominating Set and Longest Path [6], and List Coloring [102]

In these papers, complete classifications are presented giving the complexity of the problems for $\mathcal{H}$-subgraph-free graphs, where $\mathcal{H}$ is any finite set of graphs. Such classifications seem difficult to obtain. For example, for Coloring, there is only a partial classification[103]. For this reason — and also noting that the classifications for the problems above were all the same — a systematic approach was developed in Chapter 5 with the introduction of a new framework which we will describe after introducing some terminology.

A class of graphs has bounded *treewidth* if there exists a constant c such

that every graph has treewidth at most c. A graph is *subcubic* if every vertex has degree at most 3. The *subdivision* of an edge $e = uv$ replaces e by a vertex w and edges uw and wv. For an integer $k \geq 1$, the *k-subdivision* of a graph is obtained by subdividing each edge exactly k times. For a class of graphs $\mathcal{G}$ and integer k, $\mathcal{G}^k$ contains a k-subdivision of each graph in $\mathcal{G}$.

The framework of Chapter 5 has three conditions which refer to problems as "efficiently solvable" or "computationally hard" when the input is in certain graph classes. When the framework is applied, these terms are often interpreted as "polynomial-time solvable" and "NP-complete", as they will be in this paper, but the framework can also be used to, for example, distinguish different polynomial complexities. Let Π be a decision problem that takes as input a (possibly weighted) graph. We say that Π is computationally hard *under edge subdivision of subcubic graphs* if there exists an integer $k \geq 1$ such that the following holds for the class of subcubic graphs $\mathcal{G}$: if Π is computationally hard for $\mathcal{G}$, then Π is computationally hard for $\mathcal{G}^{kp}$ for every integer $p \geq 1$. Here is the framework: we say that a graph problem Π is a *C123-problem* (belongs to the framework) if it satisfies the following three conditions:

C1. Π is efficiently solvable for every graph class of bounded treewidth;

C2. Π is computationally hard for the class of subcubic graphs; and

C3. Π is computationally hard under edge subdivision of subcubic graphs.

The *claw* is the 4-vertex star. A *subdivided claw* is a graph obtained from a claw after subdividing each of its edges zero or more times. The *disjoint union* of two vertex-disjoint graphs G_1 and G_2 has vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$. The set $\mathcal{S}$ consists of the graphs that are disjoint unions of subdivided claws and paths. As shown in [120], C123-problems allow for full complexity classifications for $\mathcal{H}$-subgraph-free graphs (as long as $\mathcal{H}$ has finite size).

Theorem 6.1.10 ([120])**.** *Let Π be a C123-problem. For a finite set $\mathcal{H}$, the problem Π on $\mathcal{H}$-subgraph-free graphs is efficiently solvable if $\mathcal{H}$ contains a graph from $\mathcal{S}$ and computationally hard otherwise.*

Examples of C123-problems include Independent Set, Dominating Set, List Coloring, Odd Cycle Transversal, Max Cut, Steiner Tree and Vertex Cover; see [120] for a comprehensive list. Thus, we see the power of the framework to aid progress in deciding the complexity of problems on $\mathcal{H}$-subgraph-free graphs. But there are still many graph problems that are not C123-problems. In [146], the following question was addressed: for problems that satisfy C1 and C2 but not C3, can one find complexity classifications?

Such problems are called C12-problems and include k-Induced Disjoint Paths, C_5-Coloring, Hamilton Cycle and Star 3-Coloring, for each of which results were presented in [146].

In this paper we consider the following research question:

How do C13-problems *— that is, problems that satisfy C1 and C3 but not C2 — behave for H-subgraph-free graphs? Can we still classify their computational complexity?*

Let us immediately note some redundancy in the definition of C13-problems: if a problem does not satisfy C2, then C3 is implied. Nevertheless, we retain the terminology to preserve the link to the approach of [120] and later papers. To show a problem is a C13 problem there are two requirements: that the problem is efficiently solvable both on classes of bounded treewidth and on subcubic classes. In fact, the tractable cases of Theorem 6.1.10 rely on that the problems satisfy C1. Hence, the tractable cases also hold for C13 problems. Though this can be seen within the proof of Theorem 6.1.10 in [120], for convenience we restate and prove it here.

Theorem 6.1.11 ([120])**.** *Let Π be a problem that satisfies C1. For a finite set $\mathcal{H}$, the problem Π on $\mathcal{H}$-subgraph-free graphs is efficiently solvable if $\mathcal{H}$ contains a graph from $\mathcal{S}$.*

Proof. It is known [102, 103] that, for each $H \in \mathcal{S}$, a H-subgraph-free graph is H-minor-free. By a result of [18], as H is a tree, all H-minor-free graphs have pathwidth at most $|V(H)| - 2$. As a graph's treewidth is at most its pathwidth, the class of H-subgraph-free graphs has bounded treewidth. So, by C1, Π can be solved efficiently on $\mathcal{H}$-subgraph-free graphs. □

We are not able to provide a full dichotomy for C13 problems. We will restrict ourselves to considering H-subgraph-free graphs where H is connected and will focus on five well-known problems that we will see are C13 problems.

Our Focus. Let us introduce the candidate problems for our research question. We will see that, with one exception, we can recognize that they are C13 problems using known results.

For a graph $G = (V, E)$, a set $W \subseteq V$ is a *feedback vertex set* of G if every cycle in G contains a vertex of W. Moreover, W is an *independent feedback vertex set* if no pair of vertices of W is adjacent.

Feedback Vertex Set	
Instance:	A graph G and an integer $k \geq 0$.
Question:	Does G have a feedback vertex set of size at most k?

Independent Feedback Vertex Set	
Instance:	A graph G and an integer $k \geq 0$.
Question:	Does G have an independent feedback vertex set of size at most k?

An independent feedback vertex set is certainly a feedback vertex set. Also, a graph G might contain no independent feedback vertex set: consider, for example, a complete graph on four or more vertices. We also observe that if we transform a graph G into a graph J by subdividing every edge, then the problem of finding a minimum size feedback vertex set of G is equivalent to the problem of finding a minimum size feedback vertex set of J. Suppose that a graph class contains G if and only if it contains J. Then, for that graph class, if Independent Feedback Vertex Set can be solved in polynomial time, then so can Feedback Vertex Set, and hardness results for Feedback Vertex Set hold also for Independent Feedback Vertex Set. Both are NP-complete on general graphs [93].

It was shown in [177] that Independent Feedback Vertex Set can be solved in polynomial time on graph classes of bounded treewidth. Whereas Feedback Vertex Set does have a polynomial-time algorithm on subcubic graphs [182], a polynomial-time algorithm for Independent Feedback Vertex Set on subcubic graphs was not previously known. In the next section, we shall provide one by showing that, on this class, it is equivalent to Feedback Vertex Set, thus proving the following result.

Theorem 6.1.12. *Let G be a connected subcubic graph. Then a minimum size independent feedback vertex set of G is also a minimum size feedback vertex set of G if and only if $G \neq K_4$.*

A set $W \subseteq V$ is a vertex cover of G if every edge of E is incident with a vertex of W. And W is a *connected vertex cover* of G if the vertices of W induce a connected subgraph.

Connected Vertex Cover	
Instance:	A graph G and an integer $k \geq 0$.
Question:	Does G have a connected vertex cover of size at most k?

We note that Connected Vertex Cover can be solved in polynomial time on subcubic graphs [182] and on graphs of bounded treewidth by a result of Arnborg et al. [9].

A *proper k-coloring* of G is a function $c : V \to \{1, \ldots, k\}$ such that for each edge $uv \in E$, $c(u) \neq c(v)$.

Coloring	
Instance:	A graph G and an integer k.
Question:	Does G have a proper k-coloring?

By Brooks' Theorem [34], Coloring can be solved in polynomial time on subcubic graphs, and, by [8], also on graphs of bounded treewidth.

A *matching* of G is a subset of E that contains no adjacent edges. An *edge cut* of G is a subset of E whose removal from G creates a disconnected graph.

Matching Cut	
Instance:	A graph G.
Question:	Does G contain a set of edges that form both a matching and an edge cut?

Chvátal [51] proved that Matching Cut polynomial-time solvable for subcubic graphs, and Bonsma [31] proved the same for graphs of bounded treewidth.

Our Results

Using past results and, in Theorem 6.1.12, proving a new result on subcubic graphs, we have shown that the five problems defined above are C13 problems. We need more definitions. A *star*, denoted $K_{1,s}$, is a graph that contains a vertex of degree s whose neighbors each have degree 1. A *subdivided star* is obtained from a star by subdividing one or more of its edges.

Definition 6.1.13. *An $S_{w,x,y,z}$ is a graph formed by subdividing the edges of a $K_{1,4}$, $w-1, x-1, y-1$, and $z-1$ times respectively. Each of the subdivided edges is called a* tentacle. *The vertex of degree 4 is the* center.

In Section 6.3, we investigate the structure of H-subgraph-free graphs when H is a subdivided star and use this in Section 6.4 to show a general approach to C13 problems that requires some additional extra properties (that they can be solved on each connected component of the input graph after, possibly, the removal of bridges and the union of the respective solutions for each component is an optimum solution for the whole instance). This is sufficient to obtain the following result.

Theorem 6.1.14. *Let q and r be positive integers. The following problems can be solved in polynomial time on $S_{1,1,q,r}$-subgraph-free graphs:* Feedback Vertex Set, Independent Feedback Vertex Set, Connected Vertex Cover, Coloring *and* Matching Cut.

In Section 6.5, we obtain a hardness result.

Theorem 6.1.15. Feedback Vertex Set *and* Independent Feedback Vertex Set *are* NP*-complete on the class of* $S_{2,2,2,2}$*-subgraph free graphs that have maximum degree* 4*.*

We now state complexity classifications for each of the problems. These results, proved in Section 6.6, combine the results above with a number of other results from [51, 81, 93, 95, 103, 145, 158, 166, 176]. None of these presented general results for C13 problems though we note, for example, that hardness when H contains a cycle follows from past results on classes of bounded girth which were proved separately for each problem, but often using a similar technique. There are other results that just apply to one or two of the problems.

Theorem 6.1.16. *Let* H *be a connected graph. On* H*-subgraph-free graphs,* Feedback Vertex Set *and* Independent Feedback Vertex Set *are solvable in polynomial time if* $H \in \mathcal{S} \cup \{S_{1,1,q,r} \mid q \geq r \geq 1\}$*. They are* NP*-complete if* H *contains a cycle or more than one vertex of degree at least* 3 *or* $H \in \{K_{1,5}, S_{2,2,2,2}\}$*.*

Theorem 6.1.17. *Let* H *be a connected graph. On* H*-subgraph-free graphs,* Connected Vertex Cover *is solvable in polynomial time if* $H \in \mathcal{S} \cup \{S_{1,1,q,r} \mid q \geq r \geq 1\}$*. It is* NP*-complete if* H *contains a cycle or* $H = K_{1,5}$*.*

Theorem 6.1.18. *Let* H *be a connected graph. On* H*-subgraph-free graphs,* Coloring *is solvable in polynomial time if* $H \in \mathcal{S} \cup \{S_{1,1,q,r} \mid q \geq r \geq 1\}$ *or if* H *is a forest with maximum degree* 4 *and at most seven vertices. It is* NP*-complete if* H *contains a cycle, or* $H \in \{K_{1,5}, S_{2,2,2,2}\}$*, or if* H *contains a subdivision of the tree* T_1 *as a subgraph, or* H *contains as a subgraph the tree obtained from* T_2 *after subdividing the edge st* p *times,* $0 \leq p \leq 9$*, or* H *contains one of the trees* T_3, T_4, T_5, T_6 *as a subgraph.*

Theorem 6.1.19. *Let* H *be a connected graph. On* H*-subgraph-free graphs,* Matching Cut *is solvable in polynomial time if* $H \in \mathcal{S} \cup \{S_{1,1,q,r} \mid q \geq r \geq 1\}$*. It is* NP*-complete if* H *contains a cycle or* $H = K_{1,5}$*.*

6.2. Independent Feedback Vertex Sets of Subcubic Graphs

In [182], Ueno, Kajitani and Gotoh gave a polynomial-time algorithm for Feedback Vertex Set on subcubic graphs. In this section, we prove Theorem 6.1.12 by showing that Independent Feedback Vertex Set is also polynomial-time solvable on subcubic graphs by demonstrating that the

problems are alike as, for any subcubic graph, one can find a minimum size feedback vertex set that is also an independent set (with a single exceptional case). As the problems can be solved component-wise, we consider only connected graphs.

In fact, we are going to prove a result that is an expansion of Theorem 6.1.12 that will come in handy later. We need some definitions. A *cactus* is a graph in which no two cycles have an edge in common. A cactus is *nice* if no two cycles have a vertex in common (every subcubic cactus is nice since if two cycles share a vertex but not an edge, we can find a vertex of degree 4). A cactus is *very nice* if every vertex belongs to exactly one cycle.

Theorem 6.2.1. *Let G be a connected subcubic graph. Then a minimum size independent feedback vertex set of G is also a minimum size feedback vertex set of G if and only if $G \neq K_4$. Moreover, if $G \neq K_4$ there is a minimum size independent feedback vertex set of G that contains only vertices of degree 3 if and only if G is not a very nice cactus. There is a polynomial-time algorithm to find a minimum size independent feedback vertex set and if G is not a very nice cactus it finds a set that contains only vertices of degree 3.*

Proof. It will be seen that the proof implies a polynomial-time algorithm for finding an independent feedback vertex set of size no greater than a given feedback vertex set.

A feedback vertex set of K_4 must contain at least two vertices and so K_4 has no independent feedback vertex set. In a very nice cactus, the minimum size of a feedback vertex set is equal to the number of cycles and one can easily find such a set that is independent if one permits the inclusion of degree 2 vertices. (For example, pick an arbitrary vertex v and form an independent feedback vertex set by taking the vertex in each cycle that is farthest from v.) If there are k (disjoint) cycles, then, considering the tree-like structure of a very nice cactus, there are $2(k-1)$ vertices of degree 3 that can be considered as $k-1$ adjacent pairs. Thus, no set of k vertices of degree 3 is independent.

So suppose that $G \neq K_4$ is not a very nice cactus. Of course, we may as well also assume that G is not a tree. Let F be a feedback vertex set of G. To prove the theorem, we show that we can find an independent feedback vertex set of G that is no larger than F. We can assume that F contains only vertices of degree 3 since any vertex of degree 2 can be replaced by a nearest vertex of degree 3. As G is neither a tree nor a cycle (a cycle is a very nice cactus), we know that G has vertices of degree 3.

Let $J = \emptyset$. Our approach is to add vertices to J until it forms an independent feedback vertex set. We make some trivial but useful statements:

1. F is a feedback vertex set containing only vertices of degree 3,

2. $J \subseteq F$, and

3. J is a nonseparating independent set of G; that is, no pair of vertices of J are joined by an edge and $G - J$ is connected.

We will repeatedly modify F and J in such a way that these three statements remain true and the size of F does not increase and it remains a feedback vertex set. We can make the following changes without contradicting the three statements.

- We can add a vertex $x \in F \setminus J$ to J if x has no neighbor in J and is not a cutvertex in $G - J$.

- If $x \in F \setminus J$, then we can redefine F as $F \setminus \{x\} \cup \{y\}$ if y is a vertex that belongs to every cycle of $G - (F \setminus \{x\})$ and has degree 3 (that is, y belongs to every cycle of G that contains x but no other vertex of F).

Let $H := G - J$. Our initial aim is to make changes so that H is a graph where no two cycles have a vertex in common; that is, it is a nice cactus.

Claim 6.2.2. *We can modify F and J until H is a nice cactus.*

Proof. Assume H contains two cycles with a common vertex, and, therefore, as G is subcubic, a common edge, else we are done. Consider a subgraph K induced by two cycles of H that have a common edge (so K is 2-connected and has no cut vertex). Of course, F must contain at least one vertex of K; let r be such a vertex.

If r has degree 3 in K, then we can add it to J since it has three neighbors in H (so none in J) and is not a cutvertex in H since $K - \{r\}$ is connected.

Otherwise, r has degree 2 in K. Traversing edges of K away from r in either direction, let p and q be the first vertices of degree 3 in K that are reached (and $p \neq q$ by the definition of K).

Let r' be the first vertex of degree 3 in G reached from r on the path in K towards p.

If r has a neighbor $j \in J$, then we can redefine F as $F \setminus \{r\} \cup \{r'\}$ since every cycle in G containing r also contains either j or r'. Suppose instead that r has no neighbor in J. Let r'' be the neighbor of r in H but not K. If r is not a cutvertex in H, then we can add r to J. If r is a cutvertex in H, then no cycle of H includes the edge rr''. Thus, again, we can redefine F as $F \setminus \{r\} \cup \{r'\}$.

So we either add a vertex to J or modify F by replacing a vertex with another that is closer in K to p. By repetition, we either add a vertex to J or modify F to include p in which case, as noted above, we can add p to J.

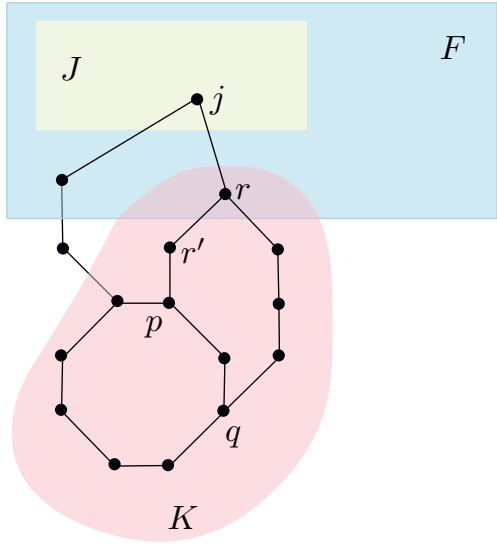

Figure 6.2: The figure shows the situation described in Claim 6.2.2.

Therefore, if H contains two cycles with a common edge, we can increase the size of J and so, ultimately, we can assume that H contains no such pair of cycles and is a nice cactus. This completes the proof of Claim 6.2.2. ◇

By Claim 6.2.2, the cycles of H are vertex disjoint and the graph has a treelike structure: if one replaces each cycle by a single vertex, then a tree is obtained. As F must contain at least one vertex of each cycle of H, if we add to J one vertex chosen from each cycle of H (in any way), it will be no larger than F. If we can do this in such a way that J is an independent set and each vertex has degree- 3, then the proof will be complete. Thus, we must describe how to choose a degree 3 vertex from each cycle of H such that the union of these vertices and J is an independent set, possibly after some further minor modifications.

The reasoning about these modifications will require that H is connected so the requirement above that J be nonseparating was needed.

If H contains no cycles, then J is already an independent feedback vertex set and there is nothing to prove. Otherwise, for any cycle C of H define the following: let $S(C)$ be the set of vertices that contains, for each cycle C' of H other than C, the vertex of C' that is nearest to C in H. See Figure 6.3. Each vertex u of $S(C)$ has degree 3 in H since it has two neighbors in a cycle C' and a neighbor not in C' on the path from u to C. Thus, no vertex of $S(C)$ has a neighbor in J. Moreover, clearly $S(C)$ is an independent set. Thus, $J \cup S(C)$ is an independent set that covers every cycle of G except C. For a vertex v in

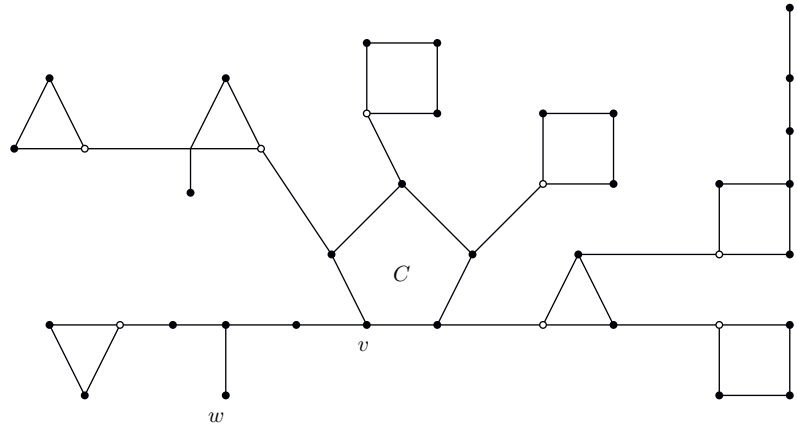

Figure 6.3: A nice subcubic cactus. The central 5-cycle is denoted C and the white vertices form the set $S(C)$. Note that w does not belong to any cycle and v is the nearest vertex to w in a cycle. Thus, $S(C) \cup \{v\}$ is an independent feedback vertex set for the graph.

C, let $F(v) = J \cup S(C) \cup \{v\}$. Note that $F(v)$ can be defined with respect to the cycle that contains v, if such a cycle exists. If we can find a cycle C that contains a vertex v of degree- 3 not adjacent to J or to another cycle in H, then $F(v)$ is an independent feedback vertex set, and we are done.

Suppose instead that no such cycle can be found. Notice that this implies that every vertex of H belongs to a cycle. (If there was a vertex w not in a cycle, then let v be a nearest vertex to w in a cycle and then $F(v)$ is an independent feedback vertex set of degree 3 vertices; again, see Figure 6.3.) So H is a very nice cactus and, moreover, $J \neq \emptyset$.

Let j be a vertex in J with neighbors v_1, v_2 and v_3 in H. Suppose that these three vertices are in the same cycle C of H. If C is a 3-cycle, then $\{j, v_1, v_2, v_3\}$ induces K_4, a contradiction. So we can assume that v_1 and v_2 are not adjacent. Then $J_1 = J \setminus \{j\} \cup \{v_1, v_2\} \cup S(C)$ is an independent feedback vertex set of degree 3 vertices and $|J_1| = |F|$. Indeed, all cycles are covered by J_1 since v_1 and $S(C)$ cover the cycles of H and every cycle containing j includes at least one of v_1 and v_2; J_1 is independent as v_1 and v_2 have degree 2 in H so no other neighbor in J and are not adjacent to vertices in H, such as those of $S(C)$, that do not belong to C, and the vertices of $S(C)$ have degree 3 in H so no neighbors in J.

Suppose instead that v_1, v_2 and v_3 do not all belong to the same cycle. Let C be the cycle that contains v_1 and suppose that v_2 and v_3 do not belong to the same cycle as each other (one might belong to C). Then $J_2 = J \setminus \{j\} \cup \{v_1\} \cup S(C)$ is an independent feedback vertex set of degree- 3

vertices and $|J_2| = |F| - 1$. Indeed all cycles are covered by J_2 since v_1 and $S(C)$ cover the cycles of H and every cycle containing j includes either v_1 or both v_2 and v_3 and all the paths from v_2 to v_3 (that do not include j) go through either a vertex of J or a vertex of $S(C)$ as they are in different cycles in H; J_2 is independent as v_1 has degree 2 in H so, as before, no other neighbor in J or $S(C)$, and the vertices of $S(C)$ have degree 3 in H so no neighbors in J. □

6.3. Graphs Excluding Subdivided Stars as a Subgraph: Structure

Recall that the treedepth of a graph G is the minimum height of a forest F such that for every pair of vertices in G one is the ancestor of the other in F. It is well known that the treewidth of a graph is at most its treedepth. In this section, we aim to show that H-subgraph-free graphs, for certain H, have bounded treedepth. Then we know that problems that are tractable on classes of bounded treewidth are also tractable on these classes. Before presenting our results, we need the following result from [159].

Theorem 6.3.1 ([159])**.** *Let G be a graph of treedepth at least d. Then G has a subgraph isomorphic to a path of length at least d.*

Our next two theorems consider graphs $S_{w,x,y,z}$. By Definition 6.1.13, this graph is four paths sharing an endpoint.

In a small abuse of terminology, we will use *leaf* to mean only a vertex of degree 1 that is adjacent to the center.

Theorem 6.3.2. *Let r be a positive integer. Then the subclass of connected $S_{1,1,1,r}$-subgraph-free graphs that are not subcubic has bounded treedepth.*

Proof. Let G be a connected $S_{1,1,1,r}$-subgraph-free graph that is not subcubic. Hence, it contains a vertex v_0 with neighbors v_1, v_2, v_3, v_4. We will show that G has treedepth at most $2r + 2$. Suppose instead that the treedepth of G is at least $2r + 3$. The graph $G \setminus \{v_0, v_1, v_2, v_3, v_4\}$ must have treedepth at least $2r - 2$ (since adding a vertex to a graph cannot increase the treedepth by more than one), and therefore, by Theorem 6.3.1, it must contain a path P of length at least $2r - 2$. Let Q be a shortest path in G between P and v_0 (which must exist as G is connected). Let z be the vertex where P and Q meet. Let P' be the longest subpath of P of which z is an endpoint. As P' is at least half the length of P, and Q contains at least one edge, the path $P' \cup Q$ contains at least r edges. Thus, there exists in G a subgraph isomorphic to $S_{1,1,1,r}$; the center is v_0, $P' \cup Q$ is the tentacle of length r, and three of v_1, v_2, v_3, v_4 are the

three leaves (since at most one of these four vertices can belong to Q and none belong to P'). This contradiction completes the proof. □

The assumption that the graphs are connected is needed: the class of all graphs that are each a disjoint union of a path and a $K_{1,4}$ is not subcubic but has unbounded treedepth.

Consider now the class of all connected graphs that are each the union of a path and a $K_{1,4}$, one of whose leaves is identified with the endpoint of the path. This is a class of graphs that are connected, not subcubic and $S_{1,1,q,r}$-subgraph-free and again has unbounded treedepth. Thus, in the following analogue of Theorem 6.3.2, we need an additional property. A bridge is *proper* if neither of its incident vertices has degree 1. A graph is *quasi-bridgeless* if it contains no proper bridge.

Theorem 6.3.3. *Let q and r be positive integers. Then the subclass of connected $S_{1,1,q,r}$-subgraph-free graphs that are not subcubic and are quasi-bridgeless has bounded treedepth.*

Proof. Let G be a connected quasi-bridgeless $S_{1,1,q,r}$-subgraph-free graph that is not subcubic. Hence, it contains a vertex v_0 with neighbors v_1, v_2, v_3, v_4. We will show that G has treedepth at most $2(q+r+3)^2+6$. Suppose instead that the treedepth of G is at least $2(q+r+3)^2+7$. The graph $J = G \setminus \{v_0, v_1, v_2, v_3, v_4\}$ must have treedepth at least $2(q+r+3)^2+2$ and therefore, by Theorem 6.3.1, it must contain a path P of length at least $2(q+r+3)^2+2$. Let z be the middle vertex of P. We prove the following claim.

Claim 6.3.4. *If there is a cycle C in G that contains z and also a vertex $v \neq z$ that has two neighbors a and b not on C, then G contains a subgraph isomorphic to $S_{1,1,q,r}$.*

Proof. A *big adorned cycle* is a graph that contains a cycle with at least $q+r+1$ edges and two further vertices each joined by an edge to the same vertex on the cycle; the latter vertex is called the center. If we find a big adorned cycle in G we are done as it contains a subgraph isomorphic to $S_{1,1,q,r}$ (the center is the same, and it is obtained by deleting one or more edges of the cycle). Let C^+ be the union of C, the vertices a and b and the edges va and vb. If $|C| \geq q+r+1$, then C^+ is a big adorned cycle.

So suppose that $|C| \leq q+r$. Consider the intersections of P with $V(C^+)$. A maximal subpath of P whose internal vertices are not in $V(C^+)$ is called an *interval* of P. Note that P has at most $|C^+|+1 \leq q+r+3$ intervals. If all intervals of P have length at most $q+r-1$, then P itself has length at most $(q+r+3)(q+r-1) < (q+r+3)^2$, a contradiction. Hence, at least one of the

intervals has length at least $q + r$; we call such an interval *long*. See Figure 6.4 for an illustration.

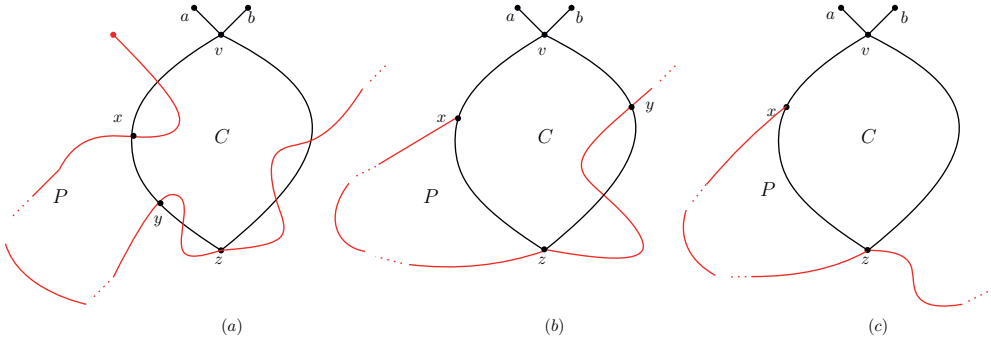

Figure 6.4: The cycle C and path P from the proof of Claim 6.3.4 illustrating the three cases (a) there is a long interval with both endpoints in P, (b) there are two vertex-disjoint long intervals, and (c) there are two long intervals that meet in a single vertex.

Suppose that there is a long interval L of which both endpoints x and y are in $V(C^+)$. Then there are shortest (possibly trivial) paths S and T on C^+ from v to x and y respectively that are vertex disjoint except for v. As x and y are distinct, the union of L, S and T is a cycle on at least $q + r + 1$ edges. As v has four neighbors in C^+, two of them do not belong to this cycle and considering these two neighbors (and the incident edges that join them to v) with the cycle, we have a big adorned cycle centered at v.

Hence, there is no long interval with both endpoints in $V(C^+)$ and we can assume any long interval has just one endpoint in $V(C^+)$.

Suppose that there are two long intervals L_1 and L_2 whose endpoints in C^+ are x and y respectively. If $x = y$, then L_1 and L_2 are the only intervals and their union is P. Since, there is no $S_{1,1,q,r}$ subgraph in G, this implies that P only intersects C^+ in x and so we must have $x = z$. Then there exists in G a subgraph isomorphic to an $S_{1,1,q,r}$ with z as its center, the neighbors of z on C as the leaves and subpaths of L_1 and L_2 as the tentacles. If $x \neq y$, then there are shortest paths S, T on C^+ from v to x and y respectively that are vertex disjoint except for v. Then there exists in G a subgraph isomorphic to an $S_{1,1,q,r}$ with v as its center, the paths S and T, possibly extended by subpaths of L_1 and L_2, as the tentacles and two neighbors of v in C^+ that do not belong to S or T as the leaves.

Hence, there is only one long interval L. As the other intervals are short, they have total length at most $|C^+| \cdot (q + r) < (q + r + 3)^2$. Hence, L has length at least $(q + r + 3)^2 + 2$. As L contains more than half the vertices of P, the middle vertex of P is an internal vertex of L and so does not belong to

C^+. This contradicts that z is the middle vertex of P and completes the proof of the claim. ◇

We now apply the above claim. Note that v_0 and z are distinct as z belongs to J but v_0 does not. Since G is quasi-bridgeless and neither v_0 nor z has degree 1, it follows from Menger's Theorem [152] that there exist two edge-disjoint paths S, T from v_0 to z. If S and T are internally vertex-disjoint paths, then their union forms a cycle that contains z. We can assume that each of S and T contain only one neighbor of v_0 else we can find shortcuts and redefine them. Hence, v_0 has two neighbors not in the cycle and we can apply Claim 6.3.4. If S and T are not internally vertex-disjoint, let v' be a vertex of $(V(S) \cap V(T)) \setminus \{z\}$ that is furthest from v on T. Consider the subpath T' of T from v' to z and the subpath S' of S from v' to z. Since T' does not intersect S by definition, S' and T' are internally vertex disjoint. Hence, their union forms a cycle that contains z. Moreover, v' has degree at least four, of which two neighbors are not on S' or T'. Hence, we can apply Claim 6.3.4 to obtain a contradiction to the assumption that the graph is $S_{1,1,q,r}$-subgraph-free. □

6.4. Graphs Excluding Subdivided Stars as a Subgraph: Algorithms

We present several applications of the structural results of the previous section.

We note that Feedback Vertex Set, Independent Feedback Vertex Set and Coloring can be solved componentwise. In a sense, so can Connected Vertex Cover and Matching Cut since disconnected graphs are NO-instances (except possibly for Connected Vertex Cover instances with edgeless components but these can be ignored).

Theorem 6.4.1. *Let r be a positive integer. A problem Π can be solved in polynomial time on $S_{1,1,1,r}$-subgraph-free graphs if the following hold:*

a) Π can be solved in polynomial time on subcubic graphs,

b) Π can be solved in polynomial time on graphs of bounded treedepth, and

c) Π can be solved componentwise on disconnected graphs.

Proof. Let C be a connected component of a $S_{1,1,1,r}$-subgraph-free graph G. If C is subcubic, then the problem can be solved in polynomial time. Otherwise, by Theorem 6.3.2, C has bounded treedepth and again the problem can be solved in polynomial time. Finally, the solutions for its connected components can be merged in polynomial time. □

Theorem 6.4.2. *Let q and r be positive integers. A problem Π can be solved in polynomial time on $S_{1,1,q,r}$-subgraph-free graphs if the following hold:*

a) Π *can be solved in polynomial time on subcubic graphs,*

b) Π *can be solved in polynomial time on graphs of bounded treedepth, and*

c) Π *can be solved on graphs with proper bridges using a polynomial-time reduction to a family of instances on graphs that are either of bounded treedepth or subcubic.*

Proof. Let H be one of the family of instances obtained from an instance G of Π. As H is either of bounded treedepth or subcubic, the problem can be solved in polynomial time. As we have a reduction, once solved on all the family of instances, we can solve Π on G. □

The simplest way to apply Theorem 6.4.2 is to show that if it is possible to solve Π on each of the family of components obtained by deleting the proper bridges of an instance, then these solutions combine to provide a solution for the initial instance (since the components are quasi-bridgeless and so certainly either of bounded treedepth or subcubic by Theorem 6.3.3),

We now use Theorem 6.4.2 to prove Theorem 6.1.14. We do not apply Theorem 6.4.1 in this paper, as the results it would give us would just be special cases of those we have obtained using Theorem 6.4.2. Nevertheless, there are potential applications of Theorem 6.4.1 as there might be C13 problems that can be solved componentwise but cannot be solved by finding the reduction required by Theorem 6.4.2. We will see, in the proof below, that to solve Independent Feedback Vertex Set via a reduction requires an intricate argument and the careful analysis of possible solutions on subcubic graphs that was provided by Theorem 6.2.1.

Theorem 6.1.14 (restated). *Let q and r be positive integers. The following problems can be solved in polynomial time on $S_{1,1,q,r}$-subgraph-free graphs:* Feedback Vertex Set, Independent Feedback Vertex Set, Connected Vertex Cover, Coloring *and* Matching Cut.

Proof of Theorem 6.1.14. To show that the result follows immediately from Theorem 6.4.2, we can show that the problems can be solved by deleting bridges and considering the resulting graph componentwise; this will be trivial for some problems, but for others we will need to find a different reduction.

For Feedback Vertex Set, as bridges do not belong to cycles, the problem is unchanged when they are deleted.

For Independent Feedback Vertex Set such a straightforward approach is not possible as if we simply delete bridges and solve the problem

on the components, the merged solution might not be independent (since we might choose both endpoints of a deleted bridge). We must argue a little more carefully.

Let G be a $S_{1,1,q,r}$-subgraph-free graph and consider the treelike structure of G when thinking of its non-trivial bridge blocks — the connected components when the bridges are deleted. In fact, consider a subgraph of G that is a block plus all its incident bridges. Some of these subgraphs might be subcubic; let us call these *C-type*. For those that are not, we can assume, by Theorem 6.3.3, that there is a constant c such that their treewidth is at most c; let us call these subgraphs *T-type* (note that this is a weaker claim that the Theorem 6.3.3 provides as we could assume that the treedepth was bounded). If such a subgraph is both subcubic and has treewidth at most c, we will think of it as T-type. We can assume $c \geq 3$ so a very nice cactus is T-type. If subgraphs of the same type overlap (because they are joined by a bridge), we observe that their union is also of that type (since the union is also either, respectively, subcubic or of treewidth at most c). So, merging overlapping subgraphs of the same type as much as possible we can consider G as being made up of C- and T-type subgraphs and bridges that each join a C-type subgraph to a T-type subgraph. As Independent Feedback Vertex Set is a C13 problem and by Theorem 6.2.1, we can solve it on these subgraphs. Before we solve it on a C-type subgraph, we can delete pendant bridges (that link to a T-type subgraph in G) so the incident vertex now has degree at most 2. As a very nice cactus is being considered as a T-type subgraph, we know, by Theorem 6.2.1, that the solutions we find for C-type subgraphs do not use the vertices incident with the bridges. Thus, the solutions can be merged for a solution for G that is also independent.

For Connected Vertex Cover, let G be a $S_{1,1,q,r}$-subgraph-free graph. Clearly, we may assume G is connected, or it has no connected vertex cover. As for Independent Feedback Vertex Set, consider each subgraph J that is a non-trivial bridge block of G and also include the bridges of G incident with the block. Observe that J is quasi-bridgeless and $S_{1,1,q,r}$-subgraph-free. Noticing that a connected vertex cover W of G must contain both vertices incident with any proper bridge, we see that the restriction of W to the vertices of J is a connected vertex cover of J that includes vertices incident with bridges of G. The construction of J means its connected vertex covers will include these vertices adjacent to bridges in G. Thus, we see that have a reduction and can solve the problem on G.

For Coloring, if for a graph G, we color the components of the graph obtained by deleting bridges, then we can merge these into a coloring of G. If the two endpoints of a bridge have been colored alike, then we just permute

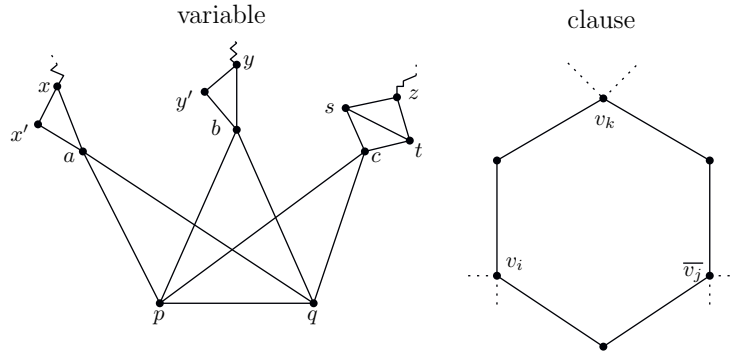

Figure 6.5: The variable and clause gadgets (for clauses of size 3) from the proof of Theorem 6.1.15. The vertices x, y, and z of a variable gadget will be identified with the (labeled) vertices of clause gadgets.

the colors on one of the components. This might create new clashes, but we move to the adjacent components and permute there. By the definition of bridge, we will never have to permute the colors on a component more than once so the process terminates.

For Matching Cut, if a graph contains a bridge, then we have immediately that it is a yes instance. □

6.5. Graphs Excluding Subdivided Stars as a Subgraph: Hardness

We show that on $S_{2,2,2,2}$-subgraph-free graph classes both Feedback Vertex Set and Independent Feedback Vertex Set are NP-complete.

Theorem 6.1.15 (restated). Feedback Vertex Set *and* Independent Feedback Vertex Set *are* NP*-complete on the class of* $S_{2,2,2,2}$*-subgraph free graphs that have maximum degree* 4*.*

Proof. Both problems belong to NP. We shall show a reduction from the following NP-complete problem 2P1N-3SAT [60].

2P1N-3SAT	
Instance:	A CNF formula Φ where each clause contains at most three literals and each variable occurs twice positively and once negatively.
Question:	Does Φ have a satisfying assignment?

Given an instance of 2P1N-3SAT with variables $\{v_1, \ldots, v_n\}$, we construct a graph G as follows. For each variable v_i, we construct the gadget shown in Figure 6.5. The triangles $xx'a$ and $yy'b$ represent the positive occurrences of the variable, while the diamond $zstc$ represents the negative occurrence. For each clause C_j, we construct a hexagon if the clause has size 3 and a square if the clause has size 2 (we may assume that no clause has size 1). Alternate vertices of this clause gadget represent literals and are identified with a vertex x, y or z of the corresponding variable gadget. Clearly, this can be done in such a way that each vertex x and y of each variable gadget is identified with exactly one vertex from a clause gadget that represents a positive literal and each vertex z of each variable gadget is identified with exactly one vertex from a clause gadget that represents a negative literal. Note that G has maximum degree 4.

Claim 6.5.1. *G does not contain $S_{2,2,2,2}$ as a subgraph.*

Proof. Let us consider where we might find the center vertex of an $S_{2,2,2,2}$ in G. Clearly, a vertex v cannot be the center vertex if its 2-neighborhood in G contains a cut of size 3 (that is, if there are three vertices each of distance at most 2 from v that form a cut in G). The center vertex cannot be the vertices p or q of a variable gadget, because the set $\{a, b, c\}$ of the same gadget forms a cut of size 3 in the 2-neighborhood of p and q. The center vertex cannot be the vertices a, b, or c of a variable gadget either, because $\{x, p, q\}$, $\{y, p, q\}$ and $\{z, p, q\}$ respectively form cuts of size 3 in their 2-neighbourhoods. The vertices x, y, and z cannot be the center vertex, as in their 2-neighborhood is a cut of size 3 that contains their two neighbors in a clause gadget and, respectively, a, b and c. The remaining vertices of G have degree less than 4. The claim is proved. ◇

Any feedback vertex set of a variable gadget has size at least 4, because it contains four disjoint cycles. So any feedback vertex set of G must contain at least $4n$ vertices. It only remains to show that G has an (independent) feedback vertex set of size at most $4n$ if and only if Φ is satisfiable.

Assume that Φ has a satisfying assignment. We construct a feedback vertex set F of G. If a variable is true, then the vertices x, y, p, and t of the variable gadget belong to F. If a variable is false, then instead z, a, b, and c belong to F. Thus, F is an independent set (vertices of distinct variable gadgets are not adjacent) and its size is exactly $4n$.

Claim 6.5.2. *F is a feedback vertex set.*

Proof. Notice that if a literal of a clause is satisfied, then, in the clause gadget, the corresponding vertex is in F. Thus, as clause is satisfied, each cycle

contained in a single variable or clause gadget contains a vertex of F. Consider a cycle of G that is not contained within a single gadget. It must include a non-trivial path of some variable gadget where the endpoints are two of $\{x, y, z\}$. If it includes x it must also include a and if it includes y it must also include b. But F contains one of $\{x, a\}$ and one of $\{y, b\}$ so such a cycle also intersects F. Thus, F intersects all the cycles of G. ◇

Conversely, suppose that G has a feedback vertex set F of size at most $4n$. Again, each variable gadget contains at least four vertices of F and so contains exactly four vertices of F. Notice that F cannot contain either $\{x, z\}$ or $\{y, z\}$ as, in each case, there remain three disjoint cycles of the gadget that would need to be covered by just two vertices.

Let us describe a satisfying assignment of Φ. If, for a variable gadget, either x or y belongs to F, we let the variable be true. If z belongs to F, we let it be false. By the preceding argument, there is no possibility that we must set a variable to be both true and false. If none of $\{x, y, z\}$ belong to F, we set the value of the variable arbitrarily. This is a satisfying assignment as every clause gadget (which is a cycle) must have at least one vertex in F and the corresponding variable is satisfied. □

6.6. Proofs of the Classifications

We prove Theorems 6.1.16–6.1.19. Noting that the theorems contain some analogous results, and wishing to avoid repetition, we make a few general comments that apply to all proofs.

We state again that the five problems under consideration are C13 problems. Thus, when $H \in \mathcal{S}$, each theorem follows from Theorem 6.1.11. When $H = S_{1,1,q,r}$, we apply Theorem 6.1.14. Thus, except for Theorem 6.1.18 on Coloring, the following proofs need only cover the NP-complete cases.

Proof of Theorem 6.1.16. We note again that Feedback Vertex Set reduces to Independent Feedback Vertex Set after subdividing each edge, so here we consider only the former.

By Poljak's construction [166], for every integer $g \geq 3$, Feedback Vertex Set is NP-complete for graphs of girth at least g (the girth of a graph is the length of its shortest cycle). Thus, Feedback Vertex Set is NP-complete for H-subgraph-free graphs whenever H contains a cycle.

Suppose that H has m vertices and more than one vertex of degree at least 3. From any graph G, if we subdivide each edge m times, we obtain a graph J that is H-subgraph-free, since the distance between any pair of vertices of degree more than 2 is at least $m + 1$. In finding a minimum size feedback

vertex set of J, we may as well restrict ourselves to selecting vertices of G. This implies that Feedback Vertex Set is NP-complete for H-subgraph-free graphs.

The problem is NP-complete on planar graphs of maximum degree 4 [176] (so for $K_{1,5}$-subgraph-free graphs).

Theorem 6.1.15 completes the proof. □

Proof of Theorem 6.1.17. For every integer $g \geq 3$, Connected Vertex Cover is NP-complete for graphs of girth at least g [158], so also for H-subgraph-free graphs whenever H contains a cycle. It is NP-complete on graphs of maximum degree 4 [93], so for $K_{1,5}$-subgraph-free graphs. □

Proof of Theorem 6.1.18. For every integer $g \geq 3$, Coloring is NP-complete for graphs of girth at least g [145], so also for H-subgraph-free graphs whenever H contains a cycle. In [95], it was shown that Coloring is NP-complete on (planar) graphs of maximum degree 4, and so too for $K_{1,5}$-subgraph-free graphs. The other cases are all proved in [103]. □

Proof of Theorem 6.1.19. For every integer $g \geq 3$, Matching Cut is NP-complete for graphs of girth at least g [81], so also for H-subgraph-free graphs whenever H contains a cycle. It is NP-complete on graphs of maximum degree 4 [51], so for $K_{1,5}$-subgraph-free graphs. □

6.7. Conclusions

We made significant progress towards classifying the complexity of five well-known C13-problems on H-subgraph-free graphs, extending previously known results. In particular, we identified a gap in the literature, and provided a polynomial-time algorithm for Independent Feedback Vertex Set for subcubic graphs.

If H is connected, then we narrowed the gap for these problems to the open case where $H = S_{1,p,q,r}$, so H is a subdivided star with one short leg and three arbitrarily long legs. To obtain a result for connected $S_{1,p,q,r}$-subgraph-free graphs similar to our previous results, we would need the graphs to be 3-edge-connected. Indeed, the statement is false without this assumption. Consider the class of all graphs that are each the union of a path and a $K_{1,4}$ two of whose leaves are identified with distinct endpoints of the path and whose other two leaves are made adjacent. This is a class of graphs that are bridgeless, not subcubic and $S_{1,p,q,r}$-subgraph-free, and again has unbounded treedepth. It is not yet clear whether a suitably modified theorem statement would indeed hold. In addition, it is unclear whether this would yield a result that could be

applied in the same way as Theorems 6.3.2 and 6.3.3 were above. We leave the case $H = S_{1,p,q,r}$ as future research.

Finally, we also leave determining the complexity of Connected Vertex Cover and Matching Cut on $S_{2,2,2,2}$-subgraph-free graphs as an open problem.

7

Conclusion

In this thesis, we studied some well known NP-hard problems and investigated the limits of their tractability. By imposing restrictions on the structure of the input graph, we were able to design efficient algorithms for some of these problems. On the other hand, we also discovered problems that are hard even on simple classes of input.

In Chapter 3, we investigated the parameterized complexity of Planar Edge Multiway Cut with respect to the parameter *terminal face-cover number* (k). We presented an algorithm with a run-time of $2^{O(k^2 \cdot \log k)} n^{O(\sqrt{k})}$. Due to the result of Marx [149], we know this algorithm to be ETH-tight. A similar complexity was observed for the closely related problem Minimum Steiner Tree on planar graphs parameterized by the *terminal face cover number* [118]. An interesting direction for further research is to investigate the parameterized complexity of these problems on other classes of embedded graphs with respect to the same parameter. More specifically, it is an open question whether there exists a subexponential parameterized algorithm (with the exponent being the parameter) solving either of the aforementioned problems on bounded-genus graphs.

Next, we showed in Chapter 4 that Edge Multiway Cut and Node Multiway Cut are NP-hard on planar subcubic graphs. Our result implies a dichotomy in terms of the maximum degree of the input graph, that is, both the problems are in P when the input graph has maximum degree at most 2 and NP-hard otherwise. It also implies that both the problems can be completely classified on $\mathcal{H}$-topological-minor-free graphs and $\mathcal{H}$-subgraph-free graphs due to the result of Robertson and Seymour [169] in the former case and Johnson *et al.* [120] in the latter.

In Chapter 5, our meta-classification was able to congregate from literature as well as prove new complexity classifications for a multitude of problems. Our classification for $\mathcal{H}$-subgraph-free graphs requires that $\mathcal{H}$ is a finite set of graphs. It is an interesting open problem to find a similar dichotomy for all those problems when $\mathcal{H}$ can be infinite. The biggest hurdle here is our lack of understanding of the treewidth of $\mathcal{H}$-subgraph-free graphs for an infinite set $\mathcal{H}$. Besides that, finding more problems that fit our complexity framework can help classify their complexity on $\mathcal{H}$-subgraph-free graphs. Finally, an interesting open problem is to investigate the complexity of problems that satisfy only two of the three conditions of our framework, on the class of $\mathcal{H}$-subgraph-free graphs.

Motivated by last question, there have been recent studies looking at the complexity of the problems that partly fit into our framework [121, 122, 146]. In Chapter 6, we investigate the complexity of problems that are tractable on graphs of bounded treewidth as well as graphs of maximum degree at most 3, on the class of H-subgraph-free graphs. Firstly, we showed that Independent Feedback Vertex Set is one such problem. There was no polynomial-time algorithm known for the problem on the class of subcubic graphs prior to our result. For any connected graph H, we came close to obtaining a complexity dichotomy for Feedback Vertex Set and Independent Feedback Vertex Set on H-subgraph-free graphs, with the only open case being $H = S_{1,p,q,r}$, that is a subdivided star with one tiny leg and three long legs. Furthermore, it remains open to determine the complexity of Connected Vertex Cover and Matching Cut on $S_{2,2,2,2}$-subgraph-free graphs.

We conclude this thesis with the hope that with our deeper understanding of the structure of various classes of graphs, in particular $\mathcal{H}$-subgraph-free graphs, we can push further the frontiers of tractability of important NP-hard problems.

Bibliography

[1] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *Proc. of SODA 2016*, pages 377–391. ACM-SIAM, 2016.

[2] Tara Abrishami, Maria Chudnovsky, and Kristina Vušković. Induced Subgraphs and Tree Decompositions I. Even-Hole-Free Graphs of Bounded Degree. *Journal of Combinatorial Theory, Series B*, 157:144–175, 2022.

[3] Tara Abrishami, Bogdan Alecu, Maria Chudnovsky, Sepehr Hajebi, and Sophie Spirkl. Induced Subgraphs and Tree Decompositions VII. Basic Obstructions in H-Free Graphs. *Journal of Combinatorial Theory, Series B*, 164:443–472, 2024.

[4] Michael O. Albertson, Glenn G. Chappell, Henry A. Kierstead, André Kündgen, and Radhika Ramamurthi. Coloring with No 2-Colored P_4's. *Electronic Journal of Combinatorics*, 11, 2004.

[5] Vladimir E. Alekseev. On Easy and Hard Hereditary Classes of Graphs with Respect to the Independent Set Problem. *Discrete Applied Mathematics*, 132:17–26, 2003.

[6] Vladimir E. Alekseev and Dmitry V. Korobitsyn. Complexity of Some Problems on Hereditary Graph Classes. *Diskretnaya Matematika*, 4: 34–40, 1992.

[7] Vladimir E. Alekseev, Rodica Boliac, Dmitry V. Korobitsyn, and Vadim V. Lozin. NP-Hard Graph Problems and Boundary Classes of Graphs. *Theoretical Computer Science*, 389:219–236, 2007.

[8] Stefan Arnborg and Andrzej Proskurowski. Linear Time Algorithms for NP-Hard Problems Restricted to Partial k-Trees. *Discrete Applied Mathematics*, 23:11–24, 1989.

[9] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy Problems for Tree-Decomposable Graphs. *Journal of Algorithms*, 12:308–340, 1991.

[10] Sanjeev Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems. *Journal of the ACM*, 45(5):753–782, 1998.

[11] Mohammad Hossein Bateni, Mohammad Taghi Hajiaghayi, and Dániel Marx. Approximation Schemes for Steiner Forest on Planar Graphs and Graphs of Bounded Treewidth. *Journal of the ACM*, 58:21:1–21:37, 2011.

[12] Mohammad Hossein Bateni, MohammadTaghi Hajiaghayi, Philip N. Klein, and Claire Mathieu. A Polynomial-time Approximation Scheme for Planar Multiway Cut. In *Proc. of SODA 2012*, pages 639–655. SIAM, 2012.

[13] Cédric Bentz. A Polynomial-Time Algorithm for Planar Multicuts with Few Source-Sink Pairs. In *Proc. of IPEC 2012*, volume 7535 of *LNCS*, pages 109–119. Springer, 2012.

[14] Cédric Bentz. An FPT Algorithm for Planar Multicuts with Sources and Sinks on The Outer Face. *Algorithmica*, 81(1):224–237, 2019.

[15] Benjamin Bergougnoux, Charis Papadopoulos, and Jan Arne Telle. Node Multiway Cut and Subset Feedback Vertex Set on Graphs of Bounded Mim-Width. *Algorithmica*, 84:1385–1417, 2022.

[16] Marshall Bern. Faster Exact Algorithms for Steiner Trees in Planar Networks. *Networks*, 20:109–120, 2006.

[17] Daniel Bienstock and Clyde L. Monma. On The Complexity of Covering Vertices by Faces in a Planar Graph. *SIAM Journal of Computing*, 17 (1):53–76, 1988.

[18] Daniel Bienstock, Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly Excluding a Forest. *Journal of Combinatoral Theory, Series B*, 52:274–283, 1991.

[19] Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM Journal of Computing*, 39(2):546–563, 2009.

[20] Hans L. Bodlaender. On Linear Time Minor Tests with Depth-First Search. *Journal of Algorithms*, 14:1–23, 1993.

[21] Hans L. Bodlaender. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[22] Hans L. Bodlaender and Ton Kloks. Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs. *Journal of Algorithms*, 21: 358–402, 1996.

[23] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *Journal of the ACM*, 63:44:1–44:69, 2016.

[24] Hans L. Bodlaender, Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, Yoshio Okamoto, Yota Otachi, and Tom C. van der Zanden. Subgraph Isomorphism on Graph Classes that Exclude a Substructure. *Algorithmica*, 82:3566–3587, 2020.

[25] Hans L. Bodlaender, Nick Brettell, Matthew Johnson, Giacomo Paesani, Daniël Paulusma, and Erik Jan van Leeuwen. Steiner Trees for Hereditary Graph Classes: A Treewidth Perspective. *Theoretical Computer Science*, 867:30–39, 2021.

[26] Hans L. Bodlaender, Édouard Bonnet, Lars Jaffke, Dusan Knop, Paloma T. Lima, Martin Milanic, Sebastian Ordyniak, Sukanya Pandey, and Ondrej Suchý. Treewidth is NP-Complete on Cubic Graphs. In *Proc. of IPEC 2023*, volume 285 of *LIPIcs*, pages 7:1–7:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[27] Hans L. Bodlaender, Matthew Johnson, Barnaby Martin, Jelle J. Oostveen, Sukanya Pandey, Daniël Paulusma, Siani Smith, and Erik Jan van Leeuwen. Complexity Framework for Forbidden Subgraphs IV: The Steiner Forest Problem. *arXiv*, abs/2305.01613, 2023.

[28] Jan Bok, Nikola Jedlicková, Barnaby Martin, Daniël Paulusma, and Siani Smith. Acyclic, Star and Injective colouring: A Complexity Picture for H-Free Graphs. In *Proc. of ESA 2020*, volume 173 of *LIPIcs*, pages 22:1–22:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[29] Rodica Boliac and Vadim V. Lozin. On the Clique-Width of Graphs in Hereditary Classes. In *Proc. of ISAAC 2002*, volume 2518 of *LNCS*, pages 44–54. Springer, 2002.

[30] Édouard Bonnet, Dibyayan Chakraborty, and Julien Duron. Cutting Barnette Graphs Perfectly is Hard. In *Proc. of WG 2023*, volume 14093 of *LNCS*, pages 116–129. Springer, 2023.

[31] Paul S. Bonsma. The Complexity of the Matching-Cut Problem for Planar Graphs and other Graph Classes. *Journal of Graph Theory*, 62: 109–126, 2009.

[32] Glencora Borradaile, Philip N. Klein, and Claire Mathieu. Steiner Tree in Planar Graphs: An $O(n \log n)$ Approximation Scheme with Singly-Exponential Dependence on Epsilon. In *Algorithms and Data Structures*, pages 275–286. Springer, 2007.

[33] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. SIAM, 1999. ISBN 978-0-89871-432-6.

[34] Rowland Leonard Brook. On Colouring the Nodes of a Network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37:194–197, 1941.

[35] Andrei A. Bulatov. A Dichotomy Theorem for Nonuniform CSPs. In *Proc. of FOCS 2017*, pages 319–330. IEEE Computer Society, 2017.

[36] Jonathan F. Buss and Judy Goldsmith. Nondterminism within P. In *Proc. of STACS 1991*, volume 480 of *LNCS*, pages 348–359. Springer, 1991.

[37] Jiazhen Cai. Counting Embeddings of Planar Graphs Using DFS Trees. *SIAM Journal of Discrete Mathematics*, 6(3):335–352, 1993.

[38] Leizhen Cai. Parameterized Complexity of Cardinality Constrained Optimization Problems. *The Computer Journal*, 51(1):102–121, 2008.

[39] Gruia Călinescu and Cristina G. Fernandes. Multicuts in Unweighted Digraphs with Bounded Degree and Bounded Tree-Width. *Electronic Notes in Discrete Mathematics*, 7:194–197, 2001.

[40] Gruia Călinescu, Howard J. Karloff, and Yuval Rabani. An Improved Approximation Algorithm for Multiway Cut. *Journal of Computer and System Sciences*, 60:564–574, 2000.

[41] Yixin Cao, Jianer Chen, and Jia-Hao Fan. An $O^*(1.84^k)$ Parameterized Algorithm for the Multiterminal Cut Problem. *Information Processing Letters*, 114:167–173, 2014.

[42] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (Complicated) Surfaces is Hard. In *Proc. of SoCG 2006*, pages 421–429. ACM, 2006.

[43] Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum Cuts and Shortest Homologous Cycles. In *Proc. of SoCG 2009*, pages 377–385. ACM, 2009.

[44] Danny Chen and Xiadong Wu. Efficient Algorithms for k-Terminal Cuts on Planar Graphs. *Algorithmica*, 38:299–316, 02 2004.

[45] Danny Z. Chen and Jinhui Xu. Shortest Path Queries in Planar Graphs. In *Proc. of STOC 2000*, pages 469–478. ACM, 2000.

[46] Jianer Chen, Yang Liu, and Songjian Lu. An Improved Parameterized Algorithm for the Minimum Node Multiway Cut Problem. *Algorithmica*, 55:1–13, 2009.

[47] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Almost-Linear-Time Algorithms for Maximum Flow and Minimum-Cost Flow. *Communications of the ACM*, 66(12):85–92, 2023.

[48] Kevin K. H. Cheung and Kyle Harvey. Revisiting a Simple Algorithm for the Planar Multiterminal Cut Problem. *Operations Research Letters*, 38(4):334–336, 2010.

[49] Rajesh Chitnis, Mohammad Taghi Hajiaghayi, and Dániel Marx. Fixed-Parameter Tractability of Directed Multiway Cut Parameterized by the Size of the Cutset. *SIAM Journal on Computing*, 42:1674–1696, 2013.

[50] Miroslav Chlebík and Janka Chlebíková. The Complexity of Combinatorial Optimization Problems on d-Dimensional Boxes. *SIAM Journal on Discrete Mathematics*, 21:158–169, 2007.

[51] Vasek Chvátal. Recognizing Decomposable Graphs. *Journal of Graph Theory*, 8:51–53, 1984.

[52] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit Disk Graphs. *Discrete Mathematics*, 86:165–177, 1990.

[53] Vincent Cohen-Addad, Éric Colin de Verdière, and Arnaud de Mesmay. A Near-Linear Approximation Scheme for Multicuts of Embedded Graphs With a Fixed Number of Terminals. *SIAM Journal of Computing*, 50(1): 1–31, 2021.

[54] Vincent Cohen-Addad, Éric Colin de Verdière, Dániel Marx, and Arnaud de Mesmay. Almost Tight Lower Bounds for Hard Cutting Problems in Embedded Graphs. *Journal of ACM*, 68(4):30:1–30:26, 2021.

[55] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 1989. ISBN 0-262-03141-8.

[56] Bruno Courcelle. The Monadic Second-order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Information and Computation*, 85:12–75, 1990.

[57] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. On Multiway Cut Parameterized Above Lower Bounds. *ACM Transactions on Computation Theory*, 5:3:1–3:11, 2013.

[58] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015. ISBN 978-3-319-21274-6.

[59] Konrad K. Dabrowski and Daniël Paulusma. Clique-Width of Graph Classes Defined by Two Forbidden Induced Subgraphs. *Computer Journal*, 59:650–666, 2016.

[60] Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

[61] David P. Dailey. Uniqueness of Colorability and Colorability of Planar 4-Regular Graphs are NP-Complete. *Discrete Mathematics*, 30:289–293, 1980.

[62] Anuj Dawar, Martin Grohe, Stephan Kreutzer, and Nicole Schweikardt. Approximation Schemes for First-order Definable Optimisation Problems. In *Proc. of LICS 2006*, pages 411–420. IEEE Computer Society, 2006.

[63] Éric Colin de Verdière. Multicuts in Planar and Bounded-Genus Graphs with Bounded Number of Terminals. In *Proc. of ESA 2015*, pages 373–385. Springer, 2015.

[64] Rina Dechter and Judea Pearl. Tree Clustering for Constraint Networks. *Artificial Intelligence*, 38:353–366, 1989.

[65] Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential Parameterized Algorithms on Bounded-Genus Graphs and H-Minor-Free Graphs. *Journal of the ACM*, 52:866–893, 2005.

[66] Xiaojie Deng, Bingkai Lin, and Chihao Zhang. Multi-Multiway Cut Problem on Graphs of Bounded Branchwidth. In *Proc. of FAW-AAIM 2013, LNCS*, volume 7924, pages 315–324. Springer, 2013.

[67] Reinhard Diestel. *Graph Theory, 3rd edition.* Springer, 2005. ISBN 978-3-642-14278-9.

[68] Yefim Dinitz. Dinitz' Algorithm: The Original Version and Even's Version. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, volume 3895 of *LNCS*, pages 218–240. Springer, 2006.

[69] Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Decompositions. *Algorithmica*, 58(3):790–810, 2010.

[70] Rodney G. Downey and Michael R. Fellows. Fixed Parameter Tractability and Completeness. In *Complexity Theory: Current Research, Dagstuhl Workshop, February 2-8, 1992*, pages 191–225. Cambridge University Press, 1992.

[71] Rodney G. Downey and Michael R. Fellows. Fixed-Parameter Intractability. In *Proc. of FCT 1992*, pages 36–49. IEEE Computer Society, 1992.

[72] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999. ISBN 978-1-4612-6798-0.

[73] S. E. Dreyfus and R. A. Wagner. The Steiner Problem in Graphs. *Networks*, 1(3):195–207, 1971.

[74] Jack Edmonds and Richard M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *Journal of ACM*, 19 (2):248–264, 1972.

[75] John A. Ellis, Ivan Hal Sudborough, and Jonathan S. Turner. The vertex Separation and Search Number of a Graph. *Information and Computation*, 113(1):50–79, 1994.

[76] Jeff Erickson and Amir Nayyeri. Shortest Non-Crossing Walks in the Plane. In *Proc. of SODA 2011*, pages 297–208. SIAM, 2011.

[77] Jeff Erickson and Amir Nayyeri. Minimum Cuts and Shortest Non-Separating Cycles via Homology Covers. In *Proc. of SODA 2011*, pages 1166–1176. SIAM, 2011.

[78] Jeff Erickson, Kyle Fox, and Amir Nayyeri. Global Minimum Cuts in Surface Embedded Graphs. In *Proc. of SODA 2012*, pages 1309–1318. SIAM, 2012.

[79] Ranel E. Erickson, Clyde L. Monma, and Arthur F. Veinott. Send-and-Split Method for Minimum-Concave-Cost Network Flows. *Mathematics of Operations Research*, 12(4):634–664, 1987.

[80] Jacob Evald and Søren Dahlgaard. Tight Hardness Results for Distance and Centrality Problems in Constant Degree Graphs. *arXiv*, abs/1609.08403, 2016.

[81] Carl Feghali, Felicia Lucke, Daniël Paulusma, and Bernard Ries. New Hardness Results for Matching Cut and Disconnected Perfect Matching. *arXiv*, abs/2212.12317, 2022.

[82] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010. ISBN 978-3-642-16532-0.

[83] Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. Measure and Conquer: Domination - A Case Study. In *Proc. of ICALP 2005*, volume 3580 of *LNCS*, pages 191–203. Springer, 2005.

[84] Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. Measure and Conquer: A Simple $O(2^{0.288n})$ Independent Set Algorithm. In *Proc. of SODA 2006*, pages 18–25. ACM Press, 2006.

[85] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Excluded Grid Minors and Efficient Polynomial-Time Approximation Schemes. *Journal of the ACM*, 65:10:1–10:44, 2018.

[86] Lester R. Ford and Delbert R. Fulkerson. Maximal Flow Through a Network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[87] Florent Foucaud, Hervé Hocquard, and Dimitri Lajou. Complexity and Algorithms for Injective Edge-Coloring in Graphs. *Information Processing Letters*, 170:106121, 2021.

[88] András Frank and Alexander Schrijver. Vertex-Disjoint Simple Paths of Given Homotopy in a Planar Graph. In *Polyhedral Combinatorics, Proc. of DIMACS Workshop 1989*, volume 1 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 139–162. DIMACS/AMS, 1990.

[89] Greg N. Frederickson. Planar Graph Decomposition and All Pairs Shortest Paths. *Journal of ACM*, 38(1):162–204, 1991.

[90] Esther Galby, Dániel Marx, Philipp Schepper, Roohani Sharma, and Prafullkumar Tale. Domination and Cut Problems on Chordal Graphs with Bounded Leafage. In *Proc. of IPEC 2022*, volume 249 of *LIPIcs*, pages 14:1–14:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[91] Anna Galluccio, Pavol Hell, and Jaroslav Nešetřil. The Complexity of H-Colouring of Bounded Degree Graphs. *Discrete Mathematics*, 222: 101–109, 2000.

[92] Michael R. Garey and David S. Johnson. The Rectilinear Steiner Tree Problem is NP-Complete. *SIAM Journal on Applied Mathematics*, 32: 826–834, 1977.

[93] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979. ISBN 0-7167-1044-7.

[94] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some Simplified NP-Complete Problems. In *Proc. of STOC 1974*, pages 47–63. ACM, 1974.

[95] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some Simplified NP-Complete Graph Problems. *Theoretical Computer Science*, 1:237–267, 1976.

[96] Michael R. Garey, David S. Johnson, and Robert Endre Tarjan. The Planar Hamiltonian Circuit Problem is NP-Complete. *SIAM Journal on Computing*, 5:704–714, 1976.

[97] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-Dual Approximation Algorithms for Integral Flow and Multicut in Trees, with Applications to Matching and Set Cover. In *Proc. of ICALP 1993*, volume 700 of *LNCS*, pages 64–75. Springer, 1993.

[98] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Multiway Cuts in Node Weighted Graphs. *Journal of Algorithms*, 50:49–61, 2004.

[99] Peter Gartland, Daniel Lokshtanov, Tomás Masarík, Marcin Pilipczuk, Michal Pilipczuk, and Pawel Rzazewski. Maximum Weight Independent Set in Graphs with no Long Claws in Quasi-Polynomial Time. *arXiv*, abs/2305.15738, 2023.

[100] Omer Giménez and Marc Noy. Asymptotic Enumeration and Limit Laws of Planar Graphs. *Journal of the American Mathematical Society*, 22(2): 309–329, 2009.

[101] Oded Goldreich. Computational Complexity: A Conceptual Perspective. *SIGACT News*, 39:35–39, 2008.

[102] Petr A. Golovach and Daniël Paulusma. List Coloring in the Absence of Two Subgraphs. *Discrete Applied Mathematics*, 166:123–130, 2014.

[103] Petr A. Golovach, Daniël Paulusma, and Bernard Ries. Coloring Graphs Characterized by a Forbidden Subgraph. *Discrete Applied Mathematics*, 180:101–110, 2015.

[104] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*. 2004. ISBN 978-0-12-289260-8.

[105] Martin Grohe. The Complexity of Homomorphism and Constraint Satisfaction Problems Seen from the Other Side. *Journal of the ACM*, 54: 1:1–1:24, 2007.

[106] Andrzej Grzesik, Tereza Klimosová, Marcin Pilipczuk, and Michal Pilipczuk. Polynomial-Time Algorithm for Maximum Weight Independent Set on P_6-Free graphs. *ACM Transaction on Algorithms*, 18:4:1–4:57, 2022.

[107] Qian-Ping Gu and Hisao Tamaki. Improved Bounds on the Planar Branchwidth with Respect to the Largest Grid Minor Size. *Algorithmica*, 64(3):416–453, 2012.

[108] Sylvain Guillemot. FPT Algorithms for Path-Transversal and Cycle-Transversal Problems. *Discrete Optimization*, 8(1):61–71, 2011.

[109] F. Hadlock. Finding a Maximum Cut of a Planar Graph in Polynomial Time. *SIAM Journal on Computing*, 4:221–225, 1975.

[110] Frank Harary. *Graph Theory*. Addison-Wesley, 1991. ISBN 978-0-201-02787-7.

[111] David Hartvigsen. The Planar Multiterminal Cut Problem. *Discrete Applied Mathematics*, 85(3):203–222, 1998.

[112] Michael Held and Richard M. Karp. The Traveling-Salesman Problem and Minimum Spanning Trees: Part II. *Mathematical Programming*, 1 (1):6–25, 1971.

[113] Robert Hickingbotham. Induced Subgraphs and Path Decompositions. *Electronic Journal of Combinatorics*, 30(2):P2.37, 2022.

[114] John E. Hopcroft and Robert Endre Tarjan. Efficient Planarity Testing. *Journal of ACM*, 21(4):549–568, 1974.

[115] Te C Hu. Integer Programming and Network Flows. Technical report, Wisconsin University Madison, Department of Computer Sciences, 1969.

[116] Lars Jaffke, O-joung Kwon, and Jan Arne Telle. Mim-Width I. Induced Path Problems. *Discrete Applied Mathematics*, 278:153–168, 2020.

[117] Bart M. P. Jansen, Marcin Pilipczuk, and Erik Jan van Leeuwen. A Deterministic Polynomial Kernel for Odd Cycle Transversal and Vertex Multiway Cut in Planar Graphs. In *Proc. of STACS 2019*, volume 126 of *LIPIcs*, pages 39:1–39:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[118] Bart M. P. Jansen, Marcin Pilipczuk, and Erik Jan van Leeuwen. A Deterministic Polynomial Kernel for Odd Cycle Transversal and Vertex Multiway Cut in Planar Graphs. *SIAM Journal on Discrete Mathematics*, 35(4):2387–2429, 2021.

[119] Klaus Jansen and Petra Scheffler. Generalized Coloring for Tree-like Graphs. *Discrete Applied Mathematics*, 75:135–155, 1997.

[120] Matthew Johnson, Barnaby Martin, Jelle J. Oostveen, Sukanya Pandey, Siani Smith, and Erik Jan van Leeuwen. Complexity Framework for Forbidden Subgraphs I: The Framework. *arXiv*, abs/2211.12887, 2022.

[121] Matthew Johnson, Barnaby Martin, Sukanya Pandey, Daniël Paulusma, Siani Smith, and Erik Jan van Leeuwen. Complexity Framework for Forbidden Subgraphs III: When Problems Are Tractable on Subcubic Graphs. In *Proc. of MFCS 2023*, volume 272 of *LIPIcs*, pages 57:1–57:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[122] Matthew Johnson, Barnaby Martin, Siani Smith, Sukanya Pandey, Daniël Paulusma, and Erik Jan van Leeuwen. Edge Multiway Cut and Node Multiway Cut are NP-Complete on Subcubic Graphs. In *Proc. of SWAT 2024*, volume 294 of *LIPIcs*, pages 29:1–29:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

[123] Marcin Kamiński. Max-Cut and Containment Relations in Graphs. *Theoretical Computer Science*, 438:89–95, 2012.

[124] David R. Karger, Philip Klein, Cliff Stein, Mikkel Thorup, and Neal E. Young. Rounding Algorithms for a Geometric Embedding of Minimum Multiway Cut. In *Proc. of STOC 1999*, pages 668–678. ACM, 1999.

[125] Richard M. Karp. Reducibility Among Combinatorial Problems. In *Proc. of CCC 1972*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[126] Richard M. Karp. Dynamic Programming Meets the Principle of Inclusion and Exclusion. *Operations Research Letters*, 1(2):49–51, 1982.

[127] Walter Kern and Daniël Paulusma. Contracting to a Longest Path in H-Free Graphs. In *Proc. of ISAAC 2020*, volume 181 of *LIPIcs*, pages 22:1–22:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[128] Nancy G. Kinnersley. The Vertex Separation Number of a Graph Equals its Path-Width. *Information Processing Letters*, 42:345–350, 1992.

[129] Philip N. Klein and Dániel Marx. Solving Planar k-Terminal Cut in $O(n^{c\sqrt{k}})$ Time. In *Proc. of ICALP 2012*, volume 7391 of *LNCS*, pages 569–580. Springer, 2012.

[130] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. A Bound on the Pathwidth of Sparse Graphs with Applications to Exact Algorithms. *SIAM Journal of Discrete Mathematics*, 23(1):407–427, 2009.

[131] Tuukka Korhonen. Grid Induced Minor Theorem for Graphs of Small Degree. *Journal of Combinatorial Theory, Series B*, 160:206–214, 2023.

[132] Nicholas Korpelainen, Vadim V. Lozin, Dmitriy S. Malyshev, and Alexander Tiskin. Boundary Properties of Graphs for Algorithmic Graph Problems. *Theoretical Computer Science*, 412:3545–3554, 2011.

[133] Stefan Kratsch and Magnus Wahlström. Representative Sets and Irrelevant Vertices: New Tools for Kernelization. In *Proc. of FOCS 2012*, pages 450–459. IEEE Computer Society, 2012.

[134] Robert Krauthgamer and Havana Inbal Rika. Refined Vertex Sparsifiers of Planar Graphs. *SIAM Journal on Discrete Mathematics*, 34(1):101–129, 2020.

[135] Robert Krauthgamer, James R. Lee, and Havana Rika. Flow-Cut Gaps and Face Covers in Planar Graphs. In *Proc. of SODA 2019*, pages 525–534. SIAM, 2019.

[136] Stephan Kreutzer. Algorithmic Meta-Theorems. *London Mathematical Society Lecture Note Series*, 379:177–270, 2011.

[137] Casimir Kuratowski. Sur le Problème des Courbes Gauches en Topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930.

[138] Richard E. Ladner. On the Structure of Polynomial Time Reducibility. *Journal of ACM*, 22(1):155–171, 1975.

[139] Van Bang Le and Jan Arne Telle. The Perfect Matching Cut Problem Revisited. In *Proc. of WG 2021*, volume 12911 of *LNCS*, pages 182–194. Springer, 2021.

[140] Richard J. Lipton and Robert E. Tarjan. A Separator Theorem for Planar Graphs. *SIAM Journal on Applied Mathematics*, 36:177–189, 1977.

[141] Daniel Lokshtanov and Jesper Nederlof. Saving Space by Algebraization. In *Proc. of STOC 2010*, pages 321–330. ACM, 2010.

[142] Daniel Lokshtanov, Martin Vatshelle, and Yngve Villanger. Independent Set in P_5-Free Graphs in Polynomial Time. In *Proc. of SODA 2014*, pages 570–581. SIAM, 2014.

[143] Vadim V. Lozin and Igor Razgon. Tree-Width Dichotomy. *European Journal of Combinatorics*, 103, 2022.

[144] Gary MacGillivray and Mark H. Siggers. On the Complexity of H-Colouring Planar Graphs. *Discrete Mathematics*, 309:5729–5738, 2009.

[145] Frédéric Maffray and Myriam Preissmann. On the NP-Completeness of the k-Colorability Problem for Triangle-free Graphs. *Discrete Mathematics*, 162:313–317, 1996.

[146] Barnaby Martin, Sukanya Pandey, Daniël Paulusma, Siani Smith, and Erik Jan van Leeuwen. Complexity Framework for Forbidden Subgraphs II: When Hardness is Not Preserved under Edge Subdivision. *arXiv*, abs/2211.14214, 2022.

[147] Barnaby Martin, Daniël Paulusma, and Siani Smith. Hard Problems that Quickly Become Very Easy. *Information Processing Letters*, 174, 2022.

[148] Dániel Marx. Parameterized Graph Separation Problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

[149] Dániel Marx. A Tight Lower Bound for Planar Multiway Cut with Fixed Number of Terminals. In *Proc. of ICALP 2012*, volume 7391 of *LNCS*, pages 677–688. Springer, 2012.

[150] Dániel Marx and Michał Pilipczuk. Optimal Parameterized Algorithms for Planar Facility Location Problems Using Voronoi Diagrams. In *Proc. of ESA 2015*, volume 9294 of *LNCS*, pages 865–877. Springer, 2015.

[151] K. Matsumoto, Takao Nishizeki, and N. Saito. An Efficient Algorithm for Finding Multicommodity Flows in Planar Networks. *SIAM Journal of Computing*, 14:289–302, 1985.

[152] Karl Menger. Zur Allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

[153] Matthias Middendorf and Frank Pfeiffer. On the Complexity of the Disjoint Paths Problem. *Combinatorica*, 13:97–107, 1993.

[154] George J. Minty. On Maximal Independent Sets of Vertices in Claw-Free Graphs. *Journal of Combinatorial Theory, Series B*, 28(3):284–304, 1980.

[155] Pranabendu Misra, Fahad Panolan, Ashutosh Rai, Saket Saurabh, and Roohani Sharma. Quick Separation in Chordal and Split Graphs. In *Proc. of MFCS 2020*, volume 170 of *LIPIcs*, pages 70:1–70:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[156] Bojan Mohar. Face Covers and the Genus Problem for Apex Graphs. *Journal of Combinatorial Theory, Series B*, 82:102–117, 2001.

[157] Burkhard Monien and Ivan Hal Sudborough. Min Cut is NP-Complete for Edge Weighted Treees. *Theoretical Computer Science*, 58:209–229, 1988.

[158] Andrea Munaro. Boundary Classes for Graph Problems Involving Non-local Properties. *Theoretical Computer Science*, 692:46–71, 2017.

[159] Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer, 2012. ISBN 978-3-642-27874-7.

[160] Jaroslav Nešetřil and Patrice Ossona de Mendez. On Nowhere Dense Graphs. *European Journal of Combinatorics*, 32:600–617, 2011.

[161] Sukanya Pandey and Erik Jan van Leeuwen. Planar Multiway Cut with Terminals on Few Faces. In *Proc. of SODA 2022*, pages 2032–2063. SIAM, 2022.

[162] Charis Papadopoulos and Spyridon Tzimas. Subset Feedback Vertex Set on Graphs of Bounded Independent Set Size. *Theoretical Computer Science*, 814:177–188, 2020.

[163] James K. Park and Cynthia A. Phillips. Finding Minimum-Quotient Cuts in Planar Graphs. In *Proc. of STOC 1993*, pages 766–775. ACM, 1993.

[164] Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network Sparsification for Steiner Problems on Planar and Bounded-Genus Graphs. *ACM Transactions in Algorithms*, 14(4): 53:1–53:73, 2018.

[165] Michał Pilipczuk, Erik Jan van Leeuwen, and Andreas Wiese. Quasi-Polynomial Time Approximation Schemes for Packing and Covering Problems in Planar Graphs. *Algorithmica*, 82(6):1703–1739, 2020.

[166] Svatopluk Poljak. A Note on Stable Sets and Colorings of Graphs. *Commentationes Mathematicae Universitatis Carolinae*, 15:307–309, 1974.

[167] John H. Reif. Minimum *s-t* Cut of a Planar Undirected Network in $O(n \log^2(n))$ time. *SIAM Journal of Computing*, 12(1):71–81, 1983.

[168] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar Tree-Width. *Journal of Combinatorial Theory, Series B*, 36:49–64, 1984.

[169] Neil Robertson and Paul D. Seymour. Graph Minors. V. Excluding a Planar Graph. *Journal of Combinatorial Theory, Series B*, 41:92–114, 1986.

[170] Neil Robertson and Paul D. Seymour. Graph Minors. XX. Wagner's Conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[171] Petra Scheffler. A Practical Linear Time Algorithm for Disjoint Paths in Graphs with Bounded Tree-Width. Preprint 396, Technische Universität Berlin, Institut für Mathematik, 1994.

[172] J. Sedlaáček. Some Properties of Interchange Graphs. In *Theory of Graphs and Its Applications*, pages 145–150. Academic Press, 1964.

[173] Detlef Seese. Linear Time Computable Problems and First-Order Descriptions. *Mathematical Structures in Computer Science*, 6:505–526, 1996.

[174] Paul D. Seymour and Robin Thomas. Call Routing and the Ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[175] Sebastian Siebertz. Nowhere Dense Graph Classes and Algorithmic Applications. A Tutorial at Highlights of Logic, Games and Automata, 2019. *arXiv*, abs/1909.06752, 2019.

[176] Ewald Speckenmeyer. *Untersuchungen zum Feedback Vertex Set Problem in Ungerichteten Graphen*. PhD thesis, Paderborn, 1983.

[177] Yuma Tamura, Takehiro Ito, and Xiao Zhou. Algorithms for the Independent Feedback Vertex Set Problem. *IEICE Transactions on Fundamental Electronics Communication and Computing Sciences*, 98-A(6):1179–1188, 2015.

[178] Robert Endre Tarjan. A Note on Finding the Bridges of a Graph. *Information Processing Letters*, 2(6):160–161, 1974.

[179] Robert Endre Tarjan and Anthony E. Trojanowski. Finding a Maximum Independent Set. *SIAM Journal of Computing*, 6(3):537–546, 1977.

[180] Jan Arne Telle and Andrzej Proskurowski. Algorithms for Vertex Partitioning Problems on Partial k-Trees. *SIAM Journal on Discrete Mathematics*, 10:529–550, 1997.

[181] Jan Arne Telle and Yngve Villanger. FPT Algorithms for Domination in Sparse Graphs and Beyond. *Theoretical Computer Science*, 770:62–68, 2019.

[182] Shuichi Ueno, Yoji Kajitani, and Shin'ya Gotoh. On The Nonseparating Independent Set Problem and Feedback Set Problem for Graphs with No Vertex Degree Exceeding Three. *Discrete Mathematics*, 72:355–360, 1988.

[183] Johan M. M. van Rooij and Hans L. Bodlaender. Design by Measure and Conquer, A Faster Exact Algorithm for Dominating Set. In *Proc. of STACS 2008*, volume 1 of *LIPIcs*, pages 657–668. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2008.

[184] Martin Vatshelle. *New Width Parameters of Graphs*. PhD thesis, University of Bergen, 2012.

[185] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001. ISBN 978-3-540-65367-7.

[186] Martin Škoviera and Peter Varša. NP-Completeness of Perfect Matching Index of Cubic Graphs. In *Proc. of STACS 2022*, volume 219 of *LIPIcs*, pages 56:1–56:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[187] Magnus Wahlström. Quasipolynomial Multicut-Mimicking Networks and Kernels for Multiway Cut Problems. *ACM Transactions in Algorithms*, 18(2):15:1–15:19, 2022.

[188] Daniel Weißauer. In Absence of Long Chordless Cycles, Large Tree-Width Becomes a Local Phenomenon. *Journal of Combinatorial Theory, Series B*, 139:342–352, 2019.

[189] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. ISBN 978-0-521-19527-0.

[190] Wenying Xi and Wensong Lin. On Maximum P_3-Packing in Claw-Free Subcubic Graphs. *Journal of Combinatorial Optimization*, 41:694–709, 2021.

[191] Mingyu Xiao. Simple and Improved Parameterized Algorithms for Multiterminal Cuts. *Theory of Computing Systems*, 46:723–736, 2009.

[192] Mihalis Yannakakis. Node and Edge Deletion NP-Complete problems. In *Proc. of STOC 1978*, pages 253–264. ACM, 1978.

[193] Mihalis Yannakakis and Fanica Gavril. Edge Dominating Sets in Graphs. *SIAM Journal on Applied Mathematics*, 38:364–372, 1980.

[194] Wei-Chang Yeh. A Simple Algorithm for the Planar Multiway Cut Problem. *Jounal of Algorithms*, 39(1):68–77, 2001.

[195] Dmitriy Zhuk. A Proof of the CSP Dichotomy Conjecture. *Journal of the ACM*, 67:30:1–30:78, 2020.

[196] Radoslaw Ziemann and Pawel Zylinski. Vertex-Edge Domination in Cubic Graphs. *Discrete Mathematics*, 343:112075, 2020.

Samenvatting

Bij het ontwerpen van algoritmen letten we op hoe efficiënt deze een probleem oplossen. We meten de efficiëntie van een algoritme in termen van het aantal stappen dat het algoritme moet uitvoeren op de meest ongunstige invoer. In de race om de efficiëntie van algoritmen te verbeteren, is het nuttig om te weten hoe ver men deze kan verbeteren. Daartoe analyseren algoritmedeskundigen graag de complexiteit van elk probleem door een ondergrens te geven aan het aantal stappen dat elk algoritme moet uitvoeren om het op te lossen. Klassiek worden de problemen gecategoriseerd als diegenen die in polynomiale tijd kunnen worden opgelost versus diegenen die dat niet kunnen, indien $\mathsf{P} \neq \mathsf{NP}$.

In dit proefschrift bestuderen we problemen die als uiterst complex worden beschouwd, dat wil zeggen: elk algoritme moet waarschijnlijk een exponentieel aantal stappen uitvoeren om deze problemen op een willekeurige invoer op te lossen. Deze problemen worden NP-moeilijk genoemd. Onze belangrijkste focus is het identificeren van klassen van invoer waarbij we efficiënte algoritmen kunnen ontwerpen voor NP-moeilijke problemen. Hiertoe hanteren we een tweeledige benadering. Ten eerste identificeren we een parameter van het probleem naast de grootte van de invoer. De looptijd van ons algoritme wordt dan gemeten als een functie van deze parameter, naast de grootte van de probleeminvoer. Een efficiënt algoritme in dit paradigma is er een die in polynomiale tijd draait als de waarde van deze parameter constant is. Ten tweede lossen we het probleem op voor beperkte klassen van invoer. Het doel hierbij is om alle klassen van invoer te vinden waarvoor men het probleem efficiënt kan oplossen, dat wil zeggen, de doenbaarheid van het probleem identificeren.

We richten ons op problemen die zich voordoen in grafen. Een graaf is een netwerk van punten en lijnen. Verschillende praktische problemen kunnen worden gemodelleerd als problemen op grafen, bijvoorbeeld het vinden van het kortste pad tussen twee punten, het vinden van de kortste route die alle steden in een provincie bezoekt, het vinden van de maximale verkeersstroom door een wegennetwerk, enzovoort. We bestuderen enkele bekende graafproblemen, beperkt tot de klasse van monotone grafen. Een graafklasse wordt monotone genoemd als voor elke graaf in de klasse de graaf die wordt gevormd na het

verwijderen van enkele van zijn punten en/of lijnen een graaf oplevert die ook lid is van deze klasse.

In de eerste helft van dit proefschrift richten we ons op het Edge (Node) Multiway Cut probleem. In dit probleem krijgen we als invoer een ongerichte graaf, een deelverzameling van zijn knopen die terminals worden genoemd en een geheel getal s. Het doel is te beslissen of er een deelverzameling van kanten (knopen) bestaat met een kardinaliteit van hoogstens s die, wanneer verwijderd uit de graaf, de terminals paargewijs niet verbonden maakt. Dit probleem is een natuurlijke generalisatie van het Minimum (s,t)-cut probleem waarbij het aantal terminals precies twee is. We onderzoeken de complexiteit van dit probleem op planaire grafen met een terminalfacetbedekkingsgetal k en planaire grafen met een maximale graad van hoogstens 3. In het eerste geval tonen we aan dat Edge Multiway Cut kan worden opgelost in $2^{O(k^2 \log k)} n^{O(\sqrt{k})}$ tijd. Onder de aanname van ETH is de looptijd van ons algoritme optimaal. Op planaire grafen van maximale graad 3 tonen we aan dat Edge (Node) Multiway Cut NP-volledig is. Ons resultaat verbetert de eerder bekende grens van 11. Bijgevolg bewijzen we dat de grens van polynomiale oplosbaarheid voor Edge (Node) Multiway Cut in termen van de maximale graad van de invoergraaf $\Delta = 2$ is, daarbuiten is het probleem NP-moeilijk.

De tweede helft van dit proefschrift is gewijd aan het ontwikkelen van een complexiteitskader voor monotone graafklassen. We presenteren een meta-classificatiestelling over de klasse van $\mathcal{H}$-deelgraafvrije grafen, waarbij $\mathcal{H}$ een eindige verzameling grafen is. Ons resultaat stelt dat elk probleem dat voldoet aan drie specifieke eigenschappen volledig kan worden geclassificeerd op H-deelgraafvrije grafen. Deze drie eigenschappen zijn de volgende: het probleem kan worden opgelost in polynomiale tijd op de klasse van grafen met begrensde boombreedte; het probleem is NP-moeilijk op subcubische grafen; en er bestaat een $\ell \geq 1$ waarvoor het probleem NP-moeilijk blijft op ℓ-onderverdelingen van subcubische grafen. In het bijzonder kan een dergelijk probleem in polynomiale tijd worden opgelost wanneer $\mathcal{H}$ een graaf bevat die een niet-lege disjuncte vereniging is van paden en onderverdeelde klauwen. Als dit niet het geval is, is het probleem NP-moeilijk. We hebben aangetoond dat een groot aantal natuurlijk voorkomende problemen op grafen binnen dit classificatiekader past.

Bovendien hebben we gestreefd naar het classificeren van problemen die niet voldoen aan een van de drie genoemde eigenschappen, namelijk: de NP-moeilijkheid op de klasse van subcubische grafen. Hierbij toonden we eerst aan dat Independent Feedback Vertex Set in polynomiale tijd kan worden opgelost op subcubische grafen. We hebben voortgang geboekt bij het classificeren van de complexiteit van dergelijke problemen op $\mathcal{H}$-deelgraafvrije grafen. Wanneer $\mathcal{H} = \{H\}$, waarbij H een verbonden graaf is, hebben we

de kloof tussen computationeel makkelijke en moeilijke gevallen voor het probleem (Independent) Feedback Vertex Set verkleind, met als enige open geval $H = S_{1,p,q,r}$, dat wil zeggen, een ster met vier bladeren, waarvan er drie respectievelijk $p-1, q-1$ en $r-1$ keer zijn onderverdeeld. We hebben ook classificaties verkregen voor problemen zoals Connected Vertex Cover, Coloring en Matching Cut, die een vergelijkbaar gedrag vertonen als Feedback Vertex Set met betrekking tot de voorwaarden van ons classificatiekader.

We sluiten dit proefschrift af met de hoop dat met ons diepere begrip van de structuur van verschillende klassen van grafen, in het bijzonder $\mathcal{H}$-deelgraafvrije grafen, we de grenzen van de doenbaarheid van belangrijke NP-moeilijke problemen verder kunnen verleggen.

Curriculum Vitae

Personal Data

Name	Sukanya Pandey
Date of Birth	29.08.1994
Place of Birth	Patna, Bihar, India.

Education

2019 – present	**Utrecht University** PhD candidate Advisor: Dr. Erik Jan van Leeuwen Promotor: Prof. Dr. Hans Bodlaender
2018–2019	**The Institue of Mathematical Sciences, Chennai, India** Master thesis: Role coloring hereditary classes of graphs Advisor: Prof. Dr. Venkatesh Raman
2014 – 2019	**Indian Institute of Science Education and Research, Pune** Integrated Bachelor and Master of Science (Mathematics)

Publications

1. Sukanya Pandey and Erik Jan van Leeuwen. Planar Multiway Cut with Terminals on Few Faces. In *Proc. of SODA 2022*, pages 2032–2063. SIAM, 2022.

2. Matthew Johnson, Barnanby Martin, Sukanya Pandey, Daniël Paulusma, Siani Smith, Erik Jan van Leeuwen. Edge Multiway Cut and Node Multiway Cut are Hard for Planar Subcubic Graphs. In *Proc. of SWAT 2024*, volume 294 of *LIPIcs*, pages 29:1–29:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

3. Matthew Johnson, Barnanby Martin, Jelle J. Oostveen, Sukanya Pandey, Daniël Paulusma, Siani Smith, Erik Jan van Leeuwen. Complexity Frame-

work for Forbidden Subgraphs I: The Framework. *arXiv*, abs/2211.12887, 2022.

4. Matthew Johnson, Barnanby Martin, Sukanya Pandey, Daniël Paulusma, Siani Smith, Erik Jan van Leeuwen. Complexity Framework for Forbidden Subgraphs III: When Problems are Tractable on Subcubic Graphs. In *Proc. of MFCS 2023*, volume 272 of *LIPIcs*, pages 29:1–29:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

5. Hans L. Bodlaender, Matthew Johnson, Barnanby Martin, Jelle J. Oostveen, Sukanya Pandey, Daniël Paulusma, Siani Smith, Erik Jan van Leeuwen. Complexity Framework for Forbidden Subgraphs IV: The Steiner Forest Problem. In *Proc. of IWOCA 2024*, volume 14764 of *LNCS*, pages 206–217. Springer 2024.

6. Hans L. Bodlaender, Isja Mannens, Jelle J. Oostveen, Sukanya Pandey, and Erik Jan van Leeuwen. The Parameterized Complexity of Integer Multicommodity Flow. In *Proc. of IPEC 2023*, volume 285 of *LIPIcs*, pages 6:1–6:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. **(Best Paper)**

7. Hans L. Bodlaender, Édouard Bonner, Lars Jaffke, Dusan Knop, Paloma T. Lima, Martin Milanic, Sebastian Ordyniak, Sukanya Pandey, and Ondrej Suchý. Treewidth is NP-Complete on Cubic Graphs. In *Proc. of IPEC 2023*, volume 285 of *LIPIcs*, pages 7:1–7:13, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

8. Nick Bretell, Jelle J. Oostveen, Sukanya Pandey, Daniël Paulusma, and Erik Jan van Leeuwen. Computing Subset Vertex Covers in H-free Graphs. In *Proc. of FCT 2023*, volume 14292 of *LNCS*, pages 88–102, Springer, 2023.

9. Sukanya Pandey and Vibha Sahlot. Role Coloring Bipartite Graphs. *Discrete Applied Mathematics*, 322:276–285, 2022.

10. Sukanya Pandey, Venkatesh Raman, and Vibha Sahlot. Parameterizing Role Coloring on Forests. In *Proc. of SOFSEM 2021*, volume 12607 of *LNCS*, pages 308–321. Springer, 2021.