Uncertainty in Discrete Optimization:
Connectivity and Covering

Manuel Streicher

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Vom Fachbereich Mathematik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat.)
genehmigte Dissertation

Gutachter:	Prof. Dr. Sven Oliver Krumke
	Prof. Dr. Rainer Schrader
Datum der Disputation:	04.12.2020

D 386

Zusammenfassung

Das Betrachten unsicherer Strukturen oder Daten bekommt mehr und mehr Aufmerksamkeit in der diskreten Optimierung. Diese Arbeit behandelt zwei unterschiedliche Problemstrukturen innerhalb der diskreten Optimierung: Zusammenhang und Überdeckungen.

In Anwendungsproblemen mit unsicheren Strukturen in Netzwerken ist es oft interessant herauszufinden, wie viele Knoten oder Kanten *ausfallen* können, sodass das Netzwerk zusammenhängend bleibt. Zusammenhang ist ein weites, gut untersuchtes Feld innerhalb der Graphentheorie. Eines der wichtigsten Resultate diesbezüglich ist der Satz von Menger. Dieser sagt aus, dass die minimale Anzahl an Knoten, die man benötigt, um zwei nicht adjazente Knoten zu trennen, gleich der maximalen Anzahl intern knotendisjunkter Wege zwischen diesen Knoten ist. In dieser Arbeit untersuchen wir unterschiedliche Formen gemischten Zusammenhangs. In diesem werden Knoten und Kanten gleichzeitig vom Graphen entfernt. Die Beineke Harary Behauptung sagt aus, dass für je zwei unterschiedliche Knoten, die mit k Knoten und l Kanten getrennt werden können, aber nicht mit $k-1$ Knoten und l Kanten oder k Knoten und $l-1$ Kanten, $k+l$ kantendisjunkte Wege existieren, von denen $k+1$ knotendisjunkt sind. Im Gegensatz zum Satz von Menger ist die Existenz der Pfade in diesem Fall nicht hinreichend für die Aussage über den Zusammenhang. Unser Hauptbeitrag is der Beweis der Beineke Harary Behauptung für den Fall $l = 2$. Ein weiterer Beitrag dieser Arbeit ist die kanonische Zerlegung von Graphen entlang von Separatoren, die nur aus einem Knoten und einer Kante bestehen.

Bezüglich Überdeckungsproblemen betrachten wir in dieser Arbeit ein Dominationsproblem, in welchem wir annehmen, dass die genaue Struktur des zu dominierenden Graphen nicht bekannt ist. Wir kennen nur einen Obergraphen und wissen, dass der zu dominierende Graph ein Spannbaum ist. Wir beweisen eine äquivalente Charakterisierung des Problems, welche keine Spannbäume benutzt und verwenden diese, um einen Lösungsalgorithmus zu entwickeln, der auf dem Block-Schnitt Baum eines Graphen operiert. Diesen Algorithmus nutzen wir, um polynomielle Lösbarkeit des Problems auf unterschiedlichen Graphenklassen zu beweisen.

Wir betrachten auch unterschiedliche Überdeckungs- und Einrichtungsplatzierungsprobleme. In diesen Problemen sind Mengen von Standorten und Regionen gegeben, wobei jede Region eine assoziierte Anzahl an Kunden hat. Wir suchen nun eine Verteilung von Lieferanten in die Standorte, sodass jeder Kunde von einem Lieferanten bedient werden kann. Der auffallende Unterschied zu anderen Problemen in diesem Forschungsfeld ist hier, dass wir annehmen, dass jeder Lieferant nur eine konstante Anzahl Kunden bedienen kann. Wir klassifizieren die Komplexität dreier solcher Probleme, die sich in der Zuteilung von Kunden zu Lieferanten unterscheiden, und entwickeln entsprechende Lösungsmethoden.

Abstract

Dealing with uncertain structures or data has lately been getting much attention in discrete optimization. This thesis addresses two different areas in discrete optimization: Connectivity and covering.

When discussing uncertain structures in networks it is often of interest to determine how many vertices or edges may *fail* in order for the network to stay connected. Connectivity is a broad, well studied topic in graph theory. One of the most important results in this area is Menger's Theorem which states that the minimum number of vertices needed to separate two non-adjacent vertices equals the maximum number of internally vertex-disjoint paths between these vertices. Here, we discuss mixed forms of connectivity in which both vertices and edges are removed from a graph at the same time. The Beineke Harary Conjecture states that for any two distinct vertices that can be separated with k vertices and l edges but not with $k - 1$ vertices and l edges or k vertices and $l - 1$ edges there exist $k + l$ edge-disjoint paths between them of which $k + 1$ are internally vertex-disjoint. In contrast to Menger's Theorem, the existence of the paths is not sufficient for the connectivity statement to hold. Our main contribution is the proof of the Beineke Harary Conjecture for the case that l equals 2. Another contribution regarding mixed connectivity is the canonical decomposition of graphs along separators consisting of a single edge and a single vertex.

Concerning covering, in this thesis, we discuss a domination problem in which we do not assume to know the exact structure of the given graph. We are merely given a supergraph and know that the graph we aim to dominate is a spanning tree of that graph. We give an equivalent characterization of the problem that does not use spanning trees. Further, we describe an algorithm that operates on the tree structure of the blocks of a graph. We exploit this to prove polynomial time solvability on special graph classes.

We also consider different problems from the area of facility location and covering. We regard problems in which we are given sets of locations and regions, where each region has an assigned number of clients. We are now looking for an allocation of suppliers into the locations, such that each client is served by some supplier. The notable difference to other covering problems is that we assume that each supplier may only serve a fixed number of clients which is not part of the input. We discuss the complexity and solution approaches of three such problems which vary in the way the clients are assigned to the suppliers.

Acknowledgments

First and foremost I thank my supervisor Prof. Dr. Sven O. Krumke, who made my research at the optimization research group at TU Kaiserslautern possible. His ideas and oversight on topics related to this thesis significantly added to its quality. I also thank the co-referee Prof. Dr. Rainer Schrader for agreeing to read and evaluate this thesis. I thank Irene Heinrich for endless, fruitful discussions on graph theory. I am thankful to Eva Schmidt for working alongside within the project *HealthFaCT* and therefore sharing ups and downs of applying math in practical applications.

All colleagues who co-authored some of the publications at the base of this thesis also deserve a special thanks: Christina Büsing, Martin Comis, Till Heller, Irene Heinrich, Sebastian Johann, Sven O. Krumke, and Eva Schmidt: Thank you for developing some of the ideas that ultimately led to the results in this thesis.

I thank all proofreaders of this thesis for their mindful reading. Your valued input greatly increased the quality of this thesis, Oliver Bachtler, Irene Heinrich, Michael Holzhauser, Sebastian Johann, Sven O. Krumke, and Eva Schmidt.

To all of the optimization research group at TU Kaiserslautern I am very thankful for making my life as a doctoral student a very pleasant experience.

Last but not least, I thank my family for supporting me throughout my academic life.

The research for this thesis was partially funded by the German Federal Ministry of Education and Research within the project *HealthFaCT – Health: Facility location, Covering and Transport* (grant number 05M16UKC).

Contents

List of Figures

Chapter 1.

Introduction

It has become common practice in optimization to consider problems where some data is assumed not to be known exactly in advance. In many applications it is the case that only rough bounds or structures of the actual input data is known. This thesis deals with two different settings of incorporating this uncertainty into problems. It is split into two parts.

In the first part of this thesis, we consider different structural graph theoretic problems. We regard a mixed form of connectivity, in which we are allowed to remove vertices and edges in a graph. Further, we use a decomposition along mixed separators to characterize the class of Eulerian graphs that decompose into a unique number of cycles. Finally, we study a version of the dominating set problem in which the graph to be covered is not exactly known and we seek to dominate all spanning trees of a graph.

The second part considers different problems from the area of facility location and covering problems. Here we assume that the number of clients to be served is not exactly known in advance and use approaches from *Robust Optimization* to cope with the uncertain data. In one of the problems the clients are assigned to the suppliers. In the other problems the clients have freedom of choice of the supplier to some extent. The problems studied in this part are motivated by the research projects *HealthFaCT* and *ONE PLAN* in which the optimization research group of the TU Kaiserslautern seeks to optimize location structures in emergency rescue services. As the focus of this thesis is on theoretic considerations we refrain from giving details on the application in this thesis. Some details on the application can be found in our publications [KSS19; Büs+20].

Part I: Structural Uncertainty in Graphs: Connectivity and Domination

When considering graphs or networks whose structure is not known exactly in practical applications it is often of interest to know how many vertices or edges may be deleted such that the graph or a pair of distinct vertices remain connected. For example, when planning a road network it is crucial to still be able to reach all places when a road is blocked. Connectivity is a means of measuring this robustness against removal of vertices and edges. One of the most famous results concerning connectivity is Menger's Theorem, cf. [Wes01]. For two distinct non-adjacent vertices s, t it states that the minimum number of vertices (edges) necessary to disconnect s and t equals the maximum number of internally vertex-disjoint (edge-disjoint) s-t paths between the two vertices. There are many known variants of Menger's Theorem, cf. [Wes01]. There are also many results concerning fixed size separators in graphs. For example it is well known that any graph uniquely decomposes into its maximal subgraphs not containing cut vertices, the *blocks* of the graph, cf. [Wes01]. These blocks induce a unique tree structure that may be computed in linear time [HT73]. The resulting structure is called the block-cutpoint tree of a graph. Similar results have been obtained by Hopcroft and Tarjan for separators consisting of two vertices, cf. [HT72].

Another well studied field in graph theory is domination. At the foundation of this research field is the dominating set problem, in which we aim to find a a minimum size subset of the vertices such that each vertex is either contained in the set or adjacent to some vertex in the set, cf. [HHS98]. There are various variants of Dominating Set in the literature. For example in Disjunctive Domination, additionally, no two vertices in the searched set may be adjacent, cf. [GH13]. In contrast, in Total Domination each vertex in the searched set has to be adjacent to another vertex in the set, cf. [Hen09]. Dominating Set is well known to be NP-complete, cf. [GJ79], and so are most of the variants of Dominating Set. For a more detailed introduction and overview on domination in graphs we refer to the textbook [HHS98].

Contributions of Part I In this thesis we consider mixed forms of connectivity in which deletion of vertices and edges is allowed at the same time. Mixed connectivity has not been getting the same attention in the literature as its pure counterparts. The conjecture of Beineke and Harary, cf. [BH67], states:

> *If two distinct vertices s, t can be disconnected with k vertices and l edges but not with $k-1$ vertices and l edges or k vertices and $l-1$ edges, then there exist $k+l$ edge-disjoint s-t paths of which $k+1$ are internally vertex-disjoint.*

It can be considered a mixed version of Menger's Theorem. The most prominent difference between the two statements is that in the Beineke Harary Conjecture the existence of the paths is not claimed to be sufficient for the connectivity statement to hold and in fact, it is not. One of the main contributions of this thesis is proving the conjecture for $l = 2$ and arbitrary k. We also exploit this result in order to prove the conjecture for arbitrary l and k on graphs with treewidth at most three.

In analogy to blocks and triconnected components we give a canonical decomposition along separators consisting of a single vertex and a single edge. We prove this decomposition to be obtainable in linear time. As an application of this we give a characterization of the class of Eulerian graphs that decompose into a unique number of cycles: It is exactly the class of Eulerian graphs in which no two edge-disjoint cycles share three or more vertices.

Finally, we regard a version of Dominating Set in which the structure of the graph is not exactly known in advance. In Simultaneous Domination of Spanning Trees, we aim to be dominating in every spanning tree of a graph. We prove equivalence of the problem to Vertex Cover on 2-connected graphs and describe a dynamic program operating on the block-cutpoint tree of a graph that solves Min-Simultaneous Domination of Spanning Trees using an algorithm for Min Vertex Cover on certain subgraphs. We use this algorithm to prove polynomial time solvability for different graph classes, such as bipartite graphs, chordal graphs or graphs with bounded treewidth. As the most surprising result concerning this topic, we prove that Simultaneous Domination of Spanning Trees is NP-hard on perfect graphs. This is in contrast to Vertex Cover being polynomial time solvable on perfect graphs, cf. [Sch03].

Outline of Part I This part of the thesis is split into four chapters. In Chapter 3 we discuss a mixed form of connectivity, called connectivity pairs. The main focus of this chapter is the conjecture of Beineke and Harary. Mixed separators consisting of a single vertex and a single edge are discussed in Chapter 4. In particular, we consider a decomposition of graphs along those separators and prove their uniqueness. In Chapter 5 we characterize the class of Eulerian graphs decomposing into a unique number of cycles as an application of the separators discussed in Chapter 4. Finally, we analyze the complexity and give solution approaches for a variant of Dominating Set in which each spanning tree of a graph is required to be dominated in Chapter 6.

For the basic knowledge and notation needed for Part I of this thesis, we refer to Chapter 2. In particular, Section 2.5 of the chapter provides the notation concerning graph theory which is relevant for this thesis.

Part II: Uncertainty in the Demand: Robust Approaches to Covering and Facility Location Problems

In the second part of this thesis we examine uncertainty in the demand of certain *covering* and *facility location* problems. When we say that *there is uncertainty in the demand* of a problem, we mean that instead of being given a fixed demand vector as input for the problem, we rather get a set of demand vectors, subsequently called the *uncertainty set* of the instance. Each demand vector in the uncertainty set represents one possible *scenario*. In this thesis we only consider finite uncertainty sets. In the literature there are different approaches of dealing with uncertainty in input data. The most important in our field of study are *robust* and *stochastic* optimization. In stochastic optimization we are often given a probability distribution on the uncertain data and aim to optimize an objective function in expectation or intend conditions to be fulfilled with high probability. In robust optimization on the other hand, the uncertain data typically comes without probabilities. We aim at hedging against the worst case or strive for solutions that fulfill certain conditions in all possible cases. In this thesis, we focus on robust optimization and therefore refrain from giving any further details on stochastic optimization here. Instead, we refer to [BL11] for a general overview on stochastic optimization.

The work on robust optimization was started by Soyster in the early 1970s [Soy73] and became more popular in the 1990s with publications from Ben-Tal and Nemirovski [BN98; BN99; BN00] and El Ghaoui et al. [GL97; GOL98]. Since then there has been a very large number of publications in the field of robust optimization. A lot of research has been done on *discrete uncertainty* where the uncertainty set is given as a collection of vectors. As in practical applications an exact collection of *scenarios* (elements of an uncertainty set) is often not known, it is also common to only give lower and upper bounds for each uncertain component, the so called *interval uncertainty*. For a recent overview on discrete and interval uncertainty we refer to [KZ16]. Interval uncertainty is, in many applications, still very conservative, as it allows for the worst case to happen in every uncertain component. Bertsimas and Sim therefore proposed the use of *budgeted uncertainty*, in which in addition to the lower and upper bounds the number of components that may vary from a given nominal value is bounded, cf. [BS03; BS04]. In this thesis we use the notion of *budgeted uncertainty* slightly differently. Instead of bounding the number of components that may vary from their nominal value, we bound the total deviation.

For an extensive introduction to robust optimization we refer to [BGN09] and for a recent overview we refer to [GMT14].

For surveys of covering problems in facility location not including uncertain data we refer to [OD98; Far+12]. In the literature there are various studies on robust covering and facility location problems. The earliest go back to the 1970s with publications from, e.g., Cooper, cf. [Coo78]. Dhamdhere et al. [Dha+05] introduce demand-robust covering problems and provide approximation algorithms. In [PA13] the set cover problem with uncertainty in the cost coefficients is considered and exact algorithms for computing the min-max regret solution are presented. For further reading on covering and facility location problems under uncertainty we refer to [Lut+17; Sny06].

Contributions of Part II In Part II of this thesis we introduce three covering, respectively facility location problems: q-Multiset Multicover, q-freeClient, and q-orderedClient. In all of the three problems we are given a set of *locations* and a set of *regions*, where each region has a number of *clients* and a subset of the locations, its *consideration set*. In the optimization versions of the problems we aim at placing a minimum number of *suppliers* in the locations such that all clients are served, where each supplier may serve up to q clients. The difference of the three problems lies in the assignment of clients to suppliers. In q-Multiset Multicover we may assign a client to any location in its consideration set. In q-freeClient we have to ensure feasibility of the solution for any assignment of clients to locations in their consideration set. Finally, in q-orderedClient the consideration set of each client is assumed to have some ordering and the client is assigned to the first location which contains at least one supplier. In the robust versions of the problems we assume that the number of clients in each region is not known exactly, but taken from some set of possible client allocations, the uncertainty set.

We establish various complexity results for all problems, their robust versions and subproblems arising in the solution process. In the setting in which no restrictions are made on the uncertain data all regarded problems are proved to be NP-hard, even when q is fixed to 1. In the non-robust case, in which the demand is known exactly, we prove q-Multiset Multicover to be NP-complete for any fixed $q \in \mathbb{N}$ with $q \geq 3$ and q-orderedClient to be NP-complete for any fixed $q \in \mathbb{N}$ with $q \geq 2$. q-freeClient, on the other hand, is linear time solvable for any $q \in \mathbb{N}$. Introducing uncertainty in the number of clients increases complexity of the problems. We prove that q-Multiset Multicover is NP-complete for $q \in \mathbb{N}$ and restricted to uncertainty sets containing k elements if $q \cdot k \geq 3$. For q-orderedClient we even show that it is NP-complete if $\max\{q, k\} \geq 2$. Loosely speaking this shows that 1-orderedClient is NP-hard as soon as any uncertainty in the number of clients is introduced. q-freeClient is again an exception: We show linear time solvability of the problem for many common restrictions on the uncertainty set, such as discrete uncertainty, interval uncertainty and budgeted uncertainty.

Further, we describe mixed integer programming models for all regarded problems and, for q-Multiset Multicover, describe a solution approach based on constraint generations. The complexity of the arising separation problem is also studied. We prove NP-hardness of the separation problem for budgeted uncertainty and polynomial time solvability for discrete and interval uncertainty.

Outline of Part II Part II of this thesis consists of two chapters. In Chapter 7 we consider q-Multiset Multicover. We classify the complexity of the non-robust problem. We then turn to the robust version of the problem and also consider its complexity for various cases with very few restrictions on the regarded uncertainty sets. Before we turn to the complexity of the problem with more restrictions on the uncertainty sets we describe a general solution approach.

In Chapter 8 we study the two problems q-freeClient and q-orderedClient. Again, we classify the complexity of the non-robust versions at the start of the chapter. Afterwards we turn to the complexity of the robust versions and the impact of common restrictions on the uncertainty sets.

For the basic knowledge and notation needed for Part II of this thesis, we refer to Chapter 2. In particular Section 2.4 of that very chapter might be worth looking into before starting this part, as some terminology and assumptions slightly differ from the standards in robust optimization.

A Note on Publications

Large parts of this thesis have been previously published. We briefly state all those publications here and name the corresponding co-authors. In the respective chapters we give more details on the relationship of the publication and the chapter in this thesis if necessary.

Chapter 3 is published in [JKS19] and joint work with Sebastian Johann and Sven O. Krumke. A version similar to [JKS19] has been submitted to Graphs and Combinatorics (Springer) and is currently under review. Chapter 4 is strongly based on [Hei+20a], which is joint work with Irene Heinrich, Till Heller, and Eva Schmidt. An extended version of [Hei+20a] is available in [Hei+20b]. The results of Chapter 5 have originally been published in [HS19] which is joint work with Irene Heinrich. The results of Chapter 4 made it possible to make some of the statements and proofs more elegant. Some of these altered statements have also been published in [Hei+20a] and are therefore joint work with Irene Heinrich, Till Heller, and Eva Schmidt. Large parts of the result in Chapter 6 are published in [JKS18]. A version similar to [JKS18] has been submitted to Electronic Journal of Graph Theory and Application and is currently under review.

Part of the results of Chapter 7 have been published in [KSS19], which is joint work with Sven O. Krumke and Eva Schmidt. Finally, Chapter 8 is based on results from [Büs+20], which is joint work with Christina Büsing, Martin Comis, and Eva Schmidt. A version similar to [Büs+20] has also been submitted to the European Journal of Operational Research (Elsevier) and is currently under review.

Chapter 2.

Preliminaries

In this chapter we recall basic definitions, make assumptions and define notations that will be used throughout this thesis. We focus on things, that deviate from standards in terminology in the field of graph theory and optimization.

2.1. Fundamentals

The number 0 is a natural number, i.e. $0 \in \mathbb{N}$. Whenever we consider the set $\mathbb{N} \setminus \{0\}$ we use the notation $\mathbb{N}_{>0}$. For $\mathbb{A} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ we write $\mathbb{A}_{\geq 0} = \{x \in \mathbb{K} \colon x \geq 0\}$ and $\mathbb{A}_{>0} = \{x \in \mathbb{A} \colon x > 0\}$.

For two sets A, B we use the notation $A \subseteq B$ to indicate that each element of A is contained in B. If we want to emphasize that additionally $A \neq B$ we use the notation $A \subsetneq B$. Further we write $A \mathbin{\dot\cup} B$ instead of $A \cup B$ to emphasize that $A \cap B = \emptyset$. We call a collection of sets $\{A_1, \ldots, A_k\}$ a *partition* of a set A if

$$\dot{\bigcup_{i=1}^{k}} A_i = A.$$

For a finite set A with k elements, unless stated otherwise, we implicitly assume that the elements have unique indices from 1 to k, i.e. $A = \{a_1, \ldots, a_k\}$. When necessary, we identify $a \in A$ with its index $i \in \{1, \ldots, k\}$. In particular we use the elements of finite sets as indices, when applicable.

Let I be a finite set, $I' \subseteq I$ and $x \in \mathbb{R}^{|I|}$. We write

$$x(I') \coloneqq \sum_{i \in I'} x_i.$$

Note that this is an example of the identification of elements of I with their indices in I.

2.2. Problems and Complexity Classes

For the definition of decision problems, basic complexity classes, and polynomial time reductions we refer to [GJ79]. We call a decision problem $\mathcal{P}$ NP-hard if there is there is a polynomial time reduction from $\mathcal{P}'$ to $\mathcal{P}$ for all $\mathcal{P}' \in$ NP. A decision problem is called NP-complete if it is NP-hard and also contained in NP. For an overview on decision problems regarded in this thesis, see Appendix 9.

Let $\mathcal{P}$ be a decision problem and $\mathcal{I}$ be an instance of the problem. Assume the question of $\mathcal{P}$ is to decide if some value $x(\mathcal{I})$ is less or equal (larger or equal) to some number $B \in \mathbb{R}$. We denote by $\textsc{Min}$-$\mathcal{P}$ ($\textsc{Max}$-$\mathcal{P}$) the corresponding *optimization problem* in which we seek to minimize (maximize) the value $x(\mathcal{I})$.

For an optimization problem $\textsc{Min}$-$\mathcal{P}$ we call an algorithm $\textsc{alg}$ an $f(\mathcal{I})$-*approximation* if for all instances $\mathcal{I}$ of $\textsc{Min}$-$\mathcal{P}$ it holds true that

$$\textsc{alg}(\mathcal{I}) \leq f(\mathcal{I}) \cdot \textsc{opt}$$

and it can be implemented to run in polynomial time. For a more detailed definition of approximation algorithms we refer to [Aus+12].

2.3. Linear and Integer Programming

In this thesis we only use standard notation and results from the field of linear and integer programming. We therefore refrain from giving any details regarding this field and refer to [GLS88] for an introduction to the topic.

2.4. Robust Optimization

In this section we recall basic definitions and notations in the field of robust optimization used throughout this thesis. As this thesis only discusses a small sub-area of robust optimization, we refrain from giving general definitions. For a general overview over the field of robust optimization we refer to [BGN09].

Let $\mathcal{P}$ be a decision problem containing some vector $d \in U \subseteq \mathbb{R}^m$ for some $m \in \mathbb{N}_{>0}$ as input parameter, such that an instance of $\mathcal{P}$ is a $\textsc{yes}$-Instance if and only if a condition $C_{\mathcal{P}}(d)$ defined by $\mathcal{P}$ evaluates to $\textsc{true}$. The according *robust decision problem* has the same input parameters, except that it contains a collection of vectors $\mathcal{U} \subseteq U$ instead of d. A given instance is a $\textsc{yes}$-Instance if and only if the condition $C_{\mathcal{P}}(\xi)$ evaluates to $\textsc{true}$ for all $\xi \in \mathcal{U}$. A *robust optimization problem* is the optimization version of a robust decision problem as defined in Section 2.2.

We call the set $\mathcal{U}$ the *uncertainty set* of the instance and an element $\xi \in \mathcal{U}$ a *scenario*. All uncertainty sets regarded in this thesis are of the form $\mathcal{U} \subseteq \mathbb{N}^m$ for some $m \in \mathbb{N}$ and assumed to be finite and non-empty.

When regarding the complexity of robust decision problems, the encoding length of the uncertainty sets is of importance. Throughout this thesis we assume that for an uncertainty set $\mathcal{U} \subseteq \mathbb{N}^m$ for some $m \in \mathbb{N}_{>0}$ the encoding length is in $\Omega(m)$. In the following we denote by $\langle x \rangle$ the encoding length of an input parameter x. We say that an uncertainty set in a problem instance is *polynomial time enumerable* if we may enumerate all elements of $\mathcal{U}$ in polynomial time in its encoding length.

Throughout this thesis we regard robust decision and optimization problems, where we restrict the uncertainty sets of the instances. We define all of the used restrictions here. Let $\mathcal{P}$ be a robust decision or optimization problem with uncertainty set $\mathcal{U} \subseteq \mathbb{N}^m$.

Discrete Uncertainty We call $\mathcal{U}$ *discrete* if its scenarios are given explicitly. We call the problem $\mathcal{P}$ with instances restricted to discrete uncertainty sets $\mathcal{P}$ with *discrete uncertainty*. We assume that a discrete uncertainty set $\mathcal{U} = \left\{\xi^1, \dots, \xi^k\right\}$ has encoding length $\sum_{i=1}^{k} \langle \xi^i \rangle$, where each scenario is encoded as a vector of integers. Note that discrete uncertainty sets are polynomial time enumerable.

Polyhedral Uncertainty We call $\mathcal{U}$ *polyhedral* if there exists a matrix $A \in \mathbb{R}^{n \times m}$ and a vector $b \in \mathbb{R}^n$ for some $n \in \mathbb{N}_{>0}$ such that

$$\mathcal{U} = \{\xi \in \mathbb{N}^m \colon A\xi \leq b\}.$$

In this case the polyhedron $P = \{x \in \mathbb{R}^m \colon Ax \leq b\}$ is called the *underlying polyhedron*. We call the problem $\mathcal{P}$ with instances restricted to polyhedral uncertainty sets $\mathcal{P}$ with *polyhedral uncertainty*. We assume that polyhedral uncertainty are encoded as the matrix A and the vector b. Its encoding length is therefore $\langle A \rangle + \langle b \rangle$.

Interval Uncertainty We call the problem $\mathcal{P}$ restricted to instances with polyhedral uncertainty sets of the form $\mathcal{U} = \{\xi \in \mathbb{N}^m \colon a \leq \xi \leq b\}$ for some $a, b \in \mathbb{N}^m$, $\mathcal{P}$ with *interval uncertainty*. The encoding length of this kind of uncertainty sets is assumed to be $\langle a \rangle + \langle b \rangle$.

Budgeted Uncertainty We call $\mathcal{U}$ *budgeted* if there exists $\Gamma \in \mathbb{N}$ and $a, b \in \mathbb{N}^m$ such that $\mathcal{U} = \left\{\xi \in \mathbb{N}^m \colon a \leq \xi \leq b, \sum_{i=1}^{m} |\xi_i| \leq \Gamma\right\}$. In addition for this thesis we assume that $a(J) \leq \Gamma \leq b(J)$ to ensure that $\mathcal{U} \neq \emptyset$. We call the problem $\mathcal{P}$ restricted to instances with budgeted uncertainty sets $\mathcal{P}$ with *budgeted uncertainty*. We assume that the encoding length of budgeted uncertainty sets is $\langle \Gamma \rangle + \langle a \rangle + \langle b \rangle$.

Subset Budgeted Uncertainty We call $\mathcal{U}$ *subset budgeted* if there exists a collection of subsets $\mathcal{S}$ of the index set $I = \{1, \dots, m\}$, i.e. $\mathcal{S} \subseteq 2^I$, and for each $S \in \mathcal{S}$ integers $\alpha_S, \beta_S \in \mathbb{N}$ such that $\mathcal{U} = \{\xi \in \mathbb{N}^m \colon \alpha_S \leq \xi(S) \leq \beta_S \ \forall S \in \mathcal{S}\}$. We call the problem $\mathcal{P}$ restricted to instances with subset budgeted uncertainty sets $\mathcal{P}$ with *subset budgeted uncertainty*. We assume that a subset budgeted uncertainty set has encoding length $\sum_{S \in \mathcal{S}} \langle \alpha_S \rangle + \langle \beta_S \rangle$.

2.5. Graph Theory

In most parts of this thesis we use standard graph terminology, cf. [Die00; KN09; Wes01]. In the following we recall some of the notation and state well known results in graph theory. All results in this chapter can be found in the mentioned standard textbooks. As directed graphs are barely used throughout this thesis we do not repeat any particular notations here and simply refer to [KN09] for formal definitions. We often omit the set brackets in our notations, when the considered set consists of only one element.

Fundamentals An (*undirected*) *graph* G is a triple that consists of a finite non-empty *vertex set* $V(G)$, a finite *edge set* $E(G)$ and a mapping that assigns each edge a set of 2 *endvertices*. We say an edge *connects* its two endvertices. In particular, all graphs regarded in this thesis do not contain loops. The number of vertices in a graph is called its *order* and the number of edges its *size*. Two edges with the same set of endvertices are called *parallel*. If a graph G does not contain any parallels, it is called *simple*. An edge is *incident* to its endvertices and two vertices are *adjacent* or *neighbors* if they are the endvertices of an edge. The *neighborhood* $N_G(v)$ of a vertex $v \in V(G)$ is the set of all neighbors of v, and $\deg_G(v) = |N_G(v)|$ is the *degree* of v in G. For both notations we omit the subscript in the if the graph is clear from context. For disjoint subsets $S, T \subseteq V(G)$, we denote by $E(S, T)$ the set of edges that have one endpoint in S and one in T.

Lemma 2.1. *Let G be a graph. It holds that*

$$\sum_{v \in V(G)} \deg(v) = 2 \cdot |E(G)| .$$

In particular, the number of vertices with odd degree is even.

We denote an edge $e \in E(G)$ with endvertices $v, w \in V(G)$ by vw. As we consider parallel edges in this thesis, we may not identify an edge with its to endvertices. However, for simplicity of notation we assume that two edges that are equal always connect the same two endvertices. If differentiation of parallels is necessary we use the notation $(vw), (vw)', \ldots$ and so forth. By $|vw|$ we denote the number of edges connecting v and w.

We call two graphs G_1, G_2 *isomorphic* if there exists a bijective mapping $\phi \colon V(G_1) \to V(G_2)$ such that $|vw| = |\phi(v)\phi(w)|$ for all $v, w \in V(G_1)$. By abuse of notation we also call two isomorphic graphs G_1 and G_2 equal and write $G_1 = G_2$.

Sub- and Supergraphs Let G be a graph. If G is simple, the graph $\overline{G}$ with $V(\overline{G}) = V(G)$ and $E(\overline{G}) = \{vw \colon v, w \in V(G),\ vw \notin E(G)\}$ is called the *complement* of G. A *subgraph* G' of a graph G is a graph such that $V(G') \subseteq V(G)$, $E(G') \subseteq E(G)$, and in G', each edge has the same set of endvertices as in G. We write $G' \sqsubseteq G$. In this case G is called a *supergraph* of G'. If additionally $G \neq G'$ we call G' a *proper* subgraph of G and G a *proper* supergraph of G'. For a subset $V' \subseteq V(G)$ of the vertices of G the graph $G[V']$ *induced* by V' is the subgraph of G on the vertex set V' that contains all edges in $E(G)$ with both endpoints in V'. Similarly for $E' \subseteq E(G)$ the graph $G[E']$ *induced* by E' is the graph consisting of all endvertices of edges in E' as vertex set and E' as edge set. For two graphs G, G' we call the graph $G \cup G'$ with vertex set $V(G) \cup V(G')$ and edge set $E(G) \cup E(G')$ the *union* of G and G'.

Addition and Removal of Vertices and Edges Let G be a graph. For $V' \subseteq V(G)$ we write $G - V'$ for the graph $G[V(G) \setminus V']$. For $E' \subseteq E(G)$ we write $G - E'$ for the graph with vertex set $V(G)$ and edge set $E(G) \setminus E'$. For $v, w \in V(G)$ we denote by $G + vw$ the graph that, in contrast to G, contains one additional edge connecting v and w.

Paths and Cycles A graph P is called a *path* if its vertex set is of the form $V(P) = \{v_1, \ldots, v_k\}$ for distinct vertices and its edge set is of the form $E(P) = \{v_1v_2, v_2v_3, \ldots v_{k-1}v_k\}$. We often denote a path by the sequence of its vertices, i.e. $P = v_1v_2 \ldots v_k$. We denote by $\Omega(P) = \{v_1v_k\}$ the set of *endvertices* of the path P and also call P an *v_1-v_k path.* All vertices of P that are not endvertices are called its *internal vertices.* Further for $1 \leq i \leq j \leq k$, we write $v_iPv_j = v_i \ldots v_j$, where we omit to write v_i (v_j) for $i = 1$ ($j = k$). We call paths $P_1, \ldots P_k$ *internally vertex-disjoint* if no two paths P_i and P_j have a common internal vertex and *edge-disjoint* if no two paths contain a common edge.

If P is a path with endvertices $\Omega(P) = \{v, w\}$ the graph $P + vw$ is called a *cycle.* We denote by C_n a cycle on n vertices.

Particular Graphs A simple graph on n vertices K_n containing one edge for each pair of distinct vertices is called *complete graph* on n vertices. A *multiedge* is a graph on two vertices containing at least one edge. The graph C_3 is a *triangle.* A graph consistin of 4 vertices and 3 edges such that one vertex is connected to all other vertices is called a *claw.*

Special Vertex and Edge Sets Let G be a graph, $S \subseteq V(G)$ and $F \subseteq E(G)$. The set S is a *dominating set* in G if for each vertex a neighbor or itself is contained in the set S. We call S an *independent set* in G if not two vertices in S are connected by an edge. S is a *vertex cover* if at least one endvertex of each edge is contained S and it is a *clique* if induces a complete graph. The set F is an *edge cover* if every vertex in G is incident to an edge in F. If no two edges in F share an endvertex, F is a *matching.*

Graph Classes Let G be a graph. If the vertex set $V(G)$ can be partitioned into two sets $A \uplus B = V(G)$, such that each edge in $E(G)$ connects a vertex from A to a vertex from B, we call G *bipartite.* In this case $A \uplus B$ is called the *bipartition* of G. We call G *chordal*, if every cycle C of length at least 4 in G contains a *chord*, i.e. an edge in G that connects two vertices not adjacent in C. If G does not contain a claw as an induced subgraph it is called *claw free.* G is called *perfect graph* if in every induced subgraph the size of a maximum clique and the minimum number of colors needed for a feasible vertex coloring coincides.

Connectivity and Separators Let G be a graph. If $S, T \subseteq V(G)$ such that there exists an s-t path in G for some $s \in S$ and $t \in T$. A set

$$W := \{v_1, \ldots, v_k\} \subseteq V(G) \setminus (S \cup T) \quad (F := \{e_1, \ldots e_k\} \subseteq E(G))$$

is called *S-T k-vertex separator* (*S-T k-edge separator*), if in $G - W$ ($G - F$) there does not exist an s-t path for any $s \in S$, $t \in T$. We say W (F) is a k-vertex separator (k-edge separator) if it is an S-T k-vertex separator (S-T k-edge separator) for some nonempty sets $S, T \subseteq V(G)$. We call the element in a 1-vertex separator, a *cut vertex*. An edge is called *bridge* in G it it is contained in a 1-edge separator.

Lemma 2.2. *Let G be a graph that does not contain a bridge and $e_1, e_2, e_3 \in E(G)$. If $\{e_1, e_2\}$ and $\{e_2, e_3\}$ are 2-edge separator, then so is $\{e_1, e_3\}$.*

A graph G is called *connected* if for any pair of distinct vertices $v, w \in V(G)$ there exists a v-w path in G. A subgraph $H \sqsubseteq G$ is called *connected component* if it is a maximal connected subgraph of G. For $k \geq 2$, we call a graph G *k-connected* if $V(G) \geq k+1$ and G does not contain a $(k-1)$-vertex separator. We call G *k-edge connected* if it does not contain a $(k-1)$-edge separator.

Theorem 2.3 (Menger (1927), cf. [Wes01]). *Let G be a graph with distinct vertices $v, w \in V(G)$. If $vw \notin E(G)$, the minimum size of an s-t vertex separator equals the maximum number of internally vertex-disjoint s-t paths.*

There is a similar version of Menger's Theorem concerning edge connectivity and different formulations with similar proofs. We refrain from stating them here and restate them when needed in this thesis.

Let G be a connected graph. The maximal subgraphs of G that do not contain a cut vertex are called *blocks*. A block is either a multiedge or a 2-connected graph. Denote by A the set of cut vertices in G and by $\mathcal{B}$ the set of blocks in G. The simple bipartite graph with vertex set $A \mathbin{\dot\cup} \mathcal{B}$ that contains an edge aB for a cut vertex a and a block B if and only if $a \in V(B)$ is called the *block-cutpoint tree* of G. We refer to the vertices of a block-cutpoint tree by *nodes*.

Lemma 2.4 ([Wes01],[HT73]). *The block-cutpoint tree of a connected graph G is a tree and can be computed in $O(|V(G)| + |E(G)|)$.*

Directed Graphs, Cuts, and Flows A directed graph G can be defined in a similar manner as an undirected graph, cf. [Wes01]. Instead of an edge set a directed graph contains an *arc set* $A(G)$ and an arc is associated with a *startvertex* and an *endvertex*. By $\delta^+_G(v)$ we denote the set of arcs that have v as an endvertex and by $\delta^-_G(v)$ we denote the set of arcs that have v as a startvertex.

Let G be a connected directed graph with edge weights $c \colon E(G) \to \mathbb{R}$. Further, let $s, t \in V(G)$ be two distinct vertices. An *s-t cut* is a partition $\{S, T\}$ of the vertices $V(G)$ such that $s \in S$ and $t \in T$. The *capacity* of the cut is $\sum_{e \in E(S,T)} c(e)$. An s-t cut is *minimum* if it has minimum capacity among all s-t cuts. An *s-t flow* with respect to c is a map $f \colon E(G) \to \mathbb{R}$ with $f(e) \leq c(e)$ for all $e \in E(G)$ and $\sum_{e \in \delta^+(v)} f(e) = \sum_{e \in \delta^-(v)} f(e)$ for all $v \in V(G) \setminus \{s, t\}$. The *value* is defined as $\sum_{e \in \delta^-(t)} f(e) - \sum_{e \in \delta^+(t)} f(e)$.

Theorem 2.5 (Max-Flow Min-Cut Theorem, cf. [Sch03]). *Let G be a directed graph, let $s, t \in V(G)$ be distinct vertices, and let $c \colon A(G) \to \mathbb{R}_{\geq 0}$. The maximum value of an s-t flow subject to c equals the minimum capacity of an s-t cut.*

Trees and Forests A graph G that does not contain a cycle is a *forest*. A connected forest is called a *tree*. A vertex of degree 1 in a forest is called *leaf*. A tree T *rooted* at a vertex $r \in V(T)$ is a tree with r designated as the *root* of T. Let $v, w \in V(T)$. We call v a *descendant* of w if w is on the unique r-v path in T. In this case we say w is an *ancestor* of v. If additionally $vw \in E(T)$ we say v is the *parent* of w and w is a *child* of v.

Contraction and Minors Let G be a graph and $vw \in E(G)$ be an edge in G. The *contraction* of vw in G is the graph, in which the vertices v and w are replaced by a single vertex which is incident to all edges which have one endvertex in $\{v, w\}$. We call a graph H a *minor* of G if it can be obtained from a subgraph of G by repeated contractions of edges. The *contraction* of a subgraph H of G is the graph, in which all vertices $V(H)$ in G are replaced by a single vertex, which is incident to all edges which have one endvertex in $V(H)$. We write ${}^{G}/_{H}$ for the contraction of H in G.

Tree Decomposition and Treewidth A *tree decomposition* of a graph G is a pair $(\mathcal{B}, \mathcal{T})$, where $\mathcal{T}$ is a tree and $\mathcal{B} = \{B_i \colon i \in V(\mathcal{T})\}$ is a family of subsets of the vertices in $\mathcal{T}$, such that

(i) $\bigcup_{B \in \mathcal{B}} B = V(G)$,

(ii) for each $vw \in E(G)$ there exists some $B \in \mathcal{B}$ with $v, w \in B$ and

(iii) for all $i, j, k \in V(\mathcal{T})$, if k is on the path from i to j in $\mathcal{T}$, then $B_i \cap B_j \subseteq B_k$.

We refer to the subsets in $\mathcal{B}$ as *bags* and usually refer to the vertices of $\mathcal{T}$ as *nodes*. The *width* of a tree decomposition $(\mathcal{B}, \mathcal{T})$ is $\max_{B \in \mathcal{B}} |B| - 1$. Finally the *treewidth*, $\text{tw}(G)$, of a graph G is the smallest integer k for which there exists a tree decomposition of width k. For a general overview on tree decompositions and treewidth we refer to [Bod98].

Even Graphs A graph G is called *even* if all vertices are of even degree. A graph that is even and connected is called *Eulerian*.

Part I.

Structural Uncertainty in Graphs: Connectivity and Domination

Chapter 3.

Connectivity Pairs and the Conjecture of Beineke and Harary

The Beineke Harary Conjecture states that any two distinct vertices that can be separated with k vertices and l edges but not with $k-1$ vertices and l edges or k vertices and $l-1$ edges can be connected by $k+l$ edges-disjoint paths of which $k+1$ are internally vertex-disjoint. Our main contribution in this chapter is to prove the conjecture for $l=2$ and any $k \in \mathbb{N}$. We exploit the result for $l=2$ in order to prove that the conjecture also holds for all graphs that have treewidth at most 3 and all $k, l \in \mathbb{N}$. Finally, we prove that deciding if two vertices can be separated by k vertices and l edges is NP*-complete.*

The results of this chapter are published in [JKS19] and are joint work with Sven O. Krumke and Sebastian Johann.

One of the most fundamental questions, when regarding uncertainty in graphs, is to ask whether a graph remains connected when deleting some of its elements. *Connectivity* is a well examined property of graphs. A famous example is Menger's Theorem where vertex- and edge-connectivity are related to the the number of internally vertex- and edge-disjoint paths, cf. Theorem 2.3. For further basic results and a literature overview on connectivity we refer to the introduction of this thesis or standard textbooks, such as [BWO12; Die00; Wes01].

On the other hand properties of graphs that remain connected when removing vertices and edges at the same time have not been getting the same attention. In [EKM91] Egawa et al. introduce a form of mixed connectivity in which two vertices are (k, l)-connected if they cannot be separated with $k-r$ vertices and at most $rl-1$ edges for any $1 \leq r \leq k$. They establish a mixed version of Menger's Theorem: Two vertices are (k, l)-connected if and only if there exist kl edge-disjoint paths between these vertices which can be partitioned into l sets each containing k internally vertex-disjoint paths. As we do not regard this kind of mixed connectivity here we refer to [BWO12] for further details.

Here we regard a form of mixed connectivity introduced by Beineke and Harary in [BH67], called *connectivity pairs*. A pair of non-negative integers (k, l) is called a connectivity pair for distinct vertices s and t if they can be separated by removing k vertices and l edges, but not with $k-1$ vertices and l edges or k vertices and $l-1$ edges, cf. Definition 3.2.

In [BH67] Beineke and Harary claim to have proved a mixed version of Menger's Theorem concerning connectivity pairs: If (k, l) is a connectivity pair for s and t, there exist $k+l$ edge-disjoint s-t paths of which k are internally vertex-disjoint. Mader pointed out in [Mad79] that the proof is erroneous. Later Sadeghi and Fan claimed in [SF19] to have proved the following statement:

When $V(G) \geq k+l+1$, $k \geq 0$ and $l \geq 1$, a graph G has $k+l$ edge-disjoint paths of which $k+1$ are internally vertex-disjoint between any two vertices, if and only if the graph cannot be disconnected by removing k vertices and $l-1$ edges.

Again the claimed proof has been found faulty as was observed by the author of this thesis, cf. [JKS19]. The paper by Sadeghi and Fan was retracted shortly after the authors were notified that their proof is erroneous. In particular, the claimed statement is not correct, as the existence of the claimed paths are not sufficient for the connectivity statement to hold. A counterexample is provided in Section 3.2.

The most meaningful result on the conjecture to date is due to Enomoto and Kaneko, cf. [EK94]. They first extended the conjecture claiming that it is possible to find $k+1$ internally vertex-disjoint paths instead of just k under the additional assumption that $l \geq 1$ and then proved their statement for certain k and l. The exact result is restated here as Theorem 3.6.

From our studies the following conjecture originally formulated by Beineke and Harary in [BH67] and extended by Enomoto and Kaneko in [EK94] may hold. In the remainder of this section we refer to the conjecture by the name *Beineke Harary Conjecture.*

Conjecture (Beineke Harary Conjecture). *Let G be a graph, $s, t \in V(G)$ distinct vertices and k, l non-negative integers with $l \geq 1$. If (k, l) is a connectivity pair for s and t in G, then there exist $k+l$ edge-disjoint paths of which $k+1$ are internally vertex-disjoint.*

Regard a graph G and two distinct vertices s and t. By $\kappa_G(s,t)$ we denote the cardinality of a minimum s-t vertex separator in G. For $k \in \{0, \ldots, \kappa_{G-E(s,t)}(s,t)\}$ it can be observed that there is a unique $l \in \mathbb{N}$ such that (k, l) is a connectivity pair for s and t in G, cf. Observation 3.17. In [BWO12], the authors mention that it is an interesting open question to consider the complexity of computing that unique integer l.

Outline We begin this chapter by giving formal definitions for disconnecting and connectivity pairs and making some basic observations concerning the topic in Section 3.1. In Section 3.2 we return to the Beineke Harary Conjecture and state basic results and observations concerning the conjecture. Section 3.3 is dedicated to proving the Beineke Harary Conjecture for $k = 2$. We prove the conjecture for graphs of treewidth at most 3 in Section 3.4. Finally, in Section 3.5 we show that no polynomial time algorithm computing the second coordinate in a connectivity pair exists, unless P = NP.

3.1. Connectivity Pairs

Definition 3.1 (Disconnecting Pair). Let G be a graph and $S, T \subseteq V(G)$ be non-empty subsets of vertices. We call a pair (W, F) with $W \subseteq V(G) \setminus (S \cup T)$ and $F \subseteq E(G)$ an *S-T disconnecting pair* if $G - W - F$ contains no path from a vertex in S to one in T.

We call the number of edges in a disconnecting pair its *size*, the number of vertices in a disconnecting pair its *order* and the number of elements $|W| + |F|$ its *cardinality*. If $S = \{s\}$ or $T = \{t\}$ consist of only one element we omit the set brackets in the notation and also write s-t disconnecting pair.

Beineke and Harary introduced *connectivity pairs* in their paper from 1967 [BH67]. We recall their definition below.

Definition 3.2 (Connectivity Pairs). Let G be a graph and $s, t \in V(G)$ be distinct vertices. We call a tuple of non-negative integers (k, l) a *connectivity pair* for s and t in G if

(i) there exists an s-t disconnecting pair of order k and size l and

(ii) there is no s-t disconnecting pair of cardinality less than $k + l$, order at most k and size at most l.

As property *(i)* implies, that there exist k vertices other than s and t and at least l edges, we may replace property *(ii)* by

$$\text{there is no } s\text{-}t \text{ disconnecting pair of order } k \text{ and size } l-1 \text{ or order } k-1 \text{ and size } l. \tag{3.1}$$

If there are fewer than l edges between s and t, we can simplify the condition even more and replace property *(ii)* by

$$\text{there is no } s\text{-}t \text{ disconnecting pair of order } k \text{ and size } l-1. \tag{3.2}$$

This is true since we may replace any edge in an s-t disconnecting pair by a vertex incident to it unless the edge connects s and t.

3.2. The Foundations of the Beineke Harary Conjecture

The Beineke Harary Conjecture is, in some sense, a mixed form of Menger's Theorem. As we make use of them, we recall three versions of Menger's theorem here.

Theorem (Menger's Theorem). Let s and t be two distinct vertices of a graph G.

(i) If $st \notin E(G)$, then the minimum number of vertices separating s and t in G is equal to the maximum number of internally vertex-disjoint s-t paths.

(ii) The minimum number of edges separating s and t in G is equal to the maximum number of edge-disjoint s-t paths in G.

(iii) the minimum cardinality of an s-t disconnecting pair is equal to the maximum number of internally vertex-disjoint s-t paths.

Proof. Proofs for the statements *(i)* and *(ii)* can, for example, be found in [Wes01]. The statement *(iii)* is a direct consequence of *(i)*: Any edge connecting s and t induces an s-t path that is internally vertex-disjoint to all other s-t paths. Also every edge connecting s and t is contained in every s-t disconnecting pair. The statement now follows considering that any edge in an s-t disconnecting pair that does not connect s and t can be replaced by one of its endvertices. □

Menger's Theorem implies the Beineke Harary Conjecture for a couple of base cases regarding the integers k and l. These have been observed before and are not hard to grasp.

Observation 3.3. *Let $k \geq 0$ and $l \geq 1$ be integers and let s and t be two distinct vertices of a graph G.*

(i) If $(k, 0)$ is a connectivity pair for s and t in G, then s is not adjacent to t. Further, the minimum number of vertices separating s and t is k and by Menger's Theorem there are k internally vertex-disjoint s-t paths.

(ii) If $(k, 1)$ is a connectivity pair for s and t, then the minimum cardinality of an s-t disconnecting pair is $k + 1$. Hence, by Menger's Theorem there are $k + 1$ internally vertex-disjoint s-t paths.

(iii) If $(0, l)$ is a connectivity pair for s and t, then the minimum number of edges separating s and t is l. By Menger's Theorem there are l edge-disjoint s-t paths.

Another rather basic result shows that it suffices to prove the Beineke Harary Conjecture for non-adjacent vertices s and t as we see in the next two lemmata.

Lemma 3.4. *Let G be a graph, $s, t \in V(G)$ be two distinct vertices and let k, l be non-negative integers. Then, (k, l) is a connectivity pair for s and t in G if and only if $(k, l - |E(s, t)|)$ is a connectivity pair for s and t in $G - E(s, t)$.*

Proof. Any s-t disconnecting pair in G has to contain all edges in $E(s, t)$. Thus, we get a one-to-one correspondence between the s-t disconnecting pairs in G and the ones in $G - E(s, t)$ by mapping a pair (W, F) to the pair $(W, F \setminus E(s, t))$. The desired result follows immediately. □

Lemma 3.5. *Let $\mathcal{G}$ be a class of graphs which is closed under deletion of edges. If the Beineke Harary Conjecture holds true for all graphs $G \in \mathcal{G}$ and all vertices $s, t \in V(G)$ such that s and t are not adjacent, then the conjecture holds true for all graphs in $\mathcal{G}$.*

Proof. Assume the Beineke Harary Conjecture holds for all graphs $G' \in \mathcal{G}$ and all vertices $s, t \in V(G')$ with $|E(s, t)| = 0$. Let $G \in \mathcal{G}$ be a graph, $s, t \in V(G)$ distinct vertices with $|E(s, t)| \geq 1$, and let (k, l) be a connectivity pair for s and t in G. By Lemma 3.4, $(k, l - |E(s, t)|)$ is a connectivity pair for s and t in $G - E(s, t)$. Thus, by assumption there exist $k + l - |E(s, t)|$ edge-disjoint s-t paths of which at least k are internally vertex-disjoint in $G - E(s, t)$. Note that we cannot assume that $k + 1$ paths are internally vertex-disjoint, as $l - |E(s, t)| = 0$ is a possibility. Nevertheless, the $k + l - |E(s, t)|$ paths together with the edges in $E(s, t)$ yield $k + l$ edge-disjoint s-t paths of which at least $k + 1$ are internally vertex-disjoint, as the edges in $E(s, t)$ are internally vertex-disjoint to all s-t paths and by assumption $|E(s, t)| \geq 1$. □

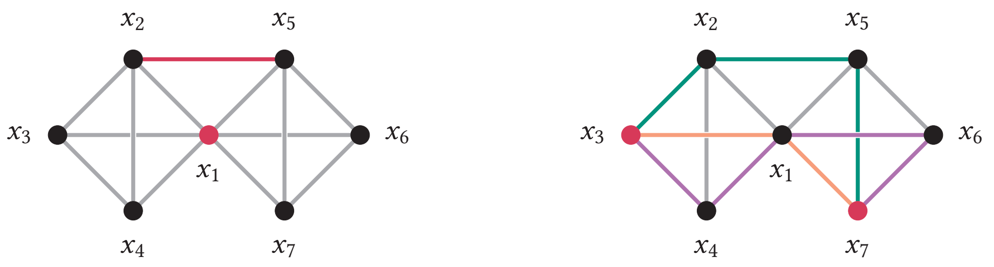

Figure 3.1.: A graph containing a vertex-edge separator, such that between any pair of vertices there exist three edge-disjoint paths of which two are internally vertex-disjoint.

Other than these simple observations the only meaningful result on the Beineke Harary Conjecture is due to Enomoto and Kaneko. Their result implies the correctness for certain integers k and l. We mention one explicit choice as a corollary, as we use this statement later on.

Theorem 3.6 ([EK94])**.** *Let q, r, k, and l be integers with $k \geq 0$ and $l \geq 1$ such that $k+l = q(k+1)+r$ and $1 \leq r \leq k+1$. Let s and t be distinct vertices of a graph G. If $q+r > k$ and if (k,l) is a connectivity pair for s and t, then G contains $k+l$ edge-disjoint s-t paths of which $k+1$ are internally vertex-disjoint.* □

Corollary 3.7. *Let $(1,l)$ be a connectivity pair for two distinct vertices s and t of a graph G. There are $l+1$ edge-disjoint s-t paths of which two are internally vertex-disjoint.*

Proof. For $l = 1$ the statement holds true due to Observation 3.3. For $l \geq 2$ and $q, r \in \mathbb{N}$ with $1 + l = q \cdot 2 + r$ and $1 \leq r \leq 2$ we have $q + r > 1$ and by Theorem 3.6 we get the desired paths. □

Before we turn to the proof of the Beineke Harary Conjecture for $l = 2$, we discuss the claim made by Sadeghi and Fan in [SF19], that we mentioned in the introduction. This serves to understand the difficulties of the Beineke Harary Conjecture and further shows why the conjecture does not claim equivalence of the existence of connectivity pairs and paths.

In [SF19] for integers $k, l \geq 1$ a graph G with at least $k+l+1$ vertices is called *(k,l)-connected* if it cannot be disconnected by removing k vertices and $l-1$ edges. The following claim is then made.

Let $k, l \geq 1$ and G be a graph with at least $k+l+1$ vertices. Then G is (k,l)-connected if and only if G is $k+1$ vertex-connected and $k+l$ edge-connected. (3.3)

If G is in fact (k,l)-connected it can readily be observed that it is also $k+1$ vertex-connected and $k+l$ edge-connected. On the other hand G being $k+1$ vertex-connected and $k+l$ edge-connected does *not* imply (k,l)-connectivity. To see this, consider the two complete graphs G_1 and G_2 on the vertex sets $\{x_1, x_2, x_3, x_4\}$ and $\{x_1, x_5, x_6, x_7\}$. We construct a graph G by regarding the union of G_1 and G_2 and additionally adding an edge between vertices x_5 and x_2. Figure 3.1 displays the constructed graph. The graph G is 2-vertex-connected and 3-edge-connected, but it is not $(1,2)$-connected as the removal of the vertex x_1 and the edge x_2x_5 disconnects the graph. Thus, the

Claim (3.3) cannot hold. As a corollary of Claim (3.3), Sadeghi and Fan state the following.

> Let $k \geq 0$, $l \geq 1$, and G be a graph with at least $k + l + 1$ vertices. Then G is (k, l)-connected if and only if it has $k + l$ edge-disjoint paths between every pair of vertices of which $k + 1$ paths are internally vertex-disjoint. (3.4)

As a corollary to Claim (3.3), Claim (3.4) cannot be considered proven. We give a counterexample to the claim in Proposition 3.8.

In the original conjecture by Beineke and Harary [BH67] and in the extension due to [EK94] it is never claimed that the existence of the desired paths is sufficient for (k, l)-connectivity and, in fact, it is not. For the sake of completeness we argue why the existence of the paths in Claim (3.4) is not sufficient.

Proposition 3.8. *The graph G constructed above contains a separator of one vertex and one edge and between any pair of vertices there exist three edge-disjoint paths of which two are internally vertex-disjoint.*

Proof. Consider the graph G above, that also provided a counterexample to Claim (3.3), see Figure 3.1. The vertex x_1 together with the edge x_5x_2 disconnects the graph. Now let $v_1, v_2 \in V(G)$. If $v_1, v_2 \in V(G_i)$ for some $i \in \{1, 2\}$ there are three internally vertex-disjoint v_1-v_2 paths. Otherwise, without loss of generality $v_1 \in \{x_2, x_3, x_4\}$ and $v_2 \in \{x_5, x_6, x_7\}$. Denote by P_1 a shortest path from v_1 to x_2 (This is either a single edge or the path without edges) and by P_2 a shortest path from x_5 to v_2. We define the v_1-v_2 path $P := (P_1 \cup P_2) + x_2x_5$. Further let $Q = v_1x_1v_2$. Finally let $w_1 \in \{x_3, x_4\} \setminus \{v_1\}$ and $w_2 \in \{x_6, x_7\} \setminus \{v_2\}$ and define the path $R = v_1w_1x_1w_2v_2$. It is easily verified that P, Q, R are three edge-disjoint v_1-v_2 paths and P and Q are also internally vertex-disjoint. □

The graph from Figure 3.1 illustrates two things. On the one hand it shows that we may not hope to prove an equivalence in the fashion of Claim (3.4). On the other hand it shows that it is not possible to replace the mixed form of connectivity by two separate statements on pure connectivity in the fashion of Claim (3.3). This is one of the reasons why the Beineke Harary Conjecture is not a consequence of Menger's Theorem and its proof has not been established as of yet. It also suggests that the usual techniques used for proofs of Menger's Theorem might not transfer to the mixed statement. In the following we use a novel technique for proving the Beineke Harary Conjecture for the case that $l = 2$. The idea is to keep the desired $k + 1$ internally vertex-disjoint paths and move from s to t along the remaining path. The statement is then proved by induction.

3.3. The Beineke Harary Conjecture for Disconnecting Pairs with Two Edges

Now that we have established some foundations for connectivity pairs and the Beineke Harary Conjecture, we are ready to turn to the main result of this chapter.

Theorem 3.9. *Let G be a graph and $s, t \in V(G)$ and let $(k, 2)$ be a connectivity pair for s and t. Then, there exist $k + 2$ edge-disjoint s-t paths of which $k + 1$ are internally vertex-disjoint.*

Before we begin with the proof, note that the result of Theorem 3.9 has only been proved for $k = 1$. In particular, the result of Enomoto and Kaneko, cf. Theorem 3.6, basically tackles the conjecture from a different angle, as in their statement for $l = 2$ and $k \geq 2$, the sum $q + r$ always equals 2, which leads to a big gap between k and $q + r$ for large k.

We prove a more general version of Theorem 3.9 and afterwards conclude the correctness of the theorem. To this end recall that for a graph G, vertices $s, t \in V(G)$, and $k \in \mathbb{N}_{>0}$ an s-t k-skein is the union of k internally vertex-disjoint s-t paths.

Theorem 3.10. *Let G be a graph, $s_1, s_2, t \in V(G)$ with $s_1 \neq t$. Further assume that*

(i) there exists an s_2-t path in G,

(ii) there exists an s_1-t $(k+1)$-skein in G, and

(iii) there is no $\{s_1, s_2\}$-t disconnecting pair of cardinality $k+1$ and order at most k in G.

Then there exist $k+2$ edge-disjoint paths of which $k+1$ are internally vertex-disjoint s_1-t paths and of which one is an s_2-t path.

Proof. Let G be a graph, $s_1, s_2, t \in V(G)$ with $s_1 \neq t$ fulfilling properties *(i)* to *(iii)*. We prove the claim by induction on the number of edges $|E(G)|$. If $|E(G)| \leq k$ there cannot be $k+1$ internally vertex-disjoint s_1-t paths, as $s_1 \neq t$. Thus, from now on we may assume the following.

Let G' be a graph with $|E(G')| < |E(G)|$ and vertices $s_1', s_2', t' \in V(G')$ with $s_1' \neq t'$. If Properties (i) to (iii) are fulfilled in G', then there exist $k+2$ edge-disjoint paths of which $k+1$ are internally vertex-disjoint s_1'-t' paths and of which one is an s_2'-t' path. (3.5)

We begin by proving the induction step for the case that s_2 is contained in an s_1-t $(k+1)$-skein and afterwards use this result to prove the induction step for the case that s_2 is not contained in such a skein.

Case 1: The vertex s_2 is contained in an s_1-t $(k+1)$-skein.

If $s_2 = t$, then the $k+1$ internally vertex-disjoint paths from Property *(ii)* together with the s_2-t path $s_2 = t$ form the desired paths. Thus, we may assume that $s_2 \neq t$. Denote by $P_1, \ldots, P_{k+1}$ the s_1-t paths of an s_1-t $(k+1)$-skein containing s_2. Without loss of generality we may assume $s_2 \in V(P_{k+1})$. Denote by s_2' the vertex after s_2 on P_{k+1}, i. e. $P_{k+1} = s_1 \ldots s_2 s_2' \ldots t$. Note that $s_1 = s_2$ is not forbidden at this point. We now want to use the induction hypothesis for $G - s_2 s_2'$ and the vertices s_1, s_2' and t, cf. Figure 3.2 a).

Property *(i)* is fulfilled as $s_2' P_{k+1}$ is an s_2'-t path in $G - s_2 s_2'$. Suppose that there do not exist $k+1$ internally vertex-disjoint s_1-t paths in $G - s_2 s_2'$. By Menger's Theorem there is an s_1-t disconnecting pair (W, F) of cardinality k. Since in $G - s_2 s_2'$ the internally vertex-disjoint paths $P_1, \ldots P_k$ still exist, all elements of (W, F) are contained in the paths $P_1, \ldots, P_k$, cf. Figure 3.2 a). Thus, the path $P_{k+1} s_2$ still exists in $G - s_2 s_2' - W - F$ and (W, F) is an $\{s_1, s_2\}$-t disconnecting pair in $G - s_2 s_2'$, cf. Figure 3.2 b). By assumption $(W, F \cup \{s_2 s_2'\})$ is not an $\{s_1, s_2\}$-t disconnecting pair in G and there exists some $\{s_1, s_2\}$-t path in $G - W - F - s_2 s_2'$, cf. Figure 3.2 c), which yields a contradiction. Hence, Property *(ii)*

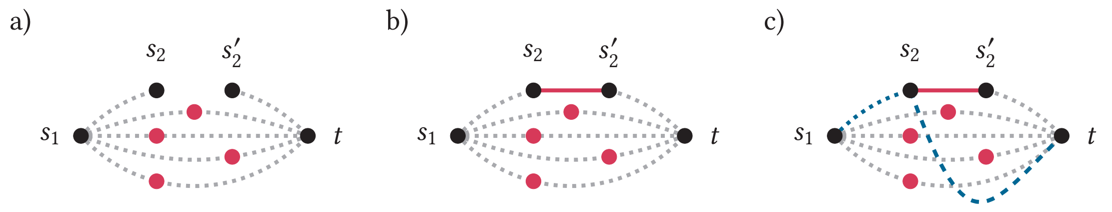

Figure 3.2.: Case 1 in the proof of Theorem 3.10: Supposed separation of s_1 and t. The colored vertices correspond to elements in (W, F). The dotted lines are mutually internally vertex-disjoint. The solid line is a single edge. The colored lines indicate a connection of vertices that does not touch colored vertices or edges.

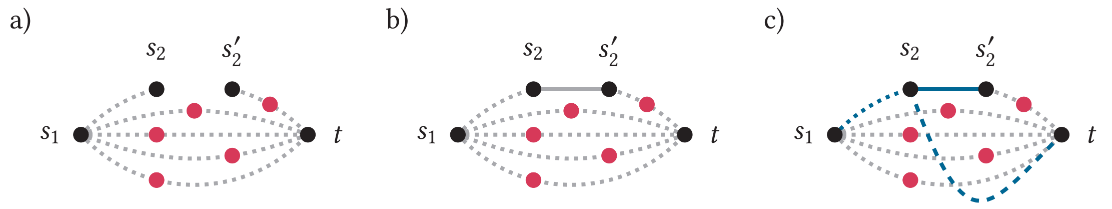

Figure 3.3.: Case 1 in the proof of Theorem 3.10: Supposed separation of $\{s_1, s_2'\}$ and t. The colored vertices correspond to elements in (W, F). The dotted lines are mutually internally vertex-disjoint. The solid line is a single edge. The colored lines indicate a connection of vertices that does not touch (W, F).

is fulfilled in $G - s_2s_2'$ and s_1, s_2', t. Now suppose there exists an $\{s_1, s_2'\}$-t disconnecting pair (W, F) of cardinality $k + 1$ and order at most k in $G - s_2s_2'$. As the paths $P_1, \ldots, P_k, s_2'P_{k+1}$ are internally vertex-disjoint, each element of (W, F) is contained in one of these paths, cf. Figure 3.3 a). Thus, $P_{k+1}s_2$ still exists in $G - s_2s_2' - W - F$ and neither s_2 nor s_2' are contained in the same component as t in $G - s_2s_2' - W - F$. This implies that (W, F) is an $\{s_1, s_2\}$-t disconnecting pair in G, cf. Figure 3.3 b). Again this is a contradiction to Property *(iii)* in G, cf. Figure 3.3 c) and hence Property *(iii)* is fulfilled for $G - s_2s_2'$ and s_1, s_2', t.

As $G - s_2s_2'$ contains $|E(G)| - 1$ edges, the statement (3.5) is applicable and there exist $k + 2$ edge-disjoint paths of which $k + 1$ are internally vertex-disjoint s_1-t paths, say $P_1', \ldots, P_{k+1}'$, and of which one is an s_2'-t path, say P_{k+2}', cf. Figure 3.4 b).

If $s_2 \in V(P_{k+2}')$ the paths $P_1', \ldots, P_{k+1}', s_2P_{k+2}'$ are the desired paths in G. Otherwise the paths $P_1', \ldots P_{k+1}', s_2s_2' \cup P_{k+2}'$ form the desired paths, cf. Figure 3.4 c).

Thus from now on, in addition to (3.5), we may assume:

Let G' be a graph with $|E(G')| = |E(G)|$ and vertices $s_1', s_2', t' \in V(G')$ such that $s_1' \neq t$ and s_2' is contained in an s_1-t $(k+1)$-skein. If Properties (i) through (iii) are fulfilled, then there exist $k + 2$ edge-disjoint paths of which $k + 1$ are internally vertex-disjoint s_1'-t' paths and of which one is an s_2'-t' path. (3.6)

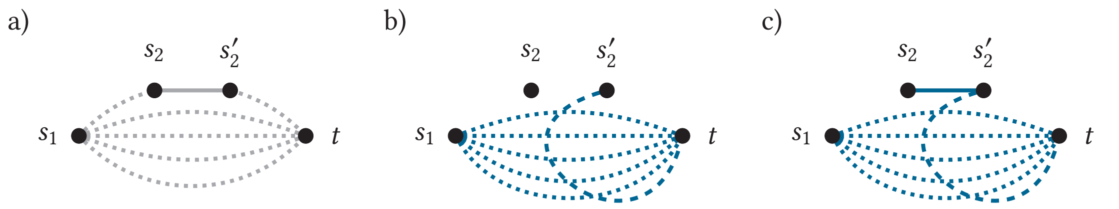

Figure 3.4.: Paths in Case 1 of the proof of Theorem 3.10. The dotted lines are mutually internally vertex-disjoint. The dashed line is edge-disjoint to the dotted lines. The solid line is a single edge not contained in any of the displayed paths. The colored lines form $k + 2$ vertex-disjoint paths of which $k + 1$ are vertex-disjoint.

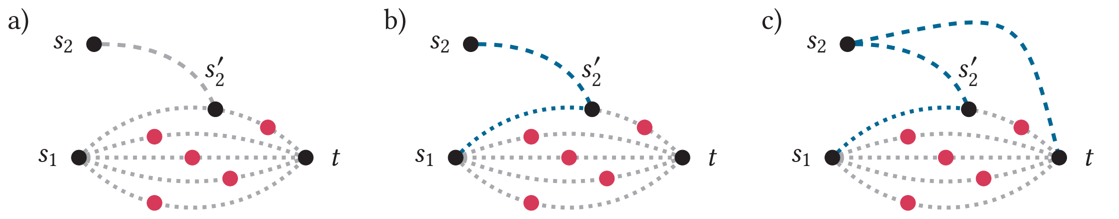

Figure 3.5.: Case 2 in the proof of Theorem 3.10: Supposed separation of $\{s_1, s_2'\}$ and t. The colored vertices correspond to (W, F). The dotted lines are mutually internally vertex-disjoint. The colored lines indicate a connection of vertices that does not touch (W, F).

Case 2: The vertex s_2 is not contained in any s_1-t $(k + 1)$-skein.

Denote by s_2' a vertex on an s_1-t $(k+1)$-skein that is closest (with respect to the number of edges) to s_2 among all vertices on s_1-t $(k + 1)$-skeins. Now we show that the assumptions still hold true if we replace s_2 by s_2'.

Observe that Properties *(i)* and *(ii)* are fulfilled when replacing s_2 by s_2'. To see that Property *(iii)* still holds, suppose that there exists an $\{s_1, s_2'\}$-t disconnecting pair (W, F) of cardinality $k + 1$ and order at most k. As there cannot be any s_1-t path left in $G - W - F$, all elements of the disconnecting pair are contained in some s_1-t $(k+1)$-skein, cf. Figure 3.5 a). The vertex set W may also not contain s_2' by definition. Thus, the vertices s_1, s_2 and s_2' are contained in the same component of $G - W - F$, cf. Figure 3.5 b). In G, the pair (W, F) cannot be $\{s_1, s_2\}$-t disconnecting by assumption, cf. Figure 3.5 c). This contradicts (W, F) being $\{s_1, s_2'\}$-t disconnecting in G and Property *(iii)* is fulfilled.

Thus, by (3.6) there exist $k + 2$ edge-disjoint paths, say $P_1, \ldots, P_{k+2}$, such that $P_1, \ldots P_{k+1}$ are internally vertex-disjoint s_1-t paths and P_{k+2} is an s_2'-t path, cf. Figure 3.6 b). Denote by P' a shortest s_2-s_2' path. Note that no element of P', except possibly s_2', is contained in $P_1, \ldots, P_{k+1}$ as no vertex or edge on an s_1-t $(k + 1)$-skein is closer to s_2 than s_2'. Further denote by s' the vertex on P' closest to s_2 that is also contained in P_{k+2}, cf. Figure 3.6 c). Then $P's' \cup s'P_{k+2}$ is an s_2-t path that is edge-disjoint to all $P_1, \ldots, P_{k+1}$ and we obtain the desired paths. □

Proof of Theorem 3.9. If s and t are adjacent, then $(k, 1)$ is a connectivity pair in $G - st$ and there

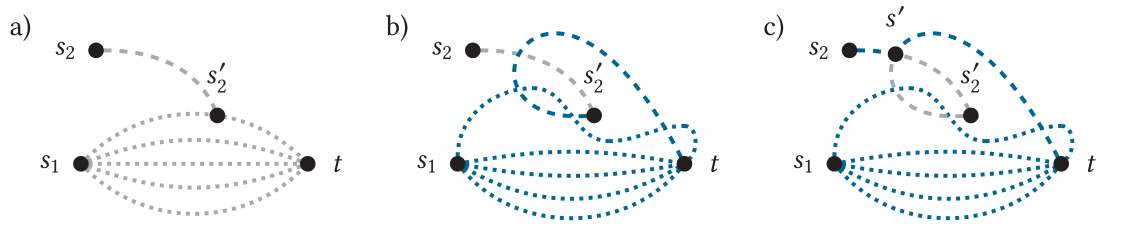

Figure 3.6.: Paths in Case 2 of the proof of Theorem 3.10. The dotted lines are mutually internally vertex-disjoint. The dashed lines are edge-disjoint to the dotted lines. The colored lines form $k + 2$ vertex-disjoint paths of which $k + 1$ are internally vertex-disjoint.

exist $k+1$ vertex-disjoint s-t paths in G by Observation 3.3 *(ii)*. Together with the deleted edge we get the desired paths in G. So assume that s and t are not adjacent. We show that Properties *(i)* through *(iii)* of Theorem 3.10 hold for G, $s_1 = s_2 = s$, and t.

By the definition of a connectivity pair, there is no s-t disconnecting pair of cardinality less than $k + 2$, order at most k and size at most 2. Thus, by Menger's Theorem there exist $k + 1$ internally vertex-disjoint s-t paths and Properties *(i)* and *(ii)* are fulfilled. Now suppose Property *(iii)* is not fulfilled and let (W, F) be an s-t disconnecting pair of cardinality $k+1$ and order at most k. As s and t are not adjacent, any edge in F has an endvertex not contained in $\{s, t\}$. Thus, replacing all but one edge in (W, F) with one of its endvertices that is contained in $\{s, t\}$ we get an s-t disconnecting pair of cardinality $k + 1$, order k, and size 1. Such a pair may not exist, as $(k, 2)$ is a connectivity pair for s and t. This yields a contradiction. Thus, the assumptions of Theorem 3.10 are fulfilled and there exist $k + 2$ edge-disjoint s-t paths in G of which $k + 1$ are internally vertex-disjoint. □

Theorem 3.9 is not only a stand-alone result, but can also be of help when proving the Beineke Harary Conjecture for some restricted graph classes. We illustrate this fact by proving the conjecture for graphs with treewidth at most 3 in the next section.

3.4. The Beineke Harary Conjecture on Graphs with Small Treewidth

In this section we prove the Beineke Harary Conjecture on graphs with treewidth at most 3. We need a couple of small lemmata and observations before we turn to the actual theorem.

The following result is well-known and can be found in [Die00]. We formulate it here as an observation:

Observation 3.11. *Let G be a graph and $(\mathcal{B}, \mathcal{T})$ a tree decomposition of G. Let $ij \in E(\mathcal{T})$ and denote by T_i and T_j the two components of $\mathcal{T} - ij$ with $i \in V(T_i)$ and $j \in V(T_j)$. If $v \in B_{i'} \setminus (B_i \cap B_j)$ for some $i' \in V(T_i)$ and $w \in B_{j'} \setminus (B_i \cap B_j)$ for some $j' \in V(T_j)$, then $B_i \cap B_j$ is a separator for v and w in G.*

Lemma 3.12. *Let G be a graph with treewidth at most k for some integer $k \geq 1$ and let $s, t \in V(G)$ be distinct and non-adjacent. Assume that every tree decomposition of width at most k has a bag* *containing s and t. Then, there exists a tree decomposition $D = (\mathcal{B}, \mathcal{T})$ of width k, such that there is some $ij \in E(\mathcal{T})$ with $s, t \in B_i \cap B_j$, $|B_i \cap B_j| \leq k$ and $G - (B_i \cap B_j)$ not connected. In particular s and t are contained in a vertex separator in G containing at most k vertices.*

Proof. Let G be a graph with treewidth k and $s, t \in V(G)$ distinct and non-adjacent. Assume that every tree decomposition of width at most k has a bag containing s and t. We begin by proving that every tree decomposition of width k has at least two bags containing s and t.

Suppose that D is a tree decomposition of G with width k, such that s and t share exactly one bag B_i. Let $j_1, \ldots, j_r$ be the neighbors of i in $\mathcal{T}$ whose bags contain s. We construct a new tree decomposition by replacing the node i with two adjacent nodes i_1 and i_2 with corresponding bags $B_{i_1} = B_i \setminus \{t\}$ and $B_{i_2} = B_i \setminus \{s\}$, making $j_1, \ldots, j_r$ adjacent to i_1 and making the remaining neighbors of i adjacent to i_2. As s and t are not adjacent, the result is a tree decomposition of width at most k in which no bag contains both, s and t. As we assumed that every tree decomposition of G with width at most k has a bag containing s and t, this is a contradiction. Thus, we may assume that every tree decomposition of width at most k has at least two bags containing s and t.

Regard some small tree decomposition $D = (\mathcal{B}, \mathcal{T})$ of G with width at most k, i.e., $B_i \subsetneq B_j$ and $B_j \subsetneq B_i$ for all $ij \in E(\mathcal{T})$. By the above arguments, there exists $ij \in E(\mathcal{T})$ such that $s, t \in B_i \cap B_j$. As the tree decomposition is small there exist $v \in B_i \setminus B_j$ and $w \in B_j \setminus B_i$. By Observation 3.11, the set $B_i \cap B_j$ separates v and w and the edge ij fulfills the conditions of the theorem. □

Lemma 3.13. *Let G be a graph with treewidth at most 3, let $s, t \in V(G)$ be distinct and non-adjacent, and let $k \geq 0$ and $l \geq 1$ be integers. Further, let $(\mathcal{B}, \mathcal{T})$ be a tree decomposition of width at most 3 such that for no bag $B \in \mathcal{B}$ we have $s, t \in B$. If (k, l) is a connectivity pair for s and t in G, then there exist $k + l$ edge-disjoint s-t paths of which $k + 1$ are internally vertex-disjoint.*

Proof. Denote by T_s (T_t) the subtree of $\mathcal{T}$ induced by all nodes corresponding to bags containing s (t). As $V(T_s) \cap V(T_t) = \emptyset$, there exists an edge $ij \in E(\mathcal{T})$ that separates $V(T_s)$ from $V(T_t)$. Thus, by Observation 3.11, the set $B_i \cap B_j$ is an s-t vertex separator in G. We may assume without loss of generality that $B_i \neq B_j$ and therefore get $|B_i \cap B_j| \leq 3$. The pair (k, l) is a connectivity pair and $l \geq 1$, which implies $k \leq 2$. If $l = 1$ the result follows from Observation 3.3. If $l = 2$ the result follows from Theorem 3.9. Further, if $k = 1$ the result follows form Corollary 3.7. Finally if $l > 2$, $k = 2$, and q, r integers such that $2 + l = q \cdot 3 + r$ with $1 \leq r \leq 3$, we get that $q + r > 2 = k$ and the desired result follows from Theorem 3.6. □

Note that in the proof of Lemma 3.13, we used our main result, Theorem 3.9, from the previous section. It is worth noting, that no other result in this thesis or another *easy* argument seems to be able to replace Theorem 3.9 in the proof of the lemma. In fact, the proof of the theorem was the only piece missing for proving the Beineke Harary Conjecture for graphs with treewidth at most 3 for some time.

We observe that the Beineke Harary Conjecture holds for graphs of treewidth 1: If for a graph G the underlying simple graph is a tree, either s and t are adjacent or there exists a vertex $a \in V(G) \setminus \{s, t\}$ separating s and t. In both cases the only possible connectivity pairs for s and t are of the form $(0, l)$ for $l \geq 1$ or $(1, 0)$. The conjecture follows from Observation 3.3.

Observation 3.14. *Let G be a graph of treewidth 1 and vertices $s, t \in V(G)$. Further let (k, l) be a connectivity pair for s and t in G, with $k \geq 0$ and $l \geq 1$. Then there exist $k + l$ edge-disjoint s-t paths, $k + 1$ of which are internally vertex-disjoint.*

In the next step we prove the conjecture for graphs with treewidth at most 2. Although Theorem 3.15 is implied by Theorem 3.16 and the proof could be included into the one of the latter, for better readability we prove the theorems separately. The structure of the two proofs is similar and therefore the proof of Theorem 3.15 can be regarded as a warm-up for the more technical one of Theorem 3.16.

Theorem 3.15. *Let G be a graph of treewidth at most 2 with distinct vertices $s, t \in V(G)$ and $k \geq 0$ and $l \geq 1$ integers. If (k, l) is a connectivity pair for s and t, then G contains $k + l$ edge-disjoint s-t paths of which $k + 1$ are internally vertex-disjoint.*

Proof. Let G be a graph, $s, t \in V(G)$ be distinct vertices and let (k, l) be a connectivity pair for s and t with $l \geq 1$. If $\mathrm{tw}(G) = 1$ the result follows from Observation 3.14. By Lemma 3.5 we may assume that s and t are not adjacent in G.

We prove the theorem by induction on the number of vertices $|V(G)|$. If $|V(G)| \leq 3$, as s and t are not adjacent, there always exists a tree decomposition of G in which no bag contains both, s and t. The claim follows from Lemma 3.13. So assume the claim holds for all graphs with less than $|V(G)|$ vertices.

If there exists a tree decomposition of G in which no bag contains both s and t, the claim is again implied by Lemma 3.13. Otherwise, by Lemma 3.12, the set $\{s, t\}$ is a vertex separator in G. Let C be a component of $G - \{s, t\}$ and denote the graph induced by $C \cup \{s, t\}$ by G_1. Let $G_2 = G - C$. Note that $|V(G_i)| < |V(G)|$ for $i \in \{1, 2\}$ and $E(G_1) \cap E(G_2) = \emptyset$. Regard some s-t disconnecting pair (W, F) of order k and size l in G. For $i \in \{1, 2\}$, the pair induces an s-t disconnecting pair (W_i, F_i) in G_i, with $W_i = W \cap V(G_i)$ and $F_i = F \cap E(G_i)$. Let $k_i = |W_i|$ and $l_i = |F_i|$. Then, (k_i, l_i) is a connectivity pair for s and t in G_i. Further $k_1 + k_2 = k$, $l_1 + l_2 = l$ and without loss of generality we may assume $l_2 \geq 1$. Thus, in G_1 there exist $k_1 + l_1$ edge-disjoint paths of which k_1 are internally vertex-disjoint. Note that we cannot assume that there exist $k_1 + 1$ internally vertex-disjoint paths as l_1 may equal 0. In G_2, by induction, we get $k_2 + l_2$ edge-disjoint paths of which $k_2 + 1$ are internally vertex-disjoint. For any two paths P_1 in G_1 and P_2 in G_2 it holds true that P_1 and P_2 are internally vertex-disjoint in G. Thus, there exist $k_1 + k_2 + l_1 + l_2 = k + l$ edge-disjoint s-t paths in G of which $k_1 + k_2 + 1 = k + 1$ are internally vertex-disjoint. □

Finally, we turn to the proof of the Beineke Harary Conjecture for graphs with treewidth at most 3. The structure of the proof is very similar to the one in Theorem 3.15. It is quite possible that this structure also generalizes graphs with larger treewidth. The main reason why we do not prove the conjecture for graphs of treewidth at most 4 (or even larger) is, that in order for Lemma 3.13 to hold for this class of graphs, we would have to prove the conjecture for $l = 3$ or find another way of proving this. The idea of the proof is to divide the graph at some separator containing s and t and use paths found in the resulting graphs by induction. In contrast to Theorem 3.15, a separator

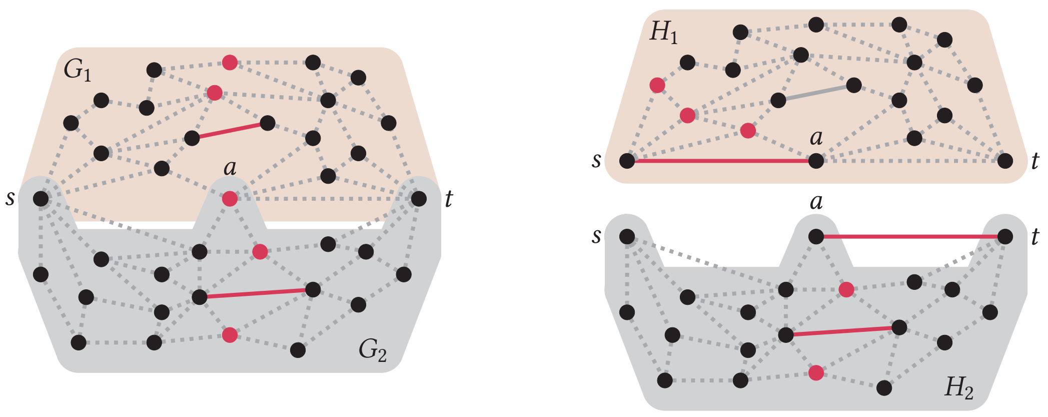

Figure 3.7.: Case 1 of the proof of Theorem 3.16: Partitioning the Graph G into G_1 and G_2 on the left. Defining the graphs H_1 and H_2 on the right. Colored elements form s-t disconnecting pair. Dotted lines indicate a connection between vertices.

containing s and t may now also contain a third vertex a. Thus, it is possible that some of the searched path actually cross at this vertex. To address this issue we introduce artificial edges that simulate part of the paths in the other component.

Theorem 3.16. *Let G be a graph with treewidth at most 3. Let $s, t \in V(G)$ be two distinct vertices and $k \geq 0, l \geq 1$ integers. If (k, l) is a connectivity pair for s and t, then G contains $k + l$ edge-disjoint s-t paths of which $k + 1$ are internally vertex-disjoint.*

Proof. Let G be a graph, $s, t \in V(G)$ distinct vertices and let (k, l) be a connectivity pair for s and t with $l \geq 1$. If $\text{tw}(G) \leq 2$ the result follows from Theorem 3.15. By Lemma 3.5 we may assume that s and t are not adjacent in G.

As in the proof of Theorem 3.15, we do the proof by induction on the number of vertices. If $|V(G)| \leq 4$, there exists a tree decomposition of G with width 3 in which no bag contains both, s and t, and the claim is implied by Lemma 3.13. So assume the claim holds for all graphs of treewidth at most 3 and $|V(G)| \leq 4$.

If there exists a tree decomposition of G width width 3 in which no bag contains both, s and t, the claim is again implied by Lemma 3.13. Otherwise, by Lemma 3.12, there exists a tree decomposition $D = (\mathcal{B}, \mathcal{T})$ containing an edge $xy \in \mathcal{T}$ such that for $B := B_x \cap B_y$ it holds true that $s, t \in B$, $|B| \leq 3$, and $G - B$ is not connected. If $B = \{s, t\}$ is a vertex separator, we may simply repeat the arguments in the proof of Theorem 3.15. So assume there is a vertex $a \in V(G) \setminus \{s, t\}$, such that $B = \{s, t, a\}$. Let C be a component of $G - B$, denote by G_1 the graph induced by $V(C) \cup B$ and let G_2 be the graph $G - V(C) - E_G(s, a) - E_G(a, t)$, cf. Figure 3.7. Note that $|V(G_i)| < |V(G)|$ for $i \in \{1, 2\}$. Further, as $B_x \cap B_y = \{s, t, a\}$, the two components of $\mathcal{T} - xy$ induce tree decompositions of G_1 and G_2 and we get $\text{tw}(G_1) \leq 3$ and $\text{tw}(G_2) \leq 3$. As the vertices s, t and a are contained in both B_x and B_y, adding edges between these vertices in G_1 or G_2 does not increase the treewidth of the graphs. We distinguish two cases.

Case 1: *The vertex a is contained in an s-t disconnecting pair of order k and size l in G.*

In $G - a$ the pair $(k-1, l)$ is a connectivity pair for s and t. Let (W, F) ba an s-t disconnecting pair of order k and size l with $a \in W$. For $i = 1, 2$, the pair (W, F) induces an s-t disconnecting pair (W_i, F_i) in $G_i - a$ of order k_i and size l_i such that $k_1 + k_2 = k - 1$ and $l_1 + l_2 = l$. Without loss of generality we may assume $l_1 \geq 1$.

Claim 1. (k_i, l_i) is a connectivity pair for s and t in $G_i - a$.

Proof. Suppose (k_i, l_i) is not a connectivity pair. As (W_i, F_i) is a disconnecting pair of order k_i and size l_i such a pair exists. Thus, there is a disconnecting pair (W', F') of order $k' \leq k_i$, size $l' \leq l_i$ and cardinality $k'+l' < k_i+l_i$, but then $(W' \cup W_j \cup \{a\}, F' \cup F_j)$ with $j \in \{1,2\} \setminus \{i\}$ is an s-t disconnecting pair in G of order at most k, size at most l and cardinality less than $k + l$ which yields a contradiction. ∎

Next we show that if (k_2, l_2) is not a connectivity pair for s and t in G_2, the desired paths exist. If (k_2, l_2) is in fact not a connectivity pair, then (k_2, l_2+p) is a connectivity pair for some integer $p \geq 1$. By induction we get $k_2 + l_2 + p$ edge-disjoint s-t paths in G_2, $k_2 + 1$ of which are internally vertex-disjoint. As (k_1, l_1) is a connectivity pair in $G_1 - a$ again by induction we get $k_1 + l_1$ edge-disjoint s-t paths in $G_1 - a$ of which $k_1 + 1$ are internally vertex-disjoint (recall that $l_1 \geq 1$). Together we get $k_2 + l_2 + p + k_1 + l_1 \geq k + l$ edge-disjoint s-t paths in G of which $k_1 + 1 + k_2 + 1 = k + 1$ are internally vertex-disjoint. Thus, we may assume that (k_2, l_2) is a connectivity pair for s and t in G_2.

Now a cannot be contained in any s-t disconnecting pair in G_2 of order k_2 and size l_2, as otherwise (k_2, l_2) would not be a connectivity pair for s and t in $G_2 - a$. We fix some s-t disconnecting pair (W_2', F_2') in G_2 that has order k_2 and size l_2. As in $G_2 - W_2' - F_2'$, the vertex a cannot be connected to both, s and t, without loss of generality we may assume that a is not connected to t. Thus, for the remainder of Case 1 we assume

(k_2, l_2) is a connectivity pair for s and t in G_2, (W_2', F_2') is an s-t disconnecting pair of order k_2 and size l_2 that does not contain a, and the vertices a and t are not connected in $G_2 - W_2' - F_2'$. (3.7)

We define $0 \leq q \leq l_1$ to be the unique integer such that $(k_1 + 1, l_1 - q)$ is a connectivity pair for s and t in G_1. Note that this is well-defined as $(W_1 \cup \{a\}, F_1)$ is an s-t disconnecting pair of order $k_1 + 1$ and size l_1.

Denote by H_1 the graph arising from G_1 by adding q parallel edges $e_1, \ldots, e_q$ between a and s, cf. Figure 3.7. Then the following holds:

Claim 2. $(k_1 + 1, l_1)$ is a connectivity pair for s and t in H_1.

Proof. As argued before, $(W_1 \cup \{a\}, F_1)$ is an s-t disconnecting pair of order $k_1 + 1$ and size l_1 in G_1 and thereby also disconnecting in H_1. Suppose that there exists an s-t disconnecting pair of order $k_1 + 1$ and size at most $l_1 - 1$. Let (W', F') be one such pair of minimal size. Suppose that $e_i \in F'$ for some $i \in \{1, \ldots, q\}$. As the size of the disconnecting pair is minimal, this implies $e_i \in F'$ for all $i \in \{1, \ldots, q\}$. If this is the case $(W', F' \setminus \{e_1, \ldots, e_q\})$ is an s-t

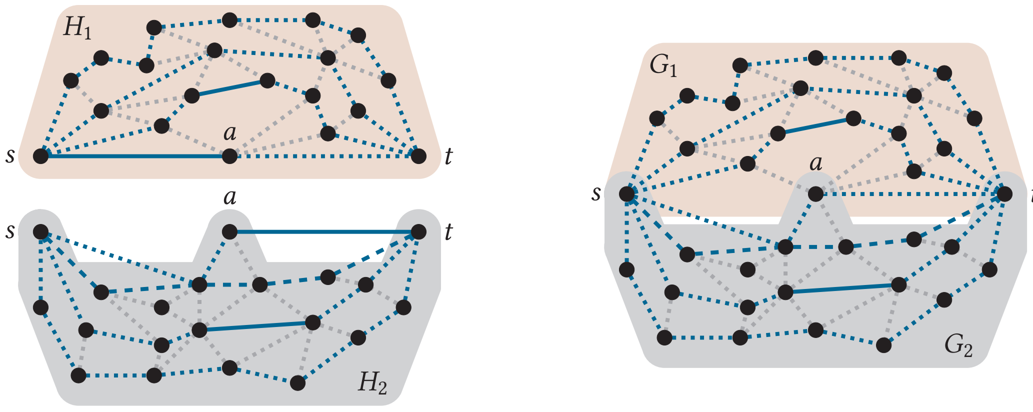

Figure 3.8.: Case 1 of the proof of Theorem 3.16: s-t paths in H_1 and H_2 given by induction on the left. Creation of desired s-t paths in G on the right. Colored dotted lines indicate internally vertex-disjoint s-t paths. Dashed line indicates an s-t path that is edge-disjoint to all other indicated paths. Solid lines are single edges.

disconnecting pair in G_1 of order at most $k_1 + 1$ and size at most $l_1 - 1 - q$ in contradiction to $(k_1 + 1, l_1 - q)$ being a connectivity pair in G_1. Thus, either $a \in W'$ or a and s are contained in the same component of $H_1 - W' - F'$. In particular there is no a-t path in $G_1 - W' - F'$. But then $(W' \cup W'_2, F' \cup F'_2)$ is an s-t disconnecting pair in G of order at most k and size at most $l - 1$ by (3.7). This contradicts (k, l) being a connectivity pair for s and t in G. ■

Next, denote by H_2 the graph arising from G_2 by adding q parallel edges $f_1, \ldots, f_q$ between a and t, cf. Figure 3.7. We prove the following:

Claim 3. $(k_2, l_2 + q)$ is a connectivity pair for s and t in H_2.

Proof. If $q = 0$ the statement holds true by (3.7), so assume that $q \geq 1$. The pair $(W'_2, F'_2 \cup \{f_1, \ldots, f_q\})$ is an s-t disconnecting pair in H_2 and a and t are not in the same component in $H_2 - W'_2 - F'_2 \cup \{f_1, \ldots, f_q\}$. So suppose there exists an s-t disconnecting pair of order k_2 and size $l_2 + q - 1$. Let (W', F') be one such pair of minimal size. With the same arguments as in the previous claim we get that $f_i \notin F'$ for all $i \in \{1, \ldots, q\}$. Thus, $a \in W'$ or a and t are in the same component in $H_2 - W' - F'$. In particular there does not exist an s-a path in $G_2 - W' - F'$. As $(k_1 + 1, l_1 - q)$ is a connectivity pair for G_1, there exists an s-t disconnecting pair (W'_1, F'_1) in G_1 of order $k_1 + 1$ and size $l_1 - q$. Suppose there exists an s-a path in $G_1 - W'_1 - F'_1$. Then, there does not exist an a-t path and $(W'_1 \cup W'_2, F'_1 \cup F'_2)$ is an s-t disconnecting pair in G of order k and size $l_2 + l_1 - q < l$ by (3.7) — a contradiction. On the other hand if there does not exist an s-a path in $G_1 - W'_1 - F'_1$, then the pair $(W'_1 \cup W', F'_1 \cup F')$ is s-t disconnecting in G and of order k and size $l - 1$, which again yields a contradiction. ■

Note that for $i \in \{1, 2\}$ it is $V(H_i) = V(G_i) < V(G)$ and $\mathrm{tw}(H_i) \leq 3$ as $\mathrm{tw}(G_i) \leq 3$ and we only added edges between s and a, respectively a and t to get to H_i from G_i. By Claim 2 and the induction hypothesis there are $k_1 + 1 + l_1$ edge-disjoint s-t paths in H_1, say $P_1, \dots, P_{k_1+l_1+1}$, of which $k_1 + 2$ are internally vertex-disjoint, cf. Figure 3.8. Without loss of generality let $P_1, \dots, P_{r_1}$ be the paths using edges from $\{e_1, \dots, e_q\}$, where from these we denote by P_1 the path that is among the $k_1 + 2$ internally vertex-disjoint paths, if one such path exists. If $q = l_2 = 0$, then $(k_2, 0)$ is a connectivity pair for s and t in $G_2 - a$ by Claim 1, and by Observation 3.3 there are k_2 internally vertex-disjoint s-t paths in $G_2 - a$. Together with $P_1, \dots, P_{k_1+l_1+1}$ we get the desired paths for G. So assume that $q + l_2 > 0$. By Claim 3 and the induction hypothesis there exist $k_2 + l_2 + q$ edge-disjoint s-t paths in H_2, say $Q_1, \dots, Q_{k_2+l_2+q}$ of which $k_2 + 1$ are internally vertex-disjoint, cf. Figure 3.8. Without loss of generality let $Q_1, \dots, Q_{r_2}$ for $r_2 \leq q$ be the paths using edges from $\{f_1, \dots, f_q\}$, where again from these we denote by Q_1 the path that is among the $k_2 + 1$ internally vertex-disjoint paths, if one such path exists. We now claim that for $r := \min\{r_1, r_2\}$ the paths

$$Q_1a \cup aP_1, \dots, Q_ra \cup aP_r, P_{r_1+1}, \dots, P_{k_1+l_1+1}, Q_{r_2+1}, \dots, Q_{k_2+l_2+q}$$

are at least $k + l$ edge-disjoint s-t paths of which at least $k + 1$ are internally vertex-disjoint, cf. Figure 3.8. First note, that the number of paths is exactly

$$r + (k_1 + l_1 + 1) - r_1 + (k_2 + l_2 + q) - r_2 = k + l + q + r - r_1 - r_2.$$

As r is equal to r_i for some i and q is greater than or equal to r_1 and r_2 we get that the number of paths is at least $k + l$. To see that among the paths above, there are at least $k + 1$ internally vertex-disjoint paths, note that we started off with a set of $k_1 + 2 + k_2 + 1 = k + 2$ internally vertex-disjoint paths $\mathcal{P} \subseteq \{P_1, \dots, P_{k_1+l_1+1}, Q_1, \dots, Q_{k_2+l_2+q}\}$.

The only vertex besides s and t that may be contained in more than one path of $\mathcal{P}$ is a. If $Q_1, P_1 \in \mathcal{P}$ they are glued together and $k + 1$ internally vertex-disjoint paths still remain. If only one of P_1 and Q_1, say P_1, is among the internally vertex-disjoint paths, then $\mathcal{P} \setminus \{P_1\}$ is a set of $k+1$ internally vertex-disjoint paths, as only one other path than P_1 may contain a. Finally if neither P_1 nor Q_1 are among the internally vertex-disjoint paths, then $\mathcal{P}$ contains a subset of internally vertex-disjoint paths of size $k + 1$ as at most two paths in $\mathcal{P}$ may contain a. This concludes Case 1.

Case 2: *The vertex a is not contained in any s-t disconnecting pair of order k and size l.*

Denote by (W, F) an s-t disconnecting pair of order k and size l and for $i \in \{1, 2\}$ let $W_i = V(G_i) \cap W$, $k_i = |W_i|$, $F_i = E(G_i) \cap E_i$, and $l_i = |F_i|$. Then $k_1 + k_2 = k$ and $l_1 + l_2 = l$. Without loss of generality we may assume that there is no s-a path in $G - W - F$ and thereby also no s-a path in $G_i - W_i - F_i$ for $i \in \{1, 2\}$.

For $i \in \{1, 2\}$ denote by $0 \leq q_i \leq l_i$ the unique integer such that $(k_i, l_i - q_i)$ is a connectivity pair for s and t in G_i. Note that this is well-defined as as (W_i, F_i) is an s-t disconnecting pair in G_i. We define $q = \max\{q_1, q_2\}$ and assume without loss of generality that $q = q_1$. Let (W_1', F_1') be an s-t disconnecting pair in G_1 of order k_1 and size $l_1 - q$ and denote by H_1 the graph arising from G_1 by adding q edges $e_1, \dots, e_q$ between a and t.

Claim 4. (k_1, l_1) is a connectivity pair for s and t in H_1.

Proof. If $q = 0$ the claim holds true by definition of q. So assume $q \geq 1$. Clearly $(W_1', F_1' \cup \{e_1, \dots, e_q\})$ is an s-t disconnecting pair in H_1 of order k_1 and size l_1. So suppose there exists an s-t disconnecting pair of order k_1 and size at most $l_1 - 1$. Let (W', F') be such a pair of minimal size. If $e_1, \dots, e_q \in F'$ the pair $(W', F' \setminus \{e_1, \dots, e_q\})$ is s-t disconnecting in G_1 and of order k_1 and size at most $l_1 - q - 1$, contradicting the fact that $(k_1, l_1 - q)$ is a connectivity pair for s and t in G_1. Thus, either $a \in W'$ or a and t are contained in the same component in $H_1 - W' - F'$. In particular there is no s-a path in $G_1 - W' - F'$ and thereby the pair $(W' \cup W_2, F' \cup F_2)$ is s-t disconnecting in G and of order k and size at most $l - 1$. A contradiction to (k, l) being a connectivity pair for s and t in G. ■

Let now H_2 be the graph arising from G_2 by adding q edges $f_1, \dots, f_q$ between a and s. For H_2 we can also find a connectivity pair.

Claim 5. $(k_2, l_2 + q)$ is a connectivity pair for H_2.

Proof. Again, if $q = 0$ the claim is immediate by definition of q. So let $q \geq 1$. Then $(W_2, F_2 \cup \{f_1, \dots, f_q\})$ is an s-t disconnecting pair of order k_2 and size $l_2 + q$ in H_2. So suppose there exists an s-t disconnecting pair of order k_2 and size at most $l_2 + q - 1$. Let (W', F') be such a pair of minimal size. If $f_1, \dots, f_q \in F'$, then there is no s-a path in $H_2 - W' - F'$. This implies that $(W' \cup W_1, F' \setminus \{f_1, \dots, f_{q'}\} \cup F_1)$ is a disconnecting pair in G of order at most k and size at most $l - 1$ yielding a contradiction. Thus, either $a \in W'$ or a and s are contained in the same component in $H_2 - W' - F'$. In particular there is no a-t path in $G_2 - W' - F'$. If there is also no a-t path in $G_1 - W_1' - F_1'$, the pair $(W' \cup W_1', F' \cup F_1')$ is disconnecting in G and of order at most k and size at most $l - 1$. This yields a contradiction to (k, l) being a connectivity pair for s and t in G. So suppose that there is an a-t path in $G_1 - W_1' - F_1'$. Then there is no s-a path in $G_1 - W_1' - F_1'$ and $(W_1' \cup W_2, F_1' \cup F_2)$ is an s-t disconnecting pair in G, that has order at most k and size at most $l_1 - q + l_2 < l$ as $q \geq 1$. Again this contradicts the fact that (k, l) is a connectivity pair for s and t in G. ■

As in the proof of Case 1 we use the induction hypothesis on H_1 and H_2 to get the desired paths. If neither $l_1 = 0$ nor $l_2 + q = 0$ we get the paths in G in the same manner as in Case 1 and therefore do not repeat the arguments here.

For the other case, let $l_1' = l_1$ and $l_2' = l_2 + q$. If we can show for $i \in \{1, 2\}$, that if $l_i' = 0$, then in $G_i - a$ the pair $(k_i, 0)$ is a connectivity pair, we can again proceed as in Case 1 and get the desired paths. To see this we simply observe that a is not contained in any k_i-vertex separator in G_i as this would imply that a is contained in an s-t disconnecting pair of order k and size l in G. □

3.5. Computing the Second Coordinate in a Connectivity Pair

In [BWO12] Beineke and Wilson proposed the question how difficult it is to compute the second coordinate in a connectivity pair. In this section we deal with this question and show that there is no polynomial time algorithm for this problem unless P = NP. We formulate the question as the following decision problem.

Connectivity Pair (CP).
Instance: An undirected graph G, vertices $s, t \in V(G)$, and $k, B \in \mathbb{N}$.
Question: Does $l \in \mathbb{N}$ with $l \leq B$ exist such that (k, l) is a connectivity pair for s and t in G?

Recall that by $\kappa_G(s,t)$ we denote the vertex-connectivity between s and t, i.e. the cardinality of a minimum vertex separator separating s and t. It can be readily observed, that for a fixed k the second coordinate in a connectivity pair is unique in the following sense.

Observation 3.17. *Let G be a graph with distinct vertices $s, t \in V(G)$ and let, $k, l \in \mathbb{N}$ with $0 \leq k \leq \kappa_{G-E(s,t)}(s,t)$. There exists a unique integer l_k such that (k, l_k) is a connectivity pair for s and t in G. In particular, if there exists an s-t disconnecting pair of order k and size l, then there exists a unique integer $l_k \leq l$ such that (k, l_k) is a connectivity pair.*

We prove that Connectivity Pair is NP-complete by a reduction from Partial Vertex Cover. Let G be a graph and q a non-negative integer. Recall the definition of a partial vertex cover in a graph: We call a subset $C \subseteq V(G)$ a *partial vertex cover* with respect to q if at least q edges in $E(G)$ are incident to a vertex in C. In [CS13] Caskurlu and Subramani showed that Partial Vertex Cover is NP-complete when restricted to instances with bipartite graphs.

Theorem 3.18. *Connectivity Pair is* NP*-complete.*

Proof. To verify that CP $\in$ NP, we use as a certificate an s-t disconnecting pair of order k and size r' for some $r' \leq B$. Clearly we may verify that the pair is disconnecting in polynomial time. Thus, by Observation 3.17 there exists an $l \leq r'$ such that (k, l) is a connectivity pair.

Let an instance of Partial Vertex Cover be given, such that the graph G of the instance is bipartite with bipartition $A \cup B$. As we can decide in polynomial time if there exists a vertex cover with at most B vertices on bipartite graphs, we may assume that any vertex cover of G has cardinality of at least $B+1$. We construct an instance of CP as follows: Set the graph of the instance to be G' with $V(G') := V(G) \cup \{s, t\}$ for some distinct vertices $s, t \notin V(G)$ and $E(G') = E(G) \cup E_s \cup E_t$, where E_s (E_t) consists of $|A| \cdot |E(G)|$ ($|B| \cdot |E(G)|$ edges of which $|E(G)|$ are parallels between s (t) and a (b) for each $a \in A$ ($b \in B$). Moreover, set $k := B$ and $l := |E(G)| - q$. All these steps can be realized in polynomial time. We note that there are at least $k+1$ internally vertex-disjoint s-t paths, as any vertex cover in G has cardinality at least $B + 1 > k$.

First assume that there exists a partial vertex cover S with respect to q in G such that $|S| \leq B$. Then (S, F) with $F := \{vw \in E(G) : v, w \notin S\}$ is an s-t disconnecting pair in G' with $|S| \leq B = k$ and $|F| \leq |E(G)| - q = l$. By Observation 3.17 there exists an $l' \leq l$ such that (k, l') is a connectivity pair for s and t in G'.

Now assume that (k, l') is a connectivity pair for s and t in G' for some $l' \leq l$. By the definition of a connectivity pair, there exists an s-t disconnecting pair (W, F) of order k and size l'. There does not exist an s-t disconnecting pair of order k and size smaller than l'. Thus, it holds true that $F \subseteq E(G)$, as $l' \leq l \leq |E(G)|$ and any disconnecting pair with a minimal amount of edges either contains an edge and all its parallels or it does not contain the particular edge. For any edge

$ab \in E(G) \setminus F$ it holds true that $a \in W$ or $b \in W$, as otherwise, $sabt$ is an s-t path in $G - W - F$. Thus, W is a partial vertex cover in G with respect to $|E(G)| - |F| \geq |E(G)| - l = q$. Further it is $|W| = k = B$, which finishes the proof. □

We get the intractability of the computation of the second coordinate of a connectivity pair as an immediate corollary: An algorithm computing the second coordinate in polynomial time could clearly be used to decide CP in polynomial time.

Corollary 3.19. *Unless* P = NP*, there is no polynomial time algorithm that, given any graph G, distinct vertices s and t and an integer $0 \leq k \leq \kappa_{G-E(s,t)}(s,t)$, returns the integer l such that (k,l) is a connectivity pair for s and t in G.*

Note that it is possible to formulate a version of CP for the first coordinate of a connectivity pair. We would get NP-completeness in a similar manner as in Theorem 3.18. We do not explicitly define this problem here, as for a fixed l, there does not necessarily exist a k such that (k,l) forms a connectivity pair and the question of computing the first coordinate in a connectivity pair is thereby undefined. To see this, for example, regard some graph where each edge has at least $l+1$ parallels.

As for any fixed k the corresponding connectivity pair is unique by Observation 3.17, the problem of computing all connectivity pairs for a graph G and vertices s and t can be solved by computing the unique connectivity pairs for all k with $0 \leq k \leq \kappa_{G-E(s,t)}(s,t)$.

Conclusion We began this chapter by briefly repeating basic observations and previous results concerning connectivity pairs and, in particular, the Beineke Harary Conjecture. We then proved the conjecture for the special case that the second coordinate in the regarded connectivity pair equals 2. This substantially differs from previous results in the literature and can be used to prove the Beineke Harary Conjecture for restricted graph classes. We illustrate the latter fact by using our results from Section 3.3 to prove the conjecture for graphs of treewidth at most 3. Finally, in the last section we consider the complexity of computing the second coordinate in a connectivity pair, which was raised by Beineke et al. in [BWO12]. We prove that there is no polynomial time algorithm performing the desired task, unless P = NP.

The most self-evident further research direction is proving the Beineke Harary Conjecture for $l = 3$ or even larger. As mentioned in Section 3.4, this could also lead to further results on graphs with bounded treewidth. Provided that the conjecture holds for $l = 3$ it should be possible to prove the conjecture for graphs of treewidth at most 4 in a similar manner as in the proofs for graphs of treewidth at most 2 and 3.

As a concluding remark, we strongly believe that the Beineke Harary Conjecture holds in its form presented in the introduction of this chapter.

Chapter 4.

2.5-connectivity

In this chapter we discuss separators containing a single vertex and a single edge — vertex-edge separators. We prove that any 2-connected graph canonically decomposes into a set of graphs, such that all resulting graphs are either cycles or do not contain vertex-edge separators. We show that this decomposition induces a tree structure and both, the graphs and the tree structure, can be computed in linear time.

This chapter is strongly based on Sections 2, 3 and 4 of our publication [Hei+20a]. The methodology used in [Hei+20a] somewhat differs from the one we use here, though the results remain largely unchanged. We comment on differences to [Hei+20a], when applicable. All of the results of this chapter are joint work with Irene Heinrich, Till Heller and Eva Schmidt.

It is well known that any connected graph uniquely decomposes into its blocks and there exists a unique block-cutpoint tree whose nodes are cut vertices and blocks, cf. Section 2.5 of Chapter 2. Hopcroft and Tarjan established a similar results for blocks: Any connected graph without a cut vertex decomposes into its *triconnected components*, which are multiedges, cycles or 3-connected graphs, cf. [HT72]. In the same publication they proved that there exists a unique tree structure on the triconnected components, which serves as a build-up manual for the original graph. This tree structure is more commonly known as SPQR-trees, which were introduced by Di Battista et al. in [BT89]. Hopcroft and Tarjan also claimed linear time computability of the triconnected components of a connected graph without a cut vertex, though their proof contained minor errors. These were corrected by Gutwenger and Mutzel in [GM01], who also argued that computing the SPQR-tree of a connected graph without a cut vertex can be implemented to run in linear time.

Triconnected components and SPQR-trees have several applications. Most of the applications are in the field of graph drawing. For example, triconnected components can be used for the recognition of planar graphs, cf. [BSW70], or the representation of all planar embeddings of a 2-connected planar graph.

More recently Grohe described a decomposition of a graph into *quasi* 4*-connected components*, cf. [Gro16]. This decomposition refines the decomposition into triconnected components. Further Grohe establishes structural results which strongly connects the decomposition into quasi 4-connected components to certain tree decompositions of a graph.

Here we introduce another set of components of a graph, that, in some sense, lies between the blocks and the triconnected components of a graph. The blocks of a connected graph can be defined by exhaustively dividing the graph at its cut vertices. The triconnected components of a graph can essentially are essentially defined by dividing the graph at 2-vertex separator. In order to obtain the components discussed here we split up the graph at (or rather along) vertex-edge separators. Vertex-edge separators are used to characterize the class of Eulerian graphs that decompose into a unique number of cycles in [HS19], cf. Chapter 5. For a summary on mixed separators we refer to the introduction of Chapter 3 or the chapter on mixed connectivity in [BWO12].

Outline In the first section of this chapter we repeat some of the results and the notation from Hopcroft and Tarjan in [HT72], as we make use of them throughout this chapter. Afterwards, in Section 4.2, we introduce vertex-edge separators and 2.5-connected components and gather some basic facts surrounding the topic. In Section 4.3 we state results on the behavior of 2-edge separator in a graph, when splitting along vertex-edge separators. In Section 4.4, we prove the uniqueness of the regarded 2.5-connected components and, afterwards, in Section 4.5 we argue that the components may be computed in linear time. Finally, we give formal proofs for the statements in Section 4.3 and consider the behavior of general separators in Section 4.6.

4.1. Triconnected Components

In this section we repeat some of the notation and results from Hopcroft and Tarjan in [HT72]. Our definitions regarding 2.5-connected components are strongly based on this.

Let G be a connected graph without a cut vertex. For two distinct vertices $v, w \in V(G)$ we define a relation on the edges of G: Two edges e_1 and e_2 relate if there exists a path P in G with $e_1, e_2 \in E(P)$ that does not contain v or w as an interior vertex. This defines an equivalence relation on $E(G)$. We call the equivalence classes $E_1, \dots, E_k$ of the relation the *separation classes* of G with respect to $\{v, w\}$. Examples of separation classes can be seen in Figure 4.1 and 4.2. The pair $\{v, w\}$ is then called *separation pair* if

(i) $k = 2$ and $\min \{|E_1|, |E_2|\} \geq 2$,

(ii) $k = 3$ and not $|E_1| = |E_2| = |E_3| = 1$, or

(iii) $k > 3$.

Examples of subsets of vertices that do not form separation pairs can be seen in Figure 4.1 and examples for sets that are separation pairs can be seen in Figure 4.2. It is worth noting, that a separation pair $\{v, w\}$ does not necessarily separate a 2-connected graph G in the sense that $G-v-w$ has more than one component, cf. Figure 4.2 c). If the two vertices of a separation pair are connected by multiple edges it is possible that the deletion of this separation pair does not lead to a graph with multiple components. However, if the graph $G - v - w$ has more than one component, $\{v, w\}$ is always a separation pair.

Let now $\{v, w\}$ be a separation pair of a graph G with separation classes $E_1, \dots, E_k$. Further, let $E' := \bigcup_{i \in I} E_i$ for some $I \subseteq \{1, \dots, k\}$, such that E', as well as $E(G) \setminus E'$ contain at least 2 edges. We

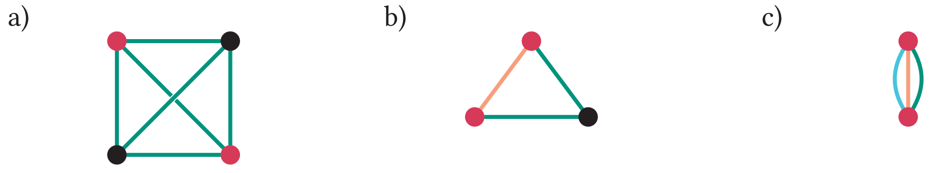

Figure 4.1.: Examples of separation classes in graphs with respect to the colored vertices. Same colored edges are contained in the same separation class.

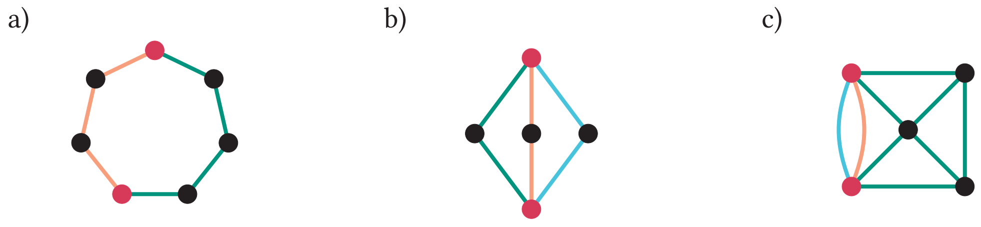

Figure 4.2.: Examples of separation pairs in graphs. The colored vertices form the separation pair. Same colored edges are contained in the same separation class.

call the two graphs $G_1 := G[E'] + vw$ and $G_2 := G[E \setminus E'] + vw$ *split graphs* of G with respect to the separation pair $\{v, w\}$. The two additional edges in $E(G_1)$ and $E(G_2)$ that are not contained in $E(G)$ are called *virtual edges*. Let $\mathcal{G}$ be a set of graphs in which no two graphs have a common edge. Replacing $G \in \mathcal{G}$ by the split graphs G_1 and G_2 corresponding to some separation pair contained in G, is called a *split* on $\mathcal{G}$. The *result* of this split is the set $\mathcal{G}' := \mathcal{G} \setminus \{G\} \cup \{G_1, G_2\}$. If $\mathcal{G} = \{G\}$ for some graph G, by abuse of notation we also say split on G instead of $\{G\}$. For an example of different splits on a graph see Figure 4.3.

Any pair of virtual edges e_1, e_2 is assigned a unique label, when created. We call e_1 the *split partner* of e_2. With this we define the backwards operation of a split. Given two graphs G_1 and G_2 with virtual edges $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$ having the same label, we call the graph $G = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2) \setminus \{e_1, e_2\})$ the *merge graph* of G_1 and G_2. Analogously to a split, replacing two graphs by their merge graph, is called a *merge*.

The sequence $s_1, \dots, s_k$ is a sequence of splits (and merges) on $\mathcal{G}$ if s_1 is a split (or a merge) on $\mathcal{G}$ and s_i is a split(or a merge) on the result of s_{i-1} for all $i \in \{2, \dots, k\}$. The result of a sequence of splits (and merges) is the result of the last split (merge) in the sequence. A set of *split components* $\mathfrak{S}$ of G is the result obtained from a sequence of splits on G, such that no graph $G' \in \mathfrak{S}$ contains a separation pair. Each graph in a set of split components is either a multiedge containing exactly 3-edges, a triangle or a 3-connected graph.

Let G be a connected graph without a cut vertex and let $\mathcal{G}$ be a set of split components of G. Perform a sequence of merges on $\mathcal{G}$, such that each merge in the sequence replaces two graphs that are either both cycles or both multiedges. If the result $\mathcal{X}$ of the sequence of merges does not allow

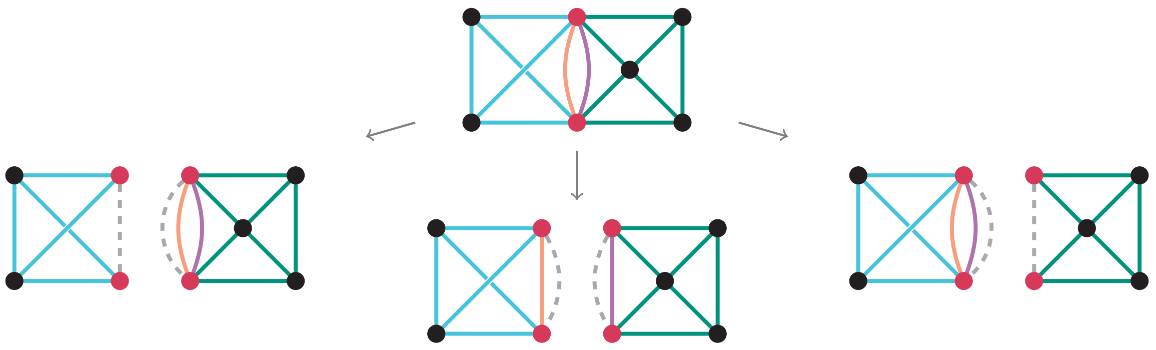

Figure 4.3.: Possible splits at the separation pair consisting of the colored vertices. Dashed edges are virtual. Edges with the same color belong to the same separation class if solid and are split partners else.

for another merge of this form, $\mathcal{X}$ is the set of *triconnected components* of G. The next theorem justifies saying the *triconnected components*.

Theorem 4.1 ([HT72]). *The triconnected components of a graph are unique.*

For a connected graph G without a cut vertex and the result $\mathcal{G}$ of a sequence of splits and merges on G Hopcroft and Tarjan [HT72] introduced the graph $T_{\text{spl}}(\mathcal{G})$ with

$$V(T_{\text{spl}}(\mathcal{G})) = \mathcal{G}$$
$$E(T_{\text{spl}}(\mathcal{G})) = \{GH\colon G \text{ and } H \text{ contain a virtual edge with the same label}\}\,.$$

Here we refer to $T_{\text{spl}}(\mathcal{G})$ as the *split tree* of $\mathcal{G}$. The following lemma is a keystone in the proof of the uniqueness of the triconnected components.

Lemma 4.2 ([HT72]). *Let G be a connected graph without a cut vertex and denote by $\mathcal{G}$ the result of a sequence of splits and merges on G. Then the following holds true.*

(i) The split tree $T_{\text{spl}}(\mathcal{G})$ is a tree.

(ii) $\mathcal{G}$ is the result of a sequence of splits on G.

As the term *uniqueness* may lead to some misunderstanding we discuss it in the following. Let $\mathcal{X}_1, \mathcal{X}_2$ be two sets of triconnected components of the same graph G. The only thing that possibly differs in $\mathcal{X}_1$ and $\mathcal{X}_2$ are the *labels* of the virtual edges. In particular we can turn $\mathcal{X}_1$ into $\mathcal{X}_2$ only by *relabeling* split partners in graphs of $\mathcal{X}_1$, i.e. replacing split partners with a new pair of virtual edges having the same endvertices as before. That being said, we may canonically label the virtual edges in a set of triconnected components in order to get *the* set of *triconnected components*. Note that this uniqueness of the components is far from obvious. The split components of a graph, for example, are not unique in this sense. To see this regard a cycle on four vertices. It can be split in two different ways leading to two different sets of split components. Note that this form of uniqueness is stronger than uniqueness up to isomorphism. When we discuss uniqueness of 2.5-connected components in Section 4.4 the term uniqueness should be understood in the same way. For an example of the triconnected components of a graph see Figure 4.4.

This concludes our summary of the results from Hopcroft and Tarjan in their paper [HT72]. We are now ready to turn to vertex-edge separators in the next section.

4.2. Vertex-Edge Separators

In this section we formally introduce vertex-edge separators and 2.5-connectivity. We give some basic results and facts on these kind of separators, establish a connection to separation pairs and finally introduce 2.5-components of a graph.

Definition 4.3. Let G be a 2-connected graph. We call the pair (v, e) a vertex-edge separator in G if $G - v - e$ is not connected. If there does not exist a vertex-edge separator in G, then we call G 2.5-connected.

Lemma 4.4. *Let G be a 2-connected graph containing a vertex-edge separator (v, uw). Then, the graph $G - v - uw$ has exactly two components.*

Proof. As G is 2-connected, the graph $G-v$ is connected. Deleting an edge from a connected graph results in a graph with no more than 2 components. As $G - v - uw$ is not connected it has exactly two components. □

Lemma 4.5. *Let G be a 2-connected graph and let $uw \in E(G)$. Then $G - uw$ is connected and the block-cutpoint tree T of $G - uw$ is a path.*

Proof. Let $uw \in E(G)$ be arbitrary. If u and w are contained in the same block of $G - uw$, the block-cutpoint tree T of $G - uw$ consists of a single block and the claim holds. So assume u and w are contained in two distinct blocks B_1, B_2 in $G - uw$. Then the block-cutpoint tree of G can be obtained by replacing all nodes on the distinct B_1-B_2 path in T by the union of all blocks on that path with the additional edge uw. As G is 2-connected its block-cutpoint tree consists of a single block and T is a path. □

Lemma 4.6. *Let G be a 2-connected graph that is not a triangle and let (v, uw) be a vertex-edge separator in G. If $v, w_1, \dots, w_k$ are the cut vertices of $G - uw$, then any two distinct vertices $x, y \in \{v, u, w, w_1, \dots, w_k\}$, such that xy is not uw and $G[xy]$ is not a block of $G - uw$, yield a separation pair (x, y) of G. In particular, $\{v, u\}$ or $\{v, w\}$ is a separation pair.*

Proof. By Lemma 4.5 the block-cutpoint tree of $G - uw$ is a path P. Let $B_1, \dots B_l$ be the blocks of $G - uw$ ordered in their appearance in P and let B_{l+1} be the path consisting of the single edge uw. Let $x, y \in \{u, v, w, w_1 \dots, w_k\}$ such that $xy \neq uw$ and $G[xy]$ is not a block in $G - uw$. If $x, y \in B_i$ for some $i \in \{1, \dots, l + 1\}$, by definition of x and y, the block B_i consists of more than one edge and $\bigcup_{j \neq i} E(B_j)$ is a separation class of G with respect to $\{x, y\}$. Thus, $\{x, y\}$ is a separation pair in G. Otherwise, each separation class of G with respect to $\{x, y\}$ contains the edges of at least two

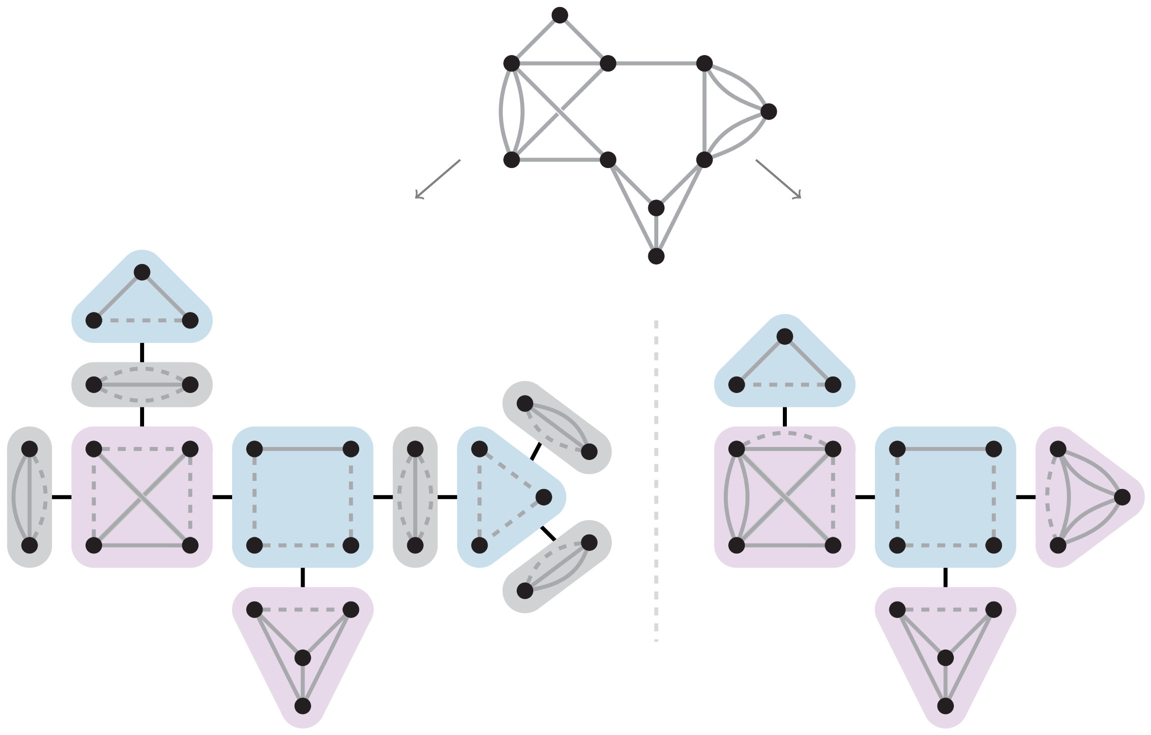

Figure 4.4.: On the left, the triconnected components of the above graph. On the right the 2.5-connected components of the graph. Dashed lines are virtual edges.

graphs in $\{B_1, \ldots, B_{l+1}\}$ and hence $\{x, y\}$ is again a separation pair. To see that $\{v, u\}$ or $\{u, w\}$ is a separation pair we observe that not both vu and vw can be a block in $G - uw$, as then G would be a triangle. □

Lemma 4.6 suggests that for each vertex-edge separator in a 2-connected graph G (not a triangle) we have at least one related separation pair. Now the main idea for the definition of 2.5-components is to split graphs along separation pairs, that are related to a vertex-edge separator in the sense of Lemma 4.6. Let us formalize what we just described.

Definition 4.7. Let G be a 2-connected graph and let $\{v, w\}$ be a separation pair in G. We call two split graphs of G with respect to $\{v, w\}$ 2.5-*split graphs* if one of the created virtual edges is contained in a 2-edge separator in its respective split graph. Any split that replaces a graph by 2.5-split graphs is called *supporting split* or 2.5-*split*. Any edge that is contained in a 2-edge separator with the created virtual edge is said to *support* the separation pair and the 2.5-split. We refer to splits that are not supporting as *non-supporting*.

A set of 2.5-*split components* $\mathfrak{S}$ of G is the result obtained from a sequence of supporting splits on G that does not allow for another supporting split. A set of 2.5-*connected components* of G is the result $\mathcal{Y}$ of a sequence of merges on a set of 2.5-split components, such that no two different

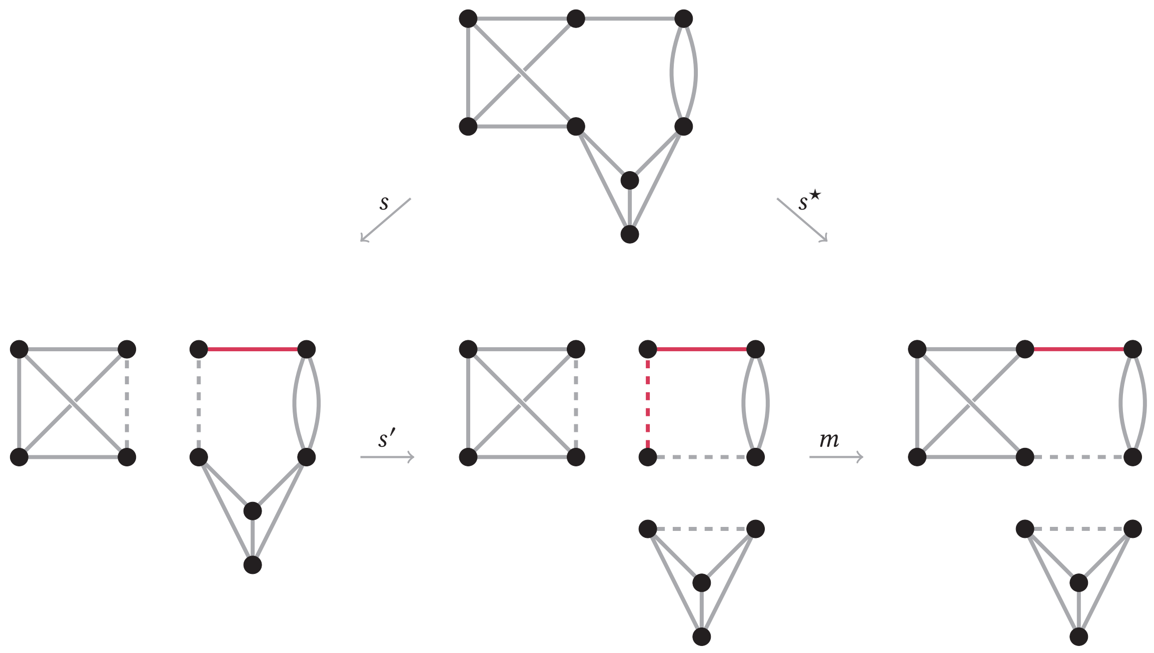

Figure 4.5.: A sequence of strongly supporting splits and merges on a graph. The result cannot be obtained by a sequence of strongly supporting splits only. Dashed lines are virtual edges. Colored edges indicate possible supports of the split leading to the displayed set of graphs.

cycles in $\mathcal{Y}$ can be merged, i.e. no two split partners are both contained in cycles of $\mathcal{Y}$.

Observe that in a 2-connected graph that is not a triangle each of the separation pairs from Lemma 4.6 induces a supporting split. Regard a set of 2.5-split components $\mathfrak{S}$ of a 2-connected graph G. Then each graph in $\mathfrak{S}$ is either a triangle, a multiedge with at least 3 edges, or a 2.5-connected graph. In particular, if a graph in $\mathfrak{S}$ is not a triangle, then it does not contain a vertex-edge separator. The same holds for a set of 2.5 components except that these may contain arbitrary large cycles instead of only triangles. For an example of 2.5-connected components of a graph G, see Figure 4.4.

In [Hei+20a], we defined 2.5-splits and the according graphs in a slightly different manner. For a split to be supporting the according separation pair had to be of the form $\{v, w\}$, where (v, uw) is a vertex-edge separator for some vertex u. The required 2-edge separator is formed by uw and one of the virtual edges created by the split. Here we refer to splits of this kind as *strongly supporting splits*. We use the former definition here, as it allows for more freedom when splitting the graph and simplifies some of the statements. In particular a result of an arbitrary sequence of strongly supporting splits and merges can in general not be obtained by a sequence of strongly supporting splits only, as Example 4.8 shows. For supporting splits this is possible as we see later on.

Example 4.8. Regard Figure 4.5. The splits s and s' indicate strongly supporting splits. The graph obtained by the merge m can however not be obtained by a strongly supporting split. The split $s^\star$ in the figure is supporting.

However, we prove that both approaches define the same set of 2.5-connected components, cf. Theorem 4.23. In order to prove the uniqueness of 2.5-connected components the transference of 2-edge separators along splits is of importance. This is what we consider in the next section.

4.3. 2-Edge Separators in Split Graphs

This section is dedicated to the behavior of 2-edge separators along splits. For Lemmata 4.9-4.11, we can formulate versions for general separators. As the statements and proofs of these are quite technical, we move them to Section 4.6 at the end of this chapter and merely state the versions which are needed for the proof of the uniqueness here. This facilitates following the proofs in the next section.

Lemma 4.9 (cf. Lemma 4.26). *Let G be a 2-connected graph and let G_1, G_2 be two split graphs of G with virtual edges $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$. If $\{e, e'\}$ is a 2-edge separator in G, then e and e' are both contained in 2-edge separators in G_1 or G_2.*

In particular if $e, e' \in E(G_i)$ for some i, $\{e, e'\}$ is a 2-edge separator in G_i. Otherwise, if $e \in E(G_1)$, $\{e, e_1\}$ and $\{e', e_2\}$ are 2-edge separators in G_1, respectively G_2.

Lemma 4.10 (cf. Lemma 4.27). *Let G be a 2-connected graph and let G_1, G_2 be two split graphs of G with virtual edges $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$. If $\{e, e'\}$ is a 2-edge separator in G_1 and $e_1 \notin \{e, e'\}$, then $\{e, e'\}$ is a 2-edge separator in G.*

Lemma 4.11 (cf. Lemma 4.28). *Let G be a 2-connected graph and let G_1, G_2 be two split graphs of G with virtual edges $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$. If $\{e, e_1\}$ and $\{e', e_2\}$ are 2-edge separators in G_1 , respectively G_2, then $\{e, e'\}$ is a 2-edge separator in G.*

For non-supporting splits and 2-edge separators we can even go further. The following lemma basically states, that a graph G has no 2-edge separator with one edge in each split graph, if the corresponding split is non-supporting. As this lemma does not generalize to arbitrary separators we prove it here.

Lemma 4.12. *Let G be a 2-connected graph and let G_1, G_2 be two split graphs of G, that are not 2.5-split graphs of G. Then e, e' is a 2-edge separator in G if and only if it is a 2-edge separator in G_1 or G_2.*

Proof. Let $\{e, e'\}$ be a 2-edge separator in G. Suppose that $e \in E(G_1)$ and $e' \in E(G_2)$. By Lemma 4.9, $\{e, e_1\}$ is a 2-edge separator in G_1, where e_1 denotes the virtual edge in G_1. As G_1, G_2 are not 2.5-split graphs of G, this is a contradiction. Otherwise $e, e' \in E(G_i)$ for some $i \in \{1, 2\}$ and again by Lemma 4.9, we get that $\{e, e'\}$ is a 2-edge separator in G_1 or G_2.

Now assume that $\{e, e'\}$ is a 2-edge separator in G_1. Denote by e_1 the virtual edge in $E(G_1)$. As G_1 and G_2 are not 2.5-split graphs $e_1 \notin \{e, e'\}$. By Lemma 4.10 we get that $\{e, e'\}$ is a 2-edge separator in G. □

4.4. Uniqueness of 2.5-connected Components

The main idea in the proof of the uniqueness of 2.5-connected components of a 2-connected graph G is to prove that any set of 2.5-connected components can be obtained from the triconnected components of G by merging the same set of virtual edges. Before we prove this, we establish some lemmata on supporting and non-supporting splits.

Lemma 4.13. *Let H be a 2-connected graph and let $t_1, \dots, t_l, s_1, \dots, s_k$ be a sequence of supporting splits on H resulting in the set $\mathcal{G}$. Let m be the merge on $\mathcal{G}$ along the virtual edges created by s_1. The result of m can be obtained by a sequence of supporting splits on H.*

Proof. Denote by $\mathcal{G}'$ the result of the sequence $t_1, \dots, t_l$. Technically the sequence of splits $s_2, \dots s_k$ may not be applied to $\mathcal{G}'$ as s_2 operates on the result of s_1. Observe however, that we may use a very similar sequence that operates on $\mathcal{G}'$: Regard the tree $T_{\mathrm{spl}}(\mathcal{G})$, cf. Lemma 4.2. The merge m corresponds to a contraction of the corresponding edge in $T_{\mathrm{spl}}(\mathcal{G})$. Denote by T' the resulting tree. Now merge along all edges in T' that correspond to some split s_i for $i \geq 3$. The remaining nodes are the result of a split s'_2 on $\mathcal{G}'$ with the same virtual edges as were created by s_2. This can be repeated for every split s_i, $i \geq 3$. We get a sequence $s'_2, \dots, s'_k$ which results in the same set of graphs as m. It remains to be shown that s'_i is supporting for all $i \geq 2$.

Denote by $e_1^\star, e_2^\star$ the virtual edges created by s_1 and let H_1, H_2 be the corresponding split graphs with $e_1^\star \in E(H_1)$. As s_1 is supporting, at least one of the two virtual edges is contained in a 2-edge separator in its corresponding split graph. Thus, we may assume that

$$e_1^\star \text{ is contained in a 2-edge separator in } H_1. \tag{4.1}$$

Let $i \geq 2$, let G be the graph that is replaced by s_i with the two split graphs G_1 and G_2, and denote by $e_i \in E(G_i)$ for $i \in \{1, 2\}$ the virtual edges created by the split. Further let G' be the graph that s'_i replaces by the two split graphs G'_1, G'_2. Note that the virtual edges created by s'_i are also e_1 and e_2. We assume $e_1 \in E(G'_1)$ and $e_2 \in E(G'_2)$. As s_i is supporting, e_1 or e_2 is contained in a 2-edge separator in G_1 or G_2. Without loss of generality assume that e_1 is contained in a 2-edge separator $\{e_1, e\}$ in G_1. If neither $e_1^\star$ nor $e_2^\star$ are contained in $E(G_1)$, then $G_1 = G'_1$ and we are done. So assume that one of them is contained in $E(G_1)$. Then G'_1 is the merge graph of G_1 and some other graph $G^\star$. Further it is $G_2 = G'_2$.

First assume that $e \notin \left\{e_1^\star, e_2^\star\right\}$. As $\{e_1, e\}$ is a 2-edge separator in G_1, by Lemma 4.10, it is also one in G'_1 and s'_i is supporting.

Next assume that $\left\{e_1, e_1^\star\right\}$ is a 2-edge separator in G_1. By (4.1), $e_1^\star$ is contained in a 2-edge separator in H_1. By Lemma 4.9 it is still contained in a 2-edge separator $\left\{e_1^\star, e^\star\right\}$ in G. If $e^\star \in E(G_2)$, by Lemma 4.9, $\left\{e^\star, e_2\right\}$ is a 2-edge separator in $G_2 = G'_2$ making s'_i supporting. Otherwise $e^\star \in E(G_1)$ and by Lemma 2.2, $\left\{e^\star, e_1\right\}$ is a 2-edge separator in G_1. Using Lemma 4.10 we get that $\left\{e^\star, e_1\right\}$ is a 2-edge separator in G'_1 making s'_i supporting.

Finally assume that $\left\{e_1, e_2^\star\right\}$ is a 2-edge separator in G'_1. By (4.1) and Lemma 4.9, $e_1^\star$ is contained in a 2-edge separator $\left\{e_1^\star, e^\star\right\}$ in $G^\star$. Further, by Lemma 4.11, $\left\{e^\star, e_1\right\}$ is a 2-edge separator in G'_1 again making s'_i supporting. □

Lemma 4.14. *Let G be a 2-connected graph and denote by $\mathcal{G}$ the result of a sequence of supporting splits and merges on G. Then the following holds true.*

(i) The split tree $T_{\text{spl}}(\mathcal{G})$ is a tree.

(ii) $\mathcal{G}$ is the result of a sequence of supporting splits on G.

Proof. As all 2.5-splits are splits, it is immediate from Lemma 4.2, that $T_{\text{spl}}(\mathcal{G})$ is a tree. Regard an arbitrary sequence of splits and merges. By Lemma 4.13, the result of the first merge in the sequence can be obtained by a sequence of supporting splits only. The claim follows by induction on the number of merges in the sequence. □

Lemma 4.15. *Let G be a 2-connected graph and let $e \in E(G)$, such that there does not exist a split on G that is supported by e. Then e does not support any split on any split graph of G.*

Proof. Let s be any split on G resulting in the the set $\{G_1, G_2\}$ with $e \in E(G_1)$ creating virtual edges $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$. We prove the statement by contraposition. So assume s' is a split on $\{G_1, G_2\}$ that is supported by e resulting in the set $\left\{G_1', G_2', G_2\right\}$ and creating the virtual edges $e_1' \in E(G_1')$, $e_2' \in E(G_2')$. Then, without loss of generality, $\left\{e, e_1'\right\}$ is a 2-edge separator in G_1'. The merge on $\left\{G_1', G_2', G_2\right\}$ along the virtual edges e_1 and e_2 now results in some set $\left\{G_1^\star, G_2^\star\right\}$, where $\left\{G_1^\star, G_2^\star\right\}$ is the result of a split $s^\star$ on $\{G\}$. As $\left\{e, e_1'\right\}$ is a 2-edge separator in G_1', by Lemma 4.10, $\left\{e, e_1'\right\}$ is a 2-edge separator in $G_1^\star$. This implies, that e supports the split $s^\star$. □

Lemma 4.16. *Let $\mathcal{Y}$ be a set of 2.5-connected components of G. There exists a sequence of splits on $\mathcal{Y}$, such that the result of the sequence are the triconnected components of G.*

Proof. Regard a sequence of splits s on $\mathcal{Y}$ leading to some set of split components of G and let m be a sequence of merges leading to the triconnected components of G. We claim, that any merge in m corresponds to a split in s.

Suppose this is not the case, i.e. there exists a merge m' in m that corresponds to a supporting split s' that is not contained in s. Denote by G_1, G_2 the split graphs corresponding to the split s' and let $e_1 \in E(G_1)$, $e_2 \in E(G_2)$ be the two virtual edges created by s'. Further, denote by G_1', G_2' the two graphs that are merged by m' with $e_1 \in E(G_1')$. As s' is a supporting split, without loss of generality we may assume that e_1 is contained in a 2-edge separator in G_1. By Lemma 4.9, e_1 is also contained in a 2-edge separator in G_1'. As no multiedge with 3 or more edges contains a 2-edge separator and only multiedges and cycles are merged in m, this implies that G_1' and G_2' are cycles and therefore

$$e_2 \text{ is contained in a 2-edge separator in } G_2'. \tag{4.2}$$

Denote by $G_1^\star, G_2^\star \in \mathcal{Y}$ the graphs in the given set of 2.5-components containing the edges e_1 and e_2. Again by Lemma 4.9, e_1 is contained in a 2-edge separator in $G_1^\star$. The only graphs with 2-edge separators in $\mathcal{Y}$ are cycles and no two cycles in $\mathcal{Y}$ share a virtual edge with the same label. Thus, $G_1^\star$ is a cycle and $G_2^\star$ does not contain a 2-edge separator. As $G_2^\star \in \mathcal{Y}$, there are no 2.5-split graphs of $G_2^\star$ and e_2 does therefore not support any split on $G_2^\star$. By Lemma 4.15, this implies that e_2 is not contained in a 2-edge separator in G_2' yielding a contradiction to (4.2). □

Lemma 4.17. *Let G be a 2-connected graph. Regard some sequence of splits $s = s_1 \dots s_k$ on G such that the result of the sequence are the triconnected components of G and let $i \in \{1, \dots, k\}$. Out of any two split partners in the result of $s_1 \dots s_i$ at most one edge is contained in a 2-edge separator in its graph.*

Proof. Suppose that there are split partners e_1 and e_2 in the result of $s_1, \dots, s_i$ that are both contained in a 2-edge separator. By Lemma 4.9, both of these are still contained in a 2-edge separator in their corresponding triconnected component. The only triconnected components containing 2-edge separators are cycles. Thus, in the triconnected components of G, there exist two virtual edges with the same label, that are both contained in a cycle. This contradicts the definition of triconnected components. □

We now turn to the actual uniqueness result. We use a coloring of the virtual edges. As we shall make use of this later on we separately define the coloring for arbitrary sequences of splits.

Definition 4.18. Let G be a 2-connected graph and let $s = s_1, \dots, s_k$ be a sequence of splits on G resulting in the set $\mathcal{G}$. For each $i \in \{1, \dots, k\}$ we assign a color from $\{\textsc{red}, \textsc{green}\}$ to the virtual edges e_1 and e_2 created by the split s_i, that may depend on colors of virtual edges created in previous splits. We denote the color of a virtual edge e by $\text{col}(e) \in \{\textsc{red}, \textsc{green}\}$.

(i) If s_i is a non-supporting split, set $\text{col}(e_1) = \text{col}(e_2) = \textsc{red}$.

(ii) If s_i is a supporting split which is supported by a non-virtual or a virtual, GREEN edge, set $\text{col}(e_1) = \text{col}(e_2) = \textsc{green}$.

(iii) Otherwise, set $\text{col}(e_1) = \text{col}(e_2) = \textsc{red}$.

We call the coloring of the virtual edges of the graphs in $\mathcal{G}$ obtained in this way the 2.5-*coloring* induced by s.

Theorem 4.19. *Let $s = s_1, \dots, s_k$ be a sequence of splits on a 2-connected graph G resulting in the triconnected components of G. Denote by* col *the 2.5-coloring induced by s. For any virtual edge e in the triconnected components of G, it holds true that $\text{col}(e) = \textsc{green}$ if and only if the triconnected component of e or its split partner is a cycle that contains at least one non-virtual edge.*

In particular, the 2.5-coloring of the virtual edges in the triconnected components does not depend on the choice of the sequence leading to the triconnected components.

Proof. For $i \in \{0, \dots, k\}$ regard the subsequence $s(i) = s_1, \dots, s_i$ resulting in the set $\mathcal{G}_i$. We prove the following.

Claim 1. Let e be a virtual edge in the result of the sequence $s(i)$. It holds true that $\text{col}(e) = \textsc{green}$ if and only if e or its split partner is contained in a 2-edge separator with a non-virtual edge in some graph $G \in \mathcal{G}_i$.

To see that Claim 1 implies the theorem, recall that in the triconnected components of G the only graphs containing 2-edge separator are cycles.

In the following, if a virtual edge fulfills the condition of Claim 1, we say that its color is *valid*. We prove Claim 1 by induction on i. For $i = 0$ the claim is trivially fulfilled, as there are no virtual edges created yet. So let $i \geq 1$, denote by $G \in \mathcal{G}_{i-1}$ the graph that is split by s_i and let G_1 and G_2 be the according split graphs, where we denote the virtual edges created by s_i by $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$. Note that we only have to validate the colors of the virtual edges in $E(G_1) \cup E(G_2)$ (and the colors of their split partners). The colors of all other virtual edges remain valid, as their respecting graphs are unchanged by the split s_i.

First assume s_i is a non-supporting split and we assign the color RED to e_1 and e_2. As s_i is non-supporting, neither e_1 nor e_2 are contained in a 2-edge separator in G_1 or G_2. Further, by Lemma 4.12, any 2-edge separator in G_1 or G_2 is also a 2-edge separator in G and vice versa. Therefore, the previously defined colors remain valid, by induction.

So from now on assume that s_i is a supporting split. By Lemma 4.17, not both, e_1 and e_2, are contained in 2-edge separators in their respective graphs G_1 and G_2. Thus, without loss of generality we may assume that

$$e_1 \text{ is contained in a 2-edge separator in } G_1, \text{ whereas } e_2 \text{ is not in } G_2. \tag{4.3}$$

As by Lemma 4.9, any 2-edge separator $\{e'_1, e'_2\}$ in G, such that $e'_1 \in E(G_1)$ and $e'_2 \in E(G_2)$, would imply that $\{e_2, e'_2\}$ is a 2-edge separator in G_2, we may assume by Lemma 4.9 that

$$\text{any 2-edge separator in } G \text{ is also a 2-edge separator in } G_1 \text{ or } G_2. \tag{4.4}$$

The last assumption we can make, is due to Lemma 4.10:

$$\text{Any 2-edge separator in } G_i, \text{ that does not contain } e_i \text{ is a 2-edge separator in } G. \tag{4.5}$$

We begin by arguing that all previously defined colors remain valid. Regard some virtual edge $e \in E(G)$ and denote by e' its split partner. If e is colored GREEN, by induction, e or e' are contained in a 2-edge separator with a non-virtual edge in their graphs in $\mathcal{G}_{i-1}$. If e is contained in a 2-edge separator with a non-virtual edge in G, by (4.4) it is contained in a 2-edge separator with a non-virtual edge in G_1 or G_2. Further, the graph of e' in $\mathcal{G}_{i-1}$ remains unchanged by s_i. Thus, the colors of e and e' remain valid. If e is colored RED, by induction, neither e nor e' are contained in a 2-edge separator with a non-virtual edge. Any 2-edge separator containing a non-virtual edge and e in G_1 or G_2 would also be a 2-edge separator in G by (4.5). As the graph in $\mathcal{G}_{i-1}$ containing e' remains unchanged by the split the color of e and e' remain valid.

It remains to validate the colors assigned to e_1 and e_2. Recall, that by (4.3) e_2 is not contained in a 2-edge separator in G_2. If e_1 is not contained in 2-edge separators with GREEN virtual or non-virtual edges, we assign the color RED to e_1 and e_2, which is clearly valid. If e_1 is contained in a 2-edge separator with a non-virtual or GREEN edge $e^\star$ we set $\text{col}(e_1) = \text{col}(e_2) = \text{GREEN}$. If $e^\star$ is non-virtual, the coloring is valid. So assume $e^\star$ is a green virtual edge. As $e^\star$ is in a 2-edge separator with e_1, by Lemma 4.17, the split partner of $e^\star$ is not contained in a 2-edge separator. Thus, by induction, there exists some non-virtual edge $e \in E(G)$ such that $\{e^\star, e\}$ is a 2-edge separator in G. By (4.4), this is also a 2-edge separator in G_1 and, by Lemma 2.2, $\{e_1, e\}$ is a 2-edge separator. Thus, all colors assigned are valid. □

Theorem 4.19 almost immediately implies the uniqueness of the 2.5-connected components. Recall that an *ear* in a graph G is a path P in G, such that all internal vertices of P are of degree 2 in G and P is not a proper subgraph of another such path. Note that each edge $e \in E(G)$ is contained in some ear of G. If an ear only contains one edge, we refer to it as *trivial ear*, otherwise it is called *non-trivial.*

Theorem 4.20. *Let G be a 2-connected graph. The 2.5-connected components of G are unique. In particular, the 2.5-connected components of G can be obtained from the triconnected components by merging along split partners that are both contained in a trivial ear or in a cycle consisting solely of virtual edges.*

Proof. Regard a sequence of 2.5-splits s on G leading to some set of 2.5-connected components $\mathcal{Y}$ of G. Such a sequence exists by Lemma 4.14. Now let s' be a sequence of splits on $\mathcal{Y}$ leading to the triconnected components of G. This sequence exists by Lemma 4.16. Denote by col the 2.5-coloring induced by the sequence ss'. As the graph G does not contain any virtual edges and all splits in s are supporting, any split of the sequence s only creates virtual edges that are colored GREEN. Note that no edge contained in any graph $H \in \mathcal{Y}$ supports a split on H. By Lemma 4.15, none of these edges support any split in s'. Thus, any split in s' is either non-supporting or only supported by virtual edges created by s'. Therefore, all virtual edges created by s' are colored RED. We know that the triconnected components of G are unique. By Theorem 4.19, the coloring of the virtual edges is independent of the choice of the sequences s and s'. Thus, any sequence of 2.5-splits on G leading to some set of 2.5-connected components produces the same set of graphs. The 2.5-connected components are therefore unique. □

Theorem 4.20 gives away two corollaries about 2.5-connected components and the according split tree.

Corollary 4.21. *Let G be a 2-connected graph with 2.5-connected components $\mathcal{Y}$ and triconnected components $\mathcal{X}$. $T_{\text{spl}}(\mathcal{Y})$ is a minor of $T(\mathcal{X})$. For any two graphs in $V(T_{\text{spl}}(\mathcal{Y}))$ that are adjacent, it holds true that one of them is a cycle containing virtual and non-virtual edges and the other is a multiedge or a 2.5-connected graph.*

Proof. Regard the 2.5-coloring of the virtual edges in $\mathcal{X}$. Now contract all edges in $E(T_{\text{spl}}(\mathcal{X}))$ that correspond to virtual edges that are colored RED and apply the corresponding merges. The resulting graph is $T_{\text{spl}}(\mathcal{Y})$. The remainder of the claim is a direct consequence of Theorem 4.20. □

Corollary 4.22. *Let G be a 2-connected graph that is not a cycle. Then G is 2.5-connected if and only if in the triconnected components of G there does not exist a cycle containing both virtual and non-virtual edges.* □

We conclude this section by revisiting the definition of 2.5-splits we used in [Hei+20a]. Recall that here we refer to these splits as strongly supporting. By Lemma 4.6, any vertex-edge separator in a 2-connected graph G that is not a triangle induces a separation pair at which a strongly supporting split is possible. Thus, it is an easy consequence that any result $\mathcal{G}$ of a sequence of strongly supporting splits such that no further strongly supporting split is possible on $\mathcal{G}$ is a set of 2.5-split components of G. As argued before an arbitrary sequence of strongly supporting splits and merges may, in general, not be replaced by a sequence of strongly supporting splits only, cf. Example 4.8. However, the next theorem shows that the 2.5-connected components can in fact be obtained by a sequence of strongly supporting splits only, cf. [Hei+20a].

Theorem 4.23. *Let $\mathcal{G}$ be a set of graphs obtained from a 2-connected graph G by a sequence of splits and merges, such that from each pair of virtual edges with the same label exactly one of the edges is contained in a non-trivial ear. If each non-trivial ear in a graph in $\mathcal{G}$ contains a non-virtual edge, then the set $\mathcal{G}$ can be obtained from G by a sequence of strongly supporting splits. In particular, the 2.5-connected components can be obtained from G by a sequence of strongly supporting splits.*

Proof. We prove the claim by the number of graphs k in $\mathcal{G}$. By Lemma 4.2, we can obtain the set $\mathcal{G}$ by a sequence of splits $s = s_1, \ldots, s_{k-1}$ on G. We prove the claim by induction on k. If $k = 1$ the claim is trivially fulfilled. So let $k \geq 2$ and regard two split partners $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$ for graphs $G_1, G_2 \in \mathcal{G}$. By assumption for some $i \in \{1, 2\}$ we have that e_i is contained in a non-trivial ear P in G_i that contains a non-virtual edge, say e_1. Then e_2 is contained in a trivial ear. We have $|E(P)| \geq 2$. Denote by $e_1', e_2' \in E(P)$ the first respectively last edge on the path P. If one of them is virtual, say e_1', set $e' := e_1'$, otherwise set $e' := e_1$. Let s_i be the split that created the virtual edge e' and denote by m_i the merge on $\mathcal{G}$ that replaces the two graphs containing the virtual edges with the same label as e'. Then, the set $\mathcal{G}'$ that is the result of the sequence $s_1, \ldots, s_k, m_i$ of splits and merges fulfills the assumptions of the theorem and contains fewer graphs than $\mathcal{G}$. By induction $\mathcal{G}'$ is the result of a sequence of 2.5-splits on G. Further, observe that the split on $\mathcal{G}'$ that reverses m_i is also a 2.5-split, which proves the claim. □

In the next section of this chapter we briefly argue how to compute the 2.5-connected components of a 2-connected graph in linear time.

4.5. Finding 2.5-connected Components in Linear Time

Triconnected components of a graph can be found in time linear in the number of edges and vertices of a graph, cf. [GM01]. Using this and Theorem 4.20 we get a procedure that computes the 2.5-connected components in linear time: First compute the triconnected components of G. Next compute the 2.5-coloring of the triconnected components. Finally, merge all virtual edges that are colored RED.

Theorem 4.24. *For a 2-connected graph G we can compute the 2.5-connected components of G in time $O(|V| + |E|)$.*

Proof. Gutwenger and Mutzel proved, that the triconnected components of a 2-connected graph G can be found in time $\mathcal{O}(|V(G)| + |E(G)|)$. To identify the virtual, GREEN edges in the 2.5-coloring of the triconnected components we use Theorem 4.19: We iterate through all triconnected components H of G. If H is a cycle that contains non-virtual edges we color all virtual edges in H and their split partners GREEN. By Theorem 4.21 all virtual edges who are not yet colored are exactly the virtual edges which have to merged. In the implementation of Gutwenger and Mutzel the edges of the triconnected components are given as lists and according pointer between split partners are given. Testing if a graph is a cycle can be done in linear time in the number of edges and vertices of the graph, e.g., by using a depth first search. Taking into account that the sum of the number of edges and vertices of all triconnected components is contained in $\mathcal{O}(|V(G)| + |E(G)|)$, cf. [HT72], the marking of the virtual edges can be completed in time $\mathcal{O}(|V(G)| + |E(G)|)$. Merging the necessary virtual edges can again be realized in time $\mathcal{O}(|V(G)| + |E(G)|)$, as a single merge can be completed in constant time, cf. [GM01]. □

4.6. Connectivity in Split Graphs

This section is dedicated to giving formal proofs for generalized versions of the claims made in Section 4.3. Recall the definition of a *disconnecting pair* from Chapter 3, cf. Definition 3.1. If G is a connected graph, then a pair (W, F) for $W \subseteq V(G)$ and $F \subseteq E(G)$ is a disconnecting pair in G if and only if $G - W - F$ is not connected. Observe that disconnecting pairs generalize the notion of vertex-, edge- and vertex-edge separators.

At the very base of all the following lemmata is the following simple observation. It will also come in handy for the application of 2.5-connectivity discussed in Chapter 5.

Observation 4.25. *Let G be a 2-connected graph and $\mathcal{H}$ be the result of a sequence of splits on G. Fix some graph $H \in \mathcal{H}$, denote by $H_1, \dots, H_k$ the graphs obtained by merging all complete pairs of virtual edges in the components of $\mathcal{H} \setminus \{H\}$ and let for $i \in \{1, \dots, k\}$ the edge e_i be the virtual edge that is left in H_i with split partner e_i'.*

Any set of edge-disjoint paths in H can be extended to a set of edge-disjoint paths in G by replacing any virtual edge e_i' in the paths by subpaths in $H_i - e_i$ connecting the endvertices of e_i, which exist by the definition of a split.

We now turn to the transference of arbitrary separators along split graphs. Lemma 4.26 deals with the transference of separators in a graph G into its split graphs, whereas Lemma 4.27 and Lemma 4.28 regard the opposite direction. A combination of Lemma 4.26 and Lemma 4.27 can be found in a similar form in our publication [HS19].

Lemma 4.26. *Let G be a 2-connected graph and let G_1, G_2 be two split graphs of G with respect to $\{v, w\}$ and virtual edges $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$. Further, let (W, F) be a disconnecting pair in G*

and let $C_1, \dots, C_k$ be the components of $G - W - F$. Set $W_1 = W \cap V(G_1)$ and

$$F_1 = \begin{cases} F \cap E(G_1) \cup \{e_1\}, & \text{if there is no } v\text{-}w \text{ path in } G - W - F, \\ F \cap E(G_1), & \text{else.} \end{cases}$$

Then each non-empty set of vertices in $\{V(C_1) \cap V(G_1), \dots V(C_k) \cap V(G_1)\}$ induces a component of $G_1 - W_1 - F_1$. In particular, if G_1 has vertices in more than one component of $G - W - F$, the pair (W_1, F_1) is a disconnecting pair in G_1 that separates two vertices in G_1 if and only if they are separated by (W, F) in G.

Proof. Denote by $\{v, w\}$ the separation pair along which the split graphs G_1 and G_2 are created. Let $x, y \in V(G_1) \setminus W$ be two distinct vertices. We show that there exists an x-y path in $G_1 - W' - F'$ if and only if there exists an x-y path in $G - W - F$.

First assume that there exists a v-w path P' in $G - W - F$. Let P be an x-y path in $G_1 - W - F$. If the path does not contain the edge e_1 it also exists in G. Otherwise, as v and w are in the same component in $G - W - F$, x and y are also in the same component in $G - W - F$. On the other hand if P is an x-y path in G it either is completely contained in G_1 or we may replace the subpath of P contained in G_1 with the edge e_1 to get an x-y path in G_1.

Otherwise v and w are not in the same component of $G - W - F$. Then no x-y path in $G - W - F$ may use vertices not contained in G_1. Thus, any x-y path in $G - W - F$ is also an x-y path in $G_1 - W' - F'$. On the other hand any x-y path in $G_1 - W' - F'$ also exists in G as $e_1 \in F'$. □

Lemma 4.27. *Let G be a 2-connected graph and let G_1, G_2 be two split graphs of G with virtual edges $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$. If (W, F) is a disconnecting pair in G_1, with $e_1 \notin F$, then (W, F) is a disconnecting pair in G.*

Proof. If (W, F) is a disconnecting pair in G_1, there exist $x, y \in V(G_1)$ such that there is no x-y path in $G_1 - W - F$. We prove the claim by contraposition. So assume that there exists an x-y path P in $G - W - F$. Then P contains a subpath P' connecting the endvertices of e_1, as otherwise it would also exist in $G_1 - W - F$. Since $e_1 \notin F$, we can replace the subpath P' in P with the edge e_1 to get an x-y path in G_1. As G is 2-connected, we get that (W, F) is a disconnecting pair in G. □

Lemma 4.28. *Let G be a 2-connected graph and let G_1, G_2 be two split graphs of G with respect to $\{v, w\}$ and with virtual edges $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$. For $i \in \{1, 2\}$, let (W_i, F_i) be a disconnecting pair in G_i such that v and w are contained in different components of $G_i - W_i - F_i$. Then $(W, F) := (W_1 \cup W_2, F_1 \cup F_2 \setminus \{e_1, e_2\})$ is a disconnecting pair in G. In this case, if $C_1, \dots, C_k$ are the components of $G - W - F$, then each non-empty set of vertices $V(C_j) \cap V(G_i)$ induces a component of $G_i - W_i - F_i$ for $j \in \{1, \dots, k\}$ and $i \in \{1, 2\}$.*

Proof. Let $x, y \in V(G_1)$ be two vertices that are contained in different components of $G_1 - W_1 - F_1$. Any x-y path in G must contain a v-w path in $G_2 - e_2$, as it otherwise also exists in G_1. Any such v-w path contains an element from (W_2, F_2) and the claim follows. The claim on the components of $G - W - F$ is now an immediate consequence of Lemma 4.26. □

Conclusion This concludes our chapter on 2.5-connectivity. For vertex cuts consisting of one, two, or three vertices, decomposition results have been established in the past. Here we succeeded in defining decomposition steps for separators consisting of a single vertex and a single edge leading to 2.5-connected components, which, in some sense, lie between triconnected components and blocks. In analogy to Hopcroft and Tarjans triconnected components, cf. [HT72], we were able to establish uniqueness results on the 2.5-connected components and discussed the transference of separators along splits on graphs. For further research, it could be of interest to examine splits along larger mixed separators and see if some of the results transfer. For example one could regard splits such that one of the created virtual edges is contained in a k-edge separator for some $k \geq 3$.

Finally we argued that we may compute the 2.5-connected components of a 2-connected graph in linear time by using a simple extension of the algorithm by Gutwenger and Mutzel that computes triconnected components in linear time, cf. [GM01].

Chapter 5.

Eulerian Graphs Decomposing into a Unique Number of Cycles

In this chapter we discuss cycle decomposition as an application to 2.5*-connected and triconnected components. We give different characterizations for the class of Eulerian graphs that decompose into a unique number of cycles. We also argue, how these characterizations can be used to recognize a graph from this class in linear time.*

The main result of this chapter, that an Eulerian graph decomposes into a unique number of cycles if and only if it does not contain two cycles sharing three or more vertices, has been published in [HS19] and [Hei20]. The means of obtaining this characterization used in [HS19] are not presented here, as the results from Chapter 4 ([Hei+20a]) make the characterization more elegant and somewhat extend the results from [HS19]. Nevertheless the ideas of the proofs in [HS19] only needed small adjustment to fit the new characterization. All of the results of this chapter are joint work with Irene Heinrich.

Problems surrounding the decomposition of even graphs into cycles or packing cycles into an arbitrary graph have been studied in the literature before. Regarding the characterization of graphs whose cycle decompositions or packings fulfill certain properties, it is worth mentioning that Harant et al. characterized all graphs whose cyclomatic number[1] differs from the maximum size of a cycle packing by a constant, cf. [Har+10]. Regarding cycles and connectivity, Otto and Recht used the triconnected components of a 2-connected graph to get an $\mathcal{O}(\log(|V(G)|))$ approximation algorithm for the maximum cycle packing problem on a graph G. This algorithm turns out to be optimal for generalized series-parallel graphs, cf. [OR19]. Here, we regard cycle decompositions as an application to the 2.5-connectivity introduced in Chapter 4. The characterizations we give are novel and fit in nicely with the above research. So far, the connection between 2.5-connectivity and cycle decomposition problems has not been studied in the literature, except in our publications [HS19; Hei+20a].

Outline In the first section we recall the definitions of minimum and maximum cycle decomposition and observe some basic results. In Section 5.2 we briefly discuss triconnected components

[1]For a graph G the cyclomatic number is defined as $|E(G)| - |V(H)| + k$, where k denotes the number of components of G.

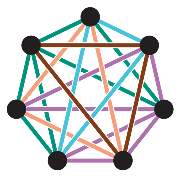
Figure 5.1.: Decomposition of the K_7 into seven triangles.

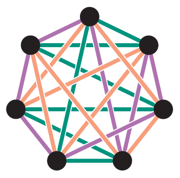
Figure 5.2.: Decomposition of the K_7 into three Hamiltonian cycles.

in connection with cycle decompositions and thereby motivate the use of 2.5-connected components. In Section 5.3 we discuss the connection of cycle decompositions and 2.5-connected components. We prove that there is a one-to-one correspondence between cycle decompositions of the 2.5-connected components of a graph and cycle decompositions of the graph itself. In the last section we characterize Eulerian graphs that decompose into a unique number of cycles using 2.5-connected components.

5.1. Cycle Decompositions: Basic Definitions and Results

It is well known that the edge set of an even graph G can be partitioned into sets $E_1, \dots, E_k$, such that $G[E_i]$ is a cycle for each $i \in \{1, \dots, k\}$, cf. [Fle90].

Definition 5.1. Let G be an even graph and let $\{E_1, \dots, E_k\}$ be a partition of the edge set $E(G)$. If $G[E_i]$ is a cycle for each $i \in \{1, \dots, k\}$ we call the set $\{G[E_1], \dots, G[E_k]\}$ a *cycle decomposition* of G. In this case we say that G can be *decomposed* into the k cycles $\{G[E_1], \dots, G[E_k]\}$.

Even graphs may have multiple cycle decompositions. In particular, two cycle decompositions of an even graph do not necessarily have the same cardinality. For example, regard the complete graph K_7 on seven vertices. This graph is Eulerian and can be decomposed into seven triangles (Figure 5.1) as well as in 3 Hamiltonian cycles (Figure 5.2).

There are two decision problems that arise from this example. For the sake of completeness we give formal definitions of both problems here.

SMALL CYCLE DECOMPOSITION (SCD).
Instance: An even graph G and an integer $B \in \mathbb{N}$.
Question: In G, does there exist a cycle decomposition with at most B cycles?

LARGE CYCLE DECOMPOSITION (LCD).
Instance: An even graph G and an integer $B \in \mathbb{N}$.
Question: In G, does there exist a cycle decomposition with at least B cycles?

Both problems are known to be NP-complete. For SCD, this is due to the fact, that it is NP-hard to decide if a directed, 4-regular graph can be decomposed into two Hamiltonian cycles, cf. [Pér84].

The NP-hardness of LCD can be proven by using the fact that deciding if the edge set of a graph can be decomposed into triangles is NP-hard, cf. [DT97]. Formal proofs for the NP-completeness of SCD and LCD can be found in Appendix B in [Hei20].

We refer to the optimization versions of the problems by Min-Cycle Decomposition, respectively Max-Cycle Decomposition and denote by $c(G)$ ($\nu(G)$) the minimum (maximum) cardinality of a cycle decomposition of an even graph G.

As a consequence of the NP-hardness results concerning SCD and LCD, there is no polynomial time algorithm solving Min-Cycle Decomposition or Max-Cycle Decomposition for general graphs, unless P = NP. Thus, it is of interest to decide if an even graph decomposes into a unique number of cycles as Min-Cycle Decomposition and Max-Cycle Decomposition are trivially solvable on this class of graphs. This is what we aim at for the remainder of this chapter. We achieve it by giving both a constructive characterization and a simple structural characterization of the class.

It is not hard to see that when computing cycle decompositions of a graph G we may essentially restrict ourselves to the blocks of a graph G. This is due to the fact that a cycle in G neither touches two different components of G, nor touches edges from different blocks of G. In particular, we get the following two observations.

Observation 5.2. *Let G be an even graph with components $D_1, \dots, D_k$ and blocks $B_1, \dots B_l$. The following two properties hold true.*

(i) $c(G) = \sum_{i=1}^{k} c(G[D_i])$ *and* $\nu(G) = \sum_{i=1}^{k} \nu(G[D_i])$,

(ii) $c(G = \sum_{i=1}^{k} c(G[B_i])$ *and* $\nu(G) = \sum_{i=1}^{k} \nu(G[B_i])$.

5.2. Cycle Decompositions in Triconnected Components

When regarding the triconnected components of a graph lemmata in this form do not come so easily. In fact, the split graphs of a given 2-connected, Eulerian graph, do not necessarily have to be Eulerian. To see this regard any two 2-connected Eulerian graphs G_1 and G_2 such that $V(G_1) \cap V(G_2) = \{v, w\}$ for distinct vertices v and w. Then $G_1 + vw$ and $G_2 + vw$ are split graphs of $G_1 \cup G_2$. Both vertices v and w are of odd degree in these split graphs. Thus, neither $G_1 + vw$ nor $G_2 + vw$ is even.

Even if the split graphs of a 2-connected graph are Eulerian G, we are in general not able to give a one-to-one correspondence between the cycle decompositions of the split graphs and the cycle decomposition of G:

Let $G := C_1 \cup \ldots \cup C_k$ for some odd integer $k \geq 3$, where $C_1, \dots, C_k$ are all edge-disjoint cycles on the vertex set $\{1, \dots, n\}$ for some $n \geq 6$. The set $\mathcal{C} := \{C_1, \dots, C_k\}$ is a cycle decomposition of G. Regarding the split graphs G_1 and G_2 with respect to $\{1, \lceil n/2 \rceil\}$, there is no cycle decomposition in G_1 and G_2 that relates to $\mathcal{C}$ in an obvious way. In particular, there is essentially only one cycle decomposition in G_1 and G_2 (up to switching parallel edges), whereas in G there exist $\lceil k/2 \rceil$ cycle decompositions that substantially differ.

This, however, does not imply that triconnected components are of no use for any cycle related problem. In fact, as in any maximum cycle packing there can be no two cycles sharing more than three vertices, cf. Lemma 5.8, we can use the triconnected components of a graph G for finding a maximum cycle packing of G. In [OR19], Otto and Recht present an $O(\log(|V(G)|))$ approximation algorithm, which can be used in a similar form to compute a maximum cycle packing using maximum cycle packings on the triconnected components of the graph. The algorithm is a dynamic program on the split tree of the triconnected components of G. We refrain from stating it here, as our focus is on the 2.5-connected components of a graph.

Regarding minimum cycle decompositions though, as the above example shows, there is no easy way of computing small cycle decompositions from minimum cycle decompositions of the triconnected components of an Eulerian graph. This is where 2.5-connected components are of better usage.

5.3. Cycle Decompositions in 2.5-connected Components

In this section we regard the correspondence between cycle decompositions of 2-connected Eulerian graphs and their 2.5-connected components. We show that they almost behave as nicely as connected components and blocks. We begin by proving that 2.5-splitting preserves nodes of even degree.

Lemma 5.3. *Let G be a 2-connected, Eulerian graph. Any 2.5-split graph of G is Eulerian.*

Proof. If G does not have 2.5-split graphs, there is nothing to prove. Denote by G_1 and G_2 two 2.5-split graphs of G with respect to a separation pair $\{v, w\}$ and let $e_1 \in E(G_1)$, $e_2 \in E(G_2)$ be the corresponding virtual edges. By the definition of a 2.5-split G_1 and G_2 are connected and we may assume without loss of generality that e_1 is contained in a 2-edge separator $\{e_1, e'\}$ in G_1.

Note that for $i \in \{1, 2\}$ and all vertices $u \in V(G) \setminus \{v, w\}$, it holds true that $\deg_{G_i}(u) = \deg_G(u)$. In particular all vertices except possibly w and v have even degree in G_i. If we can show that the degree of v and w in G_1 is even, then they also have even degree in G_2 and G_1 and G_2 are Eulerian.

Suppose v has odd degree in G_1. Regard the graph G' induced by the component of $G_1 - e_1 - e'$ containing v. As $G_1 - e_1$ is connected by the definition of a split, $w \notin V(G')$. If v is an endvertex of e_1 and e' it has odd degree in $G_1 - e_1 - e'$ and thereby in G'. All other vertices in G' have even degree, which yields a contradiction to the fact that the number of vertices with odd degree is even, cf. Lemma 2.1. Otherwise v is incident to e_1 but not incident to e' and has therefore even degree in G'. As e is incident to some other vertex in G', this vertex is of odd degree in G' again contradiction Lemma 2.1. We conclude that v has even degree in G_1 and thereby, again by Lemma 2.1, the vertex w is of even degree in G_1 and the graphs G_1 and G_2 are Eulerian. □

Let $\mathcal{G} = \{G_1, \ldots, G_k\}$ be the result of a sequence of 2.5-splits on a 2-connected Eulerian graph G, where we assume unique indices on the graphs G_i. To ease notation in the following we define a cycle decomposition of $\mathcal{G}$ to be the direct product of cycle decompositions of the graphs in the set, i.e. $(C_1, \ldots, C_k)$ for cycle decompositions C_i of G_i. Edges that are virtual in some G_i are assumed

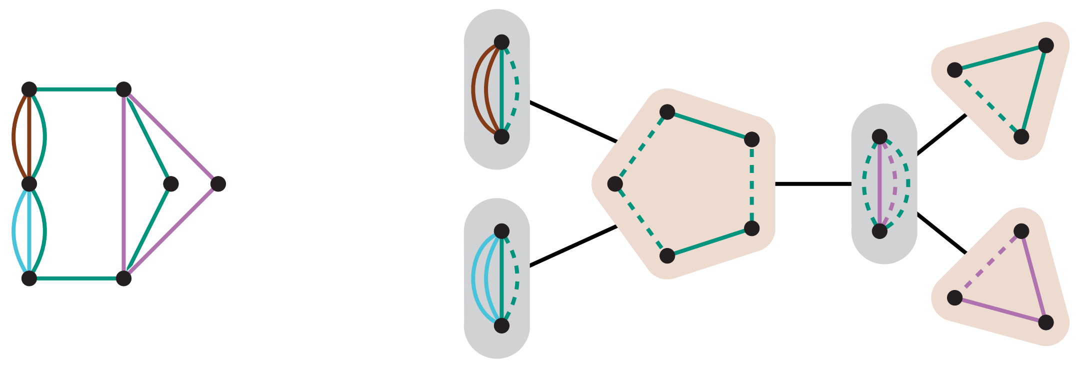

Figure 5.3.: The 2.5-connected components and its split tree of a graph G decomposing into a unique number of cycles. Virtual edges are dashed. The colors indicate a cycle decomposition of the 2.5-connected components of G and its induced cycle decomposition in G.

to be virtual in their cycles in $\mathcal{G}_i$ and we assume that their label is unchanged. Regard some cycle decomposition $\mathfrak{C} := (\mathcal{C}_1, \dots, \mathcal{C}_k)$ of $\mathcal{G}$. We call the cycle decomposition of G that is obtained by performing all possible merges on cycles in $\{C \in \mathcal{C}_i : i = 1, \dots, k\}$, the cycle decomposition *induced* by $\mathfrak{C}$. We say a cycle C in G *crosses* an edge $H_1H_2 \in T_{\mathrm{spl}}(\mathcal{G})$ if the cycle C has edges in graphs of both components of $T_{\mathrm{spl}}(\mathcal{G}) - H_1H_2$. See Figure 5.3 for an example of the above definitions.

Induced cycle decompositions will come in handy, when characterizing the class of Eulerian graphs that decompose into a unique number of cycles. We observe that the cardinality of an induced cycle decomposition is closely related to the number of cycles in the decomposition it is induced by.

Observation 5.4. *Let G be a 2-connected, Eulerian graph. Further let the result of a sequence of 2.5-splits on G be $\mathcal{G} = \{G_1, \dots, G_k\}$ and let $\mathcal{C}$ be a cycle decomposition of G induced by a cycle decomposition $(\mathcal{C}_1, \dots, \mathcal{C}_k)$ of $\mathcal{G}$. Then it holds true that*

$$|\mathcal{C}| = \sum_{i=1}^{k} (|\mathcal{C}_i|) - k + 1.$$

Note that on first sight it might be possible that arbitrarily many cycles cross a given edge of a split tree. And in fact, for arbitrary split trees this may happen as we saw before. The next lemma shows that for results of sequences of 2.5-splits on G exactly one cycle in a decomposition of G crosses an edge of the corresponding split tree. This is the main reason why 2.5-connected components turn out to be a very useful tool for cycle decomposition problems.

Lemma 5.5. *Let G be a 2-connected, Eulerian graph, let $\mathcal{G} = \{G_1, \dots, G_k\}$ be the result of a sequence of 2.5-splits on G, and let $\mathcal{C}$ be a cycle decomposition of G. For each edge $H_1H_2 \in E\left(T_{\mathrm{spl}}(\mathcal{G})\right)$, there* *exists exactly one cycle in $\mathcal{C}$ that crosses H_1H_2.*

Proof. Regard an arbitrary edge $H_1H_2 \in E\left(T_{\mathrm{spl}}(\mathcal{G})\right)$ and denote by $e_1 \in H_1$ and $H_2 \in H_2$ the virtual edges corresponding to H_1H_2. Let G_1, G_2 be the two graphs obtained from merging all virtual edges in $\mathcal{G}$ except e_1 and e_2 such that $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$. Then G_1 and G_2 are 2.5-split graphs of G and by Lemma 5.3 Eulerian. Thus, $G_1 - e_1 \cup G_2 - e_2 = G$. The graph $G_1 - e_1$ is not Eulerian, as the degree of the endvertices of e_1 are odd. We conclude that there must exist a cycle $C \in \mathcal{C}$ that uses edges of G_1 and G_2, which immediately implies that C crosses H_1H_2.

As G_1 and G_2 are 2.5-split graphs, e_1 or e_2 are contained in a 2-edge separator. Say e_1 is contained in a 2-edge separator $\{e_1, e'\}$ in G_1. Then e' is a bridge in $G_1 - e_1$ and thus, any cycle $C \in \mathcal{C}$ that crosses the edge H_1H_2, also contains the edge e'.

We get that each edge in the split tree of $\mathcal{G}$ is crossed by exactly one cycle $C \in \mathcal{C}$. □

As a consequence of Lemma 5.5 and Observation 5.4, we have a one-to-one correspondence between the cycle decompositions of a graph G and the cycle decompositions of its 2.5-connected components. In other words, any cycle decomposition of G is induced by a unique cycle decomposition of the 2.5-connected components of G. We get that computing maximum and minimum cycle decompositions may be reduced to the 2.5-connected components of a graph. See also Figure 5.3 for an example of this.

Corollary 5.6 (cf. Theorem 19 in [Hei+20a]). *Let G be a 2-connected, Eulerian graph with 2.5-connected components $\mathcal{Y} = \{H_1, \ldots, H_k\}$. It holds true that*

$$c(G) = \sum_{i=1}^{k} (c(H_i)) - k + 1 \quad \text{and} \quad \nu(G) = \sum_{i=1}^{k} (\nu(H_i)) - k + 1.$$

5.4. Characterizing Eulerian Graphs Decomposing into a Unique Number of Cycles

Before we give our characterization of Eulerian graphs decomposing into a unique number of cycles, we establish three more lemmata, which significantly simplify the characterization proof. Lemmata 5.7 and 5.8 are rather straight forward observations, whereas Lemma 5.9 is the key to the actual characterization. A similar result to Lemma 5.8 has been observed before in [DR08].

Lemma 5.7 (cf. Lemma 5.1 in [HS19]). *Let G be an Eulerian graph. If H is an even subgraph of G for which $c(H) < \nu(H)$, then $c(G) < \nu(G)$.*

Proof. Observe that $G - E(H)$ is even. Thus, given any cycle decomposition of $G - E(H)$, we can extend this decomposition to a decomposition of G with a minimum, respectively maximum, cycle decomposition of H to get cycle decompositions $\mathcal{C}_1$ and $\mathcal{C}_2$ of G such that $|\mathcal{C}_1| < |\mathcal{C}_2|$. This immediately implies that $c(G) < \nu(G)$. □

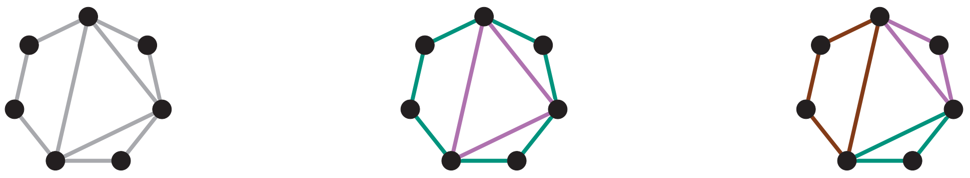
Figure 5.4.: Two cycles sharing three vertices and two different cycle decompositions.

Lemma 5.8 (cf. Lemma 5.1 in [HS19]). *Let $G = C_1 \cup C_2$, where C_1 and C_2 are edge-disjoint cycles such that $|V(C_1) \cap V(C_2)| \geq 3$. Then it holds true that $\nu(G) > 2 = c(G)$.*

Proof. See also Figure 5.4 for this proof. Note that $\{C_1, C_2\}$ is a minimum cycle decomposition of G and thereby $c(G) = 2$. Now let $u, v, w \in V(C_1) \cap V(C_2)$ three distinct vertices. Then, for $i \in \{1, 2\}$ there exists a u-v path P_i in C_i such that $w \notin V(P_i)$. $P_1 \cup P_2$ is an even subgraph of G with $\nu(P_1 \cup P_2) \geq 1$. The graph $G - E(P_1 \cup P_2)$ is even and as w is a degree 4 vertex in the graph, we have $\nu(G - E(P_1 \cup P_2)) \geq 2$. Combining two maximum cycle decompositions of these two graphs yield a maximum cycle decomposition of G with cardinality at least 3. □

Lemma 5.9 (cf. Proof of Theorem 5.2 in [HS19]). *Let G be an Eulerian, 2.5-connected graph. Then G contains two cycles that share 3 or more vertices.*

Proof. Let G be a 2.5-connected graph. Let us assume that G contains a vertex $v \in V(G)$, that it connected to two of its neighbors, say w_1 and w_2 by multiple edges, cf. Figure 5.5 a). As G is 2.5-connected we know that $G - v$ is 2-edge-connected. By Menger's Theorem there exist two edge-disjoint paths P_1, P_2 from w_1 to w_2 in $G - v$. But then the two cycles $vw_1P_1w_2v$ and $vw_1P_2w_2v$ are edge-disjoint and share more than two vertices. Thus, from now on, we may assume

$$\text{every vertex in } G \text{ is connected to at most one of its neighbors by mutliple edges.} \tag{5.1}$$

Assume next, that there is a vertex $v \in V(G)$, which has 3 or less neighbors in G. We have $|N(v)| > 1$, as G is 2-connected. Suppose for the sake of contradiction that $|N(v)| = 2$, say $N(v) = \{w_1, w_2\}$. By (5.1), v is connected to one of its neighbors by a single edge, say w_1. Then (w_2, w_1v) is a vertex-edge separator in G, which is a contradiction to G being 2.5-connected. Thus, $|N(v)| = \{w_1, w_2, w_3\}$ for distinct vertices $w_1, w_2, w_3 \in V(G)$, cf. Figure 5.5 b). By (5.1), we may further assume that v is connected to w_1 and w_2 by a single edge only. As $G - vw_1$ is 2-connected, there is a cycle C in $G - vw_1$ containing the vertices v and w_1. Since w_2 and w_3 are the only neighbours of v in $G - vw_1$, the cycle C also contains the vertices w_2 and w_3. The graph $G - E(C)$ is even and thus, vw_1 is contained in some cycle C' in $G - E(C)$. The single edge vw_2 is contained in C. Hence, v has only neighbours w_1 and w_3 in $G - E(C)$ and C' contains the vertex w_3 as well. Thereby C and C' are two edge-disjoint cycles with more than two vertices in common. We may now also assume that

$$\text{for every } v \in V(G) \text{ it holds true that } |N(v)| \geq 4. \tag{5.2}$$

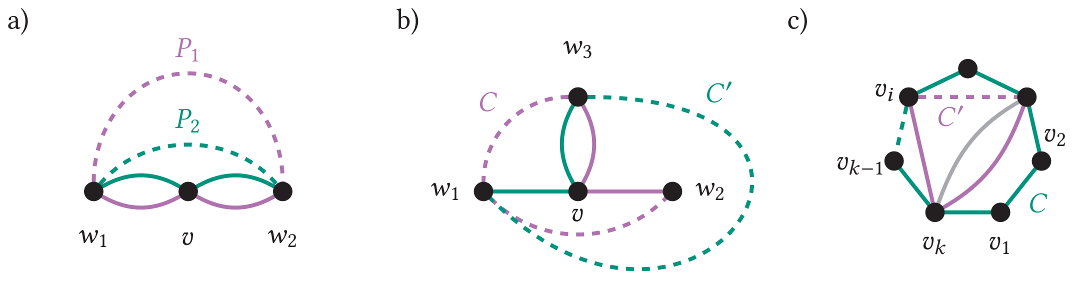

Figure 5.5.: Graphs appearing in the proof of Lemma 5.9. Solid lines are edges. Dashed lines are paths that are edge-disjoint to each other, do not contain edges that are drawn in the figure, and also do not contain vertices they do not touch in the figure.

We now exploit (5.1) and (5.2) to complete the proof. Regard a path $P = v_1v_2 \dots v_k$ with the property that $N(v_k) \subseteq \{v_1, \dots, v_{k-1}\}$ and $v_1v_k \in E(G)$, cf. Figure 5.5 c). Such a path can be found in a greedy fashion: Start at some vertex v in the graph and always move to a new vertex until all neighbors of the current vertex w have already been visited. The resulting path contains the neighborhood of w. Now simply set v_1 to be the neighbor of w that has been visited first and the subsequent vertices accordingly. By Property (5.2) we have $|N(v_k)| \geq 4$. Thus, we can find $i, j \in \{2, .., k-2\}$ with $i \neq j$ and $v_i, v_j \in N(v_k)$. (5.1) implies that v_k is connected to v_i or v_j by a single edge. Without loss of generality let this be v_i. Set $C := v_1v_2...v_kv_1$. Then $G - E(C)$ is an even graph and we can find a cycle C' in $G - E(C)$ containing the edge v_kv_i. Since $N(v_k) \subseteq \{v_1, \dots, v_{k-1}\}$ the two edge-disjoint cycles C and C' have more than two vertices in common, which concludes the proof. □

Theorem 5.10 (cf. Theorem 5.2 in [HS19]). *Let G be a 2-connected, Eulerian graph. The following statements are equivalent.*

(i) G decomposes into a unique number of cycles.

(ii) G does not contain two edge-disjoint cycles sharing three or more vertices.

(iii) The 2.5-connected components of G are a set of multiedges and cycles.

(iv) Each triconnected component of G is either a multiedge or it is a cycles that does not solely consist of virtual edges.

Proof. *(i) ⟹ (ii)*: If G contains two cycles C_1, C_2 sharing three or more vertices, then by Lemma 5.8 $\nu(C_1 \cup C_2) > c(C_1 \cup C_2)$ and by Lemma 5.7 $\nu(G) > c(G)$.

(ii) ⟹ (iii): Let $\mathcal{Y}$ be the 2.5-connected components of G and suppose $H \in \mathcal{Y}$ is a 2.5-connected graph. By Lemma 5.9, H contains two edge-disjoint cycles C_1, C_2 sharing three or more vertices. By Observation 4.25, there also exist two edge-disjoint cycles C_1', C_2' in G that share the same vertices as C_1 and C_2 — a contradiction.

(iii) $\Rightarrow$ *(i)*: For a graph H that is a cycle or an Eulerian multiedge, it holds true that $c(H) = \nu(H)$. Thus, if $\{G_1, \dots, G_k\}$ are the 2.5-connected components of G, then by Corollary 5.6, we conclude

$$c(G) = \sum_{i=1}^{k} c(G_i) + k - 1 = \sum_{i=1}^{k} \nu(G_i) + k - 1 = \nu(G).$$

(iii) $\Leftrightarrow$ *(iv)*: If the 2.5-connected components of G are a set of cycles and multiedges, then the only splits that are possible are splits of cycles or multiedges. When constructing triconnected components these splits are merged again. Thus the triconnected components of G are exactly the 2.5-connected components and by Corollary 4.21 do not contain cycles solely consisting of virtual edges. If on the other hand the triconnected components of G only contains multiedges and cycles, that not solely consist of virtual edges, by Theorem 4.20, the 2.5-connected components are the same. □

For an example of a graph that decomposes into a unique number of cycles, we refer to Figure 5.3. The figure shows one such graph and its 2.5-connected components aligned in their split tree.

Corollary 5.11. *Given an even graph G we can decide in time $\mathcal{O}(|V(G)| + |E(G)|)$ if G decomposes into a unique number of cycles. Thereby we may also decide in linear time if G contains two edge-disjoint cycles sharing three or more vertices.*

Proof. By Observations 5.2 and the fact that we can compute the components and blocks of a graph in linear time, cf. [HT73], we may restrict ourselves to 2-connected graphs. By Theorem 4.24 we can compute the 2.5-connected components of a graph in linear time. Further, it is not hard to see that we can verify that a given graph is a cycle or a multiedge in linear time in the number of edges in the graph. Note that the total number of edges in the 2.5-connected components is contained in $\mathcal{O}(|V(G)| + |E(G)|)$, cf. Chapter 4. Thus, by Theorem 5.10 we get the desired result. □

Conclusion In this chapter we discussed cycle decompositions of Eulerian graphs as an application to 2.5-connectivity in graphs. We observed that there is a one-to-one correspondence between cycle decompositions of the 2.5-connected components of a 2-connected graph and the graph itself, whereas the same does not hold for the triconnected components of a graph. As a consequence, computation of minimum and maximum cycle decompositions may be restricted to 2.5-connected graphs. As the main result of this chapter we gave two different characterizations of Eulerian graphs decomposing into a unique number of cycles: These are exactly those graphs, in which no two edge-disjoint cycles share two or more vertices or whose 2.5-connected components consist of cycles and multiedges only. The key to proving this characterization is the fact that any 2.5-connected graph contains two cycles that share three or more vertices. A direction of further research could be to classify the class of Eulerian graphs whose minimum and maximum cycle decomposition number varies by a constant. It could also be interesting to examine the maximum difference between the two graph invariants for Eulerian graphs that do not contain two edge disjoint cycles sharing k vertices for $k \geq 4$.

We also used the above characterization to argue that the class of Eulerian graphs that decompose into a unique number of cycles can be recognized in linear time.

Chapter 6.

Simultaneous Domination of Spanning Trees

In this chapter we discuss simultaneous domination in spanning trees of a graph. A dominating set is a subset of the vertices of a graph such that each vertex is either contained in the set or has a neighbor in the set. A subset of the vertices of a graph simultaneously dominates all spanning trees if the set is dominating in each spanning tree. We prove that on 2-connected graphs, any set dominating all spanning trees is a vertex cover in the graph and vice versa. This is not the case for general graphs. We show that the problem of finding a minimum size simultaneous dominating set on perfect graphs is hard, whereas it is known that a minimum vertex cover can be found in polynomial time. We present an algorithm that finds a minimum simultaneous dominating set by computing a minimum vertex cover on certain subgraphs of the blocks of the graph.

Most of the results of this section have been published in [JKS18] and are joint work with Sebastian S. Johann and Sven O. Krumke. The results for which this does not apply are marked accordingly.

Dominating sets are a very broad and much studied field in graph theory. For a general overview on domination problems we refer to [HHS98]. Here we regard a version of the problem, in which we not only seek to dominate one graph, but in particular all spanning trees of a graph.

In the literature, similar domination problems have been called *factor domination* or *global domination*. Global domination aims at dominating not only the graph itself, but also its complement, cf. [BC17]. Factor domination was originally introduced in by Brigham and Dutton in [BD90]. In their definition a set is said to be a factor dominating set for a given graph G and a partition $\{E_1, \dots E_k\}$ of the edge set of G, if it is dominating in each graph $G[E_i]$ for $i \in \{1, \dots, k\}$. Later, the term factor dominating set was also used for a set that is dominating on given factors of a graph by Dankelmann et al., cf. [DL03; Dan+06]. In the latter definition the given factors possibly share edges and do not necessarily partition the edge set. Dankelmann et al. give bounds on the smallest size of a factor dominating set. In [CH14], Caro and Henning introduce the name simultaneous domination and give bounds on the simultaneous domination number. Here we stick to the term of simultaneous domination as the original definition of factor domination does not exactly fit our problem setup. The problem of dominating all spanning trees in a graph has previously not been considered in the literature.

Outline In Section 6.1 we formally define the problem of dominating all spanning trees, give an alternative characterization and analyze its complexity. Section 6.2 deals with a dynamic programming approach to finding minimum simultaneous dominating sets operating on the block-cutpoint tree of a graph. Here, we assume that we may solved a generalized version of the problem on the blocks of a graph. We focus on solving this related problem in Section 6.3. In Section 6.4 we analyze the complexity of the problem on restricted graph classes. Finally, in Section 6.5 we present a 2-approximation based on a rounding procedure of a solution to an LP-formulation.

6.1. Problem Definition and Classification

We begin by formally defining simultaneous dominating sets of all spanning trees and the corresponding decision problem. As in this thesis we only regard simultaneous domination of spanning trees, we omit the term spanning tree in the definition of the term.

Definition 6.1. Let G be a graph and $S \subseteq V(G)$. We call the set S a *simultaneous dominating set* (*sd-set*) in G if S is a dominating set in each spanning tree of G. In this case we say S *simultaneously dominates* G. By $\gamma^T(G)$ we denote the minimum cardinality of a simultaneous dominating set in G.

We also say a vertex $v \in V(G)$ is simultaneously dominated by a set S, if it is dominated by S in every spanning tree. Similarly we call a subset $W \subseteq V(G)$ simultaneously dominated by S if all $w \in W$ are simultaneously dominated by S.

Simultaneous Domination of Spanning Trees (SDST).
Instance: A graph G and an integer $B \in \mathbb{N}$.
Question: Does there exist a subset $S \subseteq V(G)$ of the vertices of G with $|S| \leq B$ such that S is a dominating set in each spanning tree of G?

As already mentioned in the introduction of this chapter, there is a close resemblance of simultaneous dominating sets and vertex covers of graphs. As the graph consisting of a single isolated vertex is an exception to this resemblance and not of particular interest we make the following assumption for the remainder of this chapter.

Assumption 1. A connected graph contains at least two vertices.

We are now ready to clarify the resemblance between vertex covers and simultaneous dominating sets.

Proposition 6.2. *Let G be a connected graph. Then a subset $S \subseteq V(G)$ is a simultaneous dominating set in G if and only if for every $v \in V(G)$ it holds true that $v \in S$ or:*

(i) v is not a cut vertex in G and $N(v) \subseteq S$, or

(ii) v is a cut vertex in G and for one of the blocks B of G with $v \in V(B)$ it holds true that $N_B(v) \subseteq S$.

Proof. Let $v \in V \setminus S$ be a vertex that is not a cut vertex in G. If all the neighbors of v are contained in S, then v is simultaneously dominated by S, since there is at least one edge between v and one of its neighbors in each spanning tree of G. Conversely, assume that v is simultaneously dominated by S. As $G - v$ is connected, there exists a spanning tree $T$of $G - v$. We obtain a spanning tree of G from T by adding v and any edge incident to v in G. Thus, for any neighbor $w \in N_G(v)$ there is at least one spanning tree of G such that w is the only neighbor of v. Since v is dominated in every spanning tree of G and $v \notin S$ we get that all the neighbors of v are contained in S.

Next consider the case that v is a cut vertex and is further contained in the blocks $B_1, \dots, B_k$. If for some $i \in \{1, \dots, k\}$ we have $w \in S$ for all $w \in N_{B_i}(v)$, then v is simultaneously dominated by S as in every spanning tree of G there is at least one edge between v and one of its neighbors in the block B_i. Conversely, assume that v is dominated by S in every spanning tree of G. Suppose that for each block B_i there is at least one neighbor w_i of v in B_i that is not in S. We can find a spanning tree T of G by taking a spanning forest in $G - v$, adding v and for every $i \in \{1, \dots, k\}$ the edge between v and w_i. T is a spanning tree of G in which v is not dominated by S since neither the vertex v nor any of its neighbors w_i is contained in S — a contradiction. □

Corollary 6.3. *Let G be a 2-connected graph. A subset $S \subseteq V(G)$ is simultaneously dominating if and only if S is a vertex cover.*

Proof. As G is 2-connected, it does not contain any cut vertices. Thus, by Proposition 6.2, a set S is simultaneously dominating if and only if for each vertex $v \in V$ it is $v \in S$ or $N(v) \subseteq S$, which is exactly the definition of a vertex cover. □

One might think that any further investigation of SDST is obsolete. Note however, that a minimum simultaneous dominating set in a graph G cannot be found by simply taking the union of simultaneous dominating sets in the blocks of G. A difference to vertex covers is that a simultaneous dominating set in a graph G, does not need to be simultaneously dominating in all blocks of G. For examples of this see also Figure 6.1 and 6.2. We later see that SDST restricted to perfect graphs remains NP-complete, whereas Vertex Cover is known to be polynomial time solvable on perfect graphs, cf. [Sch03]. Thus, a further investigation of the problem is in line.

As, in general, Vertex Cover is NP-complete, cf. [GJ79], the next result is hardly surprising. We state it for the sake of completeness. It is not hard to see that Vertex Cover remains NP-complete when restricted to 2-connected graphs.

Theorem 6.4. *The decision problem Simultaneous Domination of Spanning Trees is NP-complete. It remains NP-complete, when restricted to 2-connected graphs.*

Proof. Given a solution to SDST, we may validate it using Proposition 6.2 in polynomial time. Thus, SDST is contained in NP. The NP-hardness is a direct consequence of Corollary 6.3 and the fact that Vertex Cover is NP-complete when restricted to 2-connected graphs. □

Using Proposition 6.2, we easily verify, that any vertex cover in a connected graph G with $|V(G)| \geq 2$ is also simultaneously dominating. Thus, it holds true that $\gamma^T(G) \leq \tau(G)$ for all

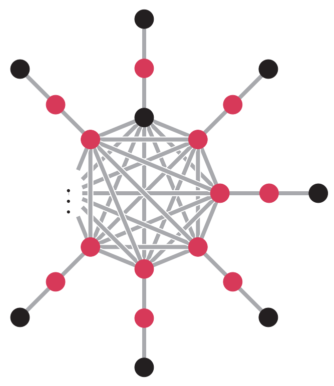
Figure 6.1.: Vertex cover of minimum size

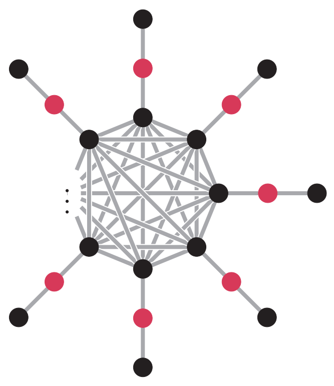
Figure 6.2.: Sd-set of minimum size

these graphs. For 2-connected graphs we also have equality by Corollary 6.3. For general graphs, however, this equality does not hold as the following example shows.

Example 6.5. For some $n \in \mathbb{N}$ denote by K_n the complete graph with vertex set $\{1, \dots, n\}$. Further, for $i \in \{1, \dots, n\}$, denote by P_i the path $i(n+i)(2n+i)$. We regard the graph $G = K_n \cup P_1 \cup \dots \cup P_n$ and observe that $\gamma^T(G) = n$ and $\tau(G) = 2n - 1$, cf. Figures 6.1, 6.2. Thus, there exists an infinite class of graphs, such that for all graphs H in this class it holds true that $\tau(H) \geq 2\gamma^T(H) - 1$.

In fact, Example 6.5 is already the worst case for the gap between the size of a minimum vertex cover number and the size of a minimum simultaneous dominating set.

Theorem 6.6. *Let G be a graph and let S be a simultaneous dominating set in G. S can be extended to a vertex cover C by adding no more than $|S| - 1$ vertices. In particular $\tau(G) \leq 2\gamma^T(G) - 1$.*

Proof. Let G be a connected graph and let S be a simultaneous dominating set in G. Denote by T the block-cutpoint tree of G. By definition, any simultaneous dominating set is non-empty. Thus, there exists some block B of G such that $V(B) \cap S \neq \emptyset$. We root the tree T at one such block B_r. Denote by X the set of cut vertices of G and define for each cut vertex $v \in X$ with children $B_1, \dots, B_k$ in T the set

$$S(v) := S \cap \left(\bigcup_{i=1}^{k} V(B_i) \right).$$

Further let $C := S \cup \{v \in X : S(v) \neq \emptyset\}$.

First we show that $|C| \leq 2 \cdot |S| - 1$. If we can find an injective mapping from $C \setminus S$ to $S \setminus V(B_r)$ we are done as $|V(B_r) \cap S| \geq 1$. So let $v \in C \setminus S$. By definition of C there exists some $w \in S(v)$. We map v to w. This induces an injective mapping: If w is not a cut vertex, then B is the only block w is contained in and B is no child of any other cut vertex than v. Otherwise w is a cut vertex and is itself a child of B in T. All blocks besides B containing w are children of w. But as $w \in S$, no other added vertex may be mapped to w. This implies that the defined mapping is injective and thereby $|C| \leq 2 \cdot |S| - 1$.

It remains to show that C is actually a vertex cover. So let vw be some edge in G. If one of the two vertices v and w is not a cut vertex, then one of them is contained in S by Proposition 6.2 and thereby also in C. So assume v and w are cut vertices. If $v \in C$ we are done, so assume $v \notin C$. By the definition of C every child B of v in T fulfills $V(B) \cap S = \emptyset$. Thus, as v is simultaneously dominated, by Proposition 6.2 this implies that all neighbors of v in the block that is the parent of v in T are contained in S and thereby in C. If w is contained in that block we are done, so assume that w is contained in some child $B^\star$ of v in T. As w is a cut vertex, by Proposition 6.2 this implies $N_{B'}(w) \subseteq S$ for some block B' containing w. In particular we have $N_{B'}(w) \cap S \neq \emptyset$, which implies $V(B') \cap S \neq \emptyset$. We cannot have $B' = B^\star$ as this would imply that $v \in S$. Thus, B' is a child of w in T and by the definition of C, it holds true that $w \in C$. Thereby the edge vw is covered by C. This implies that C is a vertex cover. □

6.2. Block-cutpoint Tree Algorithm

In this section we present a dynamic program operating on the block-cutpoint tree of a graph that solves Min-Simultaneous Domination of Spanning Trees. Throughout the section we assume that we can solve the appearing subproblem on 2-connected graphs. Solving this subproblem is then the focus of the next section.

In the following we refer to the leaves of the block-cutpoint tree of a graph as a *leaf block* and to the unique cut vertex connecting a leaf block to the rest of the block-cutpoint tree as its *connection vertex*.

To introduce the generalized version of simultaneous domination we assign colors to vertices. To get an intuition what these colors represent we now state our interpretation of them. The vertex v has color

- 1, if v is fixed to be contained in the sd-set,
- 0, if v is currently not contained in the sd-set, but assumed to be simultaneously dominated already and
- $\hat{0}$, if v is currently not contained in the sd-set and assumed not to be simultaneously dominated yet.

We say that color 1 is *better* than colors 0 and $\hat{0}$ and call color 0 *better* than color $\hat{0}$. For a subset $C \subseteq \{1, 0, \hat{0}\}$ we denote the best color of C by $\max C$.

Let us briefly describe the idea of the algorithm: We begin by regarding some leaf block B with connection vertex v of a graph G. We take among all minimum size sets $S \subseteq V(B)$ that simultaneously dominate all vertices in $V(B) \setminus \{v\}$ one with the *best coverage* for v, i.e., the best color for v. We then remove $V(B) \setminus \{v\}$ from G and continue with the next leaf-component.

In later iterations of the algorithm we have vertices, which are already simultaneously dominated or even in the sd-set for *free*. This has to be taken into account when computing a minimum size set in some block that was originally no leaf block in the whole graph. The crucial point why this procedure works is, that any given vertex can be simultaneously dominated by adding at most one vertex to the simultaneous dominating set, namely itself. Thus, if all minimum simultaneous dominating sets for some leaf block do not simultaneously dominate the connection vertex v, then we simply simultaneously dominate v later on, as we can be sure that it never costs us more than it would to simultaneously dominate it in the current leaf block.

Before we turn to the actual algorithm we formally define the previously mentioned generalized version of a simultaneous dominating set. This is the problem we aim to solve on 2-connected graphs in the next section. We also introduce some notation that simplifies the formulation and the correctness proof of the algorithm.

Definition 6.7. Let G be a connected graph and let $f\colon V(G) \to \{1, 0, \hat{0}\}$ be a mapping assigning each vertex of G one of the indicated colors. We call a subset $S \subseteq V(G)$ *f-respecting simultaneous dominating set* of G if the following conditions hold:

- $f^{-1}(1) \subseteq S$.
- S simultaneously dominates $f^{-1}(\hat{0})$.

If we do not specify the coloring we also use the term *color respecting simultaneous dominating set.*

A color respecting sd-set S is an sd-set with the property that all vertices with color 1 are contained in S and all vertices with color 0 do not have to be simultaneously dominated by S. Clearly this is a generalization of the definition of an sd-set: If we color all vertices by $\hat{0}$, a color respecting sd-set and an sd-set are the same thing.

Let G be a graph and $f\colon V(G) \to \left\{1, 0, \hat{0}\right\}$ some coloring of the nodes. For any induced subgraph H of G we denote by f^H the coloring f restricted to the nodes of H. Further, for any fixed vertex $v \in V(H)$ and $c \in \left\{1, 0, \hat{0}\right\}$ we denote by $f^H_{v=c}$ the coloring of $V(H)$ with $f^H_{v=c}(v) = c$ and $f^H_{v=c}(w) = f^H(w)$ for all $w \in V(H) \setminus \{v\}$. Finally we denote for $c \in \left\{1, 0, \hat{0}\right\}$ by $S^H_{v=c}$ a minimum $f^H_{v=c}$-respecting sd-set in H. If $H = G$ we omit the superscript H in the notation.

In the remainder of this section we assume that we are given an algorithm crSDS: The algorithm is given a 2-connected graph G and a coloring $f\colon V(G) \to \left\{1, 0, \hat{0}\right\}$. It returns an f-respecting sd-set S of minimum size and the size # of this set.

Algorithm 1 shows a pseudocode version of the complete procedure. Within it we use the *black box algorithms* crSDS and GetLeafBlock. The latter one gets a graph G which is not 2-connected as an input and returns a leaf B of the block-cutpoint tree of G and its parent $v \in B$. We save the current color of v and compute a color respecting sd-set in B for all possible colors of v. We use the simultaneous dominating set, which is the smallest among the three possibilities, where ties are broken by the best coverage of v. We then delete all vertices in $V(B) \setminus \{v\}$ from G and continue with the remaining graph. By abuse of notation we denote the graph $G - (V(B) \setminus \{v\})$ by G/B.

Before we formally prove the correctness of Algorithm 1 and discuss its running time, we prove two lemmata, which simplify arguments later on.

Algorithm 1: Computing a color respecting sd-set of minimum size

```
Input: A connected graph $G = (V, E)$ and a coloring $f\colon V(G) \to \{1, 0, \hat{0}\}$
Output: An $f$-respecting sd-set of minimum size in $G$
$S = \emptyset$
while $G$ is not 2-connected do
    $B, v$ = getLeafBlock($G$)
    $c^\star = f(v)$
    for $c \in \{1, 0, \hat{0}\}$ do
        $f(v) = c$
        $S_c, \#_c$ = crSDS($B, f^B$)
    if $\#_1 = \#_{\hat{0}} = \#_0$ then
        $f(v) = 1$
        $S = S \cup S_1$
    else if $\#_1 > \#_{\hat{0}} = \#_0$ then
        $f(v) = \max\left\{c^\star, 0\right\}$
        $S = S \cup S_{\hat{0}}$
    else
        $f(v) = c^\star$
        $S = S \cup S_0$
    $G = {}^{G}/_{B}$
$S_G, \#$ = crSDS($G, f$)
return $S \cup S_G$
```

Lemma 6.8. *Let G be a connected graph, $v \in V(G)$ a fixed vertex in G and $f\colon V(G) \to \left\{1, 0, \hat{0}\right\}$ some coloring. Then the following two statements hold true:*

(i) $|S_{v=0}| \leq |S_{v=\hat{0}}| \leq |S_{v=1}|$.

(ii) $|S_{v=1}| - |S_{v=0}| \leq 1$.

Proof. We begin by showing that every $f_{v=\hat{0}}$-respecting sd-set S is also $f_{v=0}$-respecting. Clearly we have $f_{v=0}^{-1}(1) = f_{v=\hat{0}}^{-1}(1) \subseteq S$. Further, it holds true that $f_{v=0}^{-1}(\hat{0}) \subseteq f_{v=\hat{0}}^{-1}(\hat{0})$. Thus, S simultaneously dominates $f_{v=0}^{-1}(\hat{0})$ and is thereby also $f_{v=0}$-respecting. With similar arguments we get that any $f_{v=1}$-respecting SD-set is also $f_{v=\hat{0}}$-respecting. These two small observations directly imply Property *(i)*.

To see that *(ii)* also holds, let S_0 be a minimum size $f_{v=0}$-respecting sd-set of G. Then $S_0 \cup \{v\}$ is $f_{v=1}$-respecting, as

$$f_{v=1}^{-1}(1) = f_{v=0}^{-1}(1) \cup \{v\} \subseteq S_0 \cup \{v\}$$

and $f_{v=1}^{-1}(\hat{0}) \subseteq f_{v=0}^{-1}(\hat{0})$. This already implies that the minimum size $f_{v=1}$-respecting sd-set has at most one element more than S_0. □

Lemma 6.9. *Let G be a graph with some coloring $f\colon V(G) \to \{1,0,\hat{0}\}$ and B be some leaf block of G with connection vertex $v \in V(B)$. Then the following three statements hold true:*

– *If $|S^B_{v=0}| = |S^B_{v=\hat{0}}| = |S^B_{v=1}|$, then $S^B_{v=1} \cup S^{G/B}_{v=1}$ is a minimum f-respecting simultaneous dominating set in G.*

– *If $|S^B_{v=0}| = |S^B_{v=\hat{0}}| < |S^B_{v=1}|$, then $S^B_{v=\hat{0}} \cup S^{G/B}_{v=\max\{f(v),0\}}$ is a minimum f-respecting simultaneous dominating set in G.*

– *If $|S^B_{v=0}| < |S^B_{v=\hat{0}}| = |S^B_{v=1}|$, then $S^B_{v=0} \cup S^{G/B}_{v=f(v)}$ is a minimum f-respecting simultaneous dominating set in G.*

Proof. It is easy to see that all claimed sets are f-respecting simultaneous dominating sets in G, we now focus on their minimality. To this end let S be a minimum f-respecting simultaneous dominating set in G.

We begin with the case that $|S^B_{v=0}| = |S^B_{v=\hat{0}}| = |S^B_{v=1}|$. First assume that $v \in S$. Then $S \cap V(B) \geq |S^B_{v=1}|$ and $S \cap V(G/B) \geq |S^{G/B}_{v=0}|$. By *(ii)* of Lemma 6.8 this implies the claim. Next assume $v \notin S$ and regard $S \cap V(B)$. This set simultaneously dominates all vertices in B with respect to f except possibly v. As $S^H_{v=0}$ is of minimum size among these sets we have $|S \cap V(H)| \geq |S^H_{v=0}| = |S^H_{v=1}|$ and we can replace $S \cap V(H)$ by $S^H_{v=1}$ without making it larger. We now have a minimum f-respecting simultaneous dominating set containing v and by the same arguments as above we get that $S^H_{v=1} \cup S^{G/B}_{v=1}$ is a minimum f-respecting sd-set in G.

Next assume $|S^B_{v=0}| = |S^B_{v=\hat{0}}| < |S^B_{v=1}|$. First note that this implies $v \notin S^B_{v=\hat{0}}$. Regard the set $S' := (S \cap V(G/B)) \cup S^B_{v=\hat{0}}$. Then $|S'| \leq |S|$ and S' is still simultaneously dominating with respect to f. Furthermore it holds true that $|S' \cap V(G/B)| \geq |S^{G/B}_{v=\max\{f(v),0\}}|$ and we get

$$|S| \geq |S'| = |S' \cap V(G/B)| + |S' \setminus V(G/B)| \geq |S^{G/B}_{v=\max\{f(v),0\}}| + |S^B_{v=\hat{0}}|,$$

which implies the desired result.

Finally assume that $|S^B_{v=0}| < |S^B_{v=\hat{0}}| = |S^B_{v=1}|$. If v is not simultaneously dominated by $S \cap V(B)$ in B we are done, so assume v is simultaneously dominated by S in B and hence $|S \cap V(B)| > |S^B_{v=0}|$. If $v \in S$, by Lemma 6.8 we have $|S \cap V(G/B)| \geq |S^{G/B}_{v=1}| \geq |S^{G/B}_{v=f(v)}|$ and hence,

$$|S| = |S \cap V(B)| + |S \cap V(G/B)| - 1 > |S^B_{v=0}| + |S^{G/B}_{v=f(v)}| - 1.$$

If $v \notin S$, Lemma 6.8 implies $|S \cap V(G/B)| \geq |S^{G/B}_{v=0}| \geq |S^{G/B}_{v=f(v)}| - 1$ and we get

$$|S| = |S \cap V(B)| + |S \cap V(G/B)| > |S^B_{v=0}| + |S^{G/B}_{v=f(v)}| - 1.$$

Both cases then imply $|S| \geq |S^B_{v=0}| + |S^{G/B}_{v=f(v)}|$. □

Theorem 6.10. *For a connected graph G and a coloring $f\colon V(G) \to \{1, 0, \hat{0}\}$, Algorithm 1 correctly computes a minimum f-respecting simultaneous dominating set S of G. It can be implemented to run in polynomial time if* crSDS *can be implemented to run in polynomial time.*

Proof. The proof of correctness can be regarded as a direct consequence of Lemma 6.9. Nevertheless we give a formal proof here for the sake of completeness. To this end, note that Algorithm 1 can be regarded as a recursive algorithm, where in each step one leaf-component is cut off the graph. We do induction on the number of blocks of G. If G is 2-connected the claim trivially holds. So let B be a leaf block of G with connection vertex v. In the algorithm we now compute $S^B_{v=i}$ for $i \in \{1, 0, \hat{0}\}$. By Lemma 6.8 the three case distinction made in the algorithm (concerning the sizes of these sets) are the only cases that may occur. The algorithm now handles the cases as follows:

- If $|S^B_{v=0}| = |S^B_{v=\hat{0}}| = |S^B_{v=1}|$, it adds $S^B_{v=1}$ to the current set and colors v with color 1. Thus, by induction the algorithm returns $S^B_{v=1} \cup S^{G/B}_{v=1}$, which is a minimum f-respecting simultaneous dominating set by Lemma 6.9.
- If $|S^B_{v=0}| < |S^B_{v=\hat{0}}| = |S^B_{v=1}|$, it adds $S^B_{v=0}$ to the current set and leaves the color as it was. Thus, by induction the algorithm returns $S^B_{v=0} \cup S^{G/B}_{v=f(v)}$, which is a minimum f-respecting simultaneous dominating set by Lemma 6.9.
- If $|S^B_{v=0}| = |S^B_{v=\hat{0}}| < |S^B_{v=1}|$, it adds $S^B_{v=\hat{0}}$ to the current set and sets the color of v to $\max\{f(v), 0\}$. Thus, by induction the algorithm returns $S^B_{v=\hat{0}} \cup S^{G/B}_{v=\max\{f(v),0\}}$, which is a minimum f-respecting simultaneous dominating set by Lemma 6.9.

In all considered cases the algorithm correctly computes a minimum f-respecting simultaneous dominating set.

Considering the running time of Algorithm 1, note that we can find all blocks in time $O(|V(G)| + |E(G)|)$, cf. [HT73]. With a small adjustment of the usual lowpoint algorithm by Hopcroft and Tarjan [HT73] we can get the components in order such that each time we regard the next component it is a leaf block of the remaining graph. Doing this as a preprocessing step, each call to GetLeafBlock takes constant time and the deletion of H is done implicitly. In each iteration, besides the three calls to crSDS we only do steps that can be realized in polynomial time, thus if crSDS can be implemented to run in polynomial time so can Algorithm 1. □

6.3. Color Respecting Simultaneous Dominating Sets in 2-connected Graphs

In the last section we showed how to find a minimum sd-set provided we can find a minimum size color respecting sd-set on 2-connected graphs. In this section we focus on finding such a color respecting sd-set.

Recall that a color respecting simultaneous dominating set S in a graph G that is 2-connected is, in some sense, a simultaneous dominating set in G with the additional constraint that all vertices

Algorithm 2: crSDS(G, f): Finding a minimum color respecting simultaneous dominating set on 2-connected graphs

```
    **Input:** A 2-connected graph $G = (V, E)$ and a colouring $f\colon V(G) \to \{1, 0, \hat{0}\}$
    **Output:** A minimum $f$-respecting simultaneous dominating set and its size
1   $G = G - f^{-1}(1)$
2   $G = G - E(G[f^{-1}(0)])$
3   $S = \textsc{MinVertexCover}(G)$
4   **return** $S \cup f^{-1}(1)$, $|S \cup f^{-1}(1)|$
```

with color 1 are contained in S and the exception that all vertices with color 0 do not actually have to be simultaneously dominated, cf. Definition 6.7. Algorithm 2 describes how to solve this problem using an algorithm (MinVertexCover) for solving Min Vertex Cover as a black box.

Theorem 6.11. *Given a 2-connected graph G and a coloring $f\colon V(G) \to \left\{1, 0, \hat{0}\right\}$ Algorithm 2 returns a minimum f-respecting simultaneous dominating set of G. It can be implemented to run in polynomial time if MinVertexCover can be implemented to run in polynomial time.*

Proof. We begin by proving that the set returned by the algorithm, say $S^\star$, is an f-respecting simultaneous dominating set. It is obvious that $f^{-1}(1) \subseteq S^\star$, thus by Definition 6.7, as G is 2-connected, we only need to prove that for all vertices v with $f(v) = \hat{0}$, we have $v \in S^\star$ or $N_G(v) \subseteq S^\star$. So let $v \in V$ with $f(v) = \hat{0}$. After having deleted all vertices with color 1 we do not delete edges incident to v. Thus, the vertex cover computed either contains v itself or all neighbors of v which do not have color 1. As all deleted vertices are contained in $S^\star$ the required condition follows and we conclude that $S^\star$ is indeed an f-respecting simultaneous dominating set.

Let $G' = (G - f^{-1}(1)) - E(G[f^{-1}(0)])$. To see that the algorithm actually returns a minimum f-respecting simultaneous dominating set we show that for every f-respecting simultaneous dominating set S in G it holds true that $S \setminus f^{-1}(1)$ is a vertex cover in G'. The correctness then follows immediately. So let S be an arbitrary f-respecting simultaneous dominating set in G and let $e = vw \in E(G')$. Then at least one endpoint of e, say v, has color $\hat{0}$ and neither v nor w has color 1. By Definition 6.7 this means either v or all vertices in $N_G(v)$ are contained in S. But we have $w \in N'_G(v)$. This implies $w \in S \setminus f^{-1}(1)$ or $v \in S \setminus f^{-1}(1)$. As e was an arbitrary edge in $E(G')$ we know that $S \setminus f^{-1}(1)$ is a vertex cover in G'.

It is easy to see that all steps of the algorithm, except possibly the call to MinVertexCover can be implemented to run in polynomial time. □

6.4. Special Graph Classes

This section is dedicated to analyzing the complexity of SDST when restricted to certain graph classes. From Theorem 6.10 and Theorem 6.11 we conclude polynomial time solvability of SDST for a large class of graphs.

Corollary 6.12. *Let $\mathcal{G}, \mathcal{H}$ be two classes of graphs fulfilling the properties* *(i)* *Vertex Cover restricted to instances with graphs from $\mathcal{H}$ is contained in* P *and*

(ii) *for all $G \in \mathcal{G}$ and all blocks B of G the graph $B - U - E(B[W])$ is contained in $\mathcal{H}$ for all $U, W \subseteq V(B)$.*

Then Simultaneous Domination of Spanning Trees restricted to instances with graphs from $\mathcal{G}$ is contained in P. □

In the following we give three examples of graph classes that fulfill the conditions of Corollary 6.12 and discuss two classes, *perfect graphs* and *claw free graphs*, that do not fulfill the conditions.

Bipartite Graphs It is easy to see that bipartite graphs are hereditary, *i.e.* every induced subgraph is again bipartite. Even if we delete edges in the graph it remains bipartite. With the help of König's theorem [Sch03] and for example the Hopcroft-Karp algorithm [HK71] we can compute a minimum vertex cover for bipartite graphs in polynomial time. Thus, Simultaneous Domination of Spanning Trees restricted to instances with bipartite graphs is contained in P by Corollary 6.12.

Graphs with bounded Treewidth For some fixed k regard the class $\mathcal{G}^k$ of all graphs with treewidth at most k. Bodlaender showed in [Bod93], that the class can be recognized in linear time. He also showed that a tree decomposition of width k can be found in linear time for graphs in $\mathcal{G}^k$. Arnborg and Proskurowski showed in [AP89] that a vertex cover of minimum size can be computed for a graph with bounded treewidth and given tree decomposition in linear time. As deleting vertices or edges does not increase the treewidth, by Theorem 6.12 Simultaneous Domination of Spanning Trees restricted to instances with graphs from $\mathcal{G}^k$ is contained in P.

Chordal Graphs Chordal graphs are hereditary but, if we delete edges of induced subgraphs in a chordal graph, it is possible that the resulting graph is not chordal anymore. However, with the help of the strong perfect graph theorem, cf. [Chu+06], we can show that deleting all edges of an induced subgraph from a chordal graph always results in a perfect graph. In perfect graphs we can compute a minimum vertex cover in polynomial time, cf. [Sch03]. Thus, again using Corollary 6.12 we get that Simultaneous Domination of Spanning Trees restricted to instance with chordal graphs is contained in P.

In the following we show that for any chordal graph G and $W \subseteq V(G)$ the graph $G - E(G[W])$ is perfect. Before we prove the statement, we recall the perfect graph theorem and the terminology needed to understand it as we will be needing the theorem in the proof. For a graph G an *odd hole* of G is an induced subgraph of G which is a cycle of odd length at least 5. An *odd antihole* of G is an induced subgraph of G whose complement is an odd hole in $\overline{G}$.

Theorem 6.13 (Strong perfect graph theorem, [Chu+06])**.** *A graph G is perfect if and only if G neither has an odd hole nor an odd antihole.* □

Lemma 6.14. *Let G be a chordal graph and $W \subseteq V(G)$. Let G' be the graph obtained by deleting all edges between the vertices of W in G, i.e.*

$$G' = G - E(G[W]).$$

Then G' is perfect.

Proof. Assume G' has an odd hole C_{2k+1}. Then at most k vertices of C_{2k+1} can be contained in W since W is an independent set in G'. Hence, there are two consecutive vertices on C_{2k+1} which are not in W. Since these two vertices do not have the same neighbor in C_{2k+1} and only edges between vertices of W are deleted there exits a cycle of length at least four in G that has no chord — a contradiction.

Now let us assume that the graph G' has an odd antihole $\overline{C}_{2k+1}$, where $C_{2k+1} = v_1 \dots v_{2k+1}$. We claim that the subgraph $H := G[\{v_1, \dots, v_{2k+1}\}]$ is the graph $\overline{C}_{2k+1}$ with exactly one additional edge. If there is no additional edge in H, then it follows that $H = \overline{C}_{2k+1}$. This contradicts the assumption that G is chordal and hence perfect. If there are two or more additional edges, then there are at least three vertices in $W \cap V(H)$. Since all the edges between the vertices in W are deleted to obtain G', $\overline{C}_{2k+1}$ cannot be an induced subgraph of G' (in an odd antihole, out of any three vertices, at least two are connected by an edge).

So assume that the additional edge in H is between v_2 and v_3. This implies that the vertices v_2 and v_3 are the only vertices of $V(\overline{C}_{2k+1})$ in W. Then the cycle $C = v_2v_3v_1v_4v_2$ is contained in G and has length four but no chord. Again this contradicts the assumptions and hence G' has no odd antihole. □

Corollary 6.15. *Simultaneous Domination of Spanning Trees restricted to instances with graphs that are chordal, bipartite or have bounded treewidth is contained in* P. □

We now regard two graph classes, claw free and perfect graphs, for which Corollary 6.12 does not apply directly. The results on these two graph classes are not included in [JKS18] and have not been published yet.

We begin by regarding the class of *claw free graphs.* Recall that a graph G is *claw free* if it does not contain a claw as an induced subgraph. Claw free graphs are hereditary, but the deletion of edges of an induced subgraph may create claws. Further, it is currently unknown if Min Vertex Cover is polynomial time solvable on the class of graphs that consists of all claw free graphs and all graphs obtained from a claw free graph by deleting all edges of an induced subgraph. It is well known, however, that Vertex Cover restricted to claw free graphs is contained in P, cf. [GJ79]. Before we prove that we can solve Min-SDST on claw free graphs in polynomial time, we establish two lemmata which are the foundations for the proof.

Lemma 6.16. *Let G be a claw free graph. If $v \in V(G)$ is a cut vertex of G, then it is contained in exactly two blocks, say B_1 and B_2. Further, $N_{B_1}(v)$ and $N_{B_2}(v)$ induce cliques in G.*

Proof. If v was only contained in one block, it would not be a cut vertex. Further, if v is contained in three blocks, say B_1, B_2 and B_3, then for any choice of $w_i \in N_{B_i}(v)$ for $i \in \{1, 2, 3\}$, the set $\{v, w_1, w_2, w_3\}$ induces a claw in G.

Let B_1 and B_2 be the two blocks in G that contain v. If there exist two vertices $w_1, w_2 \in N_{B_1}(v)$ such that the edge $w_1w_2 \notin E(G)$, then the set $\{v, w_1, w_2, w_3\}$ induces a claw in G for any $w_3 \in N_{B_2}(v)$. □

Lemma 6.17. *Let G be a 2-connected graph and $f: V(G) \to \{1, 0, \hat{0}\}$ be a coloring of the vertices, such that for all $v \in V(G)$ with $f(v) = 0$ it holds true that*

$$\hat{N}(v) := \left\{w \in N_G(v) : f(w) = \hat{0}\right\}$$

induces a clique in G. Then there exists a minimum f-respecting simultaneous dominating set S in G, such that $\hat{N}(v) \subseteq S$ for all $v \in V(G)$ with $f(v) = 0$.

Proof. Let S be a minimum f-respecting sd-set in G and assume there exists some $v \in V(G)$ with $f(v) = 0$ such that $\hat{N}(v)$ is not completely contained in S. Then there exists $w \in \hat{N}(v)$ such that $w \notin S$. As $f(w) = \hat{0}$, by Definition 6.7 and Proposition 6.2 all neighbors of w are contained in S. In particular $v \in S$ and as $\hat{N}(v)$ induces a clique in G, also $\hat{N}(v) \setminus \{w\} \subseteq S$. It is now easy to see that the set $S \setminus \{v\} \cup \{w\}$ is also an f-respecting sd-set in G and has the same cardinality. □

Theorem 6.18. *Simultaneous Domination of Spanning Trees restricted to instances with claw free graphs is in* P.

Proof. As claw free graphs are hereditary, any block of a given graph is claw free. Regard any call to crSDS in Algorithm 2 and denote by B and f the block and the coloring passed to crSDS. Define a new coloring $f': V(B) \to \{1, 0, \hat{0}\}$ by setting

$$f'(v) = \begin{cases} 1, & \text{if there exists } vw \in E(B) \text{ such that } f(v) = \hat{0} \text{ and } f(w) = 0 \\ f(v), & \text{else.} \end{cases} \tag{6.1}$$

First note that any f'-respecting simultaneous dominating set is also an f-respecting simultaneous dominating set. Further, as Algorithm 2 only assigns colors 0 and 1 to vertices that were once cut vertices, by Lemma 6.16, we know that the neighborhood of any vertex w with $f(w) = 0$ induces a clique. Thus, by Lemma 6.17 any vertex v with $f'(v) = 1$ and $f(v) = \hat{0}$ can be assumed to be contained in a minimum f-respecting simultaneous dominating set. This implies that the cardinality of a minimum f'-respecting sd-set is the same as the cardinality of a minimum f-respecting sd-set. Thus, we can replace a coloring f in a call to crSDS by the coloring f'. Now, in the procedure crSDS at first all vertices with color 1 are deleted from the graph. As the regarded block is claw free and claw free graphs are hereditary, the remaining graph is also claw free. After the deletion of vertices of color 1, all vertices w with color $f'(w) = 0$ only have neighbors that are also colored 0 by f'. Thus, after deleting the edges between vertices of color 0 each vertex with color 0 has degree 0. Vertices of degree 0 can be disregard for computing a minimum vertex cover.

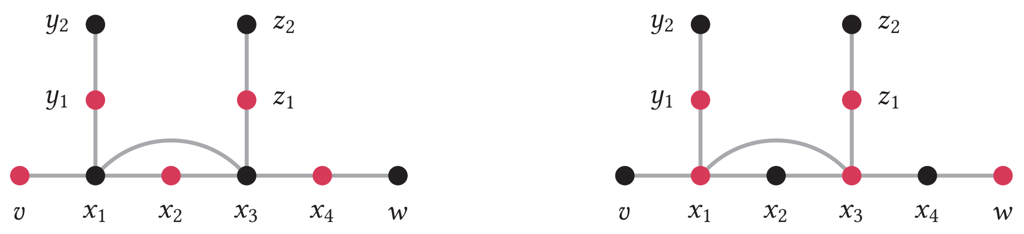

Figure 6.3.: Two sd-sets in the graph H_{vw} from the proof of Theorem 6.19.

The remaining graph is still claw free as it only contains vertices of color $\hat{0}$ and we did not delete any edges between these vertices. As Vertex Cover is contained in P on claw free graphs we can implement MinVertexCover in Algorithm 1 to run in polynomial time for the given class of graphs. □

As of now, we did not see any difference in the complexity of SDST and Vertex Cover. Note that we did not show that SDST is contained in P, when restricted to perfect graphs. We know Vertex Cover is contained in P on perfect graphs. Corollary 6.12 cannot be applied in this case as deleting the edges of an induced subgraph of a perfect graph does not preserve the perfectness and it is currently unknown (and, in fact, unlikely as we see in the following) if Min Vertex Cover can be solved in polynomial time on the obtained class of graphs.

Johann showed, that SDST is NP-complete, when restricted to perfect graphs, cf. [Joh20]. As the result is not published as of yet, we give a simplified version of Johann's proof here.

Theorem 6.19 ([Joh20]). *Simultaneous Domination of Spanning Trees is* NP*-complete when restricted to instances with perfect graphs.*

Proof. By Theorem 6.4, SDST restricted to instances with perfect graphs is contained in NP.

It is well known that Vertex Cover is NP-complete when restricted to 2-connected graphs. So let G be a simple, 2-connected graph and B' the integer of an instance of Vertex Cover. For each edge $vw \in E(G)$, denote by H_{vw} the graph with

$$V(H_{vw}) = \{v, w, x_1, x_2, x_3, x_4, y_1, y_2, z_1, z_2\} \text{ and}$$
$$E(H_{vw}) = \{vx_1, x_1x_2, x_2x_3, x_3x_4, x_4w, x_1x_3, x_1y_1, y_1y_2, x_3z_1, z_1z_2\},$$

where we assume that any vertex in H_{vw} except v and w is unique to that graph. For the construction of a graph H_{vw} see also Figure 6.3. We then regard the graph

$$H = \bigcup_{vw \in E(G)} H_{vw}$$

and show that it is perfect. First note that the only vertices in H that can possibly have degree larger than 5 are the ones also contained in G. As none of these are adjacent in H there does not exist an odd antihole of size 7 or larger. Further, the only cycle completely contained inside some

H_{vw} for some $vw \in E(G)$ is $x_1x_2x_3x_1$ and therefore of length 3. Whenever a chordless cycle passes through a graph H_{vw} it must contain the subpath $vx_1x_3x_4w$ as it otherwise contains the chord x_1x_3. This subpath has length 4 and thereby, all chordless cycles of H with length larger than 3 are of even length. Thus, H does not contain any odd holes and as any odd hole of size 5 is also an odd antihole of size 5 we get that H is perfect by the strong perfect graph theorem.

In the following we prove that H contains an sd-set of cardinality at most $B := B' + 4\,|E(G)|$ if and only if G contains a vertex cover of cardinality at most B'.

So let $S' \subseteq V(G)$ be a vertex cover in G with $|S'| \leq B'$. Consider the set S that contains all vertices in S' and for each graph H_{vw} the vertices x_1, x_3, y_1, z_1 if $v \notin S'$ and x_2, x_4, y_1, z_1 if $v \in S'$. It is $|S| \leq B' + 4\,|E(G)|$ and we claim that S simultaneously dominates the graph H. To this end, regard some $vw \in E(G)$. As $y_1, z_1 \in S$ and x_1 and x_3 are cut vertices in H, by Proposition 6.2, x_1 and x_3 are simultaneously dominated by S. For $x \in \{x_2, x_4, y_1, y_2, z_1, z_2\}$ it holds true that either x itself is contained in S or all neighbors of x are contained in S. Thus, all vertices in H_{vw} except possibly v and w are simultaneously dominated by S by Proposition 6.2. If $v \notin S$, by definition of S, all neighbors of v in H are contained in S and v is simultaneously dominated. If $w \notin S$ it must be the case that $v \in S$ as S' is a vertex cover in G. Again by the definition of S all neighbors of w are then contained in S and w is simultaneously dominated by S. We conclude that S is an sd-set in H.

Now let $S \subseteq V(H)$ be a simultaneously dominating set in H, such that $|S| \leq B = B' + 4\,|E(G)|$. We begin by showing that for each $vw \in E(G)$ we have $|S \cap (V(H_{vw}) \setminus \{v, w\})| \geq 4$. To this end note that

$$|S \cap \{y_1, y_2, z_1, z_2\}| \geq 2. \tag{6.2}$$

Further, x_2 and x_4 are not cut vertices in H as G is 2-connected. Thus, by Proposition 6.2, we either have $x_3 \notin S$ in which case $x_2, x_4 \in S$, or $x_3 \in S$ in which case $x_1 \in S$ or $x_2 \in S$. We conclude that for each $vw \in E(G)$, it holds true that

$$|S \cap V(H_{vw}) \setminus \{v, w\}| \geq 4. \tag{6.3}$$

Now assume there exists an edge $vw \in E(G)$, such that $v, w \notin S$. As G is 2-connected, neither v nor w are cut vertices in H. By Proposition 6.2, this implies that $x_1, x_4 \in S$. As x_2 is not a cut vertex in H we also have $x_2 \in S$ or $x_3 \in S$. By (6.2) we therefore have $S \cap V(H_{vw}) \geq 5$. Replacing the elements in $S \cap V(H_{vw})$ by the elements in the set $\{v, x_2, x_4, y_1, z_1\}$ yields an sd-set of no larger cardinality which contains v. Thus, we may assume that

$$\text{for each } vw \in E(G) \text{ we have } v \in S \text{ or } w \in S. \tag{6.4}$$

Set $S' = S \cap V(G)$. By (6.3), it is $|S'| \leq B'$ and by (6.4), S' is a vertex cover in G. □

The result of Theorem 6.19 is somewhat surprising, as it proves that allowing cut vertices in a graph increases the complexity. In fact SDST can be solved in polynomial time on 2-connected, perfect graphs as a direct consequence of Corollary 6.3. Also the result distinguishes SDST from

Vertex Cover complexity wise, which makes the investigation of the problem worth the while.

6.5. A 2-Approximation

It is well known that Dominating Set may not be approximated within a constant factor, cf. [Fei98]. On the other hand Vertex Cover can be approximated by a factor of 2 using all end vertices of edges in a maximal matching, cf. [Sch03]. Using this approximation combined with Theorem 6.6 we directly get a 4-approximation for SDST. In the following we see that there also exists a 2-approximation for SDST.

The following idea is deduced from a 2-approximation of Vertex Cover using the LP-relaxation of an IP-formulation for the problem, cf. [Sch03]. The approximation for Min-Simultaneous Domination of Spanning Trees is more involved than the approximation for the vertex cover problem and is therefore worth to be described in detail. We begin by formulating an integer program for Min-SDST. Then we use the solution of its LP relaxation to obtain an integral solution of at most twice the optimal objective value of the LP and, thus, also at most twice the optimal objective value of the IP.

The following IP describes Min-SDST for a graph G. Let $\mathrm{CV}(G)$ be the set of cut vertices in G, $\mathrm{NCV}(G) := V(G) \setminus \mathrm{CV}(G)$ and for each $v \in \mathrm{CV}$ denote by $\mathcal{B}_v$ the set of all blocks of G containing v. In the solution the variable x_v states if the vertex v is in the sd-set or not. The variable y_{vB} is only used if v is a cut vertex and states if v is simultaneously dominated by the block B, i.e., $x_w = 1$ for all $w \in N_B(v)$ if $y_{vB} = 1$.

$$
\begin{array}{llrll}
\text{(IP 6.5)} & \min\limits_{x,y} & \sum\limits_{v \in V} x_v & & \text{(6.5a)} \\
& \text{s.t.} & x_v + x_w \geq 1 & \forall v \in \mathrm{NCV}(G),\ w \in N_G(v) & \text{(6.5b)} \\
& & x_w \geq y_{vB} & \forall v \in \mathrm{CV}(G),\ B \in \mathcal{B}_v,\ w \in N_B(v) & \text{(6.5c)} \\
& & \sum\limits_{B \in \mathcal{B}_v} y_{vB} + x_v \geq 1 & \forall v \in \mathrm{CV}(G) & \text{(6.5d)} \\
& & x_v, y_{vB} \in \{0,1\} & \forall v \in V. & \text{(6.5e)}
\end{array}
$$

Lemma 6.20. *Let $G = (V,E)$ be a graph and let $x \in \{0,1\}^{|V|}$. Then the set $S = \{v \in V : x_v = 1\}$ is a simultaneous dominating set of minimum size in G if and only if there exists y such that (x,y) is an optimal solution for (IP 6.5).*

Proof. The lemma follows if we show that the set $S = \{v \in V : x_v = 1\}$ is an sd-set of G if and only if there is a y such that (x,y) is a feasible solution of (IP 6.5).

First let (x,y) be a feasible solution for (IP 6.5) and set $S = \{v \in V : x_v = 1\}$. Note that by (6.5e) the entries in x_v and y_{vB} can only be 0 or 1. By (6.5b) we have for every non-cut vertex that either itself or all its neighbors are contained in S which implies condition *(i)* of Proposition 6.2. For a cut vertex v, conditions (6.5d) ensure that v is contained in S or that for at least one block B containing v it is $y_{vB} = 1$. By (6.5c) we have $x_w = 1$ for all $w \in N_B(v)$ if $y_{vB} = 1$. Thus, condition *(ii)* of Proposition 6.2 is also implied and it follows that S is a feasible sd-set.

Now suppose that S is a feasible sd-set. Set $x_v = 1$ if $v \in S$ and $x_v = 0$ otherwise. For every cut vertex v we have that v itself is in S or it is simultaneously dominated, *i.e.* there is a block B such that all neighbors of v in B are in S. We set $y_{vB} = 1$ if and only if the latter case is true. This immediately shows that (6.5c) and (6.5d) are fulfilled. The conditions (6.5b) are also satisfied by condition *(i)* of Proposition 6.2. This shows that (x, y) is a feasible solution for (IP 6.5). □

In the LP-relaxation of (IP 6.5), we require the variables x and y to have real values greater or equal to 0 and less or equal to 1. We refer to the LP-relaxation by (LP 6.5). In the following we describe a procedure that rounds non-integral variables in an optimal solution to (LP 6.5) yielding an integral solution to (IP 6.5) that has objective value at most twice as large.

Let (x, y) be an optimal solution for the LP. Before we start, let us describe the outline of the rounding procedure. At first we round up at least one of the two variables in each constraint (6.5b). Afterwards we regard (6.5c) and (6.5d) and ensure their validity for the cut vertices of G. To do so we use the block-cutpoint tree T of G. We regard the cut vertices of G bottom up in the tree T and if necessary round up the variable of the cut vertex itself, while decreasing some values of neighbors of the cut vertex in order to maintain the approximation quality. During all rounding steps we ensure that the current solution remains feasible for (LP 6.5) such that after making all variables integral the resulting solution automatically induces an sd-set. Any variable that is at some point set to 1 is never changed again, implying that only fractional variables are rounded down. We will now formally describe the three different rounding steps.

First Rounding Step For all $v \in V$ we round up all x_v to 1 which have a value greater or equal to $1/2$. Moreover for each cut vertex v and each block B with $v \in V(B)$ we set

$$y_{vB} := \min \{x_w \colon w \in N_B(v)\} . \tag{6.6}$$

Whenever we change the value of a variable x_v in any rounding step we update all respective variables y_{wB}, such that (6.6) remains valid throughout the whole procedure. We ensure the following statement throughout the rounding procedure.

$$\text{Any variable with a value larger or equal to } 1/2 \text{ has value } 1. \tag{6.7}$$

In any solution to (LP 6.5) one of the two variables in a constraint (6.5b) has value at least $1/2$. Thus, after the first rounding step and by the fact that we never round down any variable x_v that is set to 1, constraints (6.5b) remain valid throughout the whole rounding procedure. Further, as we always update the y variables according to (6.6), constraints (6.5c) are never violated by (x, y) at any stage of the rounding procedure.

Now regard the block-cutpoint tree T of G and root it at any cut vertex $r \in \mathrm{CV}(G)$. It is easily observed that we may now iteratively choose a cut vertex v such that all descendants of v in T that are cut vertices have already been regarded. If for some block B containing v we have $y_{vB} \geq \frac{1}{2}$, by (6.7), it holds true that $y_{vB} = 1$. This implies that vertex v is simultaneously dominated by block B and we can safely go to the next cut vertex. So assume that $y_{vB} < \frac{1}{2}$ for all blocks containing v. We denote by B' the parent of v in T and by $B_1, \dots, B_k$ its children. As $y_{vB'} < \frac{1}{2}$, by constraint (6.5d) it holds true that

$$x_v + \sum_{i=1}^{k} y_{vB_i} \geq \frac{1}{2}. \tag{6.8}$$

Second Rounding Step For every cut vertex v moving bottom up in the block-cutpoint tree T of G, test if $y_{vB} = 1$ for some block B containing v. If none such block exists, set $x_v = 1$ and $x_{w_i} = 0$ for all $i = 1, \dots, k$ and some $w_i \in N_{B_i}(v)$ with x-value $\min \{x_w \colon w \in N_{B_i}(v)\} = y_{vB_i}$.

Note that any cut vertex w that is a descendant of v in T was processed in the rounding procedure before v. Thus, if x_w has a fractional value when rounding x_v, it must be the case that $y_{wB} = 1$ for some block B containing w, as otherwise we would have set x_w to 1. Decreasing the x variables during the second rounding step does therefore not violate any constraint (6.5d) and the solution (x, y) is still feasible for (LP 6.5).

Third Rounding Step Decrease all remaining fractional variables to 0.

After processing the root r of T in the second rounding step, for each cut vertex v in G it holds true that $x_v = 1$ or $y_{vB} = 1$ for some block B containing v. Thus, constraints (6.5d) are fulfilled after the third rounding step and (x, y) is feasible for (IP 6.5).

Theorem 6.21. *The procedure that solves (LP 6.5) and applies the described three rounding steps to the solution is a 2-approximation algorithm for* Min-Simultaneous Domination of Spanning Trees.

Proof. Let G be a graph and denote by (x, y) a solution of (LP 6.5) for G that was rounded according to the three rounding steps described before. We already argued above, that applying the three rounding steps to the solution of (LP 6.5) maintains feasibility of the solution. As all values of (x, y) are integral after rounding, it holds true that (x, y) is a solution to (IP 6.5). In the first rounding step we only increase variables that have value larger or equal to $1/2$. In the second rounding step, if we increase a variable x_v to 1, we also decrease a set of variables $\{x_{w_1}, \dots, x_{w_k}\}$ to 0. By (6.8) the total value of the affected variables before the rounding is at least $1/2$. Thus, the rounding at most doubles their summed up value. The third rounding step then only decreases the objective value. Thus, The obtained solution (x, y) induces a simultaneously dominating set $S = \{v \in V(G) \colon x_v = 1\}$ such that for any minimum simultaneous dominating set $S^\star$ it holds true that $|S| \leq 2 |S^\star|$.

We can compute an optimal solution to (LP 6.5) in polynomial time, cf. [GLS88]. It is easy to see that the rounding steps can also be implemented to run in polynomial time, taking into account that we can obtain the block-cutpoint tree of a graph G in linear time in the size of the graph G, cf. [HT73]. □

Conclusion In this chapter we thoroughly investigated Simultaneous Domination of Spanning Trees. On 2-connected graphs the problem revealed itself to be equivalent to Vertex Cover. After providing an algorithm that reduces the problem to finding a Vertex Cover on subgraphs of the blocks of a graph, we argued how we can use this to prove Simultaneous Domination of Spanning Trees to be contained in P when restricted to graphs that are bipartite, chordal, claw free or have bounded treewidth. Somewhat surprisingly we saw, that Simultaneous Domination of Spanning Trees is NP-complete when restricted to perfect graphs, whereas Vertex Cover is contained in P on the very same class of graphs. Finally we showed that for general graphs we may approximate Min-Simultaneous Domination of Spanning Trees by a factor of 2. A direction of further research could be to regard simultaneous domination of other classes. For example it could be interesting to consider the problem of finding a minimum size set on all cycles in a graph. It is very likely that some of the techniques described in this chapter can be transferred to other simultaneous dominating set problems.

Part II.

Uncertainty in the Demand: Robust Approaches to Covering and Facility Location Problems

Chapter 7.

q-Multiset Multicover

*In this chapter we discuss q-*Multiset Multicover *which is a covering problem in which every set may only cover q of its elements. We prove that it is* NP*-complete in the strong sense for any fixed $q \geq 3$, but polynomial time solvable for $q \in \{1, 2\}$. We introduce a robust version of q-*Multiset Multicover*, and prove that this version is* NP*-hard in the strong sense for any $q \in \mathbb{N}_{>0}$. Further we discuss general solution techniques based on integer programming and constraint generation. For specific choices of uncertainty sets, we see that* Robust 1-Multiset Multicover *remains* NP*-hard even when the uncertainty set is restricted to contain only three elements. We regard further common uncertainty sets, analyze their complexity, and discuss possible improvements in the solution process.*

The results of this chapter have partially been published in [KSS19]. In particular this applies to the results on q-Multiset Multicover and the results on Robust q-Multiset Multicover with *budgeted uncertainty.* All of the results in this chapter are joint work with Eva Schmidt and Sven O. Krumke.

As the literature reviews for Chapter 7 and 8 overlap in large parts, it is presented in the introduction to Part II.

Outline We begin this chapter with the formal definition, integer programming formulations of the problem and classify it by complexity class for all $q \in \mathbb{N}_{>0}$. In Section 7.2 we repeat this process for q-Multiset Multicover including uncertainty in the demand of the elements to be covered, where we only make very rough assumptions on the given uncertainty sets. Before we discuss complexity and solution approaches for specific uncertainty sets in Section 7.4, we briefly discuss general solution approaches independent of the choice of uncertainty set in Section 7.3.

7.1. Problem Definition and Classification

In the first section of this chapter we formally define q-Multiset Multicover, give different integer programming formulations for the problem and classify it by complexity class for all integers $q \in \mathbb{N}_{>0}$, where we give polynomial time algorithms whenever the problem is contained in P. Given a finite ground set and a collection of subsets, in q-Multiset Multicover we ask if there exist B subsets with multiple choices being allowed, such that the demand of each element is covered, when each subset may only cover up to q of its elements (again multiple choices are allowed). We now formally define q-Multiset Multicover for a fixed integer $q \in \mathbb{N}_{>0}$.

***q*-Multiset Multicover (*q*-MsMc).**
Instance: A finite set J, weights $d_j \in \mathbb{N}$ for all $j \in J$, a collection of subsets $\mathcal{J} \subseteq 2^J$, and an integer $B \in \mathbb{N}$.
Question: Does there exist $x \in \mathbb{N}^{|\mathcal{J}|}$ with $x(\mathcal{J}) \leq B$ such that there exists $y \in \mathbb{N}^{|\mathcal{J}|\times|J|}$ satisfying

$$\sum_{A\in\mathcal{J}\colon j\in A} y_{Aj} \geq d_j \quad \forall j \in J \quad \text{and} \quad \sum_{j\in A} y_{Aj} \leq q \cdot x_A \quad \forall A \in \mathcal{J}?$$

For a fixed subset A, the integer y_{Aj} in the problem definition models the amount of demand of element j covered by the subset A. If, instead of regarding the subsets $A \in \mathcal{J}$, we regard all multisets of cardinality q of A, we get an instance of Multiset Multicover, raising the input size only by a polynomial factor as q is not part of the input. Thereby, q-MsMc is a representation of certain Multiset Multicover instances having smaller input size.

In the sequel, it is useful to model an instance of q-MsMc as a bipartite graph with weights. We set G to be the simple bipartite graph with bipartition $I \,\dot\cup\, J$, where I is an index set for the collection $\mathcal{J} = \{A_i\colon i \in I\}$ and the neighborhood of $i \in I$ in G is exactly A_i. Using this we get the following alternative definition of q-MsMc. This definition eases notation in many cases and, in the author's opinion, gives the better intuition for the problem.

***q*-Multiset Multicover (*q*-MsMc).**
Instance: Finite, disjoint sets I, J, weights $d_j \in \mathbb{N}$ for all $j \in J$, a bipartite graph $G = (I \,\dot\cup\, J, E)$, and an integer $B \in \mathbb{N}$.
Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \leq B$ such that there exists $y \in \mathbb{N}^{|I|\times|J|}$ satisfying

$$\sum_{i\in N(j)} y_{ij} \geq d_j \quad \forall j \in J \quad \text{and} \quad \sum_{j\in N(i)} y_{ij} \leq q \cdot x_i \quad \forall i \in I?$$

Let us regard a small example so we may better grasp the equivalence of the two definitions of q-MsMc.

Example 7.1. Regard a collection of subsets $\mathcal{J} = \{\{1, 2, 3\}, \{2, 4, 6\}, \{5, 6\}\}$ on the ground set $J = \{1, \dots, 6\}$. If these are part of an instance of q-MsMc, we may also regard them as the bipartite graph G depicted in Figure 7.1. Instead of choosing integers x_A for the subsets $A \in \mathcal{J}$ we now choose integers x_i for the vertices $i \in I$ that belong to the left partition in Figure 7.1.

Arising from facility location problems we use the following notations for the remainder of this thesis.

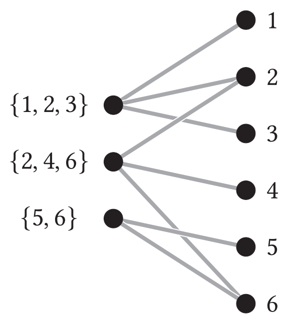

Figure 7.1.: Graph of q-MsMc instance from Example 7.1.

Notation 7.2. For an instance of q-MsMc we call the set I *locations* and the set J *regions*. Further, d_j describes the number of *clients* or the *demand* in region $j \in J$. The integers x_i denote the number of *suppliers* in location $i \in I$. The number q is interpreted as the number of clients a single supplier may serve.

In the optimization version Min-q-Multiset Multicover (Min-q-MsMc) we aim for a minimum number of suppliers. It can readily be seen that the following integer program models Min-q-MsMc. We label the MIP in dependence on a demand vector $d \in \mathbb{N}^{|J|}$ for reference purposes.

$$
\begin{aligned}
&\text{(MIP 7.1)}(d) \quad \min_{x,y} && \sum_{i\in I} x_i && && (7.1a)\\
&\text{s.t.} && \sum_{i\in N(j)} y_{ij} \geq d_j && \forall j \in J && (7.1b)\\
& && \sum_{j\in N(i)} y_{ij} \leq q \cdot x_i && \forall i \in I && (7.1c)\\
& && y_{ij} \geq 0 && \forall i \in I,\ j \in J && (7.1d)\\
& && x_i \in \mathbb{N} && \forall i \in I. && (7.1e)
\end{aligned}
$$

Note that the variables y_{ij} are not forced to be integral. Lemma 7.5 argues why this is no restriction.

Looking at (MIP 7.1) another resemblance to a well known optimization problem is revealed: The capacitated facility location problem. One could think of q-MsMc as a facility location problem, where the facility capacities are all fixed to a constant $q \in \mathbb{N}_{>0}$. Another difference to standard capacitated facility location problems is that in q-Multiset Multicover the variables y do not appear in the objective function. This will be of importance for the subsequent reformulation of (MIP 7.1). Also, this is the main reason why we deem q-MsMc a covering rather than a facility location problem.

As the y variables do not appear in the objective function they can be interpreted as auxiliary variables, which are not necessarily interesting for a solution to the problem. This fact leads us to

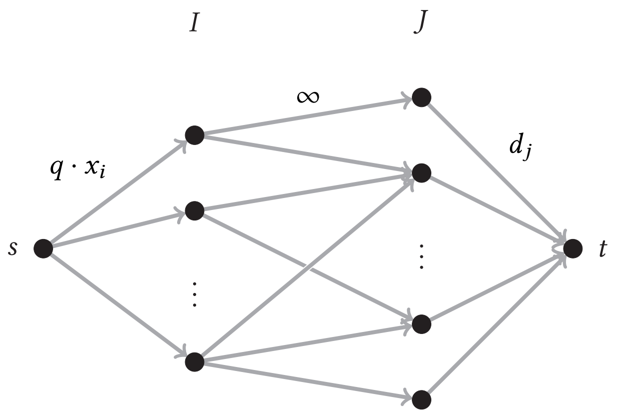

Figure 7.2.: Flow network from Definition 7.4.

the alternative integer programming formulation (IP 7.2)(d) of q-Multiset Multicover, which is proved to be equivalent to (MIP 7.1)(d) in Lemma 7.5.

$$\text{(IP 7.2)}(d) \quad \min_{x} \quad \sum_{i \in I} x_i \tag{7.2a}$$

$$\text{s.t.} \quad \sum_{i \in N(S)} q \cdot x_i \geq \sum_{j \in S} d_j \quad \forall S \subseteq J \tag{7.2b}$$

$$x_i \in \mathbb{N} \qquad \forall i \in I. \tag{7.2c}$$

Before we prove the equivalence of the two formulations, we observe trivial lower and upper bounds for an optimal solution and define a flow network that will be used in the proof of Lemma 7.5 and again later on in this chapter.

Observation 7.3. *Let an instance of Min-q-Multiset Multicover be given and let $x^\star$ be an optimal solution to the instance. It holds true that*

$$\sum_{i \in I} x_i^\star \geq \left\lceil \frac{\sum_{j \in J} d_j}{q} \right\rceil \quad \text{and} \quad x_i^\star \leq \left\lceil \frac{\sum_{j \in N(i)} d_j}{q} \right\rceil \quad \forall i \in I.$$

Definition 7.4. Let an instance $\mathcal{I}$ of q-Multiset Multicover be given. We define the *flow network* corresponding to $\mathcal{I}$ to be the directed graph

$$H_{\mathcal{I}} = (I \cup J \cup \{s\} \cup \{t\}, R \cup R_s \cup R_t),$$

where $R = \{ij : ij \in E(G)\}$, $R_s = \{si : i \in I\}$ and $R_t = \{jt : j \in J\}$. Further we define a capacity function $c\colon R(H_{\mathcal{I}}) \to \mathbb{Z} \cup \{\infty\}$ by setting

$$c(r) = \begin{cases} \infty, & r \in R, \\ q \cdot x_i, & r \in R_s, \\ d_j, & r \in R_t. \end{cases}$$

for each $r \in R(H_{\mathcal{I}})$. See also Figure 7.2 for the construction of $H_{\mathcal{I}}$.

Lemma 7.5. *The LP-relaxations of (MIP 7.1) and (IP 7.2) are equivalent in the following sense: For an instance of* Min-*q*-MsMc *it holds true that $x \in \mathbb{R}^{|I|}$ is a feasible solution to the LP-relaxation of (IP 7.2)(d) if and only if there exists $y \in \mathbb{R}^{|I|\times|J|}_{\geq 0}$ such that (x, y) is a feasible solution for the LP-relaxation of (MIP 7.1)(d). If in this case $x \in \mathbb{N}$, the values of the variables y can also be chosen integral.*

Proof. We denote by (LP 7.1) and (LP 7.2) the LP-relaxations of (MIP 7.1) and (IP 7.2). Let $\mathcal{I}$ be an instance of q-MsMc. If (x, y) is a feasible solution for (LP 7.1)(d) then x is also feasible for (LP 7.2)(d), as for any $S \subseteq J$ we have:

$$\begin{aligned}\sum_{i\in N(S)} q \cdot x_i \geq \sum_{i\in N(S)} \sum_{j\in N(i)} y_{ij} &= \sum_{i\in N(S)} \left(\sum_{j\in N(i)\cap S} y_{ij} + \sum_{j\in N(i)\setminus S} y_{ij} \right) \\ &\geq \sum_{i\in N(S)} \sum_{j\in N(i)\cap S} y_{ij} = \sum_{j\in S} \sum_{i\in N(j)} y_{ij} \geq \sum_{j\in S} d_j.\end{aligned}$$

Now assume we are given a feasible solution x for (LP 7.2)(d). Let $H := H_{\mathcal{I}}$ be the flow network corresponding to $\mathcal{I}$, cf. Definition 7.4. We claim that any maximum s-t-flow in H has flow value $d(J)$. Note that given an s-t-flow f with flow value $d(J)$ the solution (x, y) with $y_{ij} = f(ij)$ for all $ij \in R$ is feasible for (LP 7.1)(d).

The flow value of any s-t flow cannot be larger than $d(J)$ (consider the s-t cut with $T = \{t\}$). Thus, it suffices to show that a maximum s-t flow in H has flow value no less than $d(J)$. To this end let $S, T \subseteq V(H)$ be any s-t cut in H. Let $J' = J \setminus S$, possibly being the empty set. If any location in the neighborhood of J' is contained in S, the cut contains an arc with infinite capacity. Thus, we may assume $N^-_H(J') \cap S = \emptyset$, which implies $N^-_H(J') \subseteq T$. Since x is a feasible solution to (IP 7.2)(d) we obtain for any subset $Q \subseteq J$

$$\sum_{i\in N^-_H(Q)} q \cdot x_i = \sum_{i\in N_G(Q)} q \cdot x_i \geq \sum_{j\in Q} d_j.$$

This in turn implies

$$c(S, T) \geq \sum_{j\in J\cap S} d_j + \sum_{i\in N^-_H(J')} q \cdot x_i \geq \sum_{j\in J\cap S} d_j + \sum_{j\in J'} d_j = \sum_{j\in J} d_j.$$

Thus, every s-t cut has capacity larger or equal to $d(J)$ and by the *Max-Flow-Min-Cut Theorem* we obtain the desired result, cf. [AMO93].

Note that the capacities of the arcs in R_s and R_t from Definition 7.4 are integral. Thus, there exists an integral s-t flow f in H if and only if there exists a continuous s-t flow f' in H, cf. [AMO93]. Thereby, for any optimal solution (x, y) to (MIP 7.1), we can find an optimal solution (x, y'), where y' is integral. □

We now have valid integer programming formulations for q-MsMc. Note that Lemma 7.5 still holds if we do not require the x variables to have integral values. Thus, the LP-relaxations of (MIP 7.1) and (IP 7.2) also have the same objective value. One might raise the question, why a reformulation of polynomial many constraints (7.1b)-(7.1d) to exponentially many constraints (7.2b) is of any use. In fact, for q-MsMc the reformulation is rather of theoretical interest. However, when regarding demand uncertainty in the subsequent sections of this chapter, the reformulation is useful in the solution process of q-MsMc.

As of now, we may still hope for polynomial time algorithms solving q-MsMc, as we know nothing about the complexity of the decision problem. In the remainder of this section we see that for all $q \geq 3$ we will most likely not be able to find polynomial time algorithms, as q-MsMc turns out to be NP-complete in the strong sense in this case. We give polynomial time algorithms for solving Min-1-MsMc and Min-2-MsMc though.

Proposition 7.6. *Min-1-Multiset Multicover is solvable in time $O(|I| + |J|)$.*

Proof. Given an instance for Min-1-MsMc in any solution each client needs to be assigned a unique supplier. This means for each client in some region $j \in J$ we may put a single supplier in some location $i \in N(j)$. We get a feasible solution with $d(J)$ suppliers, which is optimal by Observation 7.3. We can find this solution in time $O\left(|I| + |J|\right)$. □

Regarding Min-2-MsMc we observe that placing a supplier in some location $i \in I$ could be interpreted as choosing two regions in the neighborhood of i. There seems to be a close relation to edge covers. Recall that for a graph $G = (V, E)$ an *edge cover* is a subset of the edges $E' \subseteq E$ such that each vertex $v \in V$ is incident to at least one edge $e \in E'$. In fact we can use an algorithm for Edge Cover in order to find an optimal solution to Min-2-MsMc.

Theorem 7.7. *Min-2-Multiset Multicover can be solved in* $O(|I|^{5/2}|J|^{5/2})$.

Proof. Let an instance of Min-2-MsMc be given. We begin by arguing that for any $j \in J$ we may assume that $d_j \leq |I|$. Suppose $d_j \geq |I| + 1$ for some $j \in J$. Then, in any optimal solution (x, y) to (MIP 7.1) there is some region j and location $i \in N(j)$ such that $y_{ij} \geq 2$. Removing one supplier from such a location i yields an optimal solution to the same instance with the demand of region j being $d_j - 2$. As a consequence we may also solve this instance and then afterwards add a supplier to some location connected to j in order to get an optimal solution of the original problem. We can therefore decrease the demand of all j with $d_j \geq |I| + 1$ to $|I|$, respectively $|I| - 1$ by adding $\lceil 1/2\,(d_j - |I|) \rceil$ suppliers to some location connected to j. This can be done in constant time for any region $j \in J$.

Now regard the following procedure: For a given instance of the problem, duplicate each region $j \in J$ exactly d_j times yielding a set V_j for each $j \in J$. Regard the graph $H = (V, E')$ with vertex set $V = \bigcup_{j \in J} V_j$ where, for two vertices $u \in V_{j_1}$ and $v \in V_{j_2}$, the edge uv is contained in E' if $N_G(j_1) \cap N_G(j_2) \neq \emptyset$. Note that this implies that the graph induced by some set V_j is the complete graph. Next, compute a minimum edge cover $F \subseteq E'$ in H and initially set $x_i = 0$ for all $i \in I$. For each edge $uv \in F$, with $u \in V_{j_1}$, $v \in V_{j_2}$ we increase x_i by 1 for some $i \in N_G(j_1) \cap N_G(j_2)$, meaning we add a supplier in location i who serves one client in region j_1 and one in region j_2.

We first prove the correctness of the procedure. Let $H = (V, E')$ be the graph and x be the solution defined in the procedure above. Denote by F the minimum edge cover from the procedure. As there is a vertex in H for every client and the vertices corresponding to the clients are covered by the edges in F it is clear that x defines a feasible solution for Min-q-MsMc. It remains to show that, given a solution $x \in \mathbb{N}^{|I|}$ to Min-q-MsMc, there is an edge cover with $x(I)$ edges. As x is a solution to Min-q-MsMc we can find $y_{ij} \in \mathbb{N}$ for all $i \in I$, $j \in J$ fulfilling

$$\sum_{i \in N(j)} y_{ij} \geq d_j \ \forall j \in J \text{ and } \sum_{j \in N(i)} y_{ij} \leq 2x_i \ \forall i \in I.$$

Further, we may assume equality in the latter set of equations and can thereby determine for each supplier $i \in I$ the two regions $j_1, j_2 \in J$ he serves. We initially set F to the empty set. For each supplier, we now select an edge between two vertices of V_{j_1} and V_{j_2}, and add it to F. As for each $j \in J$ it holds true that

$$\sum_{i \in N(j)} y_{ij} \geq d_j$$

we may choose the edges, such that F is an edge cover of H with $x(I)$ edges. This proves the correctness of the procedure.

To see the running time of the procedure note that the preprocessing step reducing each d_j for $j \in J$ to at most $|I|$ can be implemented to run in time $O(|J|)$. The constructed graph then has at most $N := |J| \cdot (|I| + 1)$ vertices whereas the number M of edges is upper bounded by $O(|I|^2|J|^2)$. A minimum edge cover in a graph with N vertices and M edges can be obtained by first solving a maximum matching problem in time $O(\sqrt{N}M\log_N(N^2/M))$ [GK95] and then using $O(M)$ time to augment the matching [Sch03; LP86]. This gives the claimed running time. □

The NP-completeness of q-MsMc for $q \geq 3$ is a consequence of its close resemblance to Set Cover, which is well known to be NP-complete,cf. [GJ79]. We give a formal proof for the sake of completeness.

Theorem 7.8. *For any fixed $q \geq 3$, q-Multiset Multicover is* NP*-complete in the strong sense.*

Proof. Let $q \in \mathbb{N}$ with $q \geq 3$. As a consequence of Lemma 7.5, for a given instance of q-MsMc, we may test if a given solution x is feasible in polynomial time by one Max-Flow computation. Therefore, q-MsMc is contained in NP.

To see that the problem is NP-hard in the strong sense we illustrate a reduction from Exact Cover by 3-sets, which is known to be NP-hard in the strong sense, cf. [GJ79]. Let an instance of Exact Cover by 3-sets be given. We create an instance of q-MsMc in the following way. Let $I = \mathcal{S}$, $J := U$ and define the graph of the instance via $N(S) = S$ for all $S \in \mathcal{S}$. Further, let $d_j = 1$ for all $j \in J$ and $B = |U|/3$. Now, let $\mathcal{S}'$ be a solution to the instance of Exact Cover by 3-sets. Setting x_S to one if and only if $S \in \mathcal{S}'$ and zero else yields a feasible solution to q-MsMc with $x(\mathcal{S}) = B$. On the other hand, note that in any feasible solution x to q-MsMc with $x(\mathcal{S}) \leq B$, it is $x_S \leq 1$ for all $S \in \mathcal{S}$. Thus, $\mathcal{S}' = \{S : x_S > 0\}$ is a solution to Exact Cover by 3-sets. □

We already observed before that Multiset Multicover is a generalization of q-MsMc. It is well known that Min-Multiset Multicover can be approximated within a factor of $\log(s)$ where s is the size of the largest multiset of an instance, cf. [Dob82]. If we regard Min-q-MsMc as Min-Multiset Multicover problem, all multisets have fixed size q. We therefore automatically get a $\log(q)$-approximation for Min-q-MsMc:

Observation 7.9. *There is a $\log(q)$-approximation for Min-q-MsMc.*

7.2. Problem Definition and Classification Including Demand Uncertainty

In this section, we extend q-Multiset Multicover to include uncertainty in the number of clients d_j of each region $j \in J$. In particular this means that, instead of being given one fixed demand vector in an instance, the demand vectors are now subject to uncertainty and are only known to be contained in some uncertainty set $\mathcal{U}$. In a solution to Robust q-MsMc we are looking for a distribution of suppliers, such that the demand can be satisfied in every possible scenario $\xi \in \mathcal{U}$. It now becomes evident that the y variables can be considered to be auxiliary as we are not interested in the actual assignment of clients to suppliers in every possible scenario. We are merely looking for a certificate that when the actual scenario reveals itself we are able to find such an assignment.

Here and in Section 7.3, we make very rough restrictions on the given uncertainty set such as its encoding length or its polynomial time enumerability. In Section 7.4 we then discuss specific uncertainty sets in order to improve our results for general uncertainty sets. As is customary in robust optimization, we are looking for solutions that are feasible for all possible realizations of the uncertainty set $\mathcal{U}$. This results in the definition of the following robust version of q-Multiset Multicover.

Robust q-Multiset Multicover (Robust q-MsMc).
Instance: Finite, disjoint sets I, J, an uncertainty set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \mathbin{\dot\cup} J, E)$, and a positive integer $B \in \mathbb{N}_{>0}$.
Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \leq B$ such that for each $\xi \in \mathcal{U}$ there exists $y(\xi) \in \mathbb{N}^{|I| \times |J|}$ satisfying

$$\sum_{i \in N(j)} y(\xi)_{ij} \geq \xi_j \qquad \forall j \in J, \xi \in \mathcal{U} \quad \text{and}$$
$$\sum_{j \in N(i)} y(\xi)_{ij} \leq q \cdot x_i \qquad \forall i \in I, \xi \in \mathcal{U}?$$

We begin by providing integer programming formulations for Robust Min-q-MsMc. Observe that (MIP 7.3) is a valid formulation for Robust Min-q-MsMc.

$$\begin{aligned}
\text{(MIP 7.3)}(\mathcal{U}) \quad \min_{x,y} \quad & \sum_{i \in I} x_i && && \text{(7.3a)}\\
\text{s.t.} \quad & \sum_{i \in N(j)} y_{ij}(\xi) \geq \xi_j && \forall j \in J,\ \xi \in \mathcal{U} && \text{(7.3b)}\\
& \sum_{j \in N(i)} y_{ij}(\xi) \leq q \cdot x_i && \forall i \in I,\ \xi \in \mathcal{U} && \text{(7.3c)}\\
& y_{ij}(\xi) \geq 0 && \forall i \in I,\ \forall j \in J,\ \xi \in \mathcal{U} && \text{(7.3d)}\\
& x_i \in \mathbb{N} && \forall i \in I. && \text{(7.3e)}
\end{aligned}$$

In a next step we reformulate (MIP 7.3) in the same manner as (MIP 7.1) in the previous section. Although it is a direct consequence of Lemma 7.5, we prove the equivalence of the two formulations for the sake of completeness.

$$\begin{aligned}
\text{(IP 7.4)}(\mathcal{U}) \quad \min_{x} \quad & \sum_{i \in I} x_i && && \text{(7.4a)}\\
\text{s.t.} \quad & \sum_{i \in N(S)} q \cdot x_i \geq \sum_{j \in S} \xi_j && \forall S \subseteq J,\ \forall \xi \in \mathcal{U} && \text{(7.4b)}\\
& x_i \in \mathbb{N} && \forall i \in I. && \text{(7.4c)}
\end{aligned}$$

Theorem 7.10. *The LP-relaxation of (MIP 7.3) and (IP 7.4) are equivalent in the following sense: Given an instance of Robust Min-q-MsMc it holds true that $x \in \mathbb{R}^{|I|}$ is a feasible solution to the*

LP-relaxation of (IP 7.4) if and only if for each $\xi \in \mathcal{U}$, there exists $y(\xi) \in \mathbb{R}_{\geq 0}^{|I| \times |J|}$ such that (x, y) is a feasible solution to the LP-relaxation of (MIP 7.3). If in this case the values of the x variables are integral, the values of the y variables can also be chosen integral.

Proof. Denote by (LP 7.3) and (LP 7.4) the LP-relaxations of (MIP 7.3) and (IP 7.4). Assume $x \in \mathbb{R}^{|I|}$ is a feasible solution to (LP 7.4). We fix some scenario $\xi \in \mathcal{U}$. Then x is also feasible for (LP 7.2)(ξ). By Lemma 7.5 there exist $y(\xi) \in \mathbb{R}_{\geq 0}^{|I| \times |J|}$ such that $(x, y(\xi))$ is feasible for (LP 7.1)(ξ). Defining such variables $y(\xi)$ for all $\xi \in \mathcal{U}$ we get a feasible solution (x, y) for (LP 7.3). Reversing these arguments we can also show that for any feasible solution (x, y) for (LP 7.3) it holds true that x is feasible for (LP 7.4). □

Looking at the constraints (7.4b) in (IP 7.4) we can see that, for fixed $S \subseteq J$, there is only one relevant scenario $\xi \in \mathcal{U}$, namely the one maximizing the sum $\xi(S)$. This leads to the following observation.

Observation 7.11. *Replacing the constraints* (7.4b) *in (IP 7.4) by*

$$\sum_{i \in N(S)} q \cdot x_i \geq \max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in S} \xi_j \right\} \qquad \forall S \subseteq J,$$

does not change the set of feasible solutions of (IP 7.4).

Observation 7.11 gives us a new integer programming formulation for Robust q-MsMc. We state it here for the sake of completeness and reference purposes. As it will be of use later on, we define the integer program for any collection of subsets $\mathcal{J} \subseteq 2^J$. To get a valid formulation for Robust q-MsMc, we set $\mathcal{J} = 2^J$.

$$\text{(IP 7.5)}(\mathcal{J}) \quad \min_x \quad \sum_{i \in I} x_i \tag{7.5a}$$

$$\text{s.t.} \quad \sum_{i \in N(S)} q \cdot x_i \geq \max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in S} \xi_j \right\} \quad \forall S \in \mathcal{J} \tag{7.5b}$$

$$x_i \in \mathbb{N} \qquad \forall i \in I. \tag{7.5c}$$

We are now ready to analyze the complexity of Robust q-MsMc. As we already proved q-MsMc to be NP-complete for any fixed $q \geq 3$ it is not surprising that the robust version of the problem is also NP-complete for $q \geq 3$. Unfortunately, the polynomial time solvability for the cases $q = 1$ and $q = 2$ does not transfer.

Theorem 7.12. *Robust q-Multiset Multicover is* NP*-hard in the strong sense for any fixed* $q \in \mathbb{N}_{>0}$.

Proof. We show that there exists a polynomial time reduction from Dominating Set to Robust q-MsMc. Recall that a *dominating set* in a graph is a subset of the vertices, such that each vertex is

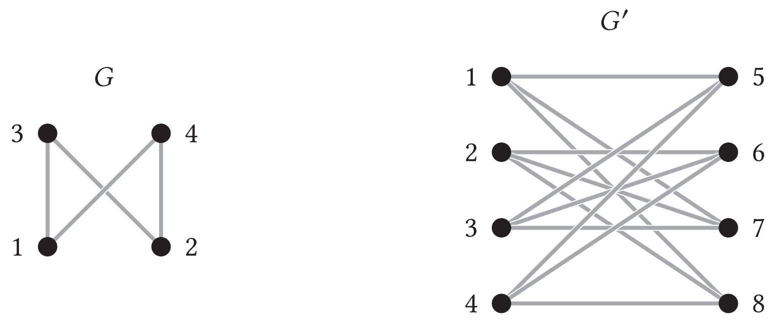

Figure 7.3.: An example of the construction of the graph G' given the displayed graph G in the proof of Theorem 7.12.

either in the set or has a neighbor in the set. To this end, let an undirected graph G with $V(G) = \{1, \dots, n\}$ and an integer $k \in \mathbb{N}$ from an instance of Dominating Set be given. To construct an instance of Robust q-MsMc we set $I = \{1, \dots, n\}$ and $J = \{n+1, \dots, 2n\}$. For every edge $uv \in E(G)$, we add the edges $u(v+n)$ and $v(u+n)$ to the bipartite graph G' with vertex set $I \cup J$. Additionally, for every $v \in V$, the edge $v(n+v)$ is added to the edge set of G'. For the construction of G' see also Figure 7.3. Moreover, we define the uncertainty set to be the set of unit vectors $\mathcal{U} = \left\{\xi \in \mathbb{N}^{|J|} : \xi(J) = 1\right\}$ and set $B = k$.

Let $T \subseteq V(G) = I$ be a dominating set in G such that $|T| \leq k$. We set $x_i = 1$ for all $i \in T$ and $x_i = 0$ else. As x fulfills $x(I) \leq B$, by Observation 7.11 it remains to be shown that, for each subset $S \subseteq J$, we have

$$\sum_{i \in N_{G'}(S)} q \cdot x_i \geq \max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in S} \xi_j \right\}. \tag{7.6}$$

If $S = \emptyset$ the constraint is trivially fulfilled. So assume $S \neq \emptyset$ and choose some arbitrary element $n + v \in S$. As v is dominated in G there exists some $u \in N_G(v) \cup \{v\}$ that is contained in T and therefore $x_u = 1$. By the construction of the graph we also have $u \in N_{G'}(n+v)$. Thus, it holds true that

$$\sum_{i \in N_{G'}(S)} q \cdot x_i \geq \sum_{i \in N_{G'}(n+v)} q \cdot x_i \geq q \geq 1.$$

As $\xi(S) \leq \xi(J) = 1$ for all scenarios $\xi \in \mathcal{U}$, equation (7.6) holds for all $S \subseteq J$.

Conversely, assume that x is a solution of Robust q-MsMc such that $x(I) \leq B$. We define T to contain all vertices $v \in V(G)$ such that $x_v \geq 1$. By this definition, we clearly have $|T| \leq B = k$. So let $u \in V(G)$ be arbitrary and regard the scenario $\xi \in \mathcal{U}$ with $\xi_j = 1$ if and only if $j = u + n$. By constraints (7.4b) for $S = \{u + n\}$ there exists some $v \in N_{G'}(u + n)$ such that $x_v \geq 1$. Thus, there exists $v \in N_G(u) \cup \{u\}$ with $v \in T$, which is what we needed to show. $\square$

Note that we did not show that Robust q-MsMc is contained in NP. In fact, it is not a priori clear if feasibility of a solution may be decided in polynomial time. Deciding this is the complement of the separation problem corresponding to the set of constraints (7.3b) and (7.3c) in (MIP 7.3) which is of interest for solving Robust q-MsMc. We give formal definitions of both problems here for the sake of completeness.

co-q-Multiset Multicover-Sep (co-q-MsMc-Sep).
Instance: Finite, disjoint sets I, J, a finite set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \cup J, E)$, and integers $x_i \in \mathbb{N}$ for $i \in I$.
Question: Are there $y(\xi) \in \mathbb{N}^{|I| \times |J|}$ for every $\xi \in \mathcal{U}$ satisfying

$$\sum_{i \in N(j)} y(\xi)_{ij} \geq \xi_j \qquad \forall j \in J, \xi \in \mathcal{U} \quad \text{and}$$
$$\sum_{j \in N(i)} y(\xi)_{ij} \leq q \cdot x_i \qquad \forall i \in I, \xi \in \mathcal{U}?$$

q-Multiset Multicover-Sep (q-MsMc-Sep).
Instance: Finite, disjoint sets I, J, a finite set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \cup J, E)$, and integers $x_i \in \mathbb{N}$ for $i \in I$.
Question: Does there exist a $\xi \in \mathcal{U}$, such that for all $y(\xi) \in \mathbb{N}^{|I| \times |J|}$ with

$$\sum_{i \in N(j)} y(\xi)_{ij} \geq \xi_j \quad \forall j \in J,$$

there exists an $i \in I$ with

$$\sum_{j \in N(i)} y(\xi)_{ij} > q \cdot x_i?$$

Regarding the complexity of co-q-MsMc-Sep and q-MsMc-Sep the size of the uncertainty set in comparison to its encoding length is of importance. The next two results handle the case in which the we may enumerate the uncertainty set in polynomial time.

Theorem 7.13. *co-q-Multiset Multicover-Sep, as well as q-Multiset Multicover-Sep, restricted to instances where $\mathcal{U}$ is polynomial time enumerable are contained in* P.

Proof. Given an instance of co-q-MsMc-Sep, we construct for each $\xi \in \mathcal{U}$ the flow network from Definition 7.4, where we set the arc capacities of all arcs $r \in R_t$ to $c(r) = \xi_j$. After one max-flow computation we can either decide that the desired $y(\xi)$ do not exist for this scenario or we obtain it in the same manner as in the proof of Lemma 7.5. Thus, after $|\mathcal{U}|$ max-flow computations we can decide co-q-MsMc-Sep. This procedure is polynomial as we may enumerate the elements of $\mathcal{U}$ in polynomial time. As q-MsMc-Sep is the complement of co-q-MsMc-Sep it is contained in P as well. □

Corollary 7.14. *Robust q-Multiset Multicover is contained in* NP *for any fixed $q \in \mathbb{N}_{>0}$ when restricted to instances where $\mathcal{U}$ is polynomial time enumerable.*

If we may not enumerate $\mathcal{U}$ in polynomially time, the procedure from the proof of Theorem 7.13 does not run in polynomial time. In fact, if we simply remove this restriction on the uncertainty set $\mathcal{U}$ q-Multiset Multicover-Sep becomes NP-complete and as a direct consequence co-q-Multiset Multicover-Sep becomes coNP-complete.

Theorem 7.15. *q-Multiset Multicover-Sep is* NP*-complete in the strong sense for any fixed $q \in \mathbb{N}_{>0}$.*

Proof. Given a fixed scenario $\xi \in \mathcal{U}$, as in the proof of Theorem 7.13, we can use the flow network from Lemma 7.5 to verify the validity of ξ by one max-flow computation. This implies that q-MsMc-Sep is contained in NP.

Let H be the graph and $k \geq 1$ the integer of an Independent Set instance. Without loss of generality let $V(H) = \{1, \ldots, n\}$. We create an instance of q-MsMc-Sep as follows. Let $J = V(H)$, $I = \{n+1\}$ and

$$\mathcal{U} = \{\xi \in \{0, q\}^n : \xi_u + \xi_v \leq q, uv \in E(H)\}.$$

Further let $G = (I \cup J, \{(n+1)j : 1 \leq j \leq n\})$ and $x_{n+1} = k - 1$.

First note that, as the created q-MsMc-Sep instance only contains one location, which is connected to all regions, the question reduces to: Does there exist a $\xi \in \mathcal{U}$ such that $\xi(J) > q \cdot x_{n+1}$.

Now assume there exists an independent set T of size at least k in H. We define a scenario ξ by setting $\xi_j = q$ if $j \in T$ and $\xi_j = 0$ else. It is $\xi \in \mathcal{U}$ and $\xi(J) \geq q \cdot k = q \cdot (x_{n+1} + 1) > q \cdot x_{n+1}$.

On the other hand assume we are given a scenario $\xi \in \mathcal{U}$ such that $\xi(J) > q \cdot x_{n+1}$. We set $T = \{v \in V(H) \colon \xi_v = q\}$. This set is independent and fulfills

$$|T| = \frac{\sum_{j \in J} \xi_j}{q} > \frac{q \cdot x_{n+1}}{q} = k - 1,$$

which finishes the proof. □

Corollary 7.16. *co-q-Multiset Multicover-Sep is* coNP*-complete in the strong sense for any fixed $q \in \mathbb{N}_{>0}$.* □

Note that the uncertainty set in the proof of Theorem 7.15 is not polyhedral. However, the proof can be adjusted, such that the regraded uncertainty set is polyhedral. We refrain from doing this here, as it does not give further insight to the problem and decreases readability of the proof.

Although the initial analysis of the complexity of Robust q-Multiset Multicover does not seem very promising, we still take a shot at solving it. The most promising approach at this point is using (IP 7.5). To achieve this, of course, it is of interest to compute the maximum in the constraints (7.5b). Although the term looks innocent enough, the part $\xi \in \mathcal{U}$ makes it unlikely to be computable efficiently in general, as it generalizes many NP-hard optimization problems. We formalize the computation of the maximum as the decision problem MaxSum.

MaxSum.
Instance: A set $\mathcal{U} \subseteq \mathbb{N}^n$ for some positive integer n and an integer $B \in \mathbb{N}$.
Question: Does there exist a $\xi \in \mathcal{U}$ such that

$$\sum_{j=1}^{n} \xi_j \geq B?$$

Observation 7.17. *If MaxSum is restricted to instances in which $\mathcal{U}$ is polynomial time enumerable it is contained in* P.

Theorem 7.18. *MaxSum is* NP*-complete in the strong sense.*

Proof. Given an element $\xi \in \mathcal{U}$ we can test if $\xi(J) \geq B$ in time $O(n)$. Thus, MaxSum is contained in NP.

To show NP-hardness, we reduce Independent Set to MaxSum. Given a graph G and an integer k from an instance of Independent Set we define an instance of MaxSum by setting $n = |V(G)|$,

$$\mathcal{U} = \{\xi \in \{0,1\}^n : \xi_u + \xi_v \leq 1, uv \in E(G)\}$$

and $B = k$. As an independent set in G corresponds to an element of $\mathcal{U}$ and vice versa we proved that MaxSum is NP-hard. □

In this section we analyzed the complexity of Robust q-MsMc and the subproblems appearing in the problem definition. Further, we gave three different integer programming formulations for the problem. Directly inputting these formulations for large uncertainty sets leads to very large integer programs, that possibly take up too much time simply being defined. In the next section we describe and analyze a general solution approach, based on the method of generating the constraints during the solution process on the fly.

7.3. General Solution Approaches

In the previous section we classified Robust q-Multiset Multicover and some of the appearing subproblems complexity wise. Although most of the regarded problems are NP-hard, in this section we are looking into general solution techniques, that work for arbitrary uncertainty sets. In the next chapter we then have a look at specific uncertainty sets to see whether we can improve the solution methods.

When aiming at solving Robust q-MsMc Theorem 7.12, stating that the problem is NP-hard in the strong sense, justifies the solution approach using an MIP or IP formulation. As for different types of uncertainty sets, the performance of (MIP 7.3) and (IP 7.4) might differ considerably, we regard both formulations.

The size of $\mathcal{U}$ as well as the size of the subsets S in (IP 7.4) may very well be exponential in the encoding size of the problem. Thus it seems to be a valid approach to generate constraints on the fly, hoping that most of them are redundant. To this end, we define the notions of *violating scenarios* and *violating subsets.*

Definition 7.19 (Violating scenarios and subsets)**.** Let an instance of Robust q-Multiset Multicover be given.

(i) Let $\mathcal{U}' \subseteq \mathcal{U}$ be a subset of the scenarios and let $\hat{x} \in \mathbb{N}^{|I|}$ be a feasible solution to (IP 7.4)($\mathcal{U}'$). In this context we call a scenario $\xi \in \mathcal{U}$ *violating*, if x is not feasible for (IP 7.4)($\mathcal{U}' \cup \{\xi\}$).

(ii) Let $\mathcal{J} \subseteq 2^J$ be a collection of subsets of the regions and let $\hat{x} \in \mathbb{N}^{|I|}$ be a feasible solution to (IP 7.5)($\mathcal{J}$). In this context we call a subset $S' \subseteq J$ *violating*, if $\hat{x}$ is not feasible for (IP 7.5)($\mathcal{J} \cup \{S'\}$)

The following observation clarifies why the definitions of violating subsets and scenarios are of interest for the computation of optimal solutions for Robust q-MsMc.

Observation 7.20. *Let an instance of Robust q-Multiset Multicover be given. The following holds true.*

(i) Let a subset of the scenarios $\mathcal{U}' \subseteq \mathcal{U}$ and a feasible solution $\hat{x}$ to (IP 7.4)($\mathcal{U}'$) be given. Assume $\xi \in \mathcal{U}$ is a violating scenario, then there is no feasible solution for (MIP 7.3)($\mathcal{U}$) using the vector $\hat{x}$.

(ii) Let a subset of the scenarios $\mathcal{U}' \subseteq \mathcal{U}$ and an optimal solution $x^\star$ to (MIP 7.3)($\mathcal{U}'$) be given. Then $x^\star$ is optimal for (MIP 7.3)($\mathcal{U}$) if and only if, there does not exist a violating scenario.

(iii) Let a collection of the regions $\mathcal{J} \subseteq 2^J$ and an optimal solution $x^\star$ to (IP 7.5)($\mathcal{J}$) be given. Then $x^\star$ is optimal for (IP 7.5)(2^J) if and only if, there does not exist a violating subset.

Since to any violating subset S', in some sense, there is a corresponding scenario, namely the one that maximizes the sum $\xi(S')$, we will also give those scenarios a name.

Definition 7.21 (Violating extreme scenario)**.** We call a violating scenario ξ' *violating extreme scenario* if there exists some violating subset S' such that

$$\xi' = \operatorname*{argmax}_{\xi \in \mathcal{U}} \left\{ \sum_{j \in S'} \xi_j \right\}.$$

Lemma 7.22. *Let an instance of Robust q-Multiset Multicover, a subset of the scenarios $\mathcal{U}' \subseteq \mathcal{U}$ and an optimal solution $x^\star$ to (IP 7.4)($\mathcal{U}'$) be given. Then $x^\star$ is optimal for (IP 7.4)($\mathcal{U}$) if and only if there does not exist a violating extreme scenario.*

Proof. If $x^\star$ is optimal for (IP 7.4)($\mathcal{U}$), then by Observation 7.20 *(ii)*, there does not exist a violating scenario and therefore no violating extreme scenario. Assume on the other hand, that there does not exist a violating extreme scenario. By definition this implies that there does not exist a violating subset. By Observation 7.20 *(iii)* this implies, that $x^\star$ is optimal for (IP 7.5)(2^J) and by Observation 7.11 we get that $x^\star$ is optimal for (IP 7.4)($\mathcal{U}$). □

Lemma 7.22 basically states that, when generating constraints in the solution process of an instance of Robust q-MsMc, we can restrict ourselves to finding violating extreme scenarios and violating subsets. An interesting question remains: Can we, given a violating extreme scenario, also compute the corresponding violating subset efficiently and vice versa. The complexity of finding violating extreme scenarios, when given violating subsets is, of course, highly dependent on the structure of the uncertainty set. As the MaxSum problem from the previous section is NP-complete, however, it is unlikely, that we can provide a polynomial time algorithm that does the job in general.

On the other hand we see in the following that, given a violating scenario, it is always possible to compute a violating subset efficiently.

Theorem 7.23. *Let an instance of Robust q-Multiset Multicover and a vector $x \in \mathbb{N}^{|I|}$ that is feasible for (IP 7.4)($\mathcal{U}'$) and (IP 7.5)($\mathcal{J}$) with sets $\mathcal{U}' \subseteq \mathcal{U}$ and $\mathcal{J} \subseteq 2^J$. Given a violating scenario $\xi' \in \mathcal{U}$ we can compute a violating subset in time $O((|I| + |J|)^3)$.*

Proof. Given an instance $\mathcal{I}$ of Robust q-MsMc, denote by $H := H_{\mathcal{I}}$ the flow network from Definition 7.4. We set the corresponding arc capacities in R_s to $q \cdot \hat{x}_i$ and in R_t to ξ'_j and compute a minimum s-t cut $\{S, T\}$. By the proof of Lemma 7.5 and the fact that ξ' is violating, the capacity of the cut is less than $\xi'(J)$. We define a subset of the regions $S' = T \cap J$. As $c(S, T) < \xi'(J)$ it is $S' \neq \emptyset$.

We claim that S' is violating. Suppose this is not the case. Then

$$\sum_{i \in N(S')} q \cdot x_i \geq \max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in S'} \xi_j \right\} \geq \sum_{j \in S'} \xi'_j. \tag{7.7}$$

As the edges with infinite capacity cannot be contained in the cut, it holds true that all arcs si for $i \in N_G(S')$ are contained in the cut. Replacing these edges with all arcs jt for $j \in S'$ also yields an s-t cut $\{\{s\} \cup I \cup J, \{t\}\}$. The capacity of the new cut is clearly $\xi'(J)$. By equations (7.7), we get

$$\sum_{j \in J} \xi'_j = c((\{s\} \cup I \cup J, \{t\})) \leq c(S, T) < \sum_{j \in J} \xi'_j.$$

This is a contradiction. Thus, the set S' is violating.

To see the running time of the above procedure, note that the minimum s-t cut can be computed in time $O((|I| + |J|)^3)$, by using a push-relabel algorithm for computing a maximum flow and using the Max-Flow-Min-Cut theorem, cf. [AMO93]. □

We now turn to the actual computation of violating scenarios and violating subsets. By Theorem 7.15 we should not hope for a polynomial time algorithm, finding either one of them in general. In fact, if we do not make any assumptions on the structure of the uncertainty set, it is unlikely that we come up with a better algorithm, than the one given in the proof of Theorem 7.13. Thus, as most of the regarded uncertainty sets in the next section are polyhedral, we restrict ourselves to polyhedral uncertainty sets for the remainder of this section.

Assumption. All uncertainty sets $\mathcal{U}$ regarded in the remainder of this section are polyhedral, i.e. $\mathcal{U} = \{\xi \in \mathbb{N}^n \colon A\xi \le b\}$ for some matrix $A \in \mathbb{R}^{n\times m}$ and vector $b \in \mathbb{R}^n$.

Under this assumption we can formulate the following integer program.

$$
\begin{aligned}
&(\text{IP 7.8})(\hat{x}, \mathcal{U}) \quad && \min_{\xi,\mu,\nu,\omega} && \sum_{i\in I} q \cdot \hat{x}_i \cdot \mu_i - \sum_{j\in J} \omega_j && && (7.8a)\\
& && \text{s.t.} && \omega_j \le \xi_j && \forall j \in J && (7.8b)\\
& && && \omega_j \le \max_{\eta\in\mathcal{U}} \left\{ \sum_{j\in J} \eta_j \right\} \cdot \nu_j && \forall j \in J && (7.8c)\\
& && && \mu_i \ge \nu_j && \forall j \in J,\ i \in N(j) && (7.8d)\\
& && && \xi \in \mathcal{U} && && (7.8e)\\
& && && \mu_i, \nu_j \in \{0,1\} && \forall i \in I,\ j \in J. && (7.8f)
\end{aligned}
$$

Note that the maximum in equation (7.8c) is merely an upper bound for the sum of all entries of any scenario. For polyhedral uncertainty sets, this could be set to the quotient of the largest entry in b and the smallest entry in A, which can be computed in linear time.

Theorem 7.24. *Let an instance of* Robust q-Multiset Multicover *be given. Then, $\hat{x} \in \mathbb{N}^{|I|}$ is feasible for (IP 7.5)(2^J) if and only if, for the optimal objective value $z^\star$ of (IP 7.8)($\hat{x}, \mathcal{U}$), it holds true that $z^\star \ge 0$. Further, if $z^\star < 0$ and $\xi^\star, \nu^\star$ are part of an optimal solution to (IP 7.8), then $\xi^\star$ is a violating extreme scenario and $S^\star = \left\{ j \in J \colon \nu^\star_j = 1 \right\}$ is a violating subset.*

Proof. Assume $\hat{x}$ is feasible for (IP 7.5)(2^J) and denote by $(\xi^\star, \mu^\star, \nu^\star, \omega^\star)$ an optimal solution to (IP 7.8)($\hat{x}, \mathcal{U}$). We set $S' = \left\{ j \in J \colon \nu^\star_j = 1 \right\}$. Note that by constraints (7.8d) and the fact that in any optimal solution the μ variables are set to zero whenever possible, we also get $N(S') = \left\{ i \in I \colon \mu^\star_i = 1 \right\}$. As it holds true that $\omega^\star_j = 0$ for all $j \notin S'$ and $\omega^\star_j = \xi^\star_j$ for all $j \in S'$, we conclude

$$
z^\star = \sum_{i\in I} q \cdot \hat{x}_i \cdot \mu^\star_i - \sum_{j\in J} \omega^\star_j = \sum_{i\in N(S')} q \cdot \hat{x}_i - \sum_{j\in S'} \xi^\star_j \ge 0,
$$

where the last inequality follows from the fact that $\xi^\star \in \mathcal{U}$ and that $\hat{x}$ is feasible for (IP 7.4).

On the other hand, if $\hat{x}$ is not feasible for (IP 7.5), there exists a subset $S' \subseteq J$ such that equation (7.5b) does not hold. We define a solution for (IP 7.8) by setting

$$
\nu^\star_j = \begin{cases} 1, & \text{if } j \in S' \\ 0, & \text{else,} \end{cases} \qquad \mu^\star_i = \begin{cases} 1, & \text{if } i \in N(S') \\ 0, & \text{else,} \end{cases} \qquad \xi^\star_j = \operatorname*{argmax}_{\xi\in\mathcal{U}} \sum_{j\in S'} \xi_j
$$

and

$$\omega_j^\star = \begin{cases} \xi_j^\star, & \text{if } j \in S' \\ 0, & \text{else.} \end{cases}$$

We get

$$z^\star = \sum_{i \in I} q \cdot \hat{x}_i \cdot \mu_i^\star - \sum_{j \in J} \omega_j^\star = \sum_{i \in N(S')} q \cdot \hat{x}_i - \sum_{j \in S'} \xi_j^\star < 0,$$

where the last inequality follows from the fact that equation (7.5b) does not hold for S'. It is a direct consequence of these arguments that S' and $\xi^\star$ as defined in the theorem statement are violating. To see that $\xi^\star$ is also a violating extreme scenario, note that increasing the value of any entry $j \in S'$ of $\xi^\star$ will lead to a smaller objective value. Thus, for all $\xi \in \mathcal{U}$, it holds true that $\xi(S') \leq \xi^\star(S')$. As S' is a violating subset, this makes $\xi^\star$ a violating extreme scenario. □

We now have a general tool at hand, which we can use to find violating subsets and scenarios. If we are only looking for violating subsets, we may also use this somewhat smaller formulation.

$$\text{(IP 7.9)}(\hat{x}, \mathcal{U}) \quad \min_{\mu, \nu} \quad \sum_{i \in I} q \cdot \hat{x}_i \cdot \mu_i - \max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in J} \xi_j \nu_j \right\} \tag{7.9a}$$

$$\text{s.t.} \quad \mu_i \geq \nu_j \quad \forall j \in J,\ i \in N(j) \tag{7.9b}$$

$$\mu_i, \nu_j \in \{0, 1\} \quad \forall i \in I,\ j \in J. \tag{7.9c}$$

Theorem 7.25. *Let an instance of* Robust *q*-Multiset Multicover *be given. Then,* $\hat{x} \in \mathbb{N}^{|I|}$ *is feasible for (IP 7.5)(2^J) if and only if, for the optimal objective value* $z^\star$ *of (IP 7.9)($\hat{x}, \mathcal{U}$), it holds true that* $z^\star \geq 0$. *Further, if* $z^\star < 0$ *and* $\nu^\star$ *is part of an optimal solution to (IP 7.9), then* $S' = \left\{ j \in J \colon \nu_j^\star = 1 \right\}$ *is a violating subset.*

Proof. The proof is along the same lines as the one for Theorem 7.24 and therefore not repeated at this point. □

The interesting part of (IP 7.9) is, of course, the computation of the maximum in the objective function. For general uncertainty sets, we will not be able to compute this maximum without computing the actual extreme scenario ξ, which simply brings us back to (IP 7.8). Nevertheless, as we see in the next section, there are uncertainty sets in which the computation of the maximum is much easier, which justifies the presence of (IP 7.9).

7.4. Specific Uncertainty Sets

By the results of the previous sections, Robust *q*-Multiset Multicover and the corresponding separation problem *q*-Multiset Multicover-Sep are NP-hard if we do not restrict the uncer-

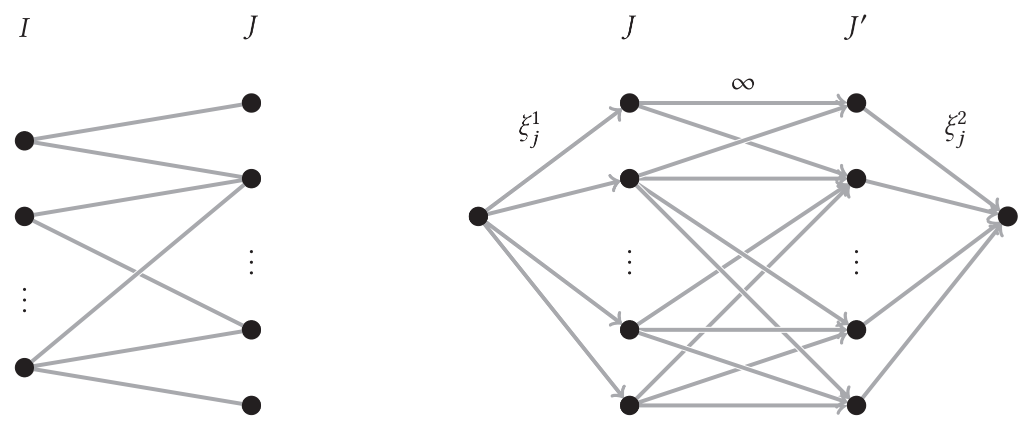

Figure 7.4.: Flow network construction from the proof of Theorem 7.27.

tainty set and the most promising tool for solving the problem in general is integer programming combined with constraint generation. In this section we regard different specific uncertainty sets, classify the complexity of the corresponding covering problems, and see if we can improve the solution methods given in Section 7.3. To this end we regard the concepts *Discrete Uncertainty*, *Interval Uncertainty* and *Budgeted Uncertainty*, which are widely spread and often used in the literature. Further we examine *Subset Budgeted Uncertainty* in order to decrease the conservatism of the solutions.

7.4.1. Discrete Uncertainty

Recall the definition of discrete uncertainty. Applied to Robust q-MsMc, this means that the uncertainty sets under consideration in this section are of the form $\mathcal{U} = \{\xi_1, \ldots, \xi_k\}$ for some $k \in \mathbb{N}$. In particular this implies that we may enumerate the elements of $\mathcal{U}$ in linear time. This fact directly gives us the first result by Corollary 7.14.

Observation 7.26. *Robust q-Multiset Multicover with discrete uncertainty is contained in* NP.

We already know that Robust q-MsMc is NP-hard in the strong sense for any fixed $q \geq 3$, even if the uncertainty set only contains one element. Further, we know that we may solve Robust q-MsMc in polynomial time if $q \leq 2$ and the uncertainty set contains only one element, cf. Section 7.1. Thus, the interesting remaining cases are the ones, where $q \leq 2$ and $|\mathcal{U}| \geq 2$. To this end note, that many polynomial time solvable problems become NP-hard when regarding robust counterparts of the problem. For example, the robust shortest path problem is NP-complete even when the uncertainty set contains only two scenarios, cf. [YY98]. We now use a similar idea to the one in Theorem 7.7 to show that we may solve Robust q-MsMc for $q = 1$ and $|\mathcal{U}| = 2$ in polynomial time.

Theorem 7.27. *Robust Min-1-Multiset Multicover restricted to uncertainty sets with two elements can be solved in time $O(|I| + |J|^3)$.*

Proof. Given an instance of Robust Min-1-MsMc with $J = \{1, \dots, n\}$ and $\mathcal{U} = \left\{\xi^1, \xi^2\right\}$, we define a directed graph $H = (V, R)$, similar to the one in Definition 7.4. We set $V(H) = J \cup J' \cup \{s\} \cup \{t\}$, where $J' = \{n+1, \dots, 2n\}$ and $s, t \notin J \cup J'$. Further we set $R = R_s \cup R_t \cup R'$, where $R_s = \{sj \colon j \in J\}$, $R_t = \{(n+j)t \colon j \in J\}$ and $R' = \{j(n+j') \colon j, j' \in J,\ N_G(j) \cap N_G(j') \neq \emptyset\}$. Intuitively there is an arc from j to $n + j'$ if there exists a supplier i that can serve clients in regions j and j'. Further we set the arc capacities $c \colon R \longrightarrow \mathbb{R}$ as

$$c(r) = \begin{cases} \xi^1_j, & \text{if } r = sj, \\ \xi^2_j, & \text{if } r = (n+j)t, \\ \infty, & \text{else.} \end{cases}$$

For the construction of H, see also Figure 7.4.

In a next step, we compute an integral maximum s-t flow f in H and define a solution to Robust Min-1-MsMc iteratively, beginning with $x_i = 0$ for all $i \in I$. For each $j(n + j') \in R'$ we increase x_i by $f(j(n + j'))$ for some $i \in N_G(j) \cap N_G(j')$. Further for each $j \in J$ we increase x_i by $\max\left\{\xi^1_j - f(sj), \xi^2_j - f(jt)\right\}$ for some $i \in N_G(j)$. We claim that x defined in this way is an optimal solution to the given instance of Robust Min-1-MsMc.

The feasibility of the solution is easily verified, considering that each unit of flow on an arc $j(n+j')$ can be considered as a supplier serving a client in j in scenario ξ^1 and a client in j' in scenario ξ^2. To prove the optimality of the procedure, let $\hat{x}$ be a feasible solution to Robust Min-1-MsMc. Denote by $\hat{y}(\xi^k) \in \mathbb{N}^{|I| \times |J|}$ for $k = 1, 2$ with

$$\begin{aligned} \sum_{i \in N(j)} \hat{y}(\xi^k) &= \xi^k_j \qquad \forall j \in J,\ k = 1, 2 \quad \text{and} \\ \sum_{j \in N(i)} \hat{y}(\xi^k) &\leq \hat{x}_i \qquad \forall i \in I,\ k = 1, 2 \end{aligned} \tag{7.10}$$

the corresponding assignment variables. These exist as $\hat{x}$ is a feasible solution to Robust Min-1-MsMc and we may assume equality in the first set of equations without loss of generality. We define an s-t flow $\hat{f} \colon R \longrightarrow \mathbb{N}$ in H iteratively, where we set $f(r) = 0$ for all $r \in R$ at the start. While there still is some $i \in I$ and $j, j' \in J$ with $\hat{y}_{ij}(\xi^1) > 0$ and $\hat{y}_{ij'}(\xi^2) > 0$, we increase $f(r)$ by $\min\left\{\hat{y}_{ij}(\xi^1), \hat{y}_{ij'}(\xi^2)\right\}$ for all $r \in \{sj, j(n + j'), (n + j')t\}$ while at the same time decreasing $\hat{y}_{ij}(\xi^1)$ and $\hat{y}_{ij'}(\xi^2)$ by the same value. By Equations (7.10), this defines a feasible s-t flow $\hat{f}$. Thus, we can also write the objective value for the solution $\hat{x}$ as

$$\sum_{j \in J} \hat{x}_j = \operatorname{val}(\hat{f}) + \sum_{j \in J} \left(\xi^1_j - \hat{f}(sj)\right) + \sum_{j \in J} \left(\xi^2_j - \hat{f}((n+j)t)\right).$$

Further note that for the sum of the demands in both scenarios it holds true that

$$\sum_{j\in J}\left(\xi_j^1+\xi_j^2\right)=2\,\mathrm{val}(f')+\sum_{j\in J}\left(\xi_j^1-f'(sj)\right)+\sum_{j\in J}\left(\xi_j^2-f'((n+j)t)\right),$$

for any feasible s-t flow f' in H. Recall that f is a maximum s-t flow and that the above equations also hold for f. We get, that

$$\sum_{j\in J}\hat{x}_j=\sum_{j\in J}\left(\xi_j^1+\xi_j^2\right)-\mathrm{val}(\hat{f})\geq\sum_{j\in J}\left(\xi_j^1+\xi_j^2\right)-\mathrm{val}(f)=\sum_{j\in J}x_j.$$

As $\hat{x}$ was an arbitrary feasible solution, this proves the optimality of x.

To see the running time of the above procedure, we note that the initialization of the x_i takes time $O(|I|)$, the maximum flow can be computed in time $O(|J|^3)$, cf.[AMO93], and the augmentation of the maximum flow in the second step can be realized in time $O(|J|)$. This leads to the overall running time of $O(|I|+|J|^3)$. □

One might be tempted to try extending the idea in the proof of Theorem 7.27 to cases where $|\mathcal{U}|\geq 3$. This does not work in the straight forward way, as adding another layer to the flow network destroys the correspondence of a doctor to a single arc. In fact we see in the following that it is unlikely that the idea extends in any way, as Robust q-Multiset Multicover remains NP-hard in the strong sense for all other remaining cases of discrete uncertainty.

Theorem 7.28. *Robust 1-Multiset Multicover restricted to instances where the uncertainty set $\mathcal{U}$ has 3 elements is* NP-*complete in the strong sense.*

Proof. By Observation 7.26, the problem is contained in NP.

Let $A=\{a_1,\dots,a_k\}$, $B=\{b_1,\dots,b_k\}$, $C=\{c_1,\dots,c_k\}$ and $M\subseteq A\times B\times C$ be an instance of Exact 3-Dimensional Matching. We define an instance of Robust 1-MsMc in the following way. Let $I=M$, $J=A\cup B\cup C$ and set $G=(I\cup J,E)$, where we have $ij\in E$ if j is an element in the tuple i. Further we set $B=k$ and $\mathcal{U}=\left\{\xi^A,\xi^B,\xi^C\right\}$ where

$$\xi_j^S=\begin{cases}1, & \text{if } j\in S\\ 0, & \text{else.}\end{cases}$$

for each $S\in\{A,B,C\}$.

Assume we are given a subset $M'\subseteq M$ with k elements, such that no element of $A\cup B\cup C$ appears in two tuples in M'. We define a solution for Robust 1-MsMc by setting $x_i=1$ if $i\in M'$ and $x_i=0$ else. As $|M'|=k=B$, this solution fulfills $x(I)\leq B$. Further, for each $S\in\{A,B,C\}$ we set $y(\xi^S)_{ij}=1$ if $j\in S$ and $x_i=1$. All remaining variables are set to zero. We can now verify that x and y fulfill the required equations.

Assume on the other hand that we are given integers $x_i\in\mathbb{N}$ with $x(I)\leq B=k$ representing a solution of Robust 1-MsMc. We set $M'=\{i\in I\colon x_i=1\}$. As for example $\xi^A(J)=k$, we know that

$|M'| = k$. Suppose for the sake of contradiction that for two different elements $i_1 = (a, b, c), i_2 = (a', b', c') \in M'$ we have $a = a'$. As $|N(i) \cap A| = 1$ for all elements $i \in I$ this directly implies $x(I) \geq k + 1$. A contradiction. If for the elements above it holds true that $b = b'$ or $c = c'$ the argument still works, as no $i \in I$ may serve more than one client from B or C. □

Observe that the proof from above extends to cases with $k > 3$ by adding $k-3$ *dummy* scenarios, regions, and locations. Each of the dummy regions is connected to a unique dummy location and the corresponding dummy scenario has demand 1 in that particular region and 0 else. The resulting problem is essentially the same as the problem with $k = 3$. We get the following corollary.

Corollary 7.29. *Robust 1-Multiset Multicover restricted to instances where the uncertainty set $\mathcal{U}$ has k elements for any fixed $k \in \mathbb{N}$ with $k \geq 3$ is* NP*-complete in the strong sense.*

Theorem 7.30. *Robust 2-Multiset Multicover restricted to instances where the uncertainty set $\mathcal{U}$ has 2 elements is* NP*-complete in the strong sense.*

Proof. By Observation 7.26, the problem is contained in NP.

The proof for NP-hardness is along the same lines as the one for Theorem 7.28. Nevertheless we state it here for the sake of completeness.

Let $A = \{a_1, \dots, a_k\}$, $B = \{b_1, \dots, b_k\}$, $C = \{c_1, \dots, c_k\}$ and $M \subseteq A \times B \times C$ be an instance of Exact 3-Dimensional Matching. We define an instance of Robust 2-MsMc in the following way. Let $I = M$, $J = A \cup B \cup C$ and set $G = (I \cup J, E)$, where we have $ij \in E$ if j is an element in the tuple i. Further we set $B = k$ and $\mathcal{U} = \{\xi^1, \xi^2\}$ where

$$\xi^1_j = \begin{cases} 1, & \text{if } j \in A, \\ 0, & \text{else} \end{cases} \quad \text{and} \quad \xi^2_j = \begin{cases} 1, & \text{if } j \in B \cup C, \\ 0, & \text{else.} \end{cases}$$

Assume we are given a subset $M' \subseteq M$ with k elements, such that no element of $A \cup B \cup C$ appears in two tuples in M'. As in the previous proof we define a solution for Robust 2-MsMc by setting $x_i = 1$ if $i \in M'$ and $x_i = 0$ else. By setting $y_{ij}(\xi^k) = \xi^k_j$ for $k \in \{1, 2\}$ if j is contained in i and $x_i = 1$, and $y_{ij}(\xi^k_j) = 0$ else, we verify the feasibility of x for Robust 2-MsMc.

Assume on the other hand that we are given integers $x_i \in \mathbb{N}$ with $x(I) \leq B = k$ representing a solution of Robust 2-MsMc. We set $M' = \{i \in I : x_i = 1\}$. As $\xi^2(J) = 2k$ it holds true that $|M'| = k$. Now suppose for the sake of contradiction, that for two different elements $i_1 = (a, b, c), i_2 = (a', b', c') \in M'$ we have $a = a'$. By the definition of M' this implies that there exists some $j \in A$ such that $x_i = 0$ for all $i \in N(j)$ contradicting the fact that x_i is a solution to Robust 2-MsMc. With similar arguments we verify that each $b \in B$ and $c \in C$ is covered exactly once by M'. □

As with Robust 1-MsMc we can easily extend the proof above to uncertainty sets with $k \geq 3$ elements by simply adding *dummy* scenarios, locations, and regions.

Corollary 7.31. *Robust 2-Multiset Multicover restricted to instances where the uncertainty set $\mathcal{U}$ has k elements for any fixed $k \in \mathbb{N}$ with $k \geq 2$ is* NP*-complete in the strong sense.*

As a final remark in this section we note that q-Multiset Multicover-Sep, co-q-Multiset Multicover-Sep and MaxSum are all contained in P when restricted to instance where the uncertainty set is given explicitly. This is due to Theorem 7.13 and Observation 7.17. In particular, in the solution process for Robust q-Multiset Multicover we may generate violating extreme scenarios or violating subsets in polynomial time.

7.4.2. Interval Uncertainty

Applying the concept of interval uncertainty to Robust q-MsMc yields uncertainty sets of the form $\mathcal{U} = \left\{\xi \in \mathbb{N}^{|J|} : a_j \leq \xi_j \leq b_j,\ \forall j \in J\right\}$ for vectors $a, b \in \mathbb{N}^{|J|}$. Loosely speaking, we give upper and lower bounds on the possible number of clients in each region.

Assume we are given an instance of Robust q-MsMc with interval uncertainty. For any subset of the regions $S \subseteq J$, we can easily compute

$$\max_{\xi \in \mathcal{U}} \left\{\sum_{j \in S} \xi_j\right\} = \sum_{j \in S} b_j.$$

Thus, (IP 7.5) is now the same as (IP 7.2)(b) and all complexity results from q-MsMc directly transfer.

As it seems very conservative to allow the worst case in every region at the same time, in the next section we try not only to give a bound on each region, but also on the total number of clients.

7.4.3. Budgeted Uncertainty

Recall that for problems with budgeted uncertainty, in addition to the bounds induced by interval uncertainty, we bound the sum of all values of entries of a scenario ξ. This leads to uncertainty sets of the form

$$\mathcal{U} = \left\{\xi \in \mathbb{N}^{|J|} : a_j \leq \xi_j \leq b_j,\ \xi(J) \leq \Gamma\right\}$$

for vectors $a, b \in \mathbb{N}^{|J|}$ and an integer $\Gamma \in \mathbb{N}$ for Robust q-MsMc. This can be interpreted as having lower and upper bound on the number of clients in each region and additionally having a bound on the total number of clients in the instance.

As Robust q-MsMc with budgeted uncertainty is a restriction of Robust q-MsMc, it is not immediate that it is also NP-hard. Nevertheless we see that Robust q-MsMc with budgeted uncertainty remains NP-hard.

Theorem 7.32. *Robust q-Multiset Multicover with budgeted uncertainty is* NP*-hard in the strong sense for any $q \in \mathbb{N}_{>0}$.*

Proof. Regarding the proof of Theorem 7.12, we observe that the constructed uncertainty set $\mathcal{U} = \left\{\xi \in \mathbb{N}^{|J|} : \sum_{j \in J} \xi_j = 1\right\}$ fits into the framework of budgeted uncertainty by setting $\Gamma = 1$ and, for all $j \in J$, setting $a_j = 0$ and $b_j = 1$. Thus, the reduction used in Theorem 7.12 also works for budgeted uncertainty and we get the desired result. □

Although we may not hope for an efficient algorithm solving Robust q-MsMc with budgeted uncertainty, there are some improvements in contrast to the general solving techniques introduced in the last section.

We begin by computing the maximum from Observation 7.11 for budgeted uncertainty.

Lemma 7.33. *Given a budgeted uncertainty set*

$$\mathcal{U} = \left\{\xi \in \mathbb{N}^{|J|} : a_j \leq \xi_j \leq b_j,\ \xi(J) \leq \Gamma\right\}$$

for some index set J and a subset $S \subseteq J$ of these indices. It is

$$\max_{\xi \in \mathcal{U}} \left\{\sum_{j \in S} \xi_j\right\} = \min \left\{\sum_{j \in S} b_j, \Gamma - \sum_{j \in J \setminus S} a_j\right\}.$$

Proof. Let $S \subseteq J$. We define the scenario ξ' by setting $\xi'_j = b_j$ if $j \in S$ and $\xi'_j = a_j$ else. If $\xi' \in \mathcal{U}$ we have

$$\begin{aligned}\max_{\xi \in \mathcal{U}} \left\{\sum_{j \in S} \xi_j\right\} &= \sum_{j \in S} b_j = \sum_{j \in S} b_j + \sum_{j \in J \setminus S} a_j - \sum_{j \in J \setminus S} a_j \\ &= \sum_{j \in J} \xi'_j - \sum_{j \in J \setminus S} a_j \\ &\leq \Gamma - \sum_{j \in J \setminus S} a_j.\end{aligned}$$

If otherwise $\xi' \notin \mathcal{U}$, it is

$$\Gamma < \sum_{j \in J} \xi'_j = \sum_{j \in S} b_j + \sum_{j \in J \setminus S} a_j$$

and thereby $b(S) > \Gamma - a(J \setminus S)$. We then easily verify that

$$\max_{\xi \in \mathcal{U}} \left\{\sum_{j \in S} \xi_j\right\} = \Gamma - \sum_{j \in J \setminus S} a_j,$$

as $a(S) \leq \Gamma - a(J \setminus S)$. □

A direct consequence of Lemma 7.33 is that for uncertainty sets representing budgeted uncertainty we can solve MaxSum in time linear in the size of the vectors in the uncertainty set.

Corollary 7.34. *MaxSum is contained in* P *for budgeted uncertainty.*

Using Lemma 7.33, we get the following formulation of Robust q-MsMc for budgeted uncertainty, with $\mathcal{J} = 2^J$.

$$\begin{aligned}
\text{(IP 7.11)}(\mathcal{J}) \quad & \min_x \quad \sum_{i \in I} x_i \\
& \text{s.t.} \quad \sum_{i \in N(S)} q \cdot x_i \geq \min \left\{ \sum_{j \in S} b_j, \Gamma - \sum_{j \in J \setminus S} a_j \right\} && \forall S \in \mathcal{J} \\
& \qquad\qquad x_i \in \mathbb{Z}_{\geq 0} && \forall i \in I.
\end{aligned}$$

As the number of constraints is still exponential in the number of regions, we are interested in the complexity of the separation problem. Although the reduction from Theorem 7.15 does not transfer, we get the following result.

Theorem 7.35. *q-Multiset Multicover-Sep with budgeted uncertainty is* NP*-complete.*

Proof. q-MsMc-Sep restricted to budgeted uncertainty is contained in NP as a restriction of q-MsMc-Sep.

We show that Knapsack reduces to q-MsMc-Sep. Let an arbitrary instance of Knapsack with a set $U = \{1, \dots, n\}$, sizes $s_u \in \mathbb{N}$ and profits $p_u \in \mathbb{N}$ associated with each element $u \in U$ and two integers $B, K \in \mathbb{N}_{>0}$ be given. For the instance of q-MsMc-Sep, we set $I = \{1, \dots, n, 2n+1\} = U \cup \{2n+1\}$ and $J = \{n+1, \dots, 2n\}$. We define a bipartite graph $G = (I \cup J, E)$, where

$$E = \{\{u, n+u\}, \{2n+1, n+u\} \text{ for } u = 1, \dots, n\}.$$

Furthermore, we set $x_u = s_u$ and $b_{n+u} = q \cdot (p_u + s_u)$ for all $u \in U$ and $x_{2n+1} = K - 1$, $\Gamma = q \cdot (B + K)$. Finally, we set $a_j = 0$ for all $j \in J$.

Now given a solution $U' \subseteq U \subseteq I$ of Knapsack with $s(U') \leq B$ and $p(U') \geq K$, we choose $S = \{n+u : u \in U'\} \subseteq J$. Then, S is nonempty since $U' \neq \emptyset$ and we have $\Gamma/q - x(U') = B + K - s(U') \geq B + K - B = K$ and

$$\begin{aligned}
\frac{b(S)}{q} - \sum_{i \in U'} x_i &= \sum_{u \in U'} \frac{b_{n+u}}{q} - \sum_{u \in U'} x_u = \sum_{u \in U'} p_u + s_u - \sum_{u \in U'} s_u \\
&= \sum_{u \in U'} p_u \geq K,
\end{aligned}$$

yielding $\min\{b(S)/q, \Gamma/q\} - x(U') \geq K$. Subtracting $K - 1 = x_{2n+1}$ on both sides we obtain:

$$\begin{aligned}
& \min\{b(S)/q, \Gamma/q\} - x(U') - x_{2n+1} \geq 1 > 0 \\
\Leftrightarrow \quad & \min\{b(S)/q, \Gamma/q\} - x(N(S)) \geq 1 > 0,
\end{aligned}$$

i.e., $\min\{b(S), \Gamma\} - q \cdot x(N(S)) > 0$. As this means, that x is not feasible for (IP 7.11), we know that there exists a $\xi' \in \mathcal{U}$, fulfilling the conditions of q-MsMc-Sep, namely

$$\xi' := \operatorname*{argmax}_{\xi \in \mathcal{U}} \left\{ \sum_{j \in S} \xi_j \right\}.$$

On the other hand, given a solution ξ' to q-MsMc-Sep. Then x is not feasible for (MIP 7.3) and by Theorem 7.10 it is not feasible for (IP 7.4). Thus, by Observation 7.11, there exists some $S \subseteq J$ with the property that

$$q \cdot \sum_{i \in N(S)} x_i < \max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in S} \xi_j \right\} = \min \left\{ \sum_{j \in S} b_j, \Gamma \right\}. \tag{7.11}$$

Set $S' := S$ and $U' = \{u : n + u \in S'\} \subseteq U$. Our aim is to show that U' is a solution for Knapsack. We have $S' \neq \emptyset$ and $N(S') = U' \cup \{2n + 1\}$. Reformulating the left hand side of (7.11) we get $x(N(S')) = x(U') + x_{2n+1} = x(U') + K - 1$, so that in total we have $\min\{b(S'), \Gamma\} - q \cdot x(U') > q \cdot (K - 1)$, i.e., $\min\{{}^{b(S')}/_{q}, {}^{\Gamma}/_{q}\} - x(U') \geq K$. When inserting the above definitions this expression becomes

$$\min \{p(U') + s(U'), B + K\} - s(U') \geq K. \tag{7.12}$$

Now, we need to differentiate between two cases: If $p(U') + s(U') \leq B + K$, equation (7.12) yields $p(U') = p(U') + s(U') - s(U') \geq K$ and $s(U') \leq B + K - p(U') \leq B + K - K = B$. If $p(U') + s(U') > B + K$, (7.12) yields $B + K - s(U') \geq K$, i.e., $s(U') \leq B$. Furthermore, $p(U') > B + K - s(U') \geq B + K - B = K$. Thus, U' is a solution for Knapsack. ☐

We now know, that separation for budgeted uncertainty is most likely not be solvable in polynomial time. But as computing the maximum from Observation 7.11 can be done efficiently, it is sufficient to compute violating subsets in the solution process of Robust q-MsMc and solely use formulation (IP 7.5). To compute violating subsets, we regard the following IP.

$$\text{(IP 7.13)}(\hat{x}, \mathcal{U}) \quad \min_{\mu, \nu, \gamma} \quad \sum_{i \in I} q \cdot \hat{x}_i \cdot \mu_i - \gamma \tag{7.13a}$$

$$\text{s.t.} \quad \gamma \leq \sum_{j \in J} b_j \cdot \nu_j \tag{7.13b}$$

$$\gamma \leq \Gamma - \sum_{j \in J} a_j + \sum_{j \in J} a_j \nu_j \tag{7.13c}$$

$$\mu_i \geq \nu_j \qquad \forall j \in J,\ i \in N(j) \tag{7.13d}$$

$$\mu_i, \nu_j \in \{0, 1\} \quad \forall i \in I,\ j \in J. \tag{7.13e}$$

Theorem 7.36. *Let an instance of Robust q-Multiset Multicover with budgeted uncertainty be given. Then, $\hat{x} \in \mathbb{N}^{|I|}$ is feasible for (IP 7.5)(2^J) if and only if, for the optimal objective value $z^\star$ of*

(IP 7.13)$(\hat{x}, \mathcal{U})$, it holds true that $z^\star \geq 0$. Further, if $z^\star < 0$ and $v^\star$ is part of an optimal solution to (IP 7.9), then $S' = \left\{ j \in J \colon v_j^\star = 1 \right\}$ is a violating subset.

Proof. By Theorem 7.25 it is enough to prove that in any optimal solution $(\mu^\star, v^\star, \gamma^\star)$ we have

$$\gamma^\star = \max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in J} \xi_j v_j^\star \right\}.$$

By Lemma 7.33 we have

$$\max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in J} \xi_j v_j^\star \right\} = \min \left\{ \sum_{j \in J} b_j v_j^\star, \Gamma - \sum_{j \in J} (1 - v_j^\star) a_j \right\}$$

Using constraints (7.13b) and (7.13c) the desired equality follows directly. □

7.4.4. Subset Budgeted Uncertainty

Last but not least we turn to subset budgeted uncertainty. In the previous section we bounded the total number of clients in all regions. For some applications this approach might still be too conservative. Let us think of the regions being distributed on a geographic map that is split into many areas that each contain a subset of the regions. Then the budgeted uncertainty approach does not necessarily forbid the worst case to happen in all regions of a given area, although this might be considered unrealistic. To tackle this problem, in subset budgeted uncertainty, we give additional bounds on a collection of subsets of the regions. In the context of Robust q-MsMc this means that the uncertainty sets are of the form

$$\mathcal{U} = \left\{ \xi \in \mathbb{N}^{|J|} \colon \alpha_{S'} \leq \sum_{j \in S'} \xi_j \leq \beta_{S'}, \ \forall S' \in \mathcal{S} \right\}$$

for a collection of subsets of the regions $\mathcal{S} \subseteq 2^J$ and integers $\alpha_{S'}, \beta_{S'} \in \mathbb{N}$ for each $S' \in \mathcal{S}$.

As Robust q-MsMc with subset budgeted uncertainty generalizes Robust q-MsMc with budgeted uncertainty by setting $\mathcal{S} = \{\{j\} : j \in J\} \cup \{J\}$, the first two results in this section are immediate.

Theorem 7.37. *Robust q-Multiset Multicover with subset budgeted uncertainty is* NP*-hard in the strong sense for any fixed $q \in \mathbb{N}_{>0}$.* □

Theorem 7.38. *q-Multiset Multicover-Sep with subset budgeted uncertainty is* NP*-complete in the strong sense for any fixed $q \in \mathbb{N}_{>0}$.* □

In contrast to budgeted uncertainty, we there is no nice characterization for the maximum in Observation 7.11. This is due to the fact that MaxSum is NP-complete for sets from subset budgeted uncertainty.

Theorem 7.39. *MaxSum is* NP*-complete in the strong sense when restricted to sets of the form*

$$\mathcal{U} = \left\{ \xi \in \mathbb{N}^n \colon \alpha_{S'} \leq \sum_{j \in S'} \xi_j \leq \beta_{S'},\ \forall S' \in \mathcal{S} \right\},$$

for a collection of subsets $\mathcal{S} \in 2^{\{1,\dots,n\}}$ and integers $\alpha_{S'}, \beta_{S'} \in \mathbb{N}$ for each $S' \subseteq \mathcal{S}$.

Proof. MaxSum with subset budgeted uncertainty is contained in NP as a restriction of a problem in NP, cf. Theorem 7.18.

To see that MaxSum with subset budgeted uncertainty is also NP-hard we show that the set $\mathcal{U}$ used in the proof of Theorem 7.18 can be modeled in the framework of subset budgeted uncertainty. For a graph G with vertex set $V(G) = \{1, \dots, n\}$ we set $\mathcal{S} = \{\{u, v\} : uv \in E(G)\} \cup \{\{u\} : u \in V(G)\}$, $\alpha_{S'} = 0$ and $\beta_{S'} = 1$ for all $S' \in \mathcal{S}$. Using these parameters we get the set

$$\mathcal{U} = \left\{ \xi \in \mathbb{N}^n \colon 0 \leq \xi_j \leq 1,\ \forall j \in \{1, \dots, n\} \text{ and } 0 \leq \xi_u + \xi_v \leq 1,\ \forall uv \in E(G) \right\}$$

which is the same as the one in the proof of Theorem 7.18. □

Theorem 7.39 shows that it is unlikely that we find an algorithm computing the maximum in Observation 7.11 efficiently. Thus, in order to do a separation step in the solution process for Robust q-MsMc with subset budgeted uncertainty we use (IP 7.8) adjusted to the specific uncertainty set.

$$\begin{array}{llll}
\text{(IP 7.14)}(\hat{x}, \mathcal{U}) & \min\limits_{\xi, \mu, \nu, \omega} & \sum\limits_{i \in I} q \cdot \hat{x}_i \cdot \mu_i - \sum\limits_{j \in J} \omega_j & \\
& \text{s.t.} & \omega_j \leq \xi_j & \forall j \in J \\
& & \omega_j \leq \max\limits_{\eta \in \mathcal{U}} \left\{ \sum\limits_{j \in J} \eta_j \right\} \cdot \nu_j & \forall j \in J \\
& & \mu_i \geq \nu_j & \forall j \in J, \ i \in N(j) \\
& & \alpha_{S'} \leq \sum\limits_{j \in S'} \xi_j \leq \beta_{S'} & \forall S' \in \mathcal{S} \\
& & \mu_i, \nu_j \in \{0, 1\} & \forall i \in I, \ j \in J.
\end{array}$$

This concludes our study on specific uncertainty sets for Robust q-Multiset Multicover. We improved the general solution techniques for all regarded uncertainty concepts. In particular we were able to find a polynomial time algorithm for the case that the uncertainty set contains only two scenarios and proved that separation for discrete uncertainty can be done in polynomial time. Further, we showed that solving Robust Min-q-MsMc with interval uncertainty is the same as solving Min-q-MsMc. For budgeted uncertainty we could show that MaxSum can be solved

$\mathcal{U}$		Robust q-MsMc	q-MsMc-Sep	co-q-MsMc-Sep	MaxSum
General		NP-hard	NP-complete	coNP-complete	NP-complete
Discrete	$q\,\lvert\mathcal{U}\rvert \leq 2$	P	P	P	P
	$q\,\lvert\mathcal{U}\rvert \geq 3$	NP-complete	P	P	P
Interval	$q \leq 2$	P	P	P	P
	$q \geq 3$	NP-complete	P	P	P
Budgeted		NP-complete	NP-complete	coNP-complete	P
Subset budgeted		NP-hard	NP-complete	coNP-complete	NP-complete

Table 7.1.: Summarized complexity results for Robust q-MsMc and the corresponding separation problems.

efficiently and gave a condensed inter programming formulation for the separation step and for subset budgeted uncertainty we were able to give an integer programming formulation.

Conclusion We conclude this chapter with a summary of the achieved complexity results. For a complete overview of these see Table 7.1. The polynomial time solvability of 2-MsMc and Robust 1-Multiset Multicover restricted to discrete uncertainty sets of cardinality two are the highlights of the positive results. It is also interesting that the q-Multiset Multicover-Sep can be solved in polynomial time for discrete uncertainty. The problem Robust q-Multiset Multicover has turned out to be very challenging. As a consequence of the proved results it is NP-hard when restricted to discrete uncertainty and $q \cdot |\mathcal{U}| \geq 3$. It also remains NP-hard for budgeted and subset budgeted uncertainty. Up to this point it is unclear if Robust q-Multiset Multicover is even contained in NP. A direction of further research to examine this question or even prove that Robust q-Multiset Multicover is Σ_2^P-complete.

We discussed different solution approaches based on constraint generation and integer programming. We formulated models for each of the discussed problem and gave various improvement possibilities for specific uncertainty sets. Here, it could be of interest to discuss transferability of approximation algorithms of q-Multiset Multicover to the robust versions of the problem.

Chapter 8.

q-Multiset Multicover with Unsteerable Demand

*In the problem setting of q-*Multiset Multicover *defined in Chapter 7 the choice of which client is served by which supplier was up to the planner. Here we focus on two related problems called q-*freeClient *and q-*orderedClient*, in which we allow the clients to choose the supplier they are served by. In q-*freeClient *the choice of the clients is unrestricted, whereas in q-*ordered-Client *the client is assumed to have a known preference list of locations. While we prove that q-*orderedClient *is* NP*-complete for any fixed* $q \geq 2$*, q-*freeClient *turns out to be linear time solvable for all* $q \in \mathbb{N}_{>0}$*. Introducing uncertainty in the demand of both problems significantly complicates the problems to the point, that* Robust *q-*freeClient *is* NP*-hard in the strong sense for any fixed* $q \in \mathbb{N}_{>0}$*. For commonly used uncertainty sets, such as discrete uncertainty sets, we show that* Robust *q-*freeClient *is contained in* P*, whereas* 1-orderedClient *remains* NP*-complete even when restricted to uncertainty sets of cardinality* 2*.*

A mixed version of q-Multiset Multicover and q-orderedClient has previously been regarded in [Büs+20]. Some of the proofs and ideas are published in similar form in that publication and are therefore joint work with Christina Büsing, Martin Comis, and Eva Schmidt. In particular this applies to the IP-formulation of q-orderedClient and the results on budgeted uncertainty. All of the results in this chapter are joint work with Eva Schmidt.

We refer to the introduction of Part II for a literature review on the topic, as it overlaps with the one of Chapter 7.

Outline The organization of this chapter is very similar to the one of Chapter 7. We begin by formally defining the problems and classifying them by their complexity class. In Section 8.2 we introduce uncertainty in the demand and analyze the influence on the complexity of the problems. Finally, in Section 8.3, we regard specific uncertainty sets for both problems and discuss changes to the complexity of the problems.

8.1. Including Unsteerable Clients into q-Multiset Multicover

Up till now, within the problem setting of q-Multiset Multicover, we assumed that all clients can be served by any supplier that is adjacent. One might wonder how the situation changes, if the client is given a choice by which of the adjacent supplier it wants to be served. This gives rise to the following problem definition.

q-freeClient (q-fC).
Instance: Finite, disjoint sets I, J, weights $d_j \in \mathbb{N}$ for all $j \in J$, a bipartite graph $G = (I \cup J, E)$, and an integer $B \in \mathbb{N}$.
Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \leq B$ such that for all $y \in \mathbb{N}^{|I|\times|J|}$ with $\sum_{i \in N(j)} y_{ij} = d_j$ for all $j \in J$, it is

$$\sum_{j \in N(i)} y_{ij} \leq q \cdot x_i \quad \forall i \in I?$$

Although q-fC may, at first glance, look like a challenging optimization problem, it turns out, that solving q-fC is rather trivial. In the optimization version of q-freeClient, Min-q-freeClient, we are looking for a minimum number of suppliers.

Theorem 8.1. *Min-q-freeClient can be solved in time $\mathcal{O}(|V(G)| + |E(G)|)$ for any fixed $q \in \mathbb{N}_{>0}$.*

Proof. Given an instance of Min-q-fC, for each $i \in I$, we set

$$\hat{x}_i = \left\lceil \frac{d(N(i))}{q} \right\rceil$$

and claim that this is an optimal solution. To show this, for each $i' \in I$, regard $y^{i'} \in \mathbb{N}^{|I|\times|J|}$ with

$$y^{i'}_{ij} = \begin{cases} d_j, & \text{if } i = i', \\ 0, & \text{else,} \end{cases}$$

for all $j \in J$. It holds true, that $\sum_{i \in N(j)} y^{i'}_{ij} = d_j$ for all $i' \in I$ and $j \in J$. Furthermore, for each $i' \in I$ we have

$$\sum_{j \in N(i')} y^{i'}_{i'j} = \sum_{j \in N(i')} d_j.$$

As $q \cdot x_{i'}$ needs to be larger or equal than this sum, we directly get that $\hat{x}$ is the optimal solution to Min-q-fC. □

Although it seems like an interesting approach to let the client decide which supplier it chooses, after looking at the proof of Theorem 8.1 the approach might be too conservative. In the remainder of this section we also consider the following alternative of q-fC in which for each region, additionally, we are given an ordering of the adjacent locations. Each client in a region would like to be served by the supplier at the front of the ordered locations. Of course, this is only possible if at least one supplier is available in the corresponding location. Such a setting might for example make sense when regarding a geographic setup in which each client wants chooses the closest available supplier.

q-orderedClient (q-oC).
Instance: Finite, disjoint sets I, J, weights $d_j \in \mathbb{N}$ for all $j \in J$, a bipartite graph $G = (I \cup J, E)$, for each $j \in J$ an ordering $\sigma_j \colon \{1, \dots, |N(j)|\} \to N(j)$, and an integer $B \in \mathbb{N}$.
Question: Does there exist $x \in \mathbb{N}^{|I|}$ for $i \in I$ with $x(I) \leq B$ such that for each $j \in J$ there exists an $i \in N(j)$ with $x_i \geq 1$ and

$$\sum_{j \in N(i)} y_{ij} \leq q \cdot x_i \quad \forall i \in I,$$

where $y \in \mathbb{N}^{|I| \times |J|}$ is defined by setting

$$y_{ij} = \begin{cases} d_j, & \text{if } i = \operatorname{argmin}_{i' \in N(j)} \left\{ \sigma_j^{-1}(i') \colon x_{i'} \geq 1 \right\}, \\ 0, & \text{else,} \end{cases}$$

for $i \in I$, $j \in J$?

The description of q-oC seems more challenging and less conservative than q-fC. In the optimization version Min-q-orderedClient we are again looking for a minimum number of suppliers. We begin by regarding the problem for $q = 1$.

Theorem 8.2. *Min-1-orderedClient can be solved in time $O(|I| + |J|)$.*

Proof. Given an instance of Min-1-oC we know that we need at least $d(J)$ suppliers. We define a solution iteratively in the following way. At the start we set $x_i = 0$ for all $i \in I$. For each $j \in J$ we now increase the value of $x_{\sigma_j(1)}$ by d_j. This procedure produces a valid solution with $d(J)$ suppliers and is therefore optimal. It can clearly be implemented to run in time $O(|I| + |J|)$. □

The reason why the solution to Min-1-oC is so simple is that we know the exact locations the clients choose. In fact this observation generalizes a bit further. If we know the subset of *opened locations*, i.e., the locations $i \in I$ with $x_i \geq 1$, the values y_{ij} in the problem definition become fixed. This fact motivates the following definitions.

Definition 8.3 (Induced solutions). Let an instance of q-orderedClient and a subset $I' \subseteq I$ of the locations be given. For each $j \in J$ with $N(j) \cap I' \neq \emptyset$ we call

$$\operatorname*{argmin}_{i' \in N(j)} \left\{ \sigma_j^{-1}(i') \colon i' \in I' \right\}$$

the *closest location* with respect to I'. Further we call

$$N'(i) = \left\{ j \in N(i) \colon i = \operatorname*{argmin}_{i' \in N(j)} \left\{ \sigma_j^{-1}(i') \colon i' \in I' \right\} \right\}$$

the *active* neighborhood of i *induced* by I'. Finally we call the solution $\hat{x} \in \mathbb{N}^{|I|}$ with

$$\hat{x}_i = \left\lceil \frac{d(N'(i))}{q} \right\rceil$$

for $i \in I$ the solution *induced* by I'.

Observation 8.4. *Let an instance of* Min-*q*-orderedClient *and a subset of the locations $I' \subseteq I$ be given. If $N(j) \cap I' \neq \emptyset$ for all $j \in J$, then $\dot{\bigcup}_{i \in I'} N'(i) = J$ and the solution $\hat{x}$ induced by I' is feasible. Denote by $\mathcal{L}$ the set of all feasible solutions x such that*

$$\{i \in I \colon \hat{x}_i \geq 1\} \subseteq \{i \in I \colon x_i \geq 1\} \subseteq I'. \tag{8.1}$$

If $\hat{x}$ is feasible, it holds true that $\hat{x}$ is the unique solution fulfilling

$$\hat{x} = \operatorname*{argmin}_{x \in \mathcal{L}} \left\{ \sum_{i \in I} x_i \right\}.$$

In particular, any optimal solution $x^\star$ to an instance of Min-*q*-orderedClient *is induced by the subset $I^\star := \left\{ i \in I \colon x_i^\star \geq 1 \right\}$.*

At first glance it might not be clear why the first inclusion in (8.1) is necessary for the statement in Observation 8.4 to hold. The reason for this is that it might be necessary to keep a location closed in order to prevent clients from choosing this location. To clarify this, regard the following situation. We are given two regions and two locations. Assume $q = 2$ and each region contains one client. If each of the two clients prefers a different location, the solution induced by all locations places one supplier in each location. But in an optimal solution we would only place a single supplier in any of the two locations.

A consequence of Observation 8.4 is that we may actually bound the number of clients in any region by a 1 as we see in Lemma 8.5.

Lemma 8.5. *Let ALG be an algorithm that solves instances of* Min-*q*-orderedClient *with $d_j = 1$ for all $j \in J$ in time $O(\mathcal{T})$. Then we can solve instances of* Min-*q*-orderedClient *in time $O(\mathcal{T} + |I| + |J|)$.*

Proof. Let an arbitrary instance $\mathcal{I}$ of Min-*q*-oC be given, where we assume that $d_j \geq 1$ for all $j \in J$. For each region $j \in J$, we denote by $r_j, k_j \in \mathbb{N}$ the unique integers fulfilling $1 \leq r_j \leq q$ and $q \cdot k_j + r_j = d_j$. Assume we are given an optimal solution $\bar{x}$ to a new instance $\mathcal{I}'$ with the same input data as $\mathcal{I}$ except that the demand in each region $j \in J$ is set to be $d'_j = r_j$. By Observation 8.4, we may assume that $\bar{x}$ is induced by some subset of the locations $I' \subseteq I$. We now claim that the solution $\hat{x}$ for $\mathcal{I}$ that is induced by I' is optimal for $\mathcal{I}$. By Observation 8.4 it is enough to prove that any feasible, induced solution has objective value at least $\hat{x}(I)$. So let $\hat{\chi}$ be any feasible solution for $\mathcal{I}$ induced by some subset $I'' \subseteq I$. We denote by $\bar{\chi}$ the solution for $\mathcal{I}'$ that is induced by I'' and

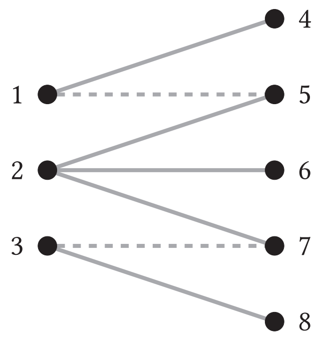

Figure 8.1.: Graph of the Min-2-oC instance in Example 8.6. Solid lines represent arcs of preference order 1 and dashed lines of preference order 2.

by $N''(i)$ the active neighborhood of $i \in I$, induced by the set I'' and calculate

$$\begin{aligned}\sum_{i\in I} \hat{\chi}_i &= \sum_{i\in I} \left\lceil \frac{\sum_{j\in N'(i)} d_j}{q} \right\rceil = \sum_{i\in I} \sum_{j\in N'(i)} k_j + \left\lceil \frac{r_j}{q} \right\rceil \\ &= \sum_{i\in I} \sum_{j\in N'(i)} k_j + \sum_{i\in I} \sum_{j\in N'(i)} \left\lceil \frac{r_j}{q} \right\rceil \\ &= \sum_{j\in J} k_j + \sum_{i\in I} \bar{\chi}_i \geq \sum_{j\in J} k_j + \sum_{i\in I} \bar{x}_i = \sum_{i\in I} \hat{x}.\end{aligned}$$

Thus, $\hat{x}$ is optimal for $\mathcal{I}$.

Now note, that $r_j \leq q$ and therefore constant. Thus, instead of solving an instance with demands r_j we can equivalently solve in instance in which each region $j \in J$ is duplicated r_j times and assigned the demand 1. This can be realized in time $O(|J|)$. The above procedure, provided we are given an optimal solution to $\mathcal{I}'$, can then be implemented to run in time $O(|I| + |J|)$ yielding the desired result. □

The difference between q-MsMc and q-oC is that in the latter we may not *force* the clients to choose the supplier we would like them to choose. This is the main reason why the procedure used to solve Min-2-MsMc in Theorem 7.7 does not work for Min-2-oC, as the following example shows.

Example 8.6. Regard an instance of Min-2-oC with the following properties. Let $I = \{1, 2, 3\}$, $J = \{4, 5, 6, 7, 8\}$ and set

$$G = (I \cup J, \{14, 15, 25, 26, 27, 37, 38\}).$$

Further, set $d_j = 1$ for all $j \in J$, $\sigma_5(1) = 2$, $\sigma_5(2) = 1$, $\sigma_7(1) = 2$ and $\sigma_7(2) = 3$. The orderings σ_4, σ_6 and σ_8 are already determined, as all of these regions only have one adjacent location. For the construction of the instance, see also Figure 8.1. As for any solution x and all $j \in J$ there

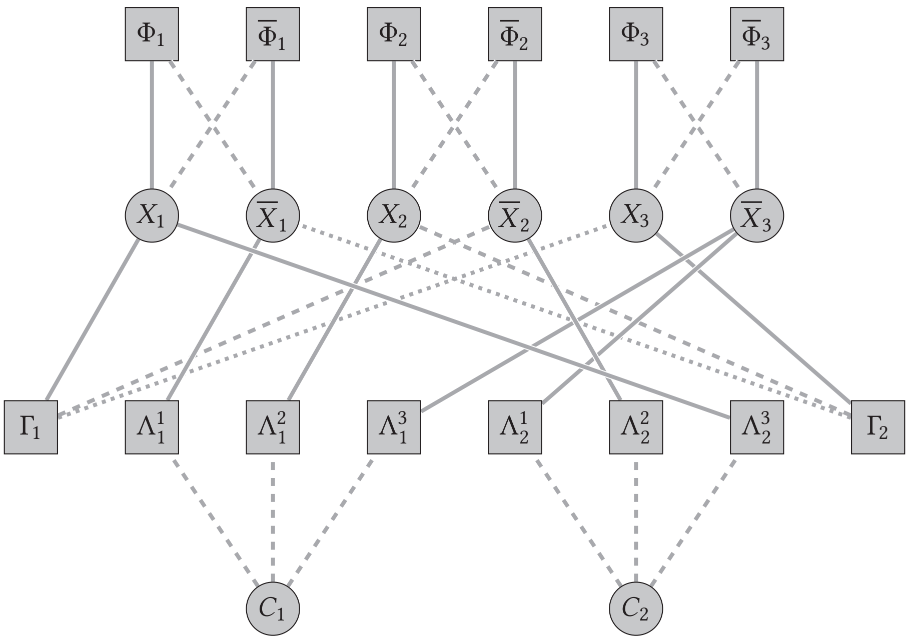

Figure 8.2.: Construction of the 2-oC instance in the proof of Theorem 8.7 given the 1-in-3-SAT instance with the formula $\left(\overline{X}_1 \vee X_2 \vee \overline{X}_3\right) \wedge \left(\overline{X}_3 \vee \overline{X}_2 \vee X_1\right)$. Circle nodes are locations and square nodes are regions. Solid lines indicate preference order 1, dashed lines indicate preference order 2 and dotted lines indicate preference order 3.

must be an $i \in N(j)$ with $x_i \geq 1$, we get that every solution fulfills $x_i \geq 1$ for all $i \in I$. Thus, by Observation 8.4, an optimal solution to Min-2-oC is given by $\hat{x}_1 = 1$, $\hat{x}_2 = 2$ and $\hat{x}_3 = 1$. If we try to solve the instance using Edge Cover as in the procedure for 2-Multiset Multicover in Theorem 7.7, we get the solution that has one supplier in each location. This solution, though, is infeasible for Min-2-oC.

Example 8.6 shows that Min-2-oC cannot be solved with the same techniques as Min-2-MsMc. In fact, unless P = NP, there is no polynomial time solving Min-2-oC as the following theorem shows.

Theorem 8.7. *2-orderedClient is* NP*-complete in the strong sense.*

Proof. We can verify a given possible solution to 2-oC in polynomial time using Observation 8.4. Therefore 2-oC is contained in NP.

We reduce 1-in-3-SAT to 2-oC to see that it is also NP-hard. So let $X_1, \dots, X_n$ be the variables of a 1-in-3-SAT instance and $C_1, \dots, C_m$ be the clauses. Further denote by L_i^j for $i \in \{1, \dots, m\}$ and $j \in \{1, 2, 3\}$ the jth literal of clause i. We may assume that each variable, as well as its negation, appear an even number of times in the given formula. Otherwise simply duplicate each clause.

We define an instance of 2-oC. By $I = I_X \cup I_C$ we denote the set of locations, where the first set $I_X = \left\{X_k, \overline{X}_k \colon k \in \{1, \dots, n\}\right\}$ contains each variable and its negation and the second set $I_C = \{C_k \colon k \in \{1, \dots, m\}\}$ contains each clause. The set of regions we set to be $J = J_X \cup J_C \cup J_L$, where $J_X = \left\{\Phi_k, \overline{\Phi}_k \colon k \in \{1, \dots, n\}\right\}$ contains an element for each variable and its negation, $J_C = \{\Gamma_k \colon k \in \{1, \dots, m\}\}$ contains an element for each clause and, finally, the set $J_L = \left\{\Lambda_k^l \colon k \in \{1, \dots, m\},\ l \in \{1, 2, 3\}\right\}$ contains an element for each literal in each clause. The graph of the instance is then set to $G = (I \cup J, E_X \cup E_C \cup E_{\overline{C}} \cup E_L)$, where

$$E_X = \bigcup_{k=1}^{n} \left\{x\phi \colon x \in \left\{X_k, \overline{X}_k\right\}, \phi \in \left\{\Phi_k, \overline{\Phi}_k\right\}\right\}, \qquad E_C = \bigcup_{k=1}^{m} \left\{C_k \Lambda_k^l \colon l \in \{1, 2, 3\}\right\},$$

$$E_{\overline{C}} = \bigcup_{k=1}^{m} \left\{\overline{L}_k^l \Gamma_k \colon l \in \{1, 2, 3\}\right\}, \qquad E_L = \bigcup_{k=1}^{m} \left\{L_k^l \Lambda_k^l \colon l \in \{1, 2, 3\}\right\}.$$

For each $j \in J$ we denote the ordering of its neighborhood σ_j as an ordered tuple. The i^{th} entry of the tuple corresponds to $\sigma_j(i)$. We set

$$\sigma_j = \begin{cases} (X_k, \overline{X}_k), & \text{if } j = \Phi_k \in J_X, \\ (\overline{X}_k, X_k), & \text{if } j = \overline{\Phi}_k \in J_X, \\ (L_k^l, C_k), & \text{if } j = \Lambda_k^l \in J_L, \\ (\overline{L}_k^1, \overline{L}_k^2, \overline{L}_k^3), & \text{if } j = \Gamma_k \in J_C, \end{cases} \quad \text{and} \quad d_j = \begin{cases} 2, & \text{if } j \in J_C, \\ 1, & \text{else.} \end{cases}$$

Finally we set

$$B = \frac{5m + 2n}{2} = \frac{1}{2} \sum_{j \in J} d_j.$$

Note that $B \in \mathbb{N}$ as all variables appear an even amount of times and therefore the number of clauses is also even. For the construction of the instance, see also Figure 8.2.

Now assume the given 1-IN-3-SAT formula is satisfiable. We define a set of locations I', the set of locations we want to open, by setting $I' = I_C \cup \left\{L_k \in \left\{X_k, \overline{X}_k\right\} \colon k = 1, \dots, n,\ L_k = \text{TRUE}\right\}$. We begin by proving that for each $j \in J$, there exists some $i \in N(j)$ with $i \in I'$. If $j \in J_L$, there is some $i \in I_C \cap N(j)$ and thus, $i \in I'$. If $j \in J_X$ it is $X_k, \overline{X}_k \in N(j)$ for some $k \in \{1, \dots, n\}$ and by definition of I' either $X_k \in I'$ or $\overline{X}_k \in I'$. Finally assume $j \in J_C$. As exactly one literal in each clause is satisfied, this means that the negations of the two other literals are set to TRUE and therefore there exist two elements in $N(j)$ that are also contained in I'. We now show that the solution $\hat{x}$ induced by I', cf. Definition 8.3, fulfills $\hat{x}(I) = B$. To this end it is enough to show that $d(N'(i))$ is even for each $i \in I'$, where $N'(i)$ denotes the active neighborhood of i induced by I'.

So let $i \in I'$ be given. We begin with the case that $i \in I_C$, say $i = C_k$ for some $k \in \{1, \dots, m\}$. The vertex C_k has the three neighbors Λ_k^l for $l \in \{1, 2, 3\}$. All of those, according to σ, prefer the connection to their respective literal location L_k^l. As exactly one literal in the clause is set to TRUE, only one of these locations is contained in I', say L_k^3. Thus, the clients in Λ_k^1 and Λ_k^2 are served by C_k, which is an even number.

Now let $i \in I_X$, say $i = L_k$ for some $k \in \{1, \dots, n\}$. By the definition of I' we have $L_k = \text{TRUE}$. Thus, $\overline{L}_k = \text{FALSE}$ and $\overline{L}_k \notin I'$ and it is $N'(i) \cap J_X = \left\{\Phi_k, \overline{\Phi}_k\right\}$. Further, $N'(i) \cap J_L$ contains an element for each time L_k appears in the given 1-IN-3-SAT formula, which is an even amount. We conclude that

$$\sum_{j \in N'(i)} d_j = |N'(i) \cap J_X| + |N'(i) \cap J_L| + 2 \cdot |N'(i) \cap J_C| \tag{8.2}$$

is even.

Now let $\hat{x}$ be a solution to 2-oC with $\hat{x}(I) \leq B$ and denote by $I' = \{i \in I : \hat{x}_i \geq 1\}$ the set of open locations for $\hat{x}$. By Observation 8.4 we may assume without loss of generality that $\hat{x}$ is the solution induced by I'. Note that $d(N'(i))$, where $N'(i)$ is the active neighborhood of i induced by I', must be even for all i, as otherwise $\hat{x}(I) > B$.

We begin by showing that, for each $k \in \{1, \dots, n\}$, it is either $X_k \in I'$ or $\overline{X}_k \in I'$, but not both. As $N(\Phi_k) = \left\{X_k, \overline{X}_k\right\}$ it is clear, that at least one of the two is contained in I'. So suppose both X_k and $\overline{X}_k$ are contained in I'. Then equation (8.2) holds for $i = X_k$ but $|N'(X_k) \cap J_X|$ is odd, while the other two summands are even. This contradicts the fact that $d(N'(X_k))$ is even. We may now formally define a truth assignment to the variables. We set $X_k = \text{TRUE}$ if and only if $X_k \in I'$. Note that by the above arguments this definition implies that $\overline{X}_k = \text{TRUE}$ if and only if $\overline{X}_k \in I'$.

In a next step we will prove that $I_C \subseteq I'$. Suppose for the sake of contradiction that for some clause C_k we have $C_k \notin I'$. As, for each $l \in \{1, 2, 3\}$, the only other neighbor of Λ_k^l is L_k^l, we must have $L_k^l \in I'$ and therefore $L_k^l = \text{TRUE}$. This implies that $\overline{L}_k^l = \text{FALSE}$ and therefore $\overline{L}_k^l \notin I'$ for all $l \in \{1, 2, 3\}$. As $N(\Gamma_k) = \left\{\overline{L}_k^1, \overline{L}_k^2, \overline{L}_k^3\right\}$, this is a contradiction to the fact that $\hat{x}$ is a feasible solution. Thus, for each k there is some $l \in \{1, 2, 3\}$ such that $L_k^l = \text{FALSE}$. If exactly one or three of these literals were to be FALSE, then $d(N'(C_k))$ would be odd, which is not possible. Thus, exactly one literal in each clause is set to TRUE, which is what we wanted to prove. □

Note that if we were to repeat the proof of Theorem 8.7 for any fixed $q \geq 3$, we would have to make some adjustments. In fact as we already know that q-MULTISET MULTICOVER is NP-hard for $q \geq 3$ it is possible to simplify the proof. In the following proof we ensure that the clients only have one choice of supplier. This way the problem becomes, in some sense, equivalent to q-MsMc. In turn this means that the preference ordering is not needed in the proof.

Theorem 8.8. *q-ORDEREDCLIENT is NP-complete in the strong sense for any fixed $q \in \mathbb{N}$ with $q \geq 3$.*

Proof. Given a possible solution to q-oC, we can verify it in polynomial time using Observation 8.4. Thus, q-oC is contained in NP.

Similarly to the proof of Theorem 7.8 we reduce EXACT COVER BY 3-SETS to q-oC in order to prove NP-hardness. From an EXACT COVER BY 3-SETS instance, let X be the finite ground set and $\mathcal{S}$ a collection of subsets of X, where $|S| = 3$ for all $S \in \mathcal{S}$.

Set $I = \mathcal{S}$, $J = X$ and define the graph $G = (I \cupdot J, E)$, where an edge $Sj \in E(G)$, if $j \in S$ for $S \in I$ and $j \in J$. The ordering of the regions $j \in J$ is defined as an arbitrary bijection. Finally set $d_j = 1$ for all $j \in J$ and $B = |X|/3$.

Assume we are given a solution $\mathcal{S}'$ to Exact Cover by 3-sets. We set $x_S = 1$ if $S \in \mathcal{S}'$ and $x_S = 0$ else. As $\mathcal{S}'$ is a solution to Exact Cover by 3-sets, for each $j \in J$ there exists $i \in N(j)$ with $x_i = 1$. Let now $i \in I$ and assume $x_i = 1$. We have

$$\sum_{j \in N(i)} y_{ij} \leq q \cdot x_i$$

as $q \geq 3$, $|N(i) = 3|$ and $d_j = 1$ for all $j \in J$. If $x_i = 0$ by the very definition of the vector y, it is $y_{ij} = 0$ for all $j \in N(i)$ and the equation holds as well.

Assume now that we are given a solution x to q-oC with $x(I) \leq |X|/3$. Then for all $i \in I$ it holds true that $x_i \leq 1$ by the feasibility of x. We set $\mathcal{S}' = \{S \in \mathcal{S} \colon x_S = 1\}$. This is a solution to Exact Cover by 3-sets, as $|S'| = |J|/3$ and for each $j \in J$ there exists some $S \in N(j)$ with $x_S = 1$ and $j \in S$. □

This finishes the classification of the complexity of q-oC. In contrast to q-MsMc, the problem is NP-hard even when $q = 2$. This confirms the intuition that q-oC is the harder problem regarding complexity. Nevertheless the problem remains polynomial time solvable for $q = 1$. As it is NP-complete for any fixed $q \geq 2$, we present an integer linear programming formulation of the problem. In the literature there are different approaches to model the orderings for each client, cf. [HP87]. As we do not compare various formulations here, we only supply one possibility here.

$$\begin{aligned}
\text{(IP 8.3)}(d) \quad \min_{v,x,z} \quad & \sum_{i \in I} x_i && && \text{(8.3a)}\\
\text{s.t.} \quad & \sum_{j \in N(i)} d_j \cdot z_{ij} \leq q \cdot x_i && \forall i \in I && \text{(8.3b)}\\
& \sum_{i \in N(j)} v_i \geq 1 && \forall j \in J && \text{(8.3c)}\\
& v_i - \sum_{k=1}^{\sigma_j^{-1}(i)-1} v_{\sigma_j(k)} \leq z_{ij} && \forall j \in J, i \in N(j) && \text{(8.3d)}\\
& v_i, z_{ij} \in \{0,1\} && \forall i \in I, j \in J && \text{(8.3e)}\\
& x_i \in \mathbb{N} && \forall i \in I. && \text{(8.3f)}
\end{aligned}$$

The variables of (IP 8.3) can be interpreted in the following way, where the meaning of the variables x should be clear from the definition of q-oC.

$$v_i = \begin{cases} 1, & \text{if } x_i \geq 1, \\ 0, & \text{else.} \end{cases} \tag{8.4}$$

$$z_{ij} = \begin{cases} 1, & \text{if } i = \operatorname{argmin}_{i' \in N(j)} \left\{ \sigma_j^{-1}(i') \colon x_{i'} \geq 1 \right\}, \\ 0, & \text{else.} \end{cases} \tag{8.5}$$

Strictly speaking, conditions (8.4) and (8.5) are not always true. The next theorem shows that we may safely ignore these cases.

Theorem 8.9. *Let an instance of* MIN-*q*-ORDEREDCLIENT *be given and let* $x \in \mathbb{N}^{|I|}$. *x is an optimal solution to* MIN-*q*-ORDEREDCLIENT *if and only if there exist* $v \in \{0,1\}^{|I|}$ *and* $z \in \{0,1\}^{|I| \times |J|}$ *such that* (v, x, z) *is optimal for (IP 8.3)(d).*

Proof. First assume that x is an optimal solution to MIN-q-oC. For $i \in I$ we set $v_i = 1$ if $x_i \geq 1$ and $v_i = 0$ else. Recall the definition of y in the problem description of q-oC. For $j \in J$ and $i \in N(j)$ it is

$$y_{ij} = \begin{cases} d_j, & \text{if } i = \operatorname{argmin}_{i' \in N(j)} \left\{ \sigma_j^{-1}(i') \colon x_{i'} \geq 1 \right\}, \\ 0, & \text{else,} \end{cases}$$

For $j \in J$ and $i \in N(j)$, we set $z_{ij} = 1$ if $y_{ij} = d_j$ and $z_{ij} = 0$ else. The only equation that is now not trivially fulfilled is equation (8.3d). If the left hand side of the equation is less or equal to zero, it is also trivially fulfilled. So assume

$$v_i - \sum_{k=1}^{\sigma_j^{-1}(i)-1} v_{\sigma_j(k)} = 1$$

for some $j \in J$ and $i \in N(j)$. This is only the case if $v_i = 1$ and $v_{\sigma_j(k)} = 0$ for all $k \in \left\{1, \ldots, \sigma_j^{-1}(i) - 1\right\}$. But this implies that

$$i = \operatorname*{argmin}_{i' \in N(j)} \left\{ \sigma_j^{-1}(i') \colon x_{i'} \geq 1 \right\}$$

and therefore $y_{ij} = d_j$. By the definition of z, it is therefore $z_{ij} = 1$ and the equation is fulfilled.

Assume now that we are given an optimal solution (v, x, z) to (IP 8.3)(d). We begin by proving that, if $x_i \geq 1$ we also have $v_i = 1$. If $x_i \geq 1$, there exist $j \in N(i)$ such that $z_{ij} = 1$, as otherwise we would set x_i to zero yielding a better objective value. Let $J' = \left\{ j \in N(i) \colon z_{ij} = 1 \right\}$. If v_i is not 1, then the solution remains feasible if for all $j \in J'$ we set $z_{ij} = 0$ and also set $x_i = 0$, again yielding a better objective value. Thus, we get that $v_i = 1$.

Next suppose there exists some $j \in J$ such that $x_i = 0$ for all $i \in N(j)$. By constraints (8.3b) this implies $z_{ij} = 0$ for all $i \in N(j)$. This, in turn, implies by constraints (8.3d) that $v_i = 0$ for all $i \in N(j)$, which is a contradiction to constraints (8.3c). Thus, for each $j \in J$ there exists some $i \in N(j)$ with $x_i \geq 1$.

Now let $j \in J$ be arbitrary and set $i = \operatorname{argmin}_{i' \in N(j)} \left\{\sigma_j^{-1}(i') \colon x_{i'} \geq 1\right\}$. If we can show that $z_{ij} = 1$, the second condition of Min-q-oC is fulfilled by constraints (8.3b) and we are done. By the very definition of i we have $x_i \geq 1$. We already argued above that this implies $v_i = 1$. Now let $k \in \left\{1, \ldots, \sigma_j^{-1}(i)\right\}$ be minimal with the condition that $v_{\sigma_j(k)} = 1$. Suppose $k < \sigma_j^{-1}(i)$ for the sake of contradiction. By constraints (8.3d) this implies $z_{\sigma_j(k)j} = 1$, which in turn implies $x_{\sigma_j(k)} \geq 1$ by constraints (8.3b) in contradiction to the definition of i. Thus, we get that $v_{\sigma_j(k)} = 0$ for all $k < \sigma_j^{-1}(i)$ and by constraints (8.3d), this implies $z_{ij} = 1$. □

This concludes the classification of the complexity of q-fC and q-oC. For the remainder of this chapter we regard robust versions of both problems and analyze the influences of the uncertainty in the data on the complexity of the problem.

8.2. Free and Ordered Client Including Demand Uncertainty

As in the chapter about Robust q-Multiset Multicover we now regard instances of the two problems q-freeClient and q-orderedClient in which the number of clients in the regions is not exactly known in advance. For q-fC this leads to the following definition.

Robust q-freeClient (Robust q-fC).
Instance: Finite, disjoint sets I, J, an uncertainty set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \mathbin{\dot\cup} J, E)$, and an integer $B \in \mathbb{N}$.
Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \leq B$ such that for all $y \in \mathbb{N}^{|I|\times|J|}$ with $\sum_{i\in N(j)} y_{ij} = \xi_j$ for some $\xi \in \mathcal{U}$ and all $j \in J$, it is

$$\sum_{j\in N(i)} y_{ij} \leq q \cdot x_i \quad \forall i \in I?$$

The solution to Robust Min-q-fC turns out to be not much more complicated than the one for Min-q-fC. In fact it is fairly obvious that for each location we have to compute the maximum amount of clients that may choose this location among all scenarios.

Theorem 8.10. *Let an instance of Robust Min-q-freeClient be given. Then $\hat{x} \in \mathbb{N}^{|I|}$, defined by setting*

$$\hat{x}_i = \left\lceil \frac{\max_{\xi\in\mathcal{U}} \{\xi(N(i))\}}{q} \right\rceil,$$

is an optimal solution to the given instance.

Proof. For each $i \in I$ the term $\max_{\xi\in\mathcal{U}} \{\xi(N(i))\}$ is the maximum number of clients that may have to be served by location i. Thus, $\hat{x}$ is feasible and $\hat{x}(I)$ is a lower bound on the optimal solution to Robust Min-q-fC. This directly implies that $\hat{x}$ is optimal. □

Although writing down the solution is not very complicated, as the computation of the maximum in Theorem 8.10 is essentially MaxSum, which is NP-complete, the next corollary is an immediate consequence.

Corollary 8.11. *Robust q-freeClient is* NP*-hard in the strong sense for any fixed q, even when restricted to instances with a single location.* □

Nevertheless, it is a direct consequence of Theorem 8.10 that we can solve Robust q-fC in polynomial time whenever we can compute

$$\max_{\xi\in\mathcal{U}} \{\xi(N(i))\}$$

in polynomial time. Next, we turn to a robust version of q-orderedClient.

Robust q-orderedClient (Robust q-oC).
Instance: Finite, disjoint sets I, J, an uncertainty set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \cupdot J, E)$, for each $j \in J$ an ordering $\sigma_j\colon \{1,\dots,|N(j)|\} \to N(j)$, and an integer $B \in \mathbb{N}$.
Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \le B$ such that for each $j \in J$ there exists an $i \in N(j)$ with $x_i \ge 1$ and

$$\sum_{j\in N(i)} y(\xi)_{ij} \le q \cdot x_i \quad \forall i \in I, \xi \in \mathcal{U},$$

where $y(\xi) \in \mathbb{N}^{|I|\times|J|}$ is defined by setting

$$y(\xi)_{ij} = \begin{cases} \xi_j, & \text{if } i = \operatorname{argmin}_{i'\in N(j)} \left\{\sigma_j^{-1}(i')\colon x_{i'} \ge 1\right\}, \\ 0, & \text{else,} \end{cases}$$

for $i \in I$, $j \in J$?

As in the non-robust case, we introduce the notion of *induced solutions*, cf. Definition 8.3, which are, in some sense, the only interesting solutions.

Definition 8.12 (Robust Induced Solutions). Let an instance of Robust Min-q-oC be given, and let $I' \subseteq I$ be a subset of the locations of the instance. We call $\hat{x} \in \mathbb{N}^{|I|}$ the solution *induced* by I' if it fulfills

$$\hat{x}_i = \left\lceil \frac{\max_{\xi\in\mathcal{U}} \left\{\sum_{j\in N'(i)} \xi_j\right\}}{q} \right\rceil,$$

where $N'(i)$ denotes the active neighborhood of i induced by I', cf. Definition 8.3.

Lemma 8.13. *Let an instance of* Robust Min-*q*-oC *and a subset $I' \subseteq I$ be given. The solution $\hat{x}$ induced by I' is feasible if and only if $I' \cap N(j) \neq \emptyset$ for all $j \in J$. Denote by $\mathcal{L}$ the set of all feasible solutions x such that $\{i \in I \colon \hat{x}_i \geq 1\} \subseteq \{i \in I \colon x_i \geq 1\} \subseteq I'$. If $\hat{x}$ is feasible, it is the unique solution fulfilling*

$$\hat{x} = \operatorname*{argmin}_{x \in \mathcal{L}} \left\{ \sum_{i \in I} x_i \right\}.$$

In particular any optimal solution $x^\star$ is induced by the set $\left\{i \in I \colon x_i^\star \geq 1\right\}$.

Proof. Observe that $\xi(N'(i))$ is the number of clients that will have to be served by location i in scenario ξ provided that I' is the set of open locations, cf. Observation 8.4. Thus, in this case $\max_{\xi \in \mathcal{U}} \{\xi(N'(i))\}$ is the largest amount of clients that will have to be served by location i. This immediately implies that $\hat{x}_i$ is the minimum possible number of suppliers needed in location i. Further we observe, that if $I' \cap N(j) \neq \emptyset$ for all $j \in J$, then $\hat{x}$ is also a feasible solution. □

Lemma 8.13 basically states that verifying a solution takes a MaxSum computation to be done. In a next step we prove NP-hardness of Robust q-oC. The proof is similar to the one in Theorem 7.8. Nevertheless, we state it here for the sake of completeness.

Theorem 8.14. *Robust q-orderedClient is* NP*-hard in the strong sense for any fixed $q \in \mathbb{N}_{>0}$.*

Proof. Let a graph H with $V(H) = \{1, \dots, n\}$ and an integer k from an instance of Dominating Set be given. We construct an instance of Robust q-oC by setting $I := V(H)$, $J = \{n+1, \dots, 2n\}$, $B = k$ and $G = (I \cup J, E)$, where we have $u(n+v) \in E$ if $u = v$ or $uv \in E(H)$. Further, for each $j \in J$, we set $\sigma_j \colon \{1, \dots, |N(j)|\} \to N(j)$ to be any bijection and $\mathcal{U} = \left\{\xi \in \mathbb{N}^{|J|} \colon \xi(J) = 1\right\}$.

Assume $S \subseteq V(H)$ is a dominating set in H, with $|S| \leq k$. We set $\hat{x}_i = 1$ if $i \in S$ and $\hat{x}_i = 0$ else. It is obvious that $\hat{x}(I) \leq B$. Further, as S is dominating for each $j \in J$ there exists an $i \in N(j)$ with $x_i \geq 1$. Finally note, that for each $\xi \in \mathcal{U}$ there exists exactly one $j \in J$ and one $i \in N(j)$ with $y(\xi)_{ij} = 1$. For this i, we also have $\hat{x}_i = 1$ and thereby $\hat{x}$ fulfills the remaining conditions for Robust q-oC.

Now assume $\hat{x}$ fulfills the conditions of Robust q-oC. We define $S' = \{i \in I \colon \hat{x}_i \geq 1\}$. We have $|S'| \leq \hat{x}(I) \leq B = k$. Now let $u \in V(H) = I$ be any node in H. We know by the conditions of Robust q-oC that there exists some $u' \in N_G(u+n)$ such that $\hat{x}_i \geq 1$ and thereby $u' \in S'$. By the definition of G we also have $u' \in \{u\} \cup N_H(u)$ and S' is a dominating set in H. □

Regardless of the NP-hardness of Robust q-oC, we still would like to have the means to solve the problem. Therefore we give a valid integer programming formulation for Robust Min-q-oC.

$$\begin{aligned}
\text{(IP 8.6)}(\mathcal{U}) \quad \min_{x,v,z} \quad & \sum_{i \in I} x_i && && \text{(8.6a)}\\
\text{s.t.} \quad & \sum_{j \in N(i)} \xi_j \cdot z_{ij} \le q \cdot x_i && \forall i \in I,\ \forall \xi \in \mathcal{U} && \text{(8.6b)}\\
& \sum_{i \in N(j)} v_i \ge 1 && \forall j \in J && \text{(8.6c)}\\
& v_i - \sum_{k=1}^{\sigma_j^{-1}(i)-1} v_{\sigma_j(k)} \le z_{ij} && \forall j \in J, i \in N(j) && \text{(8.6d)}\\
& v_i, z_{ij} \in \{0,1\} && \forall i \in I, j \in J && \text{(8.6e)}\\
& x_i \in \mathbb{N} && \forall i \in I. && \text{(8.6f)}
\end{aligned}$$

The correctness of the formulation is immediate from Theorem 8.9.

Theorem 8.15. *Let an instance of Robust Min-q-orderedClient be given. Then $x^\star \in \mathbb{N}^{|I|}$ is an optimal solution to the instance if and only if there exist $v^\star \in \{0,1\}^{|I|}$ and $z^\star \in \{0,1\}^{|I|\times|J|}$ such that $(x^\star, v^\star, z^\star)$ is an optimal solution to (IP 8.6).* □

Regard the separation problem for (IP 8.6): Given a solution $(\hat{x}, \hat{v}, \hat{z})$ to (IP 8.6)($\mathcal{U}'$) for $\mathcal{U}' \subseteq \mathcal{U}$, does there exist a $\xi \in \mathcal{U}$ such that

$$\sum_{j \in N(i)} \xi_j \cdot \hat{z}_{ij} > q \cdot \hat{x}_i.$$

This is essentially MaxSum and we cannot hope to be able to separate in polynomial time in general. However, in the following we see cases in which we may separate in polynomial time.

The first such case, of course, is the one, when we may enumerate $\mathcal{U}$ in polynomial time, as MaxSum is contained in P for these cases.

The more interesting case is the one of polyhedral uncertainty sets. The constraints (8.6b) are the ones that we would like to replace as these are possibly exponentially many. The easiest way of doing so would be to replace for each $i \in I$ the set of constraints

$$\sum_{j \in N(i)} \xi_j \cdot z_{ij} \le q \cdot x_i, \quad \forall \xi \in \mathcal{U}$$

by the single constraint

$$\max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in N(i)} \xi_j \cdot z_{ij} \right\} \le q \cdot x_i. \tag{8.7}$$

Unfortunately the result is in general not an integer program because of the maximum in (8.7).

For a special case in polyhedral uncertainty we may reformulate that maximum in order to get a new integer programming formulation with polynomially many constraints. This is exactly the case when the underlying polyhedron is integral. This procedure has been used various times in the literature, see e.g. [BS04].

Let us formalize what we have just described. We are given an instance of Robust Min-q-oC for some $q \in \mathbb{N}_{>0}$ with a polyhedral uncertainty set $\mathcal{U} = \left\{\xi \in \mathbb{N}^{|J|} : A\xi \leq b\right\}$ for some integral polyhedron $P(A, b)$ with $A \in \mathbb{Z}^{n \times |J|}$ and $b \in \mathbb{Z}^n$ for some $n \in \mathbb{N}_{>0}$. Then, for some $i \in I$ and fixed $\hat{z}_{ij} \in \{0, 1\}$ for $j \in J$ the following holds true by duality of linear programming and the fact that P is integral.

$$\max_{\xi \in \mathbb{N}^{|J|}} \left\{\sum_{j \in J} \xi_j \cdot \hat{z}_{ij} : A\xi \leq b\right\} = \min_{\eta_i \in \mathbb{R}^n} \left\{b^\top \eta_i : A^\top \eta_i \geq \hat{z}_{i\cdot}\right\}. \tag{8.8}$$

Thus, we may equivalently reformulate (IP 8.6) as (MIP 8.9).

$$\text{(MIP 8.9)} \quad \min_{x, v, z, \eta} \quad \sum_{i \in I} x_i \tag{8.9a}$$

$$\text{s.t.} \quad b^\top \eta_i \leq q \cdot x_i \qquad \forall i \in I \tag{8.9b}$$

$$A^\top \eta_i \geq z_{i\cdot} \qquad \forall i \in I \tag{8.9c}$$

$$\sum_{i \in N(j)} v_i \geq 1 \qquad \forall j \in J \tag{8.9d}$$

$$v_i - \sum_{k=1}^{\sigma_j^{-1}(i)-1} v_{\sigma_j(k)} \leq z_{ij} \qquad \forall j \in J, i \in N(j) \tag{8.9e}$$

$$v_i, z_{ij} \in \{0, 1\} \qquad \forall i \in I, j \in J \tag{8.9f}$$

$$x_i \in \mathbb{N} \qquad \forall i \in I \tag{8.9g}$$

$$\eta_i \in \mathbb{R}^n_{\geq 0} \qquad \forall i \in I. \tag{8.9h}$$

Theorem 8.16. *Let an instance of Robust Min-q-orderedClient restricted to polyhedral uncertainty sets whose underlying polyhedra are integral be given. Then $x^\star \in \mathbb{N}^{|I|}$ is an optimal solution to the instance if and only if there exist $v^\star \in \{0, 1\}^{|I|}$, $z^\star \in \{0, 1\}^{|I| \times |J|}$ and $\eta^\star \in \mathbb{R}^{n \times |I|}_{\geq 0}$ such that $(x^\star, v^\star, z^\star, \eta^\star)$ is an optimal solution to (IP 8.9).*

Proof. We argued above why (MIP 8.9) is a valid formulation for Robust Min-q-oC. Note that the minimum in equations (8.9b) can be left out, as the minimum of a set of numbers is smaller or equal to a number p if and only if there exists some number in the set that is smaller than p. Further, the variables η can be chosen continuous as P is integral. □

Corollary 8.17. *Robust q-orderedClient is contained in* NP *for any fixed $q \in \mathbb{N}_{>0}$ when restricted to polyhedral uncertainty sets whose underlying polyhedra are integral.*

We can also use equation 8.8 to separate in (IP 8.6). As a consequence of the polynomial time solvability of linear programming, cf. [GLS88], the minimum on the left hand side of the equation can be computed in polynomial time. Thus, given a solution to (IP 8.6), we can decide in polynomial time, if there exists a scenario violating a constraint from (8.6b).

In the last section of this chapter we regard specific uncertainty sets and analyze the changes in the complexity of the problems.

8.3. Specific Uncertainty Sets

In the previous sections we analyzed the complexity of Robust q-freeClient and Robust q-orderedClient for very general uncertainty sets. Further, we gave two different integer programming formulations for Robust q-oC. In this section we restrict the uncertainty sets further, and see if we can improve the results of the previous section.

8.3.1. Discrete Uncertainty

Recall the definition of discrete uncertainty. Applied to Robust q-fC and Robust q-oC this means that the uncertainty sets in this section are of the form $\mathcal{U} = \{\xi_1, \dots, \xi_k\}$ for some $k \in \mathbb{N}$.

Theorem 8.18. *Robust q-freeClient with discrete uncertainty is in* P *for any fixed $q \in \mathbb{N}_{>0}$.*

Proof. By Theorem 8.1, we can compute an optimal solution $\hat{x}$ to Robust Min-q-fC by setting

$$\hat{x}_i = \left\lceil \frac{\max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in N(i)} \xi_j \right\}}{q} \right\rceil$$

for all $i \in I$. As the maximum can clearly be computed in linear time for discrete uncertainty, the result follows immediately. □

Considering the complexity of q-oC with discrete uncertainty, we note that the negative complexity results from q-MsMc with discrete uncertainty all directly transfer, as in the reductions the instances and solutions of q-MsMc were constructed in a way that each client had only one unique choice of supplier making the two problems equivalent. Thus, the following can be proved analogously to Theorems 7.28, 7.30 and 8.7.

Theorem 8.19. *Let $q, k \in \mathbb{N}_{>0}$ with $q \geq 2$ or $k \geq 3$. Robust q-orderedClient with discrete uncertainty is* NP*-complete in the strong sense, even when restricted to instances where the uncertainty set $\mathcal{U}$ has k elements.* □

As Robust 1-oC with exactly one scenario is simply the non-robust version of the problem the only remaining case is Robust 1-oC with two scenarios. In Chapter 7 we saw that Robust 1-MsMc

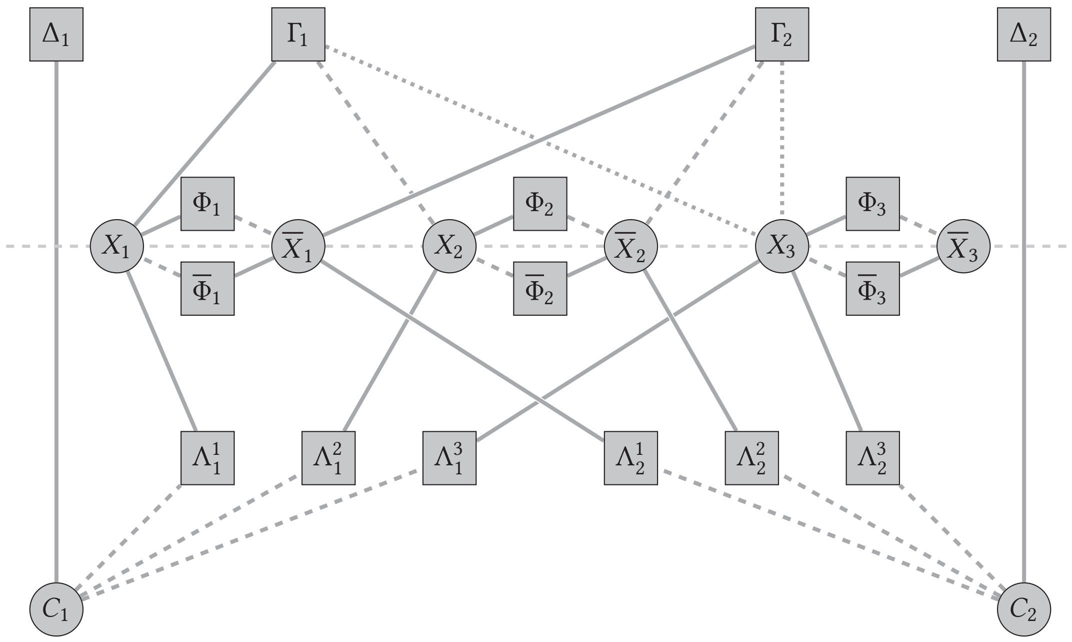

Figure 8.3.: Construction of the Robust 1-oC instance in the proof of Theorem 8.20 given the 1-in-3-SAT instance with the formula $(X_1 \vee X_2 \vee X_3) \wedge \left(\overline{X}_1 \vee \overline{X}_2 \vee X_3\right)$. Circle nodes are locations and square nodes are regions. Solid lines indicate preference order 1, dashed lines indicate preference order 2 and dotted lines indicate preference order 3. Regions above the gray dashed line contain clients in scenario ξ^1, regions below the line contain clients in scenario ξ^2.

with two scenarios has parallels to 2-MsMc. And indeed, we see that using a similar reduction as in the proof of Theorem 8.7 we can prove NP-hardness for Robust 1-oC with two scenarios.

Theorem 8.20. *Robust 1-orderedClient with discrete uncertainty restricted to uncertainty sets of cardinality 2 is* NP*-complete in the strong sense.*

Proof. We can verify a given solution of Robust 1-oC using Lemma 8.13 in polynomial time. Thus, Robust 1-oC is contained in NP.

To see that it is also NP-hard, we reduce 1-in-3-SAT to Robust 1-oC. Let $X_1, \dots, X_n$ be the variables of an 1-in-3-SAT instance and $C_1, \dots, C_m$ be the clauses. Further denote by L_i^j for $i \in \{1, \dots, m\}$ and $j \in \{1, 2, 3\}$ the j'th literal of clause i.

We define an instance of Robust 1-oC in the following way. We denote by $I = I_X \cup I_C$ the set of locations, where $I_X = \left\{X_k, \overline{X}_k \colon k \in \{1, \dots, n\}\right\}$ contains each variable and its negation and $I_C = \{C_k \colon k \in \{1, \dots, m\}\}$ contains each clause. The set of regions we set to be $J = J_X \cup J_C \cup J_L \cup J_D$, where $J_X = \left\{\Phi_k, \overline{\Phi}_k \colon k \in \{1, \dots, n\}\right\}$ contains an element for each variable and its negation, $J_C =$

$\{\Gamma_k \colon k \in \{1, \dots, m\}\}$ and $J_D = \{\Delta_k \colon k \in \{1, \dots, m\}\}$ both contain an element for each clause and $J_L = \left\{\Lambda_k^l \colon k \in \{1, \dots, m\},\ l \in \{1, 2, 3\}\right\}$ contains an element for each literal in each clause. The graph of the instance is then set to $G = (I \cup J, E_X \cup E_C \cup E_D \cup E_\Gamma \cup E_L)$, where

$$E_X = \bigcup_{k=1}^{n} \left\{ x\phi \colon x \in \left\{X_k, \overline{X}_k\right\}, \phi \in \left\{\Phi_k, \overline{\Phi}_k\right\}\right\},$$

$$E_C = \bigcup_{k=1}^{m} \left\{ C_k \Lambda_k^l \colon l \in \{1,2,3\}\right\}, \qquad E_D = \bigcup_{k=1}^{m} \{C_k \Delta_k\},$$

$$E_\Gamma = \bigcup_{k=1}^{m} \left\{ L_k^l \Gamma_k \colon l \in \{1,2,3\}\right\}, \qquad E_L = \bigcup_{k=1}^{m} \left\{ L_k^l \Lambda_k^l \colon l \in \{1,2,3\}\right\}.$$

For each $j \in J$ we denote the ordering of its neighborhood σ_j as an ordered tuple. The i^{th} entry of the tuple corresponds to $\sigma_j(i)$. For each $j \in J$ we set

$$\sigma_j = \begin{cases} (X_k, \overline{X}_k), & \text{if } j = \Phi_k \in J_X, \\ (\overline{X}_k, X_k), & \text{if } j = \overline{\Phi}_k \in J_X, \\ (L_k^l, C_k), & \text{if } j = \Lambda_k^l \in J_L, \\ (L_k^1, L_k^2, L_k^3), & \text{if } j = \Gamma_k \in J_C, \\ (C_k), & \text{if } j = \Delta_k \in J_D, \end{cases}$$

$$\xi_j^1 = \begin{cases} 2, & \text{if } j \in J_D, \\ 1, & \text{if } j \in J_C \cup \{\Phi_k \colon k \in \{1, \dots, n\}\}, \\ 0, & \text{else,} \end{cases}$$

and

$$\xi_j^2 = \begin{cases} 1, & \text{if } j \in J_L \cup \left\{\overline{\Phi}_k \colon k \in \{1, \dots, n\}\right\} \\ 0, & \text{else.} \end{cases}$$

Finally we set

$$B = 3m + n = \frac{1}{2} \sum_{j \in J} \xi_j^1 + \xi_j^2 = \sum_{j \in J} \xi_j^1 = \sum_{j \in J} \xi_j^2.$$

For the construction of the instance, also see Figure 8.3.

Assume the given 1-in-3-SAT is satisfiable. We define a set of locations I', the set of locations we want to open, by setting

$$I' = I_C \cup \left\{ L_k \in \left\{ X_k, \overline{X}_k \right\} : k \in \{1, \ldots, n\},\ L_k = \textsc{true} \right\}.$$

By the construction of the instance it is immediately seen that for each $j \in J$ there exists some $i \in N(j) \cap I'$. Recall that $N'(i)$ denotes the active neighborhood of $i \in I$ induced by the set I', cf. Definition8.3. If we can show that, for each location $i \in I'$, the equality $\xi^1(N'(i)) = \xi^2(N'(i))$ holds, we are done, as in this case we can find a feasible solution with at most B suppliers.

So let $i \in I'$ be given. We begin with the case that $i \in I_X$, say $i = L_k$ for some $k \in \{1, \ldots, n\}$. As $i \in I'$ the literal L_k is set to TRUE and thereby $(N(L_k) \cap J_L) \subseteq N'(i)$. In each clause C_k in which L_k appears, all other literals are set to FALSE. Thus, $(N(L_k) \cap J_C) \subseteq N'(i)$. Finally we have $\overline{L}_k = \textsc{false}$ and therefore also $(N(L_k) \cap J_X) \subseteq N'(i)$. We can now easily calculate $\xi^1(N'(i)) = \xi^2(N'(i))$. If on the other hand $i \in I_C$, the desired equation follows directly from the fact that exactly one literal in each clause is satisfied and therefore $|N'(i) \cap J_L| = 2$

Now assume on the other hand, that we are given a solution $\hat{x}$ to Robust 1-oC fulfilling $\hat{x}(I) \leq B$. We define a subset I' of the regions by setting $I' = \{i \in I : \hat{x}_i \geq 1\}$. Suppose there is a location $i \in I$ with $\xi^1(N'(i)) \neq \xi^2(N'(i))$, where again $N'(i)$ denotes the active neighborhood of i with respect to I'. Then $\hat{x}(I) > B$, which is a contradiction. Thus, we may assume that in both scenarios, the same amount of clients are served by any fixed location $i \in I'$. As $\xi^1(N'(C_k) \cap J_D) = 2$ for all $k \in \{1, \ldots, m\}$, we must also have $\xi^2(N'(C_k) \cap J_L) = 2$. Thus, for each clause, exactly one of its literals is contained in I'.

Let $k \in \{1 \ldots, n\}$. By the previous arguments it is $\xi^1(N'(X_k) \setminus J_X) = \xi^2(N'(X_k) \setminus J_X)$. Thus, to get equality for the complete active neighborhoods, it must hold true that $\xi^1(N'(X_k) \cap J_X) = \xi^2(N'(X_k) \cap J_X)$. If $x_i \geq 1$ for both $i = X_k$ and $i = \overline{X}_k$, this is not the case. This means setting $L_k = \textsc{true}$ if and only if $L_k \in I'$ is a valid truth assignment to the variables. Further, as pointed out above, in each clause there is exactly one literal contained in I' and therefore set to TRUE, completing the proof. □

8.3.2. Interval Uncertainty

Recall that in interval uncertainty all components of the scenarios have lower and upper bounds. For Robust q-fC and Robust q-oC this means that the uncertainty sets are of the form $\mathcal{U} = \left\{ \xi \in \mathbb{N}^{|J|} : a_j \leq \xi_j \leq b_j,\ \forall j \in J \right\}$ for vectors $a, b \in \mathbb{N}^{|J|}$.

Loosely speaking, we give upper and lower bounds on the possible number of clients in each region. Intuitively, the unique worst case scenario is the one where in each region $j \in J$ there are b_j clients. We see in the following that the intuition is correct in this case and the problems reveal to be the same as their non-robust version with demand vector b.

For Robust q-fC we observe that for each $i \in I$ it is

$$\max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in N(i)} \xi_j \right\} = \sum_{j \in N(i)} b_j.$$

Thus, by Theorem 8.10, we get the following result.

Theorem 8.21. *Let an instance of Robust Min-q-freeClient with interval uncertainty be given. Then $\hat{x} \in \mathbb{N}^{|I|}$, defined by setting*

$$\hat{x}_i = \left\lceil \frac{\sum_{j \in N(i)} b_j}{q} \right\rceil$$

for all $i \in I$, is an optimal solution to the given instance. In particular, Robust q-freeClient with interval uncertainty is in P. □

For Robust q-oC we observe that for any $\hat{z} \in \{0,1\}^{|I| \times |J|}$ we have

$$\max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in N(i)} \xi_j \cdot \hat{z}_{ij} \right\} = \sum_{j \in N(i)} b_j \cdot \hat{z}_{ij}.$$

Thus, (IP 8.3)(b) is a valid formulation for Robust Min-q-oC, cf. Equation 8.7 and Robust q-oC with interval uncertainty is essentially the same problem as q-oC.

8.3.3. Budgeted Uncertainty

Recall that in addition to the bounds given in interval uncertainty, here we are given a bound on the sum of demands in all regions. Thus, the regarded uncertainty sets for Robust q-fC and Robust q-oC are of the form $\mathcal{U} = \left\{ \xi \in \mathbb{N}^{|J|} \colon a_j \le \xi_j \le b_j,\ \xi(J) \le \Gamma \right\}$ for vectors $a, b \in \mathbb{N}^{|J|}$ and $\Gamma \in \mathbb{N}$.

Again, we begin by analyzing Robust q-fC. By Theorem 8.10 we need to compute the value $\max_{\xi \in \mathcal{U}} \{\xi(N(j))\}$ for each $i \in I$. By Lemma 7.33, this can be computed by

$$\max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in N(i)} \xi_j \right\} = \min \left\{ \sum_{j \in N(i)} b_j, \Gamma - \sum_{j \in J \setminus N(i)} a_j \right\}.$$

We get the following result immediately.

Theorem 8.22. *Let an instance of Robust Min-q-freeClient with budgeted uncertainty be given. Then $\hat{x} \in \mathbb{N}^{|I|}$, defined by setting*

$$\hat{x}_i = \left\lceil \frac{\min \left\{ b\left(N(i)\right), \Gamma - a\left(J \setminus N(i)\right) \right\}}{q} \right\rceil$$

for all $i \in I$, is an optimal solution to the given instance. In particular, Robust *q-*freeClient *with budgeted uncertainty is in* P. □

Let us turn to Robust q-oC with budgeted uncertainty. The problem turns out to be NP-hard, which perhaps is not surprising, as the reduction from Theorem 8.14 also works for budgeted uncertainty.

Theorem 8.23. *Robust q-orderedClient with budgeted uncertainty is* NP*-complete in the strong sense for any $q \in \mathbb{N}_{>0}$.*

Proof. As uncertainty sets from budgeted uncertainty are polyhedral and their underlying polyhedra are integral, we get that Robust q-oC is contained in NP by Corollary 8.17.

Regarding the proof of Theorem 8.14, we observe that the constructed uncertainty set $\mathcal{U} = \left\{\xi \in \mathbb{N}^{|J|} : \xi(J) = 1\right\}$ can be modeled as a budgeted uncertainty set by setting $\Gamma = 1$ and $a_j = 0$ and $b_j = 1$ for all $j \in J$. Thus, the reduction used in Theorem 8.14 also works for budgeted uncertainty and we get the desired result. □

In the proof of Theorem 8.23 we already observed that the regarded uncertainty sets are polyhedral and their underlying polyhedra are integral. Thus, by Theorem 8.16, we conclude that the following is a valid mixed integer programming formulation for Robust Min-q-oC with budgeted uncertainty.

$$\text{(MIP 8.10)} \quad \begin{array}{lrll} \min\limits_{x,\nu,z,\eta,\pi} & \sum\limits_{i\in I} x_i & & \\ \text{s.t.} & \pi_i\Gamma + \sum\limits_{j\in J}\eta^1_{ij}b_j - \sum\limits_{j\in J}\eta^2_{ij}a_j \le q\cdot x_i & \forall i \in I \\ & \pi_i + \eta^1_{ij} - \eta^2_{ij} \ge z_{ij} & \forall i \in I,\ j\in J \\ & \sum\limits_{i\in N(j)} v_i \ge 1 & \forall j \in J \\ & v_i - \sum\limits_{k=1}^{\sigma_j^{-1}(i)-1} v_{\sigma_j(k)} \le z_{ij} & \forall j \in J,\ i \in N(j) \\ & v_i, z_{ij} \in \{0,1\} & \forall i \in I, j \in J \\ & x_i \in \mathbb{Z}_{\ge 0} & \forall i \in I \\ & \pi_i \in \mathbb{R}_{\ge 0} & \forall i \in I \\ & \eta^1_{ij}, \eta^2_{ij} \in \mathbb{R}_{\ge 0} & \forall i \in I,\ j \in J. \end{array}$$

Theorem 8.24. *Let an instance of Robust Min-q-orderedClient with budgeted uncertainty be given. Then $x^\star \in \mathbb{N}^{|I|}$ is an optimal solution to the instance if and only if it can be extended to an optimal solution of (MIP 8.10).* □

8.3.4. Subset Budgeted Uncertainty

We now turn to subset budgeted uncertainty. In the previous section we bounded the total number of clients in all regions. As described in the last chapter, for some applications this approach might still be too conservative. To tackle this problem, in subset budgeted uncertainty, we give additional bounds on a collection of subsets of the regions. This means that the uncertainty sets in this section are of the form

$$\mathcal{U} = \left\{ \xi \in \mathbb{N}^{|J|} : \alpha_{S'} \leq \sum_{j \in S'} \xi_j \leq \beta_{S'}, \ \forall S' \in \mathcal{S} \right\}$$

for a collection of subsets of the regions $\mathcal{S} \subseteq 2^J$ and integers $\alpha_{S'}, \beta_{S'} \in \mathbb{N}$ for each $S' \in \mathcal{S}$.

Theorem 7.39 states that MaxSum is NP-complete in the strong sense for subset budgeted uncertainty. As explained in the previous chapter, subset budgeted uncertainty generalizes budgeted uncertainty. Thus, we get the following complexity results.

Theorem 8.25. *Robust q-freeClient with subset budgeted uncertainty is* NP*-hard in the strong sense for any fixed $q \in \mathbb{N}_{>0}$.*

For the solution process of Robust Min-*q*-fC with subset budgeted uncertainty we may still use Theorem 8.10. We note that

$$\max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in N(i)} \xi_j \right\}$$

is a valid integer programming formulation, which we may use in the solution process.

Theorem 8.26. *Robust q-orderedClient with subset budgeted uncertainty is* NP*-hard in the strong sense for any fixed $q \in \mathbb{N}_{>0}$.*

Further, as $\mathcal{U}$ is polyhedral, but its underlying polyhedron is not integral we may not use the reformulation (MIP 8.9) for Robust Min-*q*-oC and separation for (IP 8.6) is hard, as MaxSum is NP-hard. Nevertheless, for fixed $\hat{z} \in \{0,1\}^{|I| \times |J|}$ and $\hat{x} \in \mathbb{Z}_{\geq 0}^{|I|}$ the maximum

$$\max_{\xi \in \mathcal{U}} \left\{ \sum_{j \in N(i)} \xi_j \hat{z}_{ij} \right\}$$

is an integer programming formulation, which we can use to find violating constraints of the form

$$\sum_{j \in N(i)} \xi_j \hat{z}_{ij} > q \cdot \hat{x}_i$$

for (IP 8.6).

$\mathcal{U}$		Robust q-orderedClient	Robust q-freeClient
General		NP-hard	NP-hard
Discrete	$q = \lvert\mathcal{U}\rvert = 1$	P	P
	$\max\{q, \lvert\mathcal{U}\rvert\} \geq 2$	NP-complete	P
Interval	$q \leq 1$	P	P
	$q \geq 2$	NP-complete	P
Budgeted		NP-complete	P
Subset budgeted		NP-hard	NP-hard

Table 8.1.: Summarized complexity results for Robust q-fC and Robust q-oC.

Conclusion We conclude this chapter by giving a summary of the proved complexity results on q-freeClient and q-orderedClient. For an overview of all complexity results see Table 8.1. Note that the results for the non-robust versions are also included in the table, as these problems correspond to discrete uncertainty with only one scenario. The most surprising result from the table is that 2-oC and Robust 1-oC restricted to discrete uncertainty sets containing only 2 elements are NP-complete. Regarding complexity, this is the only difference between q-orderedClient and q-Multiset Multicover. This observation confirms the intuition that q-orderedClient is the harder problem to solve. Regarding q-freeClient, we get polynomial time solvability in almost all cases. The solutions to Min-q-freeClient may be too conservative for practical applications. For further research, it might be interesting to consider a variant of q-freeClient in which the clients are allowed to choose freely, but may only choose locations which are actually opened. This intuitively decreases conservatism of the problem and increases the complexity.

Another research direction could be to regard approximation procedures for the regarded problems. As we only proved Robust q-oC to be NP-hard but not to be contained in NP, it could be interesting to verify that the problem is contained in NP or even Σ_2^P-complete.

Chapter 9.

Conclusion

The most significant contribution of this thesis is the proof of the Beineke Harary Conjecture for $l = 2$. The techniques used for the proof are novel and focus on moving along a single one of the searched paths. The statement has previously not been proved for $l = 2$ and any $k \geq 2$. Thus, with our results we provided a further step towards proving the conjecture in general. Another key finding of this thesis is the classification of Eulerian graphs that decompose into a unique number of cycles. It is exactly the class of Eulerian graphs in which no two edge-disjoint cycles share more than three vertices. The result is a stand-alone result and illustrates very nicely the practicability of 2.5-connected components regarding cycle decomposition problems. Regarding further research it is an interesting task to classify graphs whose maximum and minimum cycle decomposition only vary by a constant.

The canonical decomposition into 2.5-connected components is a powerful tool as we illustrated in Chapter 5. A direction for further research could be to extend the definition of a supporting split to a split whose corresponding virtual edge is contained in a 3-edge separator and see if results transfer. This, however, is far from immediate which underlines the fact that the results on the uniqueness of the 2.5-connected components are interesting and surprising.

In Chapter 6 we saw an example for a well known optimization problem, Dominating Set, extended by structural uncertainty. We proved that on 2-connected graphs, a set is dominating in each spanning tree of a graph if and only if it is a vertex cover in the same graph. Nevertheless we proved that Simultaneous Domination of Spanning Trees remains NP-hard on perfect graphs, whereas Vertex Cover is polynomial time solvable on the same class. This illustrates that the problems are not computationally equivalent. Simultaneous domination is a very broad topic and has not been considered for many interesting classes of graphs.

We also examined the computational complexity of robust approaches to three covering problems with uncertainty in the demand. In all cases the problems are NP-hard in the strong sense for arbitrary uncertainty sets. In particular this is true for q-freeClient, which is almost trivial in a non-robust setting. We classified the complexity of the problems restricted to various uncertainty sets and discussed solution approaches based on constraint generation and integer programming. All considered problems were motivated by applications in the health sector. As this thesis' focus is on the theoretical aspects of the problems we did not describe the practical applications. A first step in this direction has been taken in [KSS19] and [Büs+20]. More research connecting the three regarded covering problems to practical applications is currently underway in the research projects *ONE PLAN* and *HealthFaCT*.

Decision Problem Index

1. Decision Problems

1-IN-3-SAT.

Instance: A boolean formula in conjunctive normalform containing three literals in each clause.

Question: Is there a truth assignment to the variables, such that exactly one literal in each clause is assigned the value TRUE.

Complexity: NP-complete [GJ79]. Remains NP-complete, when restricted to instances where each variable and each negation of each variable appears an even amount of times in each clause.

CO-q-MULTISET MULTICOVER-SEP (CO-q-MSMC-SEP).

Instance: Finite, disjoint sets I, J, a finite set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \cup J, E)$, and integers $x_i \in \mathbb{N}$ for $i \in I$.

Question: Are there $y(\xi) \in \mathbb{N}^{|I|\times|J|}$ for every $\xi \in \mathcal{U}$ satisfying

$$\sum_{i\in N(j)} y(\xi)_{ij} \geq \xi_j \qquad \forall j \in J, \xi \in \mathcal{U} \quad \text{and}$$
$$\sum_{j\in N(i)} y(\xi)_{ij} \leq q \cdot x_i \qquad \forall i \in I, \xi \in \mathcal{U}?$$

Complexity: coNP-complete in the strong sense for any fixed $q \in \mathbb{N}_{>0}$ (Corollary 7.16). Contained in P if cardinality of $\mathcal{U}$ is polynomial in the encoding size of the instance (Theorem 7.13).

CONNECTIVITY PAIR (CP).

Instance: An undirected graph G, vertices $s, t \in V(G)$, and $k, B \in \mathbb{N}$.

Question: Does $l \in \mathbb{N}$ with $l \leq B$ exist such that (k, l) is a connectivity pair for s and t in G?

Complexity: NP-complete (Theorem 3.18).

Dominating Set.
Instance: An undirected simple graph G and an integer $B \in \mathbb{N}$.
Question: Is there a subset $S \subseteq V(G)$ with $|S| \leq B$ such that for all $v \in V(G)$ it is $v \in S$ or $u \in S$ for some $u \in N(v)$?
Complexity: NP-complete [GJ79].

Edge Cover.
Instance: An undirected graph G and an integer $B \in \mathbb{N}$.
Question: Is there a subset of the edges $E' \subseteq E(G)$, such that each vertex in $V(G)$ is incident to an edge in E' and $|E'| \leq B$?
Complexity: Contained in P [Sch03].

Exact 3-Dimensional Matching.
Instance: Finite, disjoint sets A, B, C with $|A| = |B| = |C| = k$ for some $k \in \mathbb{N}$ and a subset $M \subseteq A \times B \times C$.
Question: Does there exist a subset $M' \subseteq M$ with $|M'| = k$, such that for each two different elements $(a, b, c), (a', b', c') \in M'$, we have $a \neq a'$, $b \neq b'$ and $c \neq c'$?
Complexity: NP-complete [GJ79].

Exact Cover by 3-sets.
Instance: An integer $n \in \mathbb{N}$, a finite set U with $|U| = 3n$, and a collection of subsets $\mathcal{S} \subseteq 2^U$, such that $|S| = 3$ for all $S \in \mathcal{S}$.
Question: Is there a subset $\mathcal{S}' \subseteq \mathcal{S}$, such that $|\mathcal{S}'| = n$ and $\bigcup_{S \in \mathcal{S}'} S = U$?
Complexity: NP-complete [GJ79].

Independent Set.
Instance: An undirected simple graph G and an integer $B \in \mathbb{N}$.
Question: Is there a subset $S \subseteq V(G)$ of the vertices of G, with $|S| \geq B$, such that there is no edge in G with both endvertices in S?
Complexity: NP-complete [GJ79].

Knapsack.
Instance: A finite set U, for each $u \in U$ integer values $s_u, p_u \in \mathbb{N}$, and two more integers $B, K \in \mathbb{N}$.
Question: Does there exist a subset $U' \subseteq U$ such that

$$\sum_{u \in U'} s_u \leq B \text{ and } \sum_{u \in U'} p_u \geq K?$$

Complexity: NP-complete [GJ79].

Large Cycle Decomposition (LCD).
Instance: An even graph G and an integer $B \in \mathbb{N}$.
Question: In G, does there exist a cycle decomposition with at least B cycles?
Complexity: NP-complete [Hei20].

MaxSum.
Instance: A set $\mathcal{U} \subseteq \mathbb{N}^n$ for some positive integer n and an integer $B \in \mathbb{N}$.
Question: Does there exist a $\xi \in \mathcal{U}$ such that

$$\sum_{j=1}^{n} \xi_j \geq B?$$

Complexity: NP-complete in the strong sense (Theorem 7.18).

Multiset Multicover.
Instance: A finite set U, integers $d_u \in \mathbb{N}$ for all $u \in U$, a collection of multisets $\mathcal{S}$ over U, and an integer $B \in \mathbb{N}$.
Question: Does there exist a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ with $|S|' \leq B$ such that for each $u \in U$ we have $\sum_{S \in \mathcal{S}'} m_S(u) \geq d_u$, where $m_S(u)$ denotes the multiplicity of u in S?
Complexity: NP-complete in the strong sense as a generalization of Set Cover.

Partial Vertex Cover.
Instance: An undirected graph G and integers $q, B \in \mathbb{N}$
Question: Does there exist a partial vertex cover C in G with respect to q such that $|S| \leq B$?
Complexity: NP-complete, even when restricted to instances with bipartite graphs [CS13].

q-freeClient (q-fC).
Instance: Finite, disjoint sets I, J, weights $d_j \in \mathbb{N}$ for all $j \in J$, a bipartite graph $G = (I \cup J, E)$, and an integer $B \in \mathbb{N}$.
Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \leq B$ such that for all $y \in \mathbb{N}^{|I|\times|J|}$ with $\sum_{i\in N(j)} y_{ij} = d_j$ for all $j \in J$, it is

$$\sum_{j\in N(i)} y_{ij} \leq q \cdot x_i \quad \forall i \in I?$$

Complexity: Can be decided in time $\mathcal{O}(|I| + |J| + |E(G)|)$ (Theorem 8.1).

q-Multiset Multicover (q-MsMc).
Instance: Finite, disjoint sets I, J, weights $d_j \in \mathbb{N}$ for all $j \in J$, a bipartite graph $G = (I \cup J, E)$, and an integer $B \in \mathbb{N}$.
Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \leq B$ such that there exists $y \in \mathbb{N}^{|I|\times|J|}$ satisfying

$$\sum_{i\in N(j)} y_{ij} \geq d_j \quad \forall j \in J \quad \text{and} \quad \sum_{j\in N(i)} y_{ij} \leq q \cdot x_i \quad \forall i \in I?$$

Complexity: NP-complete in the strong sense for any fixed $q \in \mathbb{N}_{\geq 3}$ (Theorem 7.8). Can be decided in $\mathcal{O}(|I| + |J|)$ for $q = 1$ (Proposition 7.6). Contained in P for $q = 2$ (Theorem 7.7).

q-Multiset Multicover-Sep (q-MsMc-Sep).
Instance: Finite, disjoint sets I, J, a finite set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \cup J, E)$, and integers $x_i \in \mathbb{N}$ for $i \in I$.
Question: Does there exist a $\xi \in \mathcal{U}$, such that for all $y(\xi) \in \mathbb{N}^{|I|\times|J|}$ with

$$\sum_{i\in N(j)} y(\xi)_{ij} \geq \xi_j \quad \forall j \in J,$$

there exists an $i \in I$ with

$$\sum_{j\in N(i)} y(\xi)_{ij} > q \cdot x_i?$$

Complexity: NP-complete in the strong sense for any fixed $q \in \mathbb{N}_{>0}$ (Theorem 7.15). Contained in P if cardinality of $\mathcal{U}$ is polynomial in the encoding size of the instance (Theorem 7.13).

q-orderedClient (q-oC).

Instance: Finite, disjoint sets I, J, weights $d_j \in \mathbb{N}$ for all $j \in J$, a bipartite graph $G = (I \cup J, E)$, for each $j \in J$ an ordering $\sigma_j \colon \{1, \dots, |N(j)|\} \to N(j)$, and an integer $B \in \mathbb{N}$.

Question: Does there exist $x \in \mathbb{N}^{|I|}$ for $i \in I$ with $x(I) \leq B$ such that for each $j \in J$ there exists an $i \in N(j)$ with $x_i \geq 1$ and

$$\sum_{j \in N(i)} y_{ij} \leq q \cdot x_i \quad \forall i \in I,$$

where $y \in \mathbb{N}^{|I| \times |J|}$ is defined by setting

$$y_{ij} = \begin{cases} d_j, & \text{if } i = \operatorname{argmin}_{i' \in N(j)} \left\{ \sigma_j^{-1}(i') \colon x_{i'} \geq 1 \right\}, \\ 0, & \text{else,} \end{cases}$$

for $i \in I$, $j \in J$?

Complexity: Can be decided in time $O(|I| + |J|)$ for $q = 1$, Theorem 8.2. NP-complete in the strong sense for any fixed $q \in \mathbb{N}$ with $q \geq 2$. (Theorems 8.7, 8.8).

Set Cover.

Instance: A finite set U, a collection of subsets $\mathcal{S} \subseteq 2^U$, and an integer $B \in \mathbb{N}$.

Question: Does there exist a subset $\mathcal{S}' \subset \mathcal{S}$ such that $|\mathcal{S}'| \leq B$ and $\bigcup_{S \in \mathcal{S}'} S = U$?

Complexity: NP-complete [GJ79].

Simultaneous Domination of Spanning Trees (SDST).

Instance: A graph G and an integer $B \in \mathbb{N}$.

Question: Does there exist a subset $S \subseteq V(G)$ of the vertices of G with $|S| \leq B$ such that S is a dominating set in each spanning tree of G?

Complexity: NP-complete (Theorem 6.4). Remains NP-complete when restricted to perfect graphs (Theorem 6.19). On 2-connected graphs equivalent to Vertex Cover (Corollary 6.3). Polynomial time solvable when restricted to bipartite graphs, graphs of bounded treewidth and chordal graphs (Corollary 6.12). Polynomial time solvable when restricted to claw free graphs (Theorem 6.18).

Small Cycle Decomposition (SCD).

Instance: An even graph G and an integer $B \in \mathbb{N}$.

Question: In G, does there exist a cycle decomposition with at most B cycles?

Complexity: NP-complete [Hei20].

Vertex Cover.

Instance: A graph G and an integer $B \in \mathbb{N}$.

Question: Does there exist a subset $S \subseteq V(G)$ with $|S| \leq B$, such that each edge is incident to at least one vertex in S?

Complexity: NP-complete [GJ79]. Remains NP-complete on 2-connected graphs. Polynomial time solvable when restricted to bipartite graphs, graphs of bounded treewidth, claw free graphs and perfect graphs [Sch03].

2. Robust Decision Problems

Robust q-freeClient (Robust q-fC).

Instance: Finite, disjoint sets I, J, an uncertainty set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \mathbin{\dot\cup} J, E)$, and an integer $B \in \mathbb{N}$.

Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \leq B$ such that for all $y \in \mathbb{N}^{|I|\times|J|}$ with $\sum_{i \in N(j)} y_{ij} = \xi_j$ for some $\xi \in \mathcal{U}$ and all $j \in J$, it is

$$\sum_{j \in N(i)} y_{ij} \leq q \cdot x_i \quad \forall i \in I?$$

Complexity: NP-hard in the strong sense for any fixed $q \in \mathbb{N}_{>0}$ (Theorem 8.11). Contained in P if cardinality of $\mathcal{U}$ is polynomial in the encoding size of the instance (Theorem 8.10).

Robust q-Multiset Multicover (Robust q-MsMc).

Instance: Finite, disjoint sets I, J, an uncertainty set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \mathbin{\dot\cup} J, E)$, and a positive integer $B \in \mathbb{N}_{>0}$.

Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \leq B$ such that for each $\xi \in \mathcal{U}$ there exists $y(\xi) \in \mathbb{N}^{|I|\times|J|}$ satisfying

$$\begin{aligned} \sum_{i \in N(j)} y(\xi)_{ij} &\geq \xi_j && \forall j \in J, \xi \in \mathcal{U} \quad \text{and} \\ \sum_{j \in N(i)} y(\xi)_{ij} &\leq q \cdot x_i && \forall i \in I, \xi \in \mathcal{U}? \end{aligned}$$

Complexity: NP-hard in the strong sense for any fixed $q \in \mathbb{N}_{>0}$ (Theorem 7.12).

ROBUST q-ORDEREDCLIENT (ROBUST q-OC).

Instance: Finite, disjoint sets I, J, an uncertainty set $\mathcal{U} \subseteq \mathbb{N}^{|J|}$, a bipartite graph $G = (I \cup J, E)$, for each $j \in J$ an ordering $\sigma_j \colon \{1, \ldots, |N(j)|\} \to N(j)$, and an integer $B \in \mathbb{N}$.

Question: Does there exist $x \in \mathbb{N}^{|I|}$ with $x(I) \leq B$ such that for each $j \in J$ there exists an $i \in N(j)$ with $x_i \geq 1$ and

$$\sum_{j \in N(i)} y(\xi)_{ij} \leq q \cdot x_i \quad \forall i \in I, \xi \in \mathcal{U},$$

where $y(\xi) \in \mathbb{N}^{|I| \times |J|}$ is defined by setting

$$y(\xi)_{ij} = \begin{cases} \xi_j, & \text{if } i = \operatorname{argmin}_{i' \in N(j)} \left\{ \sigma_j^{-1}(i') \colon x_{i'} \geq 1 \right\}, \\ 0, & \text{else,} \end{cases}$$

for $i \in I$, $j \in J$?

Complexity: NP-hard in the strong sense for any fixed $q \in \mathbb{N}_{>0}$ (Theorem 8.14). Contained in NP when restricted to polyhedral uncertainty sets whose underlying polyhedra are integral (Corollary 8.17).

Bibliography

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows*. Prentice Hall, 1993. DOI: 10.5555/137406.

[AP89] Stefan Arnborg and Andrzej Proskurowski. "Linear Time Algorithms for NP-hard Problems Restricted to Partial k-trees". In: *Discrete Applied Mathematics* 23.1 (1989), pp. 11–24. DOI: 10.1016/0166-218X(89)90031-0.

[Aus+12] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*. Springer, 2012. DOI: 10.1007/978-3-642-58412-1.

[BC17] Robert C. Brigham and Julie R. Carrington. "Global domination". In: *Domination in Graphs*. Routledge, 2017, pp. 301–320.

[BD90] Robert C. Brigham and Ronald D. Dutton. "Factor Domination in Graphs". In: *Discrete Mathematics* 86.1 (1990), pp. 127–136. DOI: 10.1016/0012-365X(90)90355-L.

[BGN09] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Vol. 28. Princeton University Press, 2009. DOI: 10.1515/9781400831050.

[BH67] Lowell W. Beineke and Frank Harary. "The Connectivity Function of a Graph". In: *Mathematika* 14.2 (1967), pp. 197–202. DOI: 10.1112/S0025579300003806.

[BL11] John R. Birge and François Louveaux. *Introduction to Stochastic Optimization*. Springer, 2011. DOI: 10.1007/978-1-4614-0237-4.

[BN00] Aharon Ben-Tal and Arkadi Nemirovski. "Robust Solutions of Uncertain Linear Programming Problems Contaminated with Uncertain Data". In: *Mathematical Programming* 88 (2000), pp. 411–421. DOI: 10.1007/PL00011380.

[BN98] Aharon Ben-Tal and Arkadi Nemirovski. "Robust Convex Optimization". In: *Mathematics of Operations Research* 23 (1998), pp. 769–805. DOI: 10.1287/moor.23.4.769.

[BN99] Aharon Ben-Tal and Arkadi Nemirovski. "Robust Solutions of Uncertain Linear Programs". In: *Operations Research Letters* 25.1 (1999), pp. 1–13. DOI: 10.1016/s0167-6377(99)00016-4.

[Bod93] Hans L. Bodlaender. "A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth". In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*. ACM, 1993, pp. 226–234. DOI: 10.1145/167088.167161.

[Bod98] Hans L. Bodlaender. "A Partial k-arboretum of Graphs with Bounded Treewidth". In: *Theoretical Computer Science* 209.1 (1998), pp. 1–45. DOI: 10.1016/S0304-3975(97)00228-4.

[BS03] Dimitris Bertsimas and Melvyn Sim. "Robust Discrete Optimization and Network Flows". In: *Mathematical Programming* 98.1 (2003), pp. 49–71. DOI: 10.1007/s10107-003-0396-4.

[BS04] Dimitris Bertsimas and Melvyn Sim. "The Price of Robustness". In: *Operations Research* 52.1 (2004), pp. 35–53. DOI: 10.1287/opre.1030.0065.

[BSW70] J. Bruno, Kenneth Steiglitz, and L. Weinberg. "A New Planarity Test Based on 3-connectivity". In: *IEEE Transactions on Circuit Theory* 17.2 (1970), pp. 197–206. DOI: 10.1109/tct.1970.1083101.

[BT89] Giuseppe Di Battista and Roberto Tamassia. "Incremental Planarity Testing". In: *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 1989, pp. 436–441. DOI: 10.1109/SFCS.1989.63515.

[Büs+20] Christina Büsing, Martin Comis, Eva Schmidt, and Manuel Streicher. *Robust Strategic Planning for Mobile Medical Units with Steerable and Unsteerable Demands*. 2020. arXiv: 2005.11062 [math.OC].

[BWO12] Lowell W. Beineke, Robin J. Wilson, and Ortrut R. Oellermann. *Topics in Structural Graph Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012.

[CH14] Yair Caro and Michael A. Henning. "Simultaneous Domination in Graphs". In: *Graphs and Combinatorics* 30.6 (2014), pp. 1399–1416. DOI: 10.1007/s00373-013-1353-5.

[Chu+06] Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. "The Strong Perfect Graph Theorem". In: *Annals of Mathematics* 164 (2006), pp. 51–229. DOI: 10.4007/annals.2006.164.51.

[Coo78] Leon Cooper. "Bounds on the Weber Problem Solution under Conditions of Uncertainty". In: *Journal of Regional Science* 18.1 (1978), pp. 87–92. DOI: 10.1111/j.1467-9787.1978.tb00530.x.

[CS13] Bugra Caskurlu and Kirubakran Subramani. *On Partial Vertex Cover on Bipartite Graphs and Trees*. 2013. arXiv: 1304.5934 [cs.CC].

[Dan+06] Peter Dankelmann, Michael A. Henning, Wayne Goddard, and Renu C. Laskar. "Simultaneous Graph Parameters: Factor Domination and Factor Total Domination". In: *Discrete Mathematics* 306.18 (2006), pp. 2229–2233. DOI: 10.1016/j.disc.2006.04.017.

[Dha+05] Kedar Dhamdhere, Vineet Goyal, Ramamoorthi Ravi, and Mohit Singh. "How to pay, come what may: Approximation Algorithms for Demand-robust Covering Problems". In: *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. 2005, pp. 367–376. DOI: 10.1109/SFCS.2005.42.

[Die00] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2000. DOI: 10.1007/978-3-662-53622-3_12.

[DL03] Peter Dankelmann and Renu C. Laskar. "Factor Domination and Minimum Degree". In: *Discrete Mathematics* 262.1 (2003), pp. 113–119. DOI: 10.1016/S0012-365X(02)00522-8.

[Dob82] Gregory Dobson. "Worst-Case Analysis of Greedy Heuristics for Integer Programming with Nonnegative Data". In: *Mathematics of Operations Research* 7.4 (1982), pp. 515–531. DOI: 10.1287/moor.7.4.515.

[DR08] Jan Degenhardt and Peter Recht. “On a Relation between the Cycle Packing Number and the Cyclomatic Number of a Graph”. Manuscript. 2008.

[DT97] Dorit Dor and Michael Tarsi. “Graph Decomposition is NP-complete: A Complete Proof of Holyer’s Conjecture”. In: *SIAM Journal on Computing* 26.4 (1997), pp. 1166–1187. DOI: 10.1137/s0097539792229507.

[EK94] Hikoe Enomoto and Atsushi Kaneko. “The Condition of Beineke and Harary on Edge-disjoint Paths some of which are Openly Disjoint”. In: *Tokyo Journal of Mathematics* 17.2 (1994), pp. 355–357. DOI: 10.3836/tjm/1270127958.

[EKM91] Yoshimi Egawa, Atsushi Kaneko, and Makoto Matsumoto. “A Mixed Version of Menger’s Theorem”. In: *Combinatorica* 11 (1991), pp. 71–74. DOI: 10.1007/bf01375475.

[Far+12] Reza Z. Farahani, Nasrin Asgari, Nooshin Heidari, Mahtab Hosseininia, and Mark Goh. “Covering Problems in Facility Location: A Review”. In: *Computers and Industrial Engineering* 62.1 (2012), pp. 368–407. DOI: 10.1016/j.cie.2011.08.020.

[Fei98] Uriel Feige. “A Threshold of Ln n for Approximating Set Cover”. In: *Journal of the ACM* 45.4 (1998), pp. 634–652. DOI: 10.1145/285055.285059.

[Fle90] “Chapter IV Characterization Theorems and Corollaries”. In: *Eulerian Graphs and Related Topics*. Ed. by Herbert Fleischner. Vol. 45. Annals of Discrete Mathematics. Elsevier, 1990, pp. IV.1–IV.20. DOI: 10.1016/S0167-5060(08)70949-X.

[GH13] Wayne Goddard and Michael A. Henning. “Independent Domination in Graphs: A Survey and Recent Results”. In: *Discrete Mathematics* 313.7 (2013), pp. 839–854. DOI: 10.1016/j.disc.2012.11.031.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979. DOI: 10.1137/1024022.

[GK95] Andrew V. Goldberg and Alexander V. Karzanow. “Maximum Skew-symmetric Flows”. In: *Proceedings of the 3rd Annual European Symposium on Algorithms*. Vol. 979. Lecture Notes in Computer Science. 1995, pp. 155–170. DOI: 10.1007/3-540-60313-1_141.

[GL97] Laurent El Ghaoui and Hervé Lebret. “Robust Solutions to Least-Squares Problems with Uncertain Data”. In: *SIAM Journal on Matrix Analysis and Applications* 18.4 (1997), pp. 1035–1064. DOI: 10.1137/S0895479896298130.

[GLS88] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Vol. 2. Algorithms and Combinatorics. Springer, 1988. DOI: 10.1007/978-3-642-78240-4.

[GM01] Carsten Gutwenger and Petra Mutzel. “A Linear Time Implementation of SPQR-trees”. In: *Proceedings of the 8th International Symposium on Graph Drawing*. Springer, 2001, pp. 77–90. DOI: 10.1007/3-540-44541-2_8.

[GMT14] Virginie Gabrel, Cécile Murat, and Aurélie Thiele. “Recent Advances in Robust Optimization: An Overview”. In: *European Journal of Operational Research* 235.3 (2014), pp. 471–483. DOI: 10.1016/j.ejor.2013.09.036.

[GOL98] Laurent El Ghaoui, Francois Oustry, and Hervé Lebret. “Robust Solutions to Uncertain Semidefinite Programs”. In: *SIAM Journal on Optimization* 9.1 (1998), pp. 33–52. DOI: 10.1137/S1052623496305717.

[Gro16] Martin Grohe. *Quasi-4-Connected Components.* 2016. arXiv: 1602.04505 [cs.DM].

[Har+10] Jochen Harant, Dieter Rautenbach, Peter Recht, and Friedrich Regen. "Packing Edge-disjoint Cycles in Graphs and the Cyclomatic Number". In: *Discrete Mathematics* 310.9 (2010), pp. 1456–1462. DOI: 10.1016/j.disc.2009.07.017.

[Hei+20a] Irene Heinrich, Till Heller, Eva Schmidt, and Manuel Streicher. "2.5-Connectivity: Unique Components, Critical Graphs, and Applications". In: *Graph-Theoretic Concepts in Computer Science.* Lecture Notes in Computer Science. Springer, 2020. Chap. 28. DOI: 10.1007/978-3-030-60440-0.

[Hei+20b] Irene Heinrich, Till Heller, Eva Schmidt, and Manuel Streicher. *2.5-Connectivity: Unique Components, Critical Graphs, and Applications.* 2020. arXiv: 2003.01498 [math.CO].

[Hei20] Irene Heinrich. "On Graph Decomposition: Hajós' Conjecture, the Clustering Coefficient and Dominating Sets". PhD thesis. TU Kaiserslautern, 2020.

[Hen09] Michael A. Henning. "A Survey of Selected Recent Results on Total Domination in Graphs". In: *Discrete Mathematics* 309.1 (2009), pp. 32–63. DOI: 10.1016/j.disc.2007.12.044.

[HHS98] Teresa W. Haynes, Stephen Hedetniemi, and Peter Slater. *Fundamentals of Domination in Graphs.* CRC press, 1998. DOI: 10.1201/9781482246582.

[HK71] John Hopcroft and Richard M. Karp. "A N5/2 Algorithm for Maximum Matchings in Bipartite Graphs". In: *Proceedings of the 12th Annual Symposium on Switching and Automata Theory.* IEEE Computer Society, 1971, pp. 122–125. DOI: 10.1109/SWAT.1971.1.

[HP87] Pierre Hanjoul and Dominique Peeters. "A Facility Location Problem with Clients' Preference Orderings". In: *Regional Science and Urban Economics* 17.3 (1987), pp. 451–473. DOI: 10.1016/0166-0462(87)90011-1.

[HS19] Irene Heinrich and Manuel Streicher. "Cycle Decompositions and Constructive Characterizations". In: *Electronic Journal of Graph Theroy and Applications* 7.2 (2019), pp. 411–428. DOI: 10.5614/ejgta.2019.7.2.15.

[HT72] John E. Hopcroft and Robert E. Tarjan. *Finding the triconnected components of a graph.* Tech. rep. Dept. of Computer Science, Cornell University, 1972.

[HT73] John E. Hopcroft and Robert E. Tarjan. "Algorithm 447: Efficient Algorithms for Graph Manipulation". In: *Communications of the ACM* 16.6 (1973), pp. 372–378. DOI: 10.1145/362248.362272.

[JKS18] Sebastian S. Johann, Sven O. Krumke, and Manuel Streicher. *Simultaneously Dominating all Spanning Trees of a Graph.* 2018. arXiv: 1810.12887 [math.CO].

[JKS19] Sebastian S. Johann, Sven O. Krumke, and Manuel Streicher. *On the Mixed Connectivity Conjecture of Beineke and Harary.* 2019. arXiv: 1908.11621 [math.CO].

[Joh20] Sebastian Johann. Personal communication. 2020.

[KN09] Sven O. Krumke and Hartmut Noltemeier. *Graphentheoretische Konzepte und Algorithmen.* Springer, 2009. DOI: 10.1007/978-3-8348-9592-9.

[KSS19] Sven O. Krumke, Eva Schmidt, and Manuel Streicher. "Robust Multicovers with Budgeted Uncertainty". In: *European Journal of Operational Research* 274.3 (2019), pp. 845–857. DOI: 10.1016/j.ejor.2018.11.049.

[KZ16] Adam Kasperski and Pawel Zielinski. “Robust Discrete Optimization Under Discrete and Interval Uncertainty: A Survey”. In: *Robustness analysis in decision aiding, optimization, and analytics.* Vol. 241. Springer, 2016, pp. 113–143. DOI: 10.1007/978-3-319-33121-8_6.

[LP86] Lászlo Lovász and Michael D. Plummer. *Matching Theory.* Akadémiai Kiadó, Budapest, 1986. DOI: 10.1090/chel/367.

[Lut+17] Pascal Lutter, Dirk Degel, Christina Büsing, Arie M. C. A. Koster, and Brigitte Werners. “Improved Handling of Uncertainty and Robustness in Set Covering Problems”. In: *European Journal of Operational Research* 263.1 (2017), pp. 35–49. DOI: 10.1016/j.ejor.2017.04.044.

[Mad79] Wolfgang Mader. “Connectivity and Edge-connectivity in Finite Graphs”. In: *Proceedings of the 7th British Combinatorial Conference.* Vol. 38. 1979, pp. 66–95. DOI: 10.1017/cbo9780511662133.005.

[OD98] Susan H. Owen and Mark S. Daskin. “Strategic Facility Location: A Review”. In: *European Journal of Operational Research* 111.3 (1998), pp. 423–447. DOI: 10.1016/S0377-2217(98)00186-6.

[OR19] Christin Otto and Peter Recht. “Maximum Cycle Packing Using SPR-trees.” In: *Electronic Journal of Graph Theory and Applications* 7.1 (2019), pp. 147–155. DOI: 10.5614/ejgta.2019.7.1.11.

[PA13] Jordi Pereira and Igor Averbakh. “The Robust Set Covering Problem with Interval Data”. In: *Annals of Operations Research* 207 (2013), pp. 1–19. DOI: 10.1007/s10479-011-0876-5.

[Pér84] Bernard Péroche. “NP-completeness of Some Problems of Partitioning and Covering in Graphs”. In: *Discrete Applied Mathematics* 8.2 (1984), pp. 195–208. DOI: 10.1016/0166-218X(84)90101-X.

[Sch03] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer, 2003.

[SF19] Elham Sadeghi and Neng Fan. “On the Survivable Network Design Problem with Mixed Connectivity Requirements”. In: *Annals of Operations Research* (2019). DOI: 10.1007/s10479-019-03175-5.

[Sny06] Lawrence V. Snyder. “Facility Location under Uncertainty: A Review”. In: *IIE Transactions* 38.7 (2006), pp. 547–564. DOI: 10.1080/07408170500216480.

[Soy73] Allen L. Soyster. “Convex Programming with Set-inclusive Constraints and Applications to Inexact Linear Programming”. In: *Operations Research* 21 (1973), pp. 1154–1157. DOI: 10.1287/opre.21.5.1154.

[Wes01] Douglas B. West. *Introduction to Graph Theory.* Vol. 2. Prentice-Hall, 2001.

[YY98] Gang Yu and Jian Yang. “On the Robust Shortest Path Problem”. In: *Computers and Operations Research* 25.6 (1998), pp. 457–468. DOI: 10.1016/S0305-0548(97)00085-3.

Index

This index serves as a defintion index only. The page numbers of the corresponding entries refer to the page on which the term is defined.

Curriculum Vitae

Manuel Streicher

since 01/2017	**Doctoral Studies in Mathematics**, TUK
November 4, 2016	**Master of Science in Mathematics**, TUK
since 10/2015	**Teaching and Research Assistant**, TUK
10/2014 – 02/2015	**Master Studies in Mathematics**, University of Southern Denmark, Odense
04/2014 – 11/2016	**Master Studies in Mathematics**, TUK
July 11, 2014	**Bachelor of Science in Mathematics**, TUK
09/2013 – 08/2014	**Research Assistant**, TUK
11/2012 – 08/2014	**Teaching Assistant**, TUK
10/2010 – 07/2014	**Bachelor Studies in Mathematics**, TUK
June 25, 2009	**Abitur** (high school graduation), Goetheschule Wetzlar
07/2007 – 07/2009	**High School**, Goetheschule Wetzlar
07/2006 – 06/2007	**High School**, Pleasure Ridge Park High School, USA

Wissenschaftlicher Werdegang

Manuel Streicher

seit 01/2017	**Promotionsstudium in Mathematik**, TUK
4. November 2016	**Master of Science in Mathematik**, TUK
seit 10/2015	**Lehr- und Forschungsassistent**, TUK
10/2014 – 02/2015	**Masterstudium in Mathematik**, University of Southern Denmark, Odense
04/2014 – 11/2016	**Masterstudium in Mathematik**, TUK
11. Juli 2014	**Bachelor of Science in Mathematik**, TUK
09/2013 – 08/2014	**Forschungsassistent**, TUK
11/2012 – 08/2014	**Lehrassistent**, TUK
10/2010 – 07/2014	**Bachelorstudium in Mathematik**, TUK
25. Juni 2009	**Abitur**, Goetheschule Wetzlar
07/2007 – 07/2009	**Gymnasium**, Goetheschule Wetzlar
07/2006 – 06/2007	**High School**, Pleasure Ridge Park High School, USA