Graph Modification Problems: Beyond the Known Boundaries

Akanksha Agrawal

Thesis for the degree of philosophiae doctor (PhD)
at the University of Bergen

2017

Date of defence: 17 November 2017

Scientific environment

The work of this thesis, both research and writing, was done in the Algorithms group at the Department of Informatics, University of Bergen, Bergen, Norway.

The project was done under the supervision of Prof. Saket Saurabh, and the project received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreements no. 306992 (PARAPPROX).

Acknowledgements

I offer my sincere gratitude to my advisor, Prof. Saket Saurabh, whose constant support and encouragement has been quintessential in my growth as a researcher. Certainly, the thesis could have not been accomplished without his continuous guidance. I would like to thank my co-advisor, Prof. Daniel Lokshtanov for being a never-ending source of answer to questions, guidance, and for always being open for discussions.

I would like to thank Prof. Cristina Bazgan and Prof. Peter Rossmanith for agreeing to review the thesis. Thanks to Prof. Fedor V. Fomin for being in the committee for evaluation of the thesis.

I was very fortunate to have Meirav Zehavi as my flatmate, who has answered all my untimely questions about research very patiently. I also thank you for your valuable guidance about pursuing research and all the dinner-time discussions! I would like to thank Amer E. Mouawad for around-the-clock research discussions, for helping me enhance technical skills, and as a person. I must admit I learned a lot about better writing from Sushmita Gupta. Thanks for your patience and time for explaining perks of good writing, research, life, and everything else. I would like to thank Pranabendu Misra for the last minute hassle towards end of the thesis writing and all the research discussions.

I would like to thank my other co-authors Lawqueen Kanesh, Sudeshna Kolay, R. Krithika, Diptapriyo Majumdar, Fahad Panolan, M. S. Ramanujan, Roohani Sharma, and Prafullkumar Tale for all the late-night stays during the deadlines, and for helping me develop new insights for the problems. I am thankful to Lawqueen Kanesh, Sudeshna Kolay, Sanjukta Roy, Roohani Sharma, and Prafullkumar Tale for all the enjoyable after-work time that we spent together.

Thanks to all members of the Algorithms group whose knowledge and experience, directly or indirectly has influenced me. Thanks for providing a very friendly research environment, and for all the Winter Schools, which were organized at spectacular places. I thank all the members of the office-staff for making my stay at the department comfortable and hassle-free. I am grateful to the Department of Informatics for providing necessary infrastructure and resources to accomplish my research. Many thanks to the University of Bergen for all the academic and non-academic facilities that made the PhD journey smoother. I would like to thank the Student Welfare Organisation in Bergen (SiB) for providing all the recreational facilities and various clubs, where I have improved my physical strength and spent a very enjoyable time with the club-members.

Thanks to Padmaja Barua, Marika Ivanová, and Carmeliza Rosario for all the friendship, food, and outings.

It was all made possible for me by my family's continual support and their unconditional love. I am thankful to my brother, Anmol for his love and care, taking to whom is always eclectic. I am deeply indebted to my parents Mrs. Alka Agrawal and Mr. Kanhaiya Lal Agrawal, and my grandparents Mrs. Snehlata Agrawal and Mr. Ashwani

Kumar Agrawal for all their love, support, and encouragement to pursue whatever I am interested in. I would not be able to return an iota of what I have received from them – still, I will feel content to dedicate this little piece of work to them.

Abstract

Several frameworks have been designed to cope with hard problems. Once a problem is shown to be NP-hard, while designing an algorithm for it either we have to compromise in the running time or the quality of the solution (or both). While designing algorithms in frameworks like Approximation Algorithms and Heuristics we have to be content enough with a compromised quality of the solution. Nevertheless, we might exploit the structure of the input to obtain better algorithms. In the framework of Exact Exponential Time Algorithms the objective is to reduce the exponential search space as much as possible, though it cannot be (completely) avoided, unless some Complexity Theoretic breakdown happens. Downey and Fellows introduced the framework of Parameterized Complexity in early 90's for coping with hard problems. Basically, Parameterized Complexity is a two-dimensional generalization of "P vs. NP", where in addition to the input size n, one studies how relevant a secondary measurement affects the computational complexity of problem instances. Here, the secondary measure, for instance, could be the size of solution or some structural parameter of the input. The two-dimensional analogue of P, is solvable within a time bound of $\mathcal{O}(f(k)n^c)$, where n is the size of the input, k is the parameter, $f(\cdot)$ is some (computable) function, and c is a constant that that is independent on k and n. A parameterized problem is defined by specifying the input, the parameter, and the question to be answered. Parameterized problems that can be solved in $\mathcal{O}(f(k)n^c)$ time are said to be *fixed-parameter tractable* (or in class FPT). Above FPT, there exists a hierarchy of complexity classes, known as the W-hierarchy. Analogous to the case for NP-hardness, which is used as an evidence that a problem is probably not polynomial time solvable, showing that a parameterized problem is hard for one of the classes in the W-hierarchy gives an evidence that the problem is unlikely to be in FPT.

Graph editing problems are one of the central problems in Graph Theory, and have been extensively studied in the realm of Parameterized Complexity since its inception. Some of the important and natural graph editing operations are vertex deletion, edge deletion, edge addition, and edge contraction. For a family of graphs $\mathcal{F}$, the $\mathcal{F}$-Editing problem takes as an input a graph G and an integer k, and the objective is to decide if we can obtain a graph in $\mathcal{F}$ by applying at most k edit operations in G. In fact, the $\mathcal{F}$-Editing problem, where the edit operations are restricted to one of vertex deletion, edge deletion, edge addition, or edge contraction have also received a lot of attention in Parameterized Complexity. When we restrict the operations to only deletion operation (vertex/edge deletion) then the corresponding problem is called $\mathcal{F}$-Vertex (Edge) Deletion problem. On the other hand if we only allow edge contraction then the corresponding problem is called $\mathcal{F}$-Contraction. The $\mathcal{F}$-Editing problem encompasses several NP-hard problems such as Vertex Cover, Feedback vertex set, Planar $\mathcal{F}$-Deletion, Interval Vertex Deletion, Chordal Vertex Deletion, Odd cycle transversal, Edge Bipartization, Tree Contraction,

Path Contraction, Split Contraction, Clique Contraction, etc. This list is not comprehensive but rather illustrative.

A (multi) graph $G = (V, E)$ with an (edge) coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$, is called an α-*edge-colored graph*. Edge-colored graphs are fundamental in Graph Theory and have been extensively studied in the literature, especially for alternating cycles, monochromatic subgraphs, heterochromatic subgraphs, and partitions. A natural extension of the $\mathcal{F}$-Editing problem to edge-colored graphs is the Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing problem. In the latter problem, we are given an α-edge-colored graph G with coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$, and the objective is to decide if using at most k edit operation in G, for each $i \in [\alpha]$, the graph G_i after performing the edit operations is in $\mathcal{F}_i$. Here, for $i \in [\alpha]$, G_i is the graph with vertex set $V(G)$ and edge set $\{e \in E(G) \mid i \in \mathsf{col}(e)\}$.

Chordal graphs are graphs which do not contain any induced cycle on at least four vertices. Chordal graphs and its subclasses are important classes of graphs, and have been extensively studied both from graph theoretic and algorithmic point of view. In the first part of the thesis, we focus on $\mathcal{F}$-Editing problems to (sub) classes of chordal graphs. In particular, we obtain the following results.

- We design an FPT algorithm and a polynomial kernel for the problem Block Graph Vertex Deletion, which is $\mathcal{F}$-(Vertex) Deletion where $\mathcal{F}$ is the family of block graphs.
- We give a polynomial kernel for the problem Chordal Vertex Deletion, which is $\mathcal{F}$-Vertex Deletion where $\mathcal{F}$ is the family of chordal graphs. We also design an $\mathcal{O}(\log^2 n)$-factor approximation algorithm for the problem, where n is the number of vertices in the input graph.
- We give a polynomial kernel for the problem Interval Vertex Deletion, which is $\mathcal{F}$-Vertex Deletion where $\mathcal{F}$ is the family of interval graphs.
- We show that Split Contraction when parameterized by the solution size is W[1]-hard. We also give an ETH based lower bound for the problem, when parameterized by the size of vertex cover. Furthermore, we give an FPT algorithm parameterized by the size of vertex cover, which matches the lower bound we obtain.

In the second part of the thesis, we look at Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing problems, and obtain the following results.

- We design an FPT algorithm and a polynomial kernel for the problem Simultaneous Feedback Vertex Set, which is Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing where for each $i \in [\alpha]$, $\mathcal{F}_i$ is the family of forests and the allowed edit operation is vertex deletion. We also show that the problem becomes W[1]-hard when $\alpha = \mathcal{O}(\log n)$, where n is the number of vertices in the input graph.
- We design an FPT algorithm and a polynomial kernel for the problem Simultaneous Feedback Edge Set, which is Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing where for each $i \in [\alpha]$, $\mathcal{F}_i$ is the family of forests and the allowed edit operation is edge deletion. We also show that the problem becomes W[1]-hard when $\alpha = \mathcal{O}(\log n)$, where n is the number of vertices in the input graph. We look

at another parameterization of the problem, which is the number of edges in the resulting graph, and obtain an FPT algorithm for it.

- We design an FPT algorithm for the problem Simultaneous FVS/OCT, which is Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing where for each $i \in [\alpha] \setminus \{1\}$, $\mathcal{F}_i$ is the family of forests and $\mathcal{F}_1$ is the family of bipartite graphs. Furthermore, the allowed edit operation is vertex deletion. We show that Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing is W[1]-hard when $\mathcal{F}_1$ and $\mathcal{F}_2$ are the family of bipartite graphs, and the allowed edit operation is vertex deletion.

List of Publications

The main results of this thesis are based on the following publications.

1. **Interval Vertex Deletion Admits a Polynomial Kernel [ALM$^+$17b].**

   Akanksha Agrawal, Pranabendu Misra, Saket Saurabh, Meirav Zehavi, and Daniel Lokshtanov. *Manuscript, 2017.*

2. **Polylogarithmic Approximation Algorithms for Weighted-$\mathcal{F}$-Deletion Problems [ALM$^+$17c].**

   Akanksha Agrawal, Pranabendu Misra, Saket Saurabh, Meirav Zehavi, and Daniel Lokshtanov. *Manuscript, 2017.*

3. **On the Parameterized Complexity of Simultaneous Deletion Problems [AKL$^+$17].**

   Akanksha Agrawal, R. Krithika, Daniel Lokshtanov, Amer E. Mouawad, and M. S. Ramanujan. *Accepted for publication: 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), 2017.*

4. **Split Contraction: The Untold Story [ALSZ17].**

   Akanksha Agrawal, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *In: 34th Symposium on Theoretical Aspects of Computer Science (STACS), 2017, pages 5:1–5:14.*

5. **Feedback Vertex Set Inspired Kernel for Chordal Vertex Deletion [ALM$^+$17a].**

   Akanksha Agrawal, Pranabendu Misra, Saket Saurabh, Meirav Zehavi, and Daniel Lokshtanov. *In: 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) 2017, pages 1383-1398.*

6. **Simultaneous Feedback Edge Set: A Parameterized Perspective [APSZ16].**

   Akanksha Agrawal, Fahad Panolan, Saket Saurabh, Meirav Zehavi. *In: 27th International Symposium on Algorithms and Computation (ISAAC), 2016, pages 5:1–5:13.*

7. **A Faster FPT Algorithm and a Smaller Kernel for Block Graph Vertex Deletion [AKLS16].**

   Akanksha Agrawal, Sudeshna Kolay, Daniel Lokshtanov, and Saket Saurabh. *In: 12th Latin American Symposium (LATIN), 2016, pages 1–13.*

8. **Simultaneous Feedback Vertex Set: A Parameterized Perspective [ALMS16].**

   Akanksha Agrawal, Daniel Lokshtanov, Amer E. Mouawad, and Saket Saurabh. *In: 33rd Symposium on Theoretical Aspects of Computer Science (STACS), 2016, pages 7:1–7:15.*

Most parts of the introductory chapters are based on the following publications (partially).

1. **Fine-grained Complexity of Rainbow Coloring and its Variants [Agr17a].**

   Akanksha Agrawal. *Accepted for publication: 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS), 2017.*

2. **On the Parameterized Complexity of Happy Vertex Coloring [Agr17b].**

   Akanksha Agrawal. *Accepted for publication: 28th International Workshop on Combinatorial Algorithms (IWOCA), 2017.*

Contents

List of Figures

Chapter 1

Introduction

After decades of efforts we have not been able to obtain efficient algorithms for problems like CNF-SAT, which is one of the classic NP-complete problem, or prove non-existence of efficient algorithms for them. Here, by an efficient algorithm we mean the one whose running time is bounded by a function polynomial in the size of the input instance. This has led to one of the biggest conjecture in computer science, namely the famous "P $\neq$ NP" conjecture. It is one of the most widely believed conjectures in computer science. Although, there are few communities which do not believe it to be true. For coping with NP-hardness of a problem several frameworks have been designed. These include Heuristic Algorithms, Approximation Algorithms, Exact Exponential Algorithms, and Parameterized Algorithms. Each of the above listed frameworks compromise either in the quality of the solution or in the running time (or both). In the framework of Approximation Algorithms usually the goal is to design a polynomial time algorithm but with a compromise in the quality of the solution. The goal here is to prove a guarantee on the quality of solution, i.e. it not being very far from the optimal solution. Although, for several problems we might compromise both in running time and quality of the solution. The heuristic based algorithms usually fail to provide provable (worst case) guarantee on the quality of the solution, but are usually faster. In the framework of Exact Exponential time algorithm the objective is to reduce the exponential search space as much as possible, though it cannot be (completely) avoided, unless some Complexity Theoretic breakdown happens. Downey and Fellows introduced the framework of Parameterized Complexity in early 90's for dealing with hard problems. This framework measures the computational complexity of the problem using the input size and an additional measurement which is called the parameter. The most natural candidate for the parameter is the size of the solution we are looking for. Although, in many cases we exploit the structural properties of the input instances.

A typical input to a parameterized problem comprises of an instance x of the classical problem and an integer k called the parameter. One of the goals in Parameterized Complexity is to design an algorithm for a parameterized problems where the exponential dependence of the search space is limited by some (computable) function of the parameter rather than the input size. That is, for a parameterized problem Π, given an instance (x, k) decidability in time $f(k)|x|^c$, where $f(\cdot)$ is some (computable) function and c is a constant (independent of k). The parameterized problems that admit such an algorithm are said to be *fixed-parameter tractable* (FPT). Another central notion in the field is *kernelization*, which mathematically captures the efficiency of pre-processing

(or data reduction) algorithm. Here, the goal is to output an equivalent instance of the same problem whose size is bounded $g(k)$, where k is the parameter, and $g(\cdot)$ is some (computable) function. Such a pre-processing algorithm is called a *kernelization algorithm* or a *kernel*. The size of the kernel is the function $g(\cdot)$. Usually, we are interested in polynomial functions.

Not all parameterized problems are fixed-parameter tractable under reasonable Complexity Theoretic assumptions. There exists a hierarchy of complexity classes, known as the W-hierarchy, which captures the hardness of parameterized problems. Analogous to the case for NP-hardness, which provides an evidence that a problem is unlikely to be polynomial time solvable, showing that a parameterized problem is hard for one of the classes in the W-hierarchy provides an evidence that the problem is unlikely to be in FPT.

1.1 $\mathcal{F}$-Editing problems

Given their tremendous modelling power, graphs have become an integral part of Theoretical Computer Science in general, and of Algorithm Design in particular. One graph problem which encapsulates many problems of both practical and theoretical interest is $\mathcal{F}$-Editing. The input of a typical $\mathcal{F}$-Editing problem for a family of graphs $\mathcal{F}$, is a graph G and an integer k, and the objective is to decide if we can obtain a graph in $\mathcal{F}$ by editing at most k vertices or edges. The most natural edit operations are vertex/ edge deletion, vertex/ edge addition, and edge contraction. In fact, the $\mathcal{F}$-Editing problem, where the edit operations are restricted to one of vertex deletion, edge deletion, edge addition, or edge contraction have also received a lot of attention in Parameterized Complexity. When we restrict the operations to only deletion operation (vertex/edge deletion) then the corresponding problem is called $\mathcal{F}$-Vertex (Edge) Deletion problem. On the other hand if we only allow edge contraction then the corresponding problem is called $\mathcal{F}$-Contraction. In this thesis, we restrict ourselves to vertex deletion, edge deletion, and edge contraction as the edit operations. A typical problem of this class is associated with a family of graphs, $\mathcal{F}$, such as edgeless graphs, forests, cluster graphs, chordal graphs, interval graphs, bipartite graphs, split graphs, planar graphs, etc. The $\mathcal{F}$-Editing problem generalizes several NP-hard problems such as Vertex Cover, Feedback vertex set, Planar $\mathcal{F}$-Deletion, Interval Vertex Deletion, Chordal Vertex Deletion, Odd cycle transversal, Edge Bipartization, Tree Contraction, Path Contraction, Split Contraction, Clique Contraction, etc., to name a few. The $\mathcal{F}$-Editing problems are not only mathematically and structurally challenging, but have also led to the discovery of several important techniques in the field of Parameterized Complexity. It would be completely appropriate to say that solutions to these problems played a central role in the growth of the field.

$\mathcal{F}$-Vertex (Edge) Deletion problems. Edit operation restricted to vertex/ edge deletion are among the most basic problems in Graph Theory and Graph Algorithms. Most of these problems are NP-complete [Yan78, LY80], and thus they were subject to intense study in various algorithmic paradigms to cope with their intractability [Fuj98, LY94, FLMS12, MOR13]. These include, considering a restricted class of

inputs, Approximation Algorithms, and Parameterized Complexity. Recently, a methodology for proving lower bounds on running times of algorithms for such parameterized graph editing problems was proposed by Bliznets et al. [BCK+16]. Furthermore, a well-known result by Cai [Cai96] states that in case $\mathcal{F}$ is a hereditary family of graphs with a finite set of forbidden induced subgraphs, then the graph modification problem defined by $\mathcal{F}$ and the aforementioned edit operations admits a simple FPT algorithm. We note that the result of Cai [Cai96] also holds when edge addition is allowed (in addition to vertex/ edge deletion).

Over the course of the last couple of years, parameterized algorithms have been developed for Chordal Vertex (Edge) Deletion [CM16], Unit Vertex (Edge) Editing [Cao17], Interval Vertex (Edge) Deletion [CM15a, Cao16], Cluster Vertex (Edge) Deletion [FKP+14], Threshold Vertex (Edge) Deletion [DDLS15], Chain Vertex (Edge) Deletion [DDLS15], Trivially Perfect Vertex (Edge) Deletion [DFPV15, DP15] and Split Vertex (Edge) Deletion [GKK+15]. This list is not comprehensive but rather illustrative.

Also, the study of kernelization, centred on the above question, has been one of the main areas of research in Parameterized Complexity, yielding many new important contributions to theory. These include general results showing that certain classes of parameterized problems have polynomial kernels, and as well as other results that utilize advanced techniques from algebra, matroid theory and topology for data reduction [AGK+11, BFL+16, FLST10, FLMS12, GJLS15, Jan17, KLP+15, Kra12, KW14, KW12, PPSvL14, Tho10]. The development of a framework for ruling out polynomial kernels under certain complexity-theoretic assumptions [BDFH09, BJK13, DvM14, FS11] has added a new dimension to the area, and strengthened its connections to classical complexity theory. We refer to the following surveys [Kra14, LMS12] and the corresponding chapters in the books [CFK+15, DF13, FG06, Nie06], for a detailed introduction to the field of kernelization.

Some of the most well known results in kernelization are polynomial kernels for graph deletion problems such as Feedback Vertex Set [Tho10], Odd Cycle Transversal [KW14, KW12], Vertex Cover [ACF+04, CKJ01], Planar-$\mathcal{F}$-Deletion [FLMS12], and Treedepth-η-Deletion [GJLS15]. A common thread among all these problems, with the exception of Odd Cycle Transversal, is that the corresponding family $\mathcal{F}$ can be characterized by a finite set of forbidden minors that include at least one connected planar graph. It is known that, if $\mathcal{F}$ is characterized by a finite set of forbidden induced subgraphs, then the corresponding $\mathcal{F}$-Vertex Deletion problem immediately admits an FPT algorithm as well as polynomial sized kernel because of its connection to d-Hitting Set [AK10]. However, if $\mathcal{F}$ is characterized by an *infinite* set of forbidden induced subgraphs, which is the case when $\mathcal{F}$ is the class of chordal graphs, chordal bipartite graphs, interval graphs, proper interval graphs and permutation graphs, our understanding of these problems in the realm of Parameterized Complexity and Kernelization, is still at a nascent stage. While Chordal Vertex Deletion (CVD) was known to be in FPT for some time [Mar10, CM16], the parameterized complexity of Interval Vertex Deletion was settled only recently [CM15a, Cao16]. The parameterized complexity of Permutation Vertex Deletion and Chordal Bipartite Vertex Deletion is still unknown. Coming to the question of polynomial kernels for these problems, the situation is even more grim. Until recently, the only known result was a polynomial kernel for Proper Interval Vertex Deletion: Fomin et

al. [FSV13] obtained a $\mathcal{O}(k^{53})$ sized polynomial kernel for Proper Interval Vertex Deletion, which has recently been improved to $\mathcal{O}(k^4)$ [KCOW16]. A dearth of further results in this area has led to the questions of kernelization complexity of Chordal Vertex Deletion and Interval Vertex Deletion becoming prominent open problems [CM16, CKP13, FSV13, HvtHJ$^+$13, Mar10]. Jansen and Pilipzuck [JP17] recently resolved one of these open questions. They showed that CVD admits a polynomial kernel of size $\mathcal{O}(k^{161}\log^{58} k)$. In this thesis we also look at an improved kernel for the CVD problem. Towards obtaining this result we use an approximate solution to the CVD problem. Therefore, in the following we talk about $\mathcal{F}$-Vertex Deletion problems from the view point of Approximation Algorithms.

Characterizing the graph properties, for which the corresponding vertex deletion problems can be approximated within a bounded factor in polynomial time, is a long standing open problem in Approximation Algorithms [Yan94]. In spite of a long history of research, we are still far from a complete characterization. Constant factor approximation algorithms for Weighted Vertex Cover are known since 1970s [BYE81, NJ74]. Lund and Yannakakis observed that the vertex deletion problem for any hereditary property with a "finite number of minimal forbidden induced subgraphs" can be approximated within a constant ratio [LY93]. They conjectured that for every nontrivial, hereditary property Π with an infinite forbidden set, the corresponding vertex deletion problem cannot be approximated within a constant ratio. However, it was later shown that Weighted Feedback Vertex Set, which doesn't have a finite forbidden set, admits a constant factor approximation [BBF99, BYGNR98], thus disproving their conjecture. On the other hand a result by Yannakakis [Yan79] shows that, for a wide range of graph properties Π, approximating the minimum number of vertices to delete in order to obtain a *connected* graph with the property Π within a factor $n^{1-\varepsilon}$ is NP-hard. We refer to [Yan79] for the precise list of graph properties to which this result applies to, but it is worth mentioning the list includes the class of acyclic graphs and the class of outerplanar graphs.

$\mathcal{F}$-Contraction problems. The vast majority of papers on parameterized graph editing problems, has so far been on edit operations that delete vertices, delete edges (or add edges). However, in recent years, edge contraction as an edit operation has begun to attract significant scientific attention. Edge contraction is a fundamental operation in the theory of graph minors. Given a graph G and an integer k, $\mathcal{F}$-Edge Contraction asks if we can contract at most k edges in G so that the resulting graph belongs to the family $\mathcal{F}$. For several families of graphs $\mathcal{F}$, early papers by Watanabe et al. [WAN81, WAN83] and Asano and Hirata [AH83] showed that $\mathcal{F}$-Edge Contraction is NP-complete. In the framework of Parameterized Complexity, these problems exhibit properties that are quite different from those of problems where we only delete or add vertices and edges. Indeed, for these problems, the result by Cai [Cai96] does not hold. In particular, Lokshtanov et al. [LMS13] and Cai and Guo [CG13] independently showed that if $\mathcal{F}$ is either the family of P_ℓ-free graphs for some $\ell \geq 5$ or the family of C_ℓ-free graphs for some $\ell \geq 4$, then $\mathcal{F}$-Edge Contraction is W[2]-hard. To the best of our knowledge, Heggernes et al. [HvtHL$^+$14] were the first to explicitly study $\mathcal{F}$-Edge Contraction from the viewpoint of Parameterized Complexity. They showed that in case $\mathcal{F}$ is the family of trees, $\mathcal{F}$-Edge Contraction is in FPT but does not admit a polynomial kernel, while in case $\mathcal{F}$ is the family of paths, the corresponding problem admits a

faster algorithm and an $\mathcal{O}(k)$-vertex kernel. Agrawal et al. [AKST17] considered another parameterization for contracting to the family of trees, which is the number of leaves, ℓ and number of edges, k in the solution. They designed a polynomial kernel with $\mathcal{O}(k\ell)$ vertices and also showed a matching kernel lower bound. Golovach et al. [GvtHP13] proved that if $\mathcal{F}$ is the family of planar graphs, then $\mathcal{F}$-Edge Contraction is in FPT. Moreover, Cai and Guo [CG13] showed that in case $\mathcal{F}$ is the family of cliques, $\mathcal{F}$-Edge Contraction is solvable in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$, while in case $\mathcal{F}$ is the family of chordal graphs, the problem is W[2]-hard. Heggernes et al. [HvtHLP13] developed an FPT algorithm for the case where $\mathcal{F}$ is the family of bipartite graphs. Later, a faster algorithm was proposed by Guillemot and Marx [GM13].

1.2 Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing problems

A graph G with a coloring function $\mathsf{col} : E(G) \to 2^{[\alpha]}$ is called an *α-edge-colored graph*. For an α-edge-colored graph G with coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$, and $i \in [\alpha]$, by G_i we denote the graph with vertex set $V(G)$ and edge set $\{e \in E(G) \mid i \in \mathsf{col}(e)\}$. As stated by Cai and Ye [CY14], "edge-colored graphs are fundamental in Graph Theory and have been extensively studied in the literature, especially for alternating cycles, monochromatic subgraphs, heterochromatic subgraphs, and partitions". A natural extension of the $\mathcal{F}$-Editing problem to edge-colored graphs is the Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing problem. In the latter problem, we are given an α-edge-colored graph G with coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$ and an integer k, and the objective to decide if using at most k edit operation in G, for each $i \in [\alpha]$ the modified graph G_i belongs to the family $\mathcal{F}_i$. Recently, Cai and Ye [CY14] studied several problems restricted to 2-edge-colored graphs and the edit operation being vertex deletion. In particular, they consider the Dually Connected Induced Subgraph problem, i.e. find a set S of k vertices in G such that both induced graphs $G_1[S]$ and $G_2[S]$ are connected, and the Dual Separator problem, i.e. delete a set S of at most k vertices to simultaneously disconnect the graphs G_1 and G_2. They show, among other results, that Dual Separator is NP-complete and Dually Connected Induced Subgraph is W[1]-hard even when both G_1 and G_2 are trees. On the positive side, they prove that Dually Connected Induced Subgraph is solvable in time polynomial in the input size when G is a complete graph.

1.3 An Overview of the Thesis

In Chapter 2, we setup basic notations that will be used throughout the thesis. Also, we define terminologies related to Graph Theory and Algorithms that are used in the thesis. Part I of the thesis gives a brief introduction to Parameterized Complexity, which comprises of the following chapters. In Chapter 3, we formally define parameterized problems, fixed-parameter tractable algorithms and kernelization. Furthermore, we illustrate each of them with some examples. In Chapter 4, we give a brief introduction to fixed-parameter intractability and W-hierarchy. Furthermore, we give an introduction to ETH based lower bounds. Each of these notions are illustrated with examples.

Part II of the thesis comprises of new results on $\mathcal{F}$-Editing problems. This part comprises of the following results.

- In Chapter 5, we consider the $\mathcal{F}$-Deletion problem for the family of block graphs, and is based on the paper [AKLS16]. We call this problem Block Graph Vertex Deletion (BGVD), which takes as an input a graph G and an integer k, and the objective is to decide if there is a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a block graph. We design an FPT algorithm for BGVD that runs in time $\mathcal{O}(4^k|V(G)|^{\mathcal{O}(1)})$. This improves over the previous best algorithm for the problem, which runs in time $10^k n^{\mathcal{O}(1)}$ [KK17]. We also give a (vertex) kernel of size $\mathcal{O}(k^4)$ for the problem, which is an improvement over the previously best known (vertex) kernel, which was of size $\mathcal{O}(k^6)$ [KK17].

- In Chapter 6, we explore the approximability of Weighted $\mathcal{F}$-Vertex Deletion for several families $\mathcal{F}$, and is based on the manuscript [ALM+17c]. The main result of this chapter is $\mathcal{O}(\log^2 n)$-factor approximation algorithm for Weighted Chordal Vertex Deletion. On the way to this algorithm, we also obtain a constant factor approximation algorithm for Weighted Multicut in chordal graphs. The methodology that we develop for obtaining approximation algorithm for Weighted Chordal Vertex Deletion is extendable to other problems as well. Let $\mathscr{F}$ be a finite set of graphs that includes a planar graph. Let $\mathcal{F} = \mathscr{G}(\mathscr{F})$ be the family of graphs such that every graph $H \in \mathscr{G}(\mathscr{F})$ does not contain a graph from $\mathscr{F}$ as a minor. The vertex deletion problem corresponding to $\mathcal{F} = \mathscr{G}(\mathscr{F})$ is known as the Weighted Planar $\mathscr{F}$-Minor-Free Deletion (WP$\mathscr{F}$-MFD). We give a randomized $\mathcal{O}(\log^{1.5} n)$-factor (deterministic $\mathcal{O}(\log^2 n)$-factor) approximation algorithm for WP$\mathscr{F}$-MFD, for any finite $\mathscr{F}$ that contains a planar graph. We remark that a different approximation algorithm for the same class of problems with a slightly better approximation ratio of $\mathcal{O}(\log n \log\log n)$ follows from recent work in [BRU17]. Next, we give an $\mathcal{O}(\log^3 n)$-factor approximation algorithm for Weighted Distance Hereditary Vertex Deletion (WDHVD). This is the vertex deletion problem corresponding to the family of distance hereditary graphs, or equivalently graphs of rankwidth 1.

- In Chapter 7, we design a polynomial kernel for Chordal Vertex Deletion (CVD), which is $\mathcal{F}$-Deletion problem corresponding to the family of chordal graphs. This chapter is based on the paper [ALM+17a]. The existence of a polynomial kernel for CVD was a well-known open problem in the field of Parameterized Complexity, which was resolved recently by Jansen and Pilipczuk [JP17]. They designed a polynomial kernel for CVD of size $\mathcal{O}(k^{161}\log^{58} k)$, and asked whether one can design a kernel of size $\mathcal{O}(k^{10})$. While we do not completely resolve this question, we design a significantly smaller kernel of size $\mathcal{O}(k^{12}\log^{10} k)$.

- In Chapter 8, we design a polynomial kernel for Interval Vertex Deletion (IVD for short), which is $\mathcal{F}$-Deletion problem corresponding to the family of interval graphs. This chapter is based on the paper [ALM+17b]. The existence of a polynomial kernel for the problem remained a well-known open problem in Parameterized Complexity. We settle this problem in the affirmative by exhibiting a polynomial kernel for the problem.

- In Chapter 9, we look at the problem Split Contraction, which takes as an input a graph G on n vertices and an integer k, and the objective is to decide

if there is a set $X \subseteq E(G)$ of size at most k such that G/X is a split graph. This chapter is based on the paper [ALSZ17]. Firstly, we study the problem when parameterized by the size a minimum vertex cover. We show that unless ETH fails, Split Contraction parameterized by ℓ, does not have an algorithm running in time $2^{o(\ell^2)} \cdot n^{\mathcal{O}(1)}$. We complement this result by giving an FPT algorithm that runs in time $2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$. Finally, we show that Split Contraction when parameterized by the size of solution, k is W[1]-hard.

Part III of the thesis comprises of new results on Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing problems. This part comprises of the following results.

- In Chapter 10, we consider the problem Simultaneous Feedback Vertex Set, which takes as an input an α-colored graph G and an integer k, and the objective is to decide if there is a set $S \subseteq V(G)$ of size at most k such that for each $i \in [\alpha]$, $G_i - S$ is a forest. This chapter is based on the paper [ALMS16]. We design an FPT algorithm running in time $\mathcal{O}^\star(23^{\alpha k})$ for the problem. Moreover, the special case of $\alpha = 2$, we give a faster algorithm running in time $\mathcal{O}^\star(81^k)$. We give a polynomial kernel for the problem with $\mathcal{O}(\alpha k^{3(\alpha+1)})$ vertices. Finally, we show that Sim-FVS is W[1]-hard, when parameterized by k.

- In Chapter 11, we consider the problem Simultaneous Feedback Edge Set, which takes as an input an α-colored graph G and an integer k, and the objective is to decide if there is a set $S \subseteq E(G)$ of size at most k such that for each $i \in [\alpha]$, $G_i - S$ is a forest. This chapter is based on the paper [APSZ16]. We also show that (unlike the vertex counterpart) for $\alpha \leq 2$ (2-edge-colored graphs) Sim-FES is polynomial time solvable. Next, we show that for $\alpha = 3$, Sim-FES is NP-hard. The same reduction proves that the problem cannot be solved in $2^{o(k)}n^{\mathcal{O}(1)}$ time unless ETH fails. Also, we give an FPT algorithm for the problem that runs in time $\mathcal{O}(2^{\omega k\alpha + \alpha \log k}|V(G)|^{\mathcal{O}(1)})$, where ω is the exponent in the running time of matrix multiplication. We complement our FPT algorithm by showing that Sim-FES is W[1]-hard when parameterized by the solution size k. We give a kernel with $\mathcal{O}((k\alpha)^{\mathcal{O}(\alpha)})$ vertices for the problem. Finally, we study the problem when parameterized by the dual parameter, i.e. the number of remaining edges, q in the graph. We call this problem Max-Sim-Subgraph. We design an algorithm running in time $\mathcal{O}(2^{\omega q\alpha}|V(G)|^{\mathcal{O}(1)})$ for Max-Sim-Subgraph.

- In Chapter 12, we investigate the complexity of Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion in two settings. First, we consider the problem with $\mathcal{F}_1$ being the family of all bipartite graphs and $\mathcal{F}_2 = \mathcal{F}_3 = \ldots = \mathcal{F}_\alpha$ being the family of all forests. We call this problem Simultaneous FVS/OCT (Sim-FVS/OCT). We design an FPT algorithm for Sim-FVS/OCT that runs in time $\mathcal{O}(k^{\mathsf{poly}(\alpha,k)}n^{\mathcal{O}(1)})$. In the second setting, we consider the Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion problem where $\mathcal{F}_1 = \ldots = \mathcal{F}_\alpha$ is the family of bipartite graphs. We call this problem Simultaneous OCT (Sim-OCT). We show that even for $\alpha = 2$, Sim-OCT is W[1]-hard, when parameterized by the size of the solution. This chapter is based on the paper [AKL+17].

Chapter 2

Notations and Definitions

In this chapter, we state some basic definitions and introduce terminologies that are used in this thesis. We also establish some of the notation that will be used throughout.

Sets and Functions

We denote the set of natural numbers, integers, rational numbers, and real numbers by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$, respectively. For $n \in \mathbb{N}$, by $[n]$ we denote the set $\{1, 2, \ldots, n\}$. We use $-\infty$ to denote an infinitesimals number (minus infinity) and use the convention that for any $n \in \mathbb{N}$, we have $-\infty + n = -\infty$ and $-\infty + -\infty = -\infty$. For a set X, by 2^X we denote the set of all subsets of X. For a set $\mathcal{U}$ (which we call *universe* or *ground set*), a subset $S \subseteq \mathcal{U}$, and a family of subsets $\mathcal{F} = \{F_1, \ldots, F_\ell\}$ of $\mathcal{U}$, we let $\mathcal{F}\big|_S$ denote the *restriction* of F to S, i.e. $\mathcal{F}\big|_S = \{F_1 \cap S, \ldots, F_\ell \cap S\}$. For two sets X, Y, by $X \setminus Y$ we denote the set $\{x \in X \mid x \notin Y\}$. Let $f : X \to Y$ be a function. For $y \in Y$, by $f^{-1}(y)$ we denote the set $\{x \in X \mid f(x) = y\}$. For $X' \subseteq X$, by $f|_{X'}$ we denote the function $f|_{X'} : X' \to Y$ such that $f|_{X'}(x) = f(x)$, for all $x \in X'$. For an ordered set $R = X \times Y$, a function $f : R \to Z$, and an element $r = (x, y) \in R$, we slightly abuse the notation to denote $f(r) = f((x, y))$ by $f(x, y)$. For a set X, a weight function $w : X \to \mathbb{R}$, and a set $Y \subseteq X$, by weight of Y we mean $w(Y) = \sum_{y \in Y} w(y)$.

Algorithms and Running time Analysis

We use standard notions from the book of Cormen et al. [CSRL01] for algorithm and runtime related terminologies that we use in this thesis. We use $\mathsf{poly}(x_1, x_2, \ldots, x_t)$ to denote (some) polynomial (but fixed) function on $x_1, x_2, \ldots, x_t$. To describe the running times of our algorithms, we often use the $\mathcal{O}^\star$ notation. Given $f : \mathbb{N} \to \mathbb{N}$, we define $\mathcal{O}^\star(f(n))$ to be $\mathcal{O}(f(n) \cdot \mathsf{poly}(n))$, where $\mathsf{poly}(\cdot)$ is some fixed polynomial function. That is, the $\mathcal{O}^\star$ notation suppresses polynomial factors in the running time expression. We use ω to denote the exponent in the running time of matrix multiplication, the current best known bound for ω is < 2.373 [Wil12].

Graph Theory

We use standard terminology from the book of Diestel [Die12] for the graph related terminologies which are not explicitly defined here. A *graph* $G = (V, E)$ is a tuple, where V is a finite non-empty set and E is a (multi) subset of $V \times V$. Here, V is

called the vertex set and E is the edge set of G. For a graph G, by $V(G)$ and $E(G)$ we denote the vertex and edge sets of the graph G, respectively. A *self-loop* in G is an edge $(v, v) \in E(G)$. If an edge (u, v) appears at least twice in E then it is called a *multi-edge*. The *multiplicity* of an element $(u, v) \in V \times V$ is the number of times it appears in $E(G)$. This is often called *multiplicity of the edge* (u, v). A graph is a *multi graph* if contains a self-loop or a multi-edge. A graph whose vertex set is of size n is often referred as an n-vertex graph. For $v \in V(G)$, by $N_G(v)$ we denote the set $\{u \in V(G) \mid (v, u) \in E(G)\}$, and by $N_G[v]$ we denote the set $N_G(v) \cup \{v\}$. Furthermore, by the degree of v, namely $d_G(v)$, we denote the cardinality of the (multi) set $N_G(v)$. Here, we use the convention that a loop at a vertex v contributes two to the degree of v, and multi-edges contribute to the degree which is same as its multiplicity. For a set $S \subseteq V(G)$, $N_G(S)$ denotes the set $(\cup_{v \in S} N_G(v)) \setminus S$, and $N_G[S]$ denotes the set $N_G(S) \cup S$. We drop the subscript G from above notations when the context is clear. For any non-empty subset $W \subseteq V(G)$, the subgraph of G induced by W is denoted by $G[W]$; its vertex set is W and its edge set consists of all those edges of $E(G)$ with both endpoints in W. For $W \subseteq V(G)$, by $G - W$ we denote the graph $G[V(G) \setminus W]$. For $F \subseteq E(G)$, the subgraph of G induced by F is denoted by $G[F]$; its vertex set is $V(G)$ and its edge set is F. For $F \subseteq E(G)$, by $G - F$ we denote the graph obtained by deleting the edges in F. For graphs G and H, by $G \cap H$, we denote the graph with vertex set as $V(G) \cap V(H)$ and edge set as $E(G) \cap E(H)$.

For $C, C' \subseteq V(G)$, we say that there is an edge between C and C' in G if there exists $u \in C$ and $v \in C'$ such that $(u, v) \in E(G)$. Moreover, in such cases we also say that C and C' are *adjacent* in G. A *path* $P = (v_1, v_2, \ldots, v_\ell)$ in a graph G is a subgraph of G, with vertex set $\{v_1, v_2, \ldots, v_\ell\} \subseteq V(G)$ and edge set $\{(v_i, v_{i+1}) \mid i \in [\ell - 1]\} \subseteq E(G)$. We refer to the path $P = (v_1, v_2, \ldots, v_\ell)$ as a path between v_1 and v_ℓ or a (v_1, v_ℓ)-path. The length of a path is the number of edges in it. For $X, Y \subseteq V(G)$, an (X, Y)-path in G is a path $v_1, v_2, \ldots, v_\ell$ such that $v_1 \in X$ and $v_\ell \in Y$. We say that X and Y are *linked* in G if there exists an (X, Y)-path in G. We say that vertices in Y are *reachable* from X if, for all $y \in Y$, there exists $x \in X$ such that there is a path from x to y. A graph is called *connected* if there is a path between every pair of vertices. A maximal connected subgraph is called a *connected component* or a *component* in a graph. For a subset $\mathcal{C}$ of the set of connected components in the graph G, by $V(\mathcal{C})$ we denote the set $\cup_{C \in \mathcal{C}} V(C)$.

A *cycle* $C = (v_1, v_2, \ldots, v_\ell)$ in a graph G is a subgraph of G, with vertex set $\{v_1, v_2, \ldots, v_\ell\} \subseteq V(G)$ and edge set $\{(v_i, v_{(i+1)}) \mid i \in [\ell]\} \subseteq E(G)$. Here, the indices in the edges set are computed modulo ℓ. The length of a cycle is the number of edges in it. We note that both a double edge and a loop are cycles. For a cycle C on at least four vertices, we say that $(u, v) \in E(G)$ is a *chord* of C if $u, v \in V(C)$ but $(u, v) \notin E(C)$. A cycle C is *chordless* if it contains at least four vertices and has no chords. Given a subset $U \subseteq V(G)$, we say that U *intersects* a cycle C in G if $U \cap V(C) \neq \emptyset$.

For $(v, u) \in E(G)$, the result of *contracting* the edge (v, u) in G is the graph obtained by the following operation. We add a vertex $vu^\star$ and make it adjacent to the vertices in $(N(v) \cup N(u)) \setminus \{v, u\}$ and delete v, u from the graph. We often call such an operation *contraction* of the edge (v, u). For $E' \subseteq E(G)$, we denote by G/E' the graph obtained by contracting the edges of E' in G. Here, we note that the order in which the edges in E' are contracted is insignificant.

A graph G is *isomorphic* to a graph H if there exists a *bijective* function $\phi : V(G) \to V(H)$ such that for $v, u \in V(G)$, $(v, u) \in E(G)$ if and only if $(\phi(v), \phi(u)) \in E(H)$.

A graph G is *contractible* to a graph H if there exists $E' \subseteq E(G)$ such that G/E' is *isomorphic* to H. In other words, G is contractible to H if there exists a *surjective* function $\varphi : V(G) \to V(H)$ with the following properties.

- For all $h, h' \in V(H)$, $(h, h') \in E(H)$ if and only if $W(h), W(h')$ are *adjacent* in G. Here, $W(h) = \{v \in V(G) \mid \varphi(v) = h\}$.
- For all $h \in V(H)$, $G[W(h)]$ is *connected*.

Let $\mathcal{W} = \{W(h) \mid h \in V(H)\}$. Observe that $\mathcal{W}$ defines a partition of the vertex set of G. We call $\mathcal{W}$ an *H-witness structure* of G. The sets in $\mathcal{W}$ are called *witness-sets*.

Given graphs G and H, H is a *contraction* of G if H can be obtained by a sequence of edge contractions in G. A graph H is a *minor* of a G if H is the contraction of some subgraph of G. We say that a graph G is F-minor free when F is not a minor of G. Given a family $\mathcal{F}$ of graphs, we say that a graph G is $\mathcal{F}$-minor free, if for all $F \in \mathcal{F}$, F is not a minor of G. It is well known that if H is a minor of G, then $\mathsf{tw}(H) \leq \mathsf{tw}(G)$. A graph is *planar* if it is $\{K_5, K_{3,3}\}$-minor free [Die12].

Consider a graph G. A set $S \subseteq V(G)$ is an *independent set* in G if for each $u, v \in S$ we have $(u, v) \notin E(G)$. Moreover, we call a graph an independent set if $V(G)$ is an independent set in G. A set $S \subseteq V(G)$ is a *clique* in G if for each $u, v \in S$ we have $(u, v) \in E(G)$. Moreover, we call a graph a clique if $V(G)$ is a clique in G. For $n \in \mathbb{N} \setminus \{0\}$, by K_n we denote the clique on n vertices. The graph G is a *bipartite graph* if there is a partition A, B of $V(G)$ such that A and B are both independent sets in G, respectively. Moreover, it is a *complete bipartite graph* if for each $a \in A$ and $b \in B$ we have $(a, b) \in E(G)$. For $m, n \in \mathbb{N} \setminus \{0\}$, by $K_{m,n}$ we denote the complete bipartite graph on $m + n$ vertices, where its vertex bipartition A, B have sizes m and n, respectively.

A vertex subset $S \subseteq V(G)$ is said to *cover* an edge $(u, v) \in E(G)$ if $Y \cap \{u, v\} \neq \emptyset$. A vertex subset $S \subseteq V(G)$ is called a *vertex cover* in G if it covers all the edges in G. A *minimum vertex cover* is $S \subseteq V(G)$ such that S is a vertex cover and for all $S' \subseteq V(G)$ such that S' is a vertex cover, we have $|S| \leq |S'|$. A vertex subset $S \subseteq V(G)$ is a *feedback vertex set (fvs)* in G if $G - S$ is a forest. If there is no $S' \subset S$ such that $G - S'$ is a forest then S is a *minimal feedback vertex set (minimal fvs)* in G. A vertex subset $S \subseteq V(G)$ is an *odd cycle transversal (oct)* in G if $G - S$ is bipartite. If there is no $S' \subset S$ such that $G - S'$ is a bipartite graph then S is a *minimal odd cycle transversal (minimal oct)* in G. A set $U \subseteq V(G)$ is a *module* in G if for all $u, u' \in U$ and $v \in V(G) \setminus U$, either both u and u' are adjacent to v or both u and u' are not adjacent to v. For the sake of simplicity, we also call $G[U]$ a module. A set $S \subseteq V(G)$ is called a separator in G if the number of connected components in $G - S$ is more than the number of connected components in G. Given a function $f : V(G) \to \mathbb{Q}$ and a set $S \subseteq V(G)$, we denote $f(S) = \sum_{v \in S} f(v)$. Moreover, we say that a subset $S \subseteq V(G)$ is a *balanced separator for* G if for each connected component C in $G - S$, it holds that $|V(C)| \leq \frac{2}{3}|V(G)|$.

A *chordal graph* is a graph G that has no chordless cycle as an induced subgraph. An *interal graph* is a graph is the intersection graph of vertices on the real line. More precisely, given a set of intervals $\mathcal{I}$ on real line, the interval graph G for $\mathcal{I}$ is a graph with vertex set $\mathcal{I}$ and for $u, v \in V(G)$, the edge $(u, v) \in E(G)$ if and only if the intervals u and v intersect in the real line. A *split graph* is a graph G whose vertex set $V(G)$ can be partitioned into two sets, A and B, such that $G[A]$ is a clique while B is an independent set, i.e. $G[B]$ is an edgeless graph.

A vertex *coloring* of a graph G with $k \in \mathbb{N}$ colors is a function $\varphi : V(G) \to [k]$. For a vertex coloring, we call the sets $C_1, C_2, \ldots C_k$ as color classes, where $C_i = \{v \in V(G) \mid \varphi(v) = i\}$ for $i \in [k]$. A vertex coloring φ of G is said to be *proper* if for each $(u, v) \in E(G)$, $\varphi(u) \neq \varphi(v)$. A *harmonious coloring* of a graph G is a (proper) vertex coloring $\varphi : V(G) \to [k]$, with color classes $C_1, C_2, \ldots, C_k$ such that for each $i \in [k]$, C_i is an independent set in G and for all $i, j \in [k]$, there is at most one edge between C_i and C_j in G. We use the following result for computing a harmonious coloring on bounded degree graphs.

Proposition 2.1. [CFG+17, Edw97, LM87, MX91] *Given a G with the degree of each vertex bounded by d, where d is a fixed constant. A harmonious coloring of G can be computed in time $\mathcal{O}(n^{\mathcal{O}(1)})$ using $\mathcal{O}(\sqrt{n})$ colors with each color class having at most $\mathcal{O}(\sqrt{n})$ vertices.*

An edge coloring of a graph G with k colors is a function $\phi : E(G) \to [k]$. A path P in G is said to be a *rainbow path* if for all $e, e' \in E(P)$ with $e \neq e'$ we have $\phi(e) \neq \phi(e')$. An edge coloring is said to be a *rainbow coloring* if for all $u, v \in V(G)$ there is a rainbow path between u and v in G. We drop the prefix vertex and edge from vertex coloring and edge coloring whenever the context is clear.

For a graph G, a set of edges $M \subseteq E(G)$ is called a *matching* in G if each vertex in $G[M]$ is either an isolated vertex or has degree exactly one in $G[M]$. Furthermore, the size of matching is $|M|$. A matching $M \subseteq E(G)$ is a *maximal matching* if no $M' \supset M$ is a matching in G. A matching $M \subseteq E(G)$ is a *maximum matching* if for each matching $M' \subseteq E(G)$ we have $|M'| \leq |M|$. Is is known that a maximum matching in a graph can be computed in time $\mathcal{O}(\sqrt{|V(G)|}|E(G)|)$ [MV80].

A *q-star*, $q \geq 1$, is a graph with $q + 1$ vertices, one vertex of degree q and all other vertices of degree 1. Let G be a bipartite graph with vertex bipartition (A, B). A set of edges $M \subseteq E(G)$ is called a *q-expansion* of A into B if (i) every vertex of A is incident with exactly q edges of M and (ii) exactly $q|A|$ vertices in B are incident with edges of M.

Lemma 2.1 (Expansion Lemma [CFK+15])**.** *Let q be a positive integer and G be a bipartite graph with vertex bipartition (A, B) such that $|B| \geq q|A|$ and there are no isolated vertices in B. Then, there exist nonempty vertex sets $X \subseteq A$ and $Y \subseteq B$ such that:*

(1) X has a q-expansion into Y and

(2) no vertex in Y has a neighbour outside X, i.e. $N(Y) \subseteq X$.

Furthermore, the sets X and Y can be found in time polynomial in the size of G.

Forest Decompositions. A *forest decomposition* of a graph G is a pair (F, β) where F is forest, and $\beta : V(T) \to 2^{V(G)}$ is a function that satisfies the following,

(i) $\bigcup_{v \in V(F)} \beta(v) = V(G)$,

(ii) for any edge $\{v, u\} \in E(G)$ there is a node $w \in V(F)$ such that $v, u \in \beta(w)$,

(iii) and for any $v \in V(G)$, the collection of nodes $T_v = \{u \in V(F) \mid v \in \beta(u)\}$ is a subtree of F.

For $v \in V(F)$, we call $\beta(v)$ the *bag* of v, and for the sake of clarity of presentation, we sometimes use v and $\beta(v)$ interchangeably. We refer to the vertices in $V(F)$ as *nodes*. A *tree decomposition* is a forest decomposition where F is a tree. For a graph G, by $\mathsf{tw}(G)$ we denote the minimum over all possible *tree decompositions* of G, the maximum size of a bag minus one in that *tree decomposition*. A *clique forest* of a graph G is a forest decomposition of G where every bag is a maximal clique in G. A *path decomposition* is a forest decomposition where F is a path. A *clique path* of a graph G is a path decomposition of G where every bag is a maximal clique in G.

We use the following lemmata regarding the class of chordal graphs and interval graphs.

Proposition 2.2 ([Gol04])**.** *A graph G is a chordal graph if and only if G has a clique forest. Moreover, a clique forest of a chordal graph can be constructed in polynomial time.*

Proposition 2.3 ([Gol04])**.** *A graph G is an interval graph if and only if G has a clique path.*

Edge-Colored Graphs. An *α-edge-colored graph* (or α-colored graph for short) is a graph G with a coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$. For an α-edge-colored graph G and $i \in [\alpha]$, by G_i we denote the graph with vertex set $V(G_i) = V(G)$ and edge set $E(G_i) = \{e \in E(G) \mid i \in \mathsf{col}(e)\}$. In the following, let G be an α-edge-colored graph with the coloring function $\mathsf{col} : E(G) \to [k]$. For $v \in V(G)$, the *total degree* of v is $\sum_{i \in [\alpha]} d_{G_i}(v)$. By color i edge (or i-color edge) we refer to an edge in $E(G_i)$, where $i \in [\alpha]$. A vertex $v \in V(G)$ is said to have a color i neighbor if there is an edge (v, u) in $E(G_i)$, furthermore u is a color i neighbor of v. We say a path or a cycle in G is *monochromatic* if all the edges on the path or cycle have the same color.

An *α-edge-colored graph* G with coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$ can also be viewed as a tuple $(V, E_1, \ldots, E_\alpha)$, where $V = V(G)$, and for $i \in [\alpha]$, $E_i = E(G_i)$. This representation is more handy for certain situations. Also, we slightly abuse the notation to denote an *α-edge-colored graph* G with coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$ as the tuple $G = (V, E_1, \ldots, E_\alpha)$.

For $v \in V(G)$, a *v-flower* of order k is a set of k cycles in G whose pairwise intersection is exactly $\{v\}$. If all cycles in a v-flower are monochromatic then we have a *monochromatic v-flower*. An α-colored graph G is an *α-forest* if each G_i is a forest, for $i \in [\alpha]$.

Linear Algebra

For a set A and X, by an operation of A onto X we mean a function $f : A \times X \to X$. For an element $(a, x) \in A \times X$ by ax we denote the element $f(a, x) \in X$. For a field $\mathbb{F}$ with $+$ as the additive operation and $\cdot$ as the multiplicative operation, and a commutative group $(V, +)$ with an operation of $\mathbb{F}$ onto V is called a vector space over $\mathbb{F}$ if for all $a, b \in \mathbb{F}$ and $x, y \in V$ we have: 1) $a(bx) = (ab)x$, 2) $a(x + y) = ax + ay$, 3) $(a + b)x = ax + bx$, and 4) $1 \cdot x = x$. Here, 1 is the additive identity of the field $\mathbb{F}$. If V is a vector space over $\mathbb{F}$ then the elements of V are called vectors. One of the natural candidates for vector

spaces over a field $\mathbb{F}$ is $\mathbb{F}^n$, where $n \in \mathbb{N}$ and the function $f(\cdot)$ being the component-wise multiplication. In this paper, we restrict ourselves only to such types of vector spaces. In the following, consider a field $\mathbb{F}$ and a vector space $V = \mathbb{F}^n$, where $n \in \mathbb{N}$. For a vector $\mathbf{v} = (b_1, b_2, \ldots, b_n) \in \mathbb{F}^n$ and an integer $i \in [n]$, by $\mathbf{v}[i]$ we denote the ith element (or entry) of $\mathbf{v}$, i.e., the element b_i. For vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t \in \mathbb{F}^n$, a linear combination of them is a vector $a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_t\mathbf{v}_t$, where $a_1, a_2, \ldots, a_t \in \mathbb{F}$. A set of vectors V' is said to be *linearly dependent* if there if $\mathbf{v} \in V'$, and vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t \in V' \setminus \{\mathbf{v}\}$ such that $\mathbf{v}$ is a linear combination of them. If V' is not linearly dependent then it is *linearly independent*. An inclusion-wise maximal set of linearly independent vectors is called as a *basis* of the vector space. It is known that for bases B, B' of a vector space, we have $|B| = |B'|$. By $\mathbb{F}_2$ we denote the field with exactly two elements, namely 0 and 1 with the usual addition and multiplication modulo 2 as the field operations. We refer the reader to [Lay06] for more details.

Matroids

A pair $M = (E, \mathcal{I})$, where E is a ground set and $\mathcal{I}$ is a family of subsets (called independent sets) of E, is a *matroid* if it satisfies the following conditions:

(I1) $\phi \in \mathcal{I}$,

(I2) if $A' \subseteq A$ and $A \in \mathcal{I}$ then $A' \in \mathcal{I}$, and

(I3) if $A, B \in \mathcal{I}$ and $|A| < |B|$, then there is $e \in (B \setminus A)$ such that $A \cup \{e\} \in \mathcal{I}$.

The axiom (I2) is also called the hereditary property and a pair $(E, \mathcal{I})$ satisfying only (I2) is called hereditary family. An inclusion wise maximal subset of $\mathcal{I}$ is called a *basis* of the matroid. Using axiom (I3) it is easy to show that all the bases of a matroid have the same size. This size is called the *rank* of the matroid M, and is denoted by $\mathsf{rank}(\mathrm{M})$. We refer the reader to [Oxl06] for more details about matroids.

Representable Matroids Let A be a matrix over an arbitrary field $\mathbb{F}$ and let E be the set of columns of A. For A, we define matroid $M = (E, \mathcal{I})$ as follows. A set $X \subseteq E$ is independent (that is $X \in \mathcal{I}$) if the corresponding columns are linearly independent over $\mathbb{F}$. The matroids that can be defined by such a construction are called *linear matroids*, and if a matroid can be defined by a matrix A over a field $\mathbb{F}$, then we say that the matroid is representable over $\mathbb{F}$. A matroid $M = (E, \mathcal{I})$ is called *representable* or *linear* if it is representable over some field $\mathbb{F}$.

Direct Sum of Matroids Let $M_1 = (E_1, \mathcal{I}_1)$, $M_2 = (E_2, \mathcal{I}_2)$, $\ldots$, $M_t = (E_t, \mathcal{I}_t)$ be t matroids with $E_i \cap E_j = \emptyset$ for all $1 \leq i \neq j \leq t$. The direct sum $M_1 \oplus \cdots \oplus M_t$ is a matroid $M = (E, \mathcal{I})$ with $E := \bigcup_{i=1}^{t} E_i$ and $X \subseteq E$ is independent if and only if $X \cap E_i \in \mathcal{I}_i$ for all $i \in [t]$. Let A_i be the representation matrix of $M_i = (E_i, \mathcal{I}_i)$ over field $\mathbb{F}$. Then,

$$A_M = \begin{pmatrix} A_1 & 0 & 0 & \cdots & 0 \\ 0 & A_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_t \end{pmatrix}$$

is a representation matrix of $M_1 \oplus \cdots \oplus M_t$. The correctness of this is proved in [Mar09, Oxl06].

Uniform Matroid A pair $M = (E, \mathcal{I})$ over an n-element ground set E, is called a uniform matroid if the family of independent sets is given by $\mathcal{I} = \{A \subseteq E \mid |A| \leq k\}$, where k is some constant. This matroid is also denoted as $U_{n,k}$.

Proposition 2.4 ([CFK+15, Oxl06]). *Uniform matroid $U_{n,k}$ is representable over any field of size strictly more than n and such a representation can be found in time polynomial in n.*

Graphic and Cographic Matroid Given a graph G, the graphic matroid $M = (E, \mathcal{I})$ is defined by taking the edge set $E(G)$ as universe and $F \subseteq E(G)$ is in $\mathcal{I}$ if and only if $G[F]$ is a forest. Let G be a graph and η be the number of components in G. The co-graphic matroid $M = (E, \mathcal{I})$ of G is defined by taking the the edge set $E(G)$ as universe and $F \subseteq E(G)$ is in $\mathcal{I}$ if and only if the number of connected components in $G - F$ is η.

Proposition 2.5 ([Oxl06]). *Graphic and co-graphic matroids are representable over any field of size ≥ 2 and such a representation can be found in time polynomial in the size of the graph.*

Elongation of Matroid Let $M = (E, \mathcal{I})$ be a matroid and k be an integer such that $\mathsf{rank}(M) \leq k \leq |E|$. A k-elongation matroid M_k of M is a matroid with the universe as E and $S \subseteq E$ is a basis of M_k if and only if, it contains a basis of M and $|S| = k$. Observe that the rank of the matroid M_k is k.

Proposition 2.6 ([LMPS15]). *Let M be a linear matroid of rank r, over a ground set of size n, which is representable over a field $\mathbb{F}$. Given a number $\ell \geq r$, we can compute a representation of the ℓ-elongation of M, over the field $\mathbb{F}(X)$ in $\mathcal{O}(nr\ell)$ field operations over $\mathbb{F}$.*

α-Matroid Parity In our algorithms we use a known algorithm for α-Matroid Parity. Below we define α-Matroid Parity problem formally and state its algorithmic result.

α-Matroid Parity	**Parameter:** α, q
Input: A representation A_M of a linear matroid $M = (E, \mathcal{I})$, a partition $\mathcal{P}$ of E into blocks of size α and a positive integer q.	
Question: Does there exist an independent set which is a union of q blocks?	

Proposition 2.7 ([LMPS15, Mar09]). *There is a deterministic algorithm for α-Matroid Parity, running in time $\mathcal{O}(2^{\omega q \alpha}||A_M||^{\mathcal{O}(1)})$, where $||A_M||$ is the total number of bits required to describe all the elements of matrix A_M.*

For $\alpha = 2$, the problem α-Matroid Parity is called Matroid Parity. The weighted version of this problem takes an addition input, which is a weight function $w : E \to \mathbb{Q}$, and the corresponding problem is called Weighted Matroid Parity. The problem Weighted Matroid Parity is solvable in polynomial time on linear matroids [IK17].

q-Representative Family. Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid and $\mathcal{B}$ be a family comprising of subsets of size p of E. We say that $\widehat{\mathcal{B}} \subseteq \mathcal{B}$ is a q-representative for $\mathcal{B}$ if for every set $Y \subseteq E$ of size at most q, if there is a set $X \in \mathcal{B}$, such that $X \cap Y = \emptyset$ and $X \cup Y \in \mathcal{I}$, then there is a set $\widehat{X} \in \widehat{\mathcal{B}}$ such that $\widehat{X} \cap Y = \emptyset$ and $\widehat{X} \cup Y \in \mathcal{I}$. If $\widehat{\mathcal{B}} \subseteq \mathcal{B}$ is a q-representative for $\mathcal{B}$ then we denote it by $\widehat{\mathcal{B}} \subseteq^q_{rep} \mathcal{B}$. We use the following result related to the computation of representative family.

Theorem 2.1 ([FLPS16])**.** *Let $\mathcal{M} = (E, \mathcal{I})$ be a linear matroid of rank $k = p + q$, and matrix $A_{\mathcal{M}}$ be a representation of $\mathcal{M}$ over a field $\mathbb{F}$. Also, let $\mathcal{B} = \{B_1, B_2, \ldots, B_t\}$ be a family of independent sets in E of size p. Then, there exists $\widehat{\mathcal{B}} \subseteq^q_{rep} \mathcal{B}$ ($\widehat{\mathcal{B}} \subseteq^q_{minrep} \mathcal{B}$) of size at most $\binom{p+q}{p}$. Moreover, $\widehat{\mathcal{B}} \subseteq^q_{rep} \mathcal{B}$ can be computed in at most $\mathcal{O}(\binom{p+q}{p}tp^\omega + t\binom{p+q}{p}^{\omega-1})$ operations over $\mathbb{F}$. Here, ω is the exponent in the running time of matrix multiplication.*

Definitions of Selected Problems

In the following, we give formal definitions of some problems. For all other problems their respective definitions will be given in the relevant sections.

Vertex Cover
Input: A graph G and an integer k.
Question: Is there $X \subseteq V(G)$ of size at most k such that, X is a vertex cover in G?

Independent Set
Input: A graph G and an integer k.
Question: Is there $X \subseteq V(G)$ of size at least k such that, X is an independent set in G?

Multi-Colored Clique (MCC)
Input: A graph G with a partition $V_1, V_2, \ldots, V_k$ of $V(G)$.
Question: Is there $X \subseteq V(G)$ such that, for all $i \in [k]$, $|X \cap V_i| = 1$ and $G[X]$ is a clique?

Multi-Colored Independent Set (MIS)
Input: A graph G with a partition $V_1, V_2, \ldots, V_k$ of $V(G)$.
Question: Is there $X \subseteq V(G)$ such that, for all $i \in [k]$, $|X \cap V_i| = 1$ and $G[X]$ is an independent set?

d-Hitting Set
Input: A family $\mathcal{F}$ of sets of size at most d over (finite) universe $\mathcal{U}$ and an integer k.
Question: Is there $X \subseteq \mathcal{U}$ of size at least k such that, for each $F \in \mathcal{F}$ we have $F \cap X \neq \emptyset$?

Weighted d-Hitting Set
Input: A family $\mathcal{F}$ of sets of size at most d over (finite) universe $\mathcal{U}$ and a weight function $w : \mathcal{U} \to \mathbb{Q}$.
Output: A set $X \subseteq \mathcal{U}$ of minimum possible weight such that, for each $F \in \mathcal{F}$ we have $F \cap X \neq \emptyset$.

Hitting Set
Input: A family $\mathcal{F}$ of sets over (finite) universe $\mathcal{U}$ and an integer k.
Question: Is there $X \subseteq \mathcal{U}$ of size at least k such that, for each $F \in \mathcal{F}$ we have $F \cap X \neq \emptyset$?

Part I

Introduction to Parameterized Complexity

Chapter 3

Fixed Parameter Tractability and Kernelization

Imagine you are a teacher in a school, and this time you have to deal with a more complicated job. You have to organize a week-long picnic. To add to the fun, you are organizing a forest-treasure hunt. Since it is a big forest, you have kept some contingencies like food and water at certain places, which students can pick up when they need. Also, you want to make them acquainted with the terrain. Due to limited time you can only train few students, and then assign them as team leaders for their respective teams. Students tend to happier when all their friends are around. As you are a good teacher you want to maximize the number of students who are happy during the trip. While dividing students into various teams you want to partition them in a way that maximizes the number of students whose all friends are in their team. You are not only a good teacher, but also a sagacious techie! Having given the limited time frame for dividing students into various teams and your computational expertise, you want to employ a computer to do the job for you.

The above computational scenario is captured by the problem called Happy Vertex Coloring. Here, you are given a friendship graph say G, where the vertex set is the set of students, and there is an edge between two students if they are friends of each other. Some of these vertices are colored, and this partial coloring will be denoted by $c : S \to [k]$, where k is the number of colors used. In the above (special) scenario the set S represents the selected students who are given special training in the forest, and k represent the number of teams. For the (special) scenario you are dealing with, the set of selected students are given distinct colors. A student will be considered happy if all her/his friends are in the same team, in other word they are given the same color. The objective is to find a coloring $\tilde{c}$ of student, which maximize the number of students whose all friends belong to their team. Moreover, the coloring $\tilde{c}$ must be same as the coloring c when restricted to the set S. To capture the above in the friendship graph, a vertex $v \in V(G)$ will be called *happy* if for all $u \in N(v)$ we have $\tilde{c}(u) = \tilde{c}(v)$. Furthermore, we have an integer ℓ such that when we divide students into various teams, we have at least ℓ student who are happy. The goal will be to compute a coloring $\tilde{c}$ such that $\tilde{c}|_S = c$ and the number of happy vertices is at least ℓ.

Having modelled the scenarios as a mathematical problem, now you want to design an algorithm that solves the problem. As it often turn out, the problem Happy Vertex Coloring is NP-complete, whenever $k \geq 3$ [ZL15]. Observe that, although the problem

is NP-hard, there is a brute force algorithm running in time $\mathcal{O}(k^n n^{\mathcal{O}(1)})$, which tries all possible colorings using at most k colors of a graph on n vertices. If the number of students who participate in the forest-treasure hunt is 100 and the number of teams is 10, then the brute force algorithm would roughly require executing 10^{100} instructions, which is not possible to compute in few days using most modern laptops. Therefore, you turn to various mechanisms for coping with NP-hardness. Some of the widely used coping mechanisms are Approximation Algorithms, Heuristics, Exact Exponential Algorithms, and Parameterized Algorithms. This time the coping mechanism you choose to employ is the theory of Parameterized Complexity. Before moving on to how we employ the theory of Parameterized Complexity to obtain better algorithms, we formally define parameterized problems and important notions in Parameterized Complexity that are used for dealing with NP-hard problems.

3.1 Parameterized Problems

We now move away from the "forest-treasure hunt" problem, and indulge in formal definitions and notions in Parameterized Complexity. Later, we illustrate some of these notions using our toy problem. Next, we move to the formal definition of *parameterized problems*.

Definition 3.1. A *parameterized problem* Π is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a (fixed) finite alphabet set. An instance of a parameterized problem is a tuple (x, k), where k is called the parameter.

Let Π be a parameterized problem, and $I = (x, k)$ be an instance of Π. We always assume that x is encoded using the alphabets in Σ. For example, x could be a graph, and a potential representation of it for $\Sigma = \{0, 1\}$ could be the adjacency matrix. By the size of I we mean $|I| = |x| + k$. Here, we can interpret k as being encoded in unary.

Next, we look at some examples of parameterized problems. The classical problem of Vertex Cover can be parameterized by the solution size, which is the size of vertex cover we are looking for. Here, a tuple (G, k) is an instance of Vertex Cover, where G is a graph and k is an integer, and the objective is to decide whether or not G admits a vertex cover of size at most k. One of the natural parameters for parameterizing a classical problem instance is the solution size. Moreover, one can also use some structural properties of the input/ output as the parameter. For our example of Happy Vertex Coloring, the potential parameters could be the number of colors, the number of happy vertices, the number of unhappy vertices, the number of pre-colored vertices, the number of uncolored vertices, some structural properties like maximum degree of a vertex in the input graph, or a combinations of some of them.

Next, we move to some important notions and complexity classes for parameterized problems.

3.2 Fixed-Parameter Tractability

A central notion in Parameterized Complexity is *fixed-parameter tractability* (FPT), which is defined below.

Definition 3.2. A parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is called *fixed-parameter tractable* if there is an algorithm $\mathcal{A}$, a computable (non-decreasing) function $f : \mathbb{N} \to \mathbb{N}$, and a constant c such that, given an instance $I = (x, k)$ of Π, the algorithm $\mathcal{A}$ correctly decides whether or not $(x, k) \in \Pi$ in time bounded by $f(k) \cdot |I|^c$. The complexity class containing all fixed-parameter tractable problems is called FPT.

In the following sections, we look at some examples of designing FPT algorithms for parameterized problems. In Section 3.2.1, we design an FPT algorithm for the problem called Weighted Feedback Vertex Set, which also illustrates the use of iterative compression technique and branching in designing FPT algorithms. The technique of iterative compression was introduced by Reed et al. [RSV04] for designing FPT algorithm for Odd Cycle Transversal. We will be using the technique of iterative compression and branching in later chapters in the thesis. Moreover, we use the algorithm for Weighted Feedback Vertex Set as a blackbox in later parts of the thesis. We also give illustrative example for some techniques which are not explicitly used in the thesis, but are quite useful in designing FPT algorithms. In Section 3.2.2, we illustrate an example of designing FPT algorithm using dynamic programming over tree decomposition for Maximum Happy Vertices. Finally, in Section 3.2.3, using the example of Subset Rainbow k-Coloring we illustrate the technique of color coding.

3.2.1 Improved algorithm for Weighted Feedback Vertex Set

The Feedback Vertex Set problem is one of the most well studied problems. Given a graph G and an integer k, the objective is to decide whether or not there is $S \subseteq V(G)$ of size at most k such that $G - S$ is a forest. Thus, S is a vertex subset that intersects every cycle in G. In the Parameterized Complexity setting, Feedback Vertex Set parameterized by k, has an FPT algorithm. The best known FPT algorithm runs in time $\mathcal{O}(3.618^k n^{\mathcal{O}(1)})$ [CFK+15, KP14]. The problem also admits a kernel on $\mathcal{O}(k^2)$ vertices [Tho10]. Another variant of Feedback Vertex Set that has been studied in Parameterized Complexity is Weighted Feedback Vertex Set (Weighted-FVS), where each vertex in the graph has some rational number as its weight. The problem Weighted-FVS is formally defined below.

Weighted-FVS **Parameter:** k
Input: A graph G, a weight function $w : V(G) \to \mathbb{Q}$, and an integer k.
Output: A set $S \subseteq V(G)$ of size at most k of minimum possible weight such that $G - S$ is a forest.

Weighted-FVS is known to be in FPT with an algorithm of running time $5^k n^{\mathcal{O}(1)}$ [CFL+08]. In this section, we obtain a faster FPT algorithm for Weighted-FVS. We use the method of iterative compression together with branching to reduce an instance of Weighted-FVS to an instance of Weighted-Matroid Parity. It is known that Weighted-Matroid Parity can be solved in polynomial time [IK17]. In fact, our algorithm is very similar to the algorithm for Feedback Vertex Set in [CFK+15, KP14].

We give an algorithm only for the disjoint variant of the problem, which we call Disjoint Weighted-FVS. In the Disjoint Weighted-FVS, we are given a graph G, a weight function $w : V(G) \to \mathbb{Q}$, an integer k, and a feedback vertex set $R \subseteq V(G)$

of size $k+1$. The objective is to find a set $X \subseteq V(G) \setminus R$ such that X is a minimum weighted-fvs of size at most k in G.

Before describing the algorithm for Disjoint Weighted-FVS, we briefly discuss how we can use it to solve the problem Weighted-FVS. Fix an arbitrary ordering $(v_1, v_2, \ldots, v_n)$ on the vertices of G. For $j \in [n]$, we let $G^{(j)}$ denote the subgraph of G induced on the first j vertices. Note that when $j \leq k+1$, we can take the vertex set of $G^{(k+1)}$ as a feedback vertex set in $G^{(k+1)}$ of size $k+1$. Now suppose that for some $j > k+1$, we have constructed a minimum weighted-fvs $S^{(j)}$ of $G^{(j)}$ of size at most k. Then, in the graph $G^{(j+1)}$, the set $Z^{(j+1)} = S^{(j)} \cup \{v_{j+1}\}$ is an feedback vertex set of size at most $k+1$. If in fact $|Z^{(j+1)}| \leq k$ we can arbitrarily add some vertices to make its size k. In each of the cases we described, we have $|Z^{(j+1)}| = k+1$ and we need to "compress" into a smaller solution (if it exists). To that end, we first guess the intersection X of $S^{(j+1)}$ with $Z^{(j+1)}$. In other words, for every $q \in \{0\} \cup [k]$ and every subset X of $Z^{(j+1)}$ of size q, we construct an instance (G', w', k', R') of Disjoint Weighted-FVS as follows. We let $G' = G^{(j+1)} - X$, $w' = w|_{V(G')}$, $R' = Z^{(j+1)} \setminus X$, and $k' = k - q$. Note that $|R'| = k - q + 1$, so $|R'|$ is one larger than k'. If G' does not admit an Disjoint Weighted-FVS of size at most k, then of course neither does G, and we may terminate (and declare a *no* instance). If on the other hand $S^{(j+1)}$ has been successfully found (which is the union of X and the solution to instance (G', w', k', R')), then we proceed to the next graph $G^{(j+2)}$, and so on. Finally, observe that $G^{(n)} = G$, so we eventually either find an minimum weighted-fvs of size at most k in G or conclude that no solution of size at most k exists. We rely on the following observation for the running time of Weighted-FVS using the algorithm for Disjoint Weighted-FVS.

Observation 1 ([CFK+15]). *The existence of an algorithm for* Disjoint Weighted-FVS *with running time $c^k \cdot n^{\mathcal{O}(1)}$, for a constant c, implies that* Weighted-FVS *can be solved in time $c^{k+1} \cdot n^{\mathcal{O}(1)}$.*

Next, we focus on designing an algorithm for Disjoint Weighted-FVS.

Algorithm for Disjoint Weighted-FVS

Let $(G, w : V(G) \to \mathbb{Q}, k, R)$ be an instance of Disjoint Weighted-FVS, and let $F = G - R$. We call a vertex $v \in V(F)$ a *nice* vertex if $d_G(v) = 2$ and both its neighbors are in R. A vertex $v \in V(F)$ is called *tent* if $d_G(v) = 3$ and all its neighbors are in R. We only solve decision version of the problem, however the algorithm can easily be modified to obtain a solution. We start with some simple reduction rules that preprocesses the graph. The reduction rules are applied in the order in which they are described.

Reduction Rule 3.1. *If $k < 0$ or $k = 0$ and G is not a forest, then return that (G, w, k, R) is a no instance of* Disjoint Weighted-FVS.

Reduction Rule 3.2. *If $k \geq 0$ and G is a forest, then return that (G, w, k, R) is a yes instance of* Disjoint Weighted-FVS.

Reduction Rule 3.3. *If there is $v \in V(G)$ which is of degree one, then delete v. The resulting instance is $(G - \{v\}, w|_{V(G)\setminus\{v\}}, k, R \setminus \{v\})$.*

Reduction Rule 3.4. *If there is $v \in V(F)$ such that $G[R \cup \{v\}]$ has a cycle, then delete v from G and degrease k by 1. The resulting instance is $(G - \{v\}, w|_{V(G)\setminus\{v\}}, k-1, R)$.*

Reduction Rule 3.5. *If there is an edge of multiplicity larger than* 2 *in* $E(G)$*, then reduce its multiplicity to* 2*.*

The correctness of the above reduction rules do not depend on the weights, and thus it is similar to the one for unweighted version of Weighted-FVS, and therefore, their safeness follows from [CFK+15].

Reduction Rule 3.6. *Let* $x \in V(F)$ *be a leaf with the only neighbor as* y *in* F*. Also,* x *has at most* 2 *neighbors in* R*. Subdivide the edge* (x, y)*, and add the newly created vertex* x^* *to* R*. We define the new weight function* $w^* : V(G) \cup \{x^*\} \to \mathbb{Q}$*, as follows:* $w^*(x^*) = 1$ *and* $w^*(v) = w(v)$*, for* $v \in V(G)$*. Let* G^* *be the newly created graph after subdivision of the edge* (x, y)*. Our reduced instance for* Disjoint Weighted-FVS *is* $(G^*, w^*, k, R \cup \{x^*\})$*.*

Lemma 3.1. *Reduction Rule 3.6 is safe.*

Proof. Let $x \in V(F)$ be a leaf with the only neighbor as y in F. Also, x has at most 2 neighbors in R. Let G^* be the graph after subdivision of the edge (x, y) with the newly added vertex as x^*. Furthermore, we define $w^* : V(G) \cup \{x^*\} \to \mathbb{Q}$, as follows: $w^*(x^*) = 1$ and $w^*(v) = w(v)$, for $v \in V(G)$. We will show that G has a weighted-fvs of size at most k and weight at most W if and only if G^* has a weighted-fvs of size at most k and weight at most W.

In the forward direction, let $S \subseteq V(G) \setminus R$ be a weighted-fvs of G of size at most k such that $w(S) \leq W$. We will show that indeed S is a weighted-fvs in G^* of size at most k and $w^*(S) \leq W$. We first bound the weight of S in G^*. Note that $w^*(v) = w(v)$, for $v \in V(G)$. Therefore $w^*(S) = w(S) \leq W$. We now show that S is a feedback vertex set in G^*. Suppose not, then there is a cycle C in $G^* - S$. If C does not contain x^*, then it contains none of the edges in $\{(x, x^*), (x^*, y)\}$. This by definition implies that C is also a cycle of $G - S$, which is a contradiction. On the other hand, let C contain the vertex x^*. By construction, C must also contain both (x, x^*) and (x^*, y). However, this means that $C' = (C - \{(x, x^*), (x^*, y)\}) \cup \{(x, y)\}$ is a cycle in $G - S$, which is a contradiction. Therefore, $S \subseteq V(G) \setminus R$ is also a weighted-fvs in G^*.

In the reverse direction, consider a weighted-fvs $S \subseteq V(G^*) \setminus (R \cup \{x^*\})$ of size at most k and $w^*(S) \leq W$. As S is a weighted-fvs disjoint from $R \cup \{x^*\}$, $x^* \notin S$. Thus, $w(S) = w^*(S)$, . We now show that S is a feedback vertex set in G. Suppose not, then there is a cycle C in $G - S$. If C does not contain the edge (x, y), then by construction, C is also a cycle in $G^* - S$, which is a contradiction to the fact that S is a weighted-fvs of G^*. Otherwise, C contains the edge (x, y). But then, $C' = (C - \{(x, y)\}) \cup \{(x, x^*), (x^*, y)\}$ is a cycle $G^* - S$, which is a contradiction. Hence, S must is a weighted-fvs for G.

□

Now, we are ready to describe the main algorithm. To measure the running time of our algorithm for an instance $I = (G, w, k, R)$, we define the following measure.

$$\mu(I) = k + \rho(R) - (\eta + \tau)$$

Here, $\rho(R)$ is the number of connected components in $G[R]$ and η, τ are the number of nice vertices and tents in F, respectively.

Let $I = (G, w, k, R)$ be an instance of Disjoint Weighted-FVS where none of Reduction Rules 3.1 to 3.6 are applicable. It is clear that Reduction Rules 3.1 to 3.5

do not increase the measure. We only need to argue about Reduction Rule 3.6, which increases the number of vertices in R. The number of vertices in R^* is one more than the number of vertices in R, and thus the number of connected components in $G^*[R^*]$ increases by (at most) one. However, we create either a *nice* vertex or a *tent* in G^*, therefore one of η or τ increases by 1. Hence, $\mu(I^*) = k+(\rho(R)+1)-(\eta+\tau+1) \leq \mu(I)$. Note that we do not increase the number of vertices in F, therefore we can apply the Reduction Rule 3.6 at most $|V(F)|$ times.

In Lemma 3.2, we show that if $\mu < 0$, then (G, w, k, R) is a *no* instance. This will form one of the base cases in our branching algorithm.

Lemma 3.2. *For an instance $I = (G, w, k, R)$ of* Disjoint Weighted-FVS*, if $\mu < 0$, then I is a no instance.*

Proof. Suppose that I is a *yes* instance of Disjoint Weighted-FVS and $\mu < 0$. Let S be a weighted-fvs in G of size at most k, and $F' = G - S$, which is a forest. Let $N \subseteq V(G) \setminus R$, $T \subseteq V(G) \setminus R$ be the set of nice vertices and tents in $V(G) \setminus R$, respectively. Since F' is a forest we have that $G' = G[(R \cup N \cup T) \setminus S]$ is a forest. In G', we contract each of the connected components in R to a single vertex to obtain a forest $\tilde{F}$. Observe that $\tilde{F}$ has at most $|V(\tilde{F})| \leq \rho(R) + |N \setminus S| + |T \setminus S|$ vertices and thus can have at most $\rho(R) + |N \setminus S| + |T \setminus S| - 1$ many edges. The vertices in $(N \cup T) \setminus S \subseteq V(G) \setminus R$ forms an independent set in $\tilde{F}$, since they are nice vertices or tents. The vertices in $N \setminus S$ and $T \setminus S$ have degree 2 and degree 3 in $\tilde{F}$, respectively, since their degree cannot drop while contracting the components of $G[R]$. Thus we obtain the following.

$$2|N \setminus S| + 3|T \setminus S| \leq |E(\tilde{F})| \leq \rho(R) + |N \setminus S| + |T \setminus S| - 1$$

Therefore, $|N \setminus S| + |T \setminus S| < \rho(R)$. But $N \cap T = \emptyset$, and thus we have the following.

$$|N| + |T| < \rho(R) + |S| \leq \rho(R) + k \tag{3.1}$$

However, by our assumption, $\mu(I) = \rho(R) + k - (|N| + |T|) < 0$, and thus $|N| + |T| > \rho(R) + k$. This, contradicts the inequality given in Equation 3.1 contradicting our assumption that I is a *yes* instance of Disjoint Weighted-FVS. This completes the proof.

□

Next, we conjure all that we have developed, and give the description of the algorithm for Disjoint Weighted-FVS, prove its correctness, and analyze its running time assuming a polynomial time procedure that we explain in the next subsection.

Description of the Algorithm. Let $I = (G, w, k, R)$ be an instance of Disjoint Weighted-FVS. If $G[R]$ is not a forest, then return that (G, w, k, R) is a *no* instance of Disjoint Weighted-FVS. Hereafter, we will assume that $G[R]$ is a forest. First, the algorithm exhaustively applies the Reduction Rules 3.1 to 3.6. If at any point $\mu(I) < 0$, then we return that (G, w, k, R) is a *no* instance. For sake of clarity, we will denote the reduced instance by (G, w, k, R). If all the vertices $v \in V(G) \setminus R$ are either nice vertices or tents then we solve the problem in polynomial time by using Theorem 3.2. We defer the proof of Theorem 3.2 to the following subsection, where we solve the instance using Weighted Matroid Parity. Otherwise, we apply the following Branching rule.

Branching Rule 3.2.1. If there is a leaf vertex $v \in V(G) \setminus R$, which is neither a nice vertex nor a tent then, we branch as follows.

i) v belongs to the solution. In this branch we delete v from G and decrease k by 1. The resulting instance is $(G - \{v\}, w', k-1, R)$, where $w' = w|_{V(G)\setminus\{v\}}$.

`The measure` μ `decreases by at least 1.`

ii) v does not belong to the solution. Note that v is neither a nice vertex nor a tent, and none of the reduction rules are applicable. Therefore, v has at least 3 neighbors in R. We add the vertex v to R. As a result, the number of components in $G[R]$ decreases by at least 2. The resulting instance is $(G, w, k, R \cup \{v\})$.

`The measure` μ `decreases by at least 2.`

The worst case branching vector corresponding to the above branching rule is $(1, 2)$.

Lemma 3.3. *The algorithm presented is correct.*

Proof. Let $I = (G, w, k, R)$ be an instance of Disjoint Weighted-FVS. We prove the correctness of the algorithm by induction on the measure $\mu = \mu(I)$. By Lemma 3.2 when $\mu < 0$, then we correctly conclude that I is a *no* instance.

For the induction hypothesis, let us assume that the algorithm correctly decides if the input is a *yes*/*no* instance for $\mu = t$. We will prove it for $\mu = t + 1$. If any of the reduction rules are applicable, then either we correctly decide the instance or create an equivalent instance, which follows from their safeness. If we correctly decide the instance then the algorithm is trivially correct for $\mu = t + 1$. Otherwise, we obtain an instance I'. If $\mu(I') < \mu(I)$ (the case when Reduction Rule 3.4 is applied) then by the induction hypothesis the algorithm correctly decides for the measure $\mu = t$. Otherwise, we have an instance with the same measure. If none of the reduction rules are applicable, then we have the following cases:

- Each $v \in V(G)\setminus R$ is either a nice vertex or a tent. In this case, we solve the problem in polynomial time, and the correctness of this step follows from Theorem 3.2.
- There is a leaf $v \in V(G) \setminus R$ such that v is neither a nice vertex nor a tent. The existence of such a vertex is guaranteed by the non-applicability of the reduction rules. In this case, we apply the Branching Rule 3.2.1. The Branching Rule is exhaustive. Moreover, at each branch the measure decreases at least by one. Hence, by the induction hypothesis it follows that the algorithm correctly decides whether or not I is a *yes* instance.

This completes the proof of correctness. □

Next, we obtain an FPT algorithm for Disjoint Weighted-FVS using Lemma 3.3.

Lemma 3.4. Disjoint Weighted-FVS *can be solved in time* $\mathcal{O}^\star(2.618^k)$.

Proof. The correctness of the algorithm follows from Lemma 3.3. All of the Reduction rules 3.1 to 3.6 can be applied in polynomial time. Also, at each branch we spend a polynomial amount of time. For each of the recursive calls at a branch, the measure μ decreases at least by 1. When $\mu < 0$, then we are able to correctly conclude that the

given input is a no instance by Lemma 3.2. The number of leaves, and thus the size of the branching tree is upper bounded by the solution to the following recurrence.

$$T(\mu) \leq T(\mu - 1) + T(\mu - 2)$$

The above recurrence solves to 1.618^{μ}. Since at the start of the algorithm $\mu \leq 2k$, we have that the number of leaves is upper bounded by $\mathcal{O}(1.618^{2k})$. Therefore, Disjoint Weighted-FVS can be solved in time $\mathcal{O}^{\star}(2.618^{k})$.

□

Using Lemma 3.4 and Observation 1, we obtain the following theorem.

Theorem 3.1. Weighted-FVS *has an* FPT *algorithm which runs in time* $\mathcal{O}^{\star}(3.618^{k})$, *where* n *is the number of vertices in the input graph.*

Algorithm for sub-cubic Disjoint Weighted-FVS

Let (G, w, k, R) be an instance of Disjoint Weighted-FVS where each vertex in $V(G) \setminus R$ is either a nice vertex or a tent. An instance of the Matroid Parity problem we create is same as that in [KP14]. In fact, what we use is the Weighted Matroid Parity problem.

The Weighted Matroid Parity problem for the graphic matroid $\mathcal{M}_H$ of a graph H is defined as follows. Let H be a graph with even number of edges, i.e. $|E(H)| = 2m$, where $m \in \mathbb{N}$, and we have a partition of $E(H)$ into pairs, say $E(H) = \{e_1^1, e_2^1\} \cup \{e_1^2, e_2^2\} \cup \cdots \cup \{e_1^m, e_2^m\}$. Furthermore, for each pair $\{e_1^i, e_2^i\}$, where $i \in [m]$ there is a positive (rational) weight $w_{\mathcal{M}}(\{e_1^i, e_2^i\})$. That is, $w_{\mathcal{M}}$ is a weight function on pairs. We want to find a set $I \subseteq [m]$ of maximum weight such that $\cup_{i \in I}\{e_1^i, e_2^i\}$ is an independent set in $\mathcal{M}_H$. Equivalently, $\cup_{i \in I}\{e_1^i, e_2^i\}$ is acyclic in H. The Weighted Matroid Parity problem is polynomial time solvable on linear matroids, and hence in graphic matroids [IK17].

For each vertex $v \in V(G) \setminus R$, we arbitrarily label the edges incident to v. If v is a nice vertex then we label it as $\{e_1^v, e_2^v\}$; otherwise if v is a tent vertex then we label it as $\{e_0^v, e_1^v, e_2^v\}$. We let $w_{\mathcal{M}}(\{e_1^v, e_2^v\}) = w(v)$. Note that $F = E(G[R]) \cup \{e_0^v : v \in V(G) \setminus R\}$ is a forest. We contract all the edges in G which are in F to obtain a (new) graph H. In the process of contraction, we have not contracted any multiple edge or self-loops. Also, we have $E(H) = \bigcup_{v \in V(G) \setminus R}\{e_1^v, e_2^v\}$. The input to the Weighted Matroid Parity algorithm for graphical matroid is the graph H, the set of pairs $\{e_1^v, e_2^v\}$ with weight $w_{\mathcal{M}}(\{e_1^v, e_2^v\})$, for $v \in V(G) \setminus R$. In Lemma 3.5 we prove that finding a minimum weighted-fvs $X \subseteq V(G) \setminus R$ in (G, w, k, R) is equivalent to computing a maximum weight subset $I \subseteq \{\{e_1^v, e_2^v\} \mid v \in V(G) \setminus R\}$ such that $\cup_{v \in I}\{e_1^v, e_2^v\}$ is an independent set in $\mathcal{M}_H$.

Lemma 3.5. *For a set* $I \subseteq V(G) \setminus R$, $\cup_{i \in I}\{e_1^i, e_2^i\}$ *is an independent set in* $\mathcal{M}_H$ *of maximum weight if and only if* $(V(G) \setminus R) \setminus I$ *is a feedback vertex set in* G *of minimum weight.*

Proof. By the definition of H, $\cup_{i \in I}\{e_1^i, e_2^i\}$ is an independent set in $\mathcal{M}_H$ if and only if $F \cup (\cup_{i \in I}\{e_1^i, e_2^i\})$ is acyclic in G. Recall that $F = E(G[R]) \cup \{e_0^v : v \in V(G) \setminus R\}$ is a forest. Therefore, if $\cup_{i \in I}\{e_1^i, e_2^i\}$ is an independent set in $\mathcal{M}_H$, then $G' = G - ((V(G) \setminus R) \setminus I)$ is a forest. In other words, $(V(G) \setminus R) \setminus I$ is a feedback vertex set in G.

In the reverse direction, consider $I \subseteq V(G) \setminus R$ such that $(V(G) \setminus R) \setminus I$ is a feedback vertex set in G. This implies that $G[I \cup R]$ is a forest. Define $F' = \bigcup_{i \in I} \{e_1^i, e_2^i\}$. Clearly, $F' \subseteq E(G[I \cup R])$. Suppose, F' contains a cycle in H. This means that on uncontracting the edges in F, there is a cycle contained in $G[I \cup R]$, which is a contradiction. Therefore, $\cup_{i \in I} \{e_1^i, e_2^i\}$ is an independent set in $\mathcal{M}_H$.

Note that by definition of $w_{\mathcal{M}}$, $w_{\mathcal{M}}(I) = w(I)$. Therefore, $w((V(G) \setminus R) \setminus I) = w(V(G)) - w(R) - w(I) = w(V(G)) - w(R) - w_{\mathcal{M}}(I)$. This implies that whenever $w_{\mathcal{M}}(I)$ is maximized then $w((V(G) \setminus R) \setminus I)$ is minimized and vice-versa. This completes the proof.

□

Lemma 3.5 immediately implies the following theorem.

Theorem 3.2. *Let (G, w, k, R) be an instance of* Disjoint Weighted-FVS. *If each $v \in V(G) \setminus R$ is either a nice vertex or a tent, then* Disjoint Weighted-FVS *in (G, w, k, R) can be solved in polynomial time.*

3.2.2 Algorithm for Maximum Happy Vertices on graphs of bounded treewidth

In this section, we design a dynamic programming based FPT algorithm for Maximum Happy Vertices when parameterized by the treewidth of input graph and the number of colors in the pre-coloring of a subset of vertices. The problem Maximum Happy Vertices is formally defined below.

Maximum Happy Vertices	**Parameter:** $k, \mathsf{tw}(G)$
Input: A graph G, an integer k, a vertex subset $S \subseteq V(G)$, and a (partial) coloring $c : S \subseteq V(G) \to [k]$.	
Output: An integer ℓ such that for all $\tilde{c} \in \{\hat{c} \mid \hat{c}\|_S = c\}$, we have $\|H_{\tilde{c}}\| \leq \ell$, where $H_{\tilde{c}}$ is the set of happy vertices in G with respect to $\tilde{c}$. Furthermore, there exist $\tilde{c} \in \{\hat{c} \mid \hat{c}\|_S = c\}$ such that $\|H_{\tilde{c}}\| = \ell$.	

Let $(G, k, S, c : S \to [k])$ be an instance of Maximum Happy Vertices, $n = |V(G)|$ and $m = |E(G)|$. Without loss of generality we assume that for each $i \in [k]$, we have $c^{-1}(i) \neq \emptyset$, otherwise we can adjust the instance appropriately by adding isolated vertices. For each $i \in [k]$, we arbitrarily choose a vertex from $c^{-1}(i)$, which we denote by $v_i^\star$. We let $S^\star = \{v_i^\star \mid i \in [k]\}$, and $\mathcal{S}^\star = (\{v_1^\star\}, \{v_2^\star\}, \ldots, \{v_k^\star\})$. We start by computing a tree decomposition $(\overline{\mathcal{T}}, \bar{\beta})$ of width at most $w \leq 6 \cdot \mathsf{tw}(G)$ in time $\mathcal{O}(2^{\mathcal{O}(\mathsf{tw}(G))}n)$, using the algorithm of Bodlaender et al. [BDD+16]. Next, we find a more structure tree decomposition called a *nice tree decomposition*, which is defined below.

Definition 3.3. A (rooted) tree decomposition $(\mathcal{T}, \beta)$ of a graph G, where $\mathcal{T}$ is a tree rooted at $r \in \mathcal{V}(\mathcal{T})$ and $\beta : V(\mathcal{T}) \to 2^{V(G)}$, is called a *nice tree decomposition* if the following conditions are satisfied.

1. $\beta(r) = \emptyset$ and $\beta(\ell) = \emptyset$ for every leaf node ℓ in $\mathcal{T}$;
2. Every non-leaf node t of $\mathcal{T}$ is of one of the following type:
   - **Introduce node:** The node t has exactly one child t' in $\mathcal{T}$ and $\beta(t) = \beta(t') \cup \{v\}$, where $v \notin \beta(t')$.

- **Forget node:** The node t has exactly one child t' in $\mathcal{T}$ and $\beta(t) = \beta(t') \setminus \{v\}$, where $v \in \beta(t')$.
- **Join node:** The node t has exactly two children t_1, t_2 in $\mathcal{T}$ and $\beta(t) = \beta(t_1) = \beta(t_2)$.

We rely on the following lemma to compute a nice tree decomposition using the given tree decomposition $(\bar{\mathcal{T}}, \bar{\beta})$.

Lemma 3.6 ([CFK+15, Klo94]). *If a G has a tree decomposition $(\mathcal{T}, \beta)$ of width at most w then there is a nice tree decomposition of G of width at most w. Moreover, given a tree decomposition $(\mathcal{T}, \beta)$ of G of width at most w, in time $\mathcal{O}(w^2 \cdot \max(|V(\mathcal{T})|, |V(G)|))$ we can compute a nice tree decomposition of G of width at most w with at most $\mathcal{O}(w|V(G)|)$ nodes.*

Using Lemma 3.6 we compute a nice tree decomposition $(\mathcal{T}', \beta')$ of G with the root node as r' and width at most w, in time $\mathcal{O}(\mathsf{tw}(G)^2 n)$. We modify the nice tree decomposition $(\mathcal{T}', \beta')$ to obtain even more structured rooted tree decomposition $(\mathcal{T}, \beta)$ with root node $r = r'$ as follows. We let $\mathcal{T} = \mathcal{T}'$, and for $t \in V(\mathcal{T})$, we have $\beta(t) = \beta'(t) \cup S^\star$, *i.e.* $(\mathcal{T}, \beta)$ is obtained from $(\mathcal{T}', \beta')$ by adding all the vertices in $S^\star$ to each bag of $t \in \mathcal{V}(\mathcal{T})$. Note that width of $(\mathcal{T}, \beta)$ is bounded by $w + k \leq 6 \cdot \mathsf{tw}(G) + k$. The purpose of adding all vertices in $S^\star$ to each bag is to ensure the subgraph induced by the subtree rooted at a node contains vertices of all k colors, which simplifies the proof. We note that the notion of introduce node, forget node, and join node naturally extends to the tree decomposition $(\mathcal{T}, \mathcal{X})$.

For a node $t \in V(\mathcal{T})$, by $\mathsf{desc}(t)$ we denote the set of nodes which are descendants of t (including t) in $\mathcal{T}$. Furthermore, for $t \in V(\mathcal{T})$, by G_t we denote the graph $G[V_t]$, where $V_t = \cup_{t' \in \mathsf{desc}(t)} \beta(t')$.

We now move to the description of the entries of the dynamic programming table. Consider a node $t \in V(\mathcal{T})$, and an ordered partition $\mathcal{P} = (P_1, P_2, \ldots P_k)$ of $\beta(t)$ into k sets. We call $\mathcal{P}$ a *valid* ordered partition if and only if for all $i \in [k]$, $c^{-1}(i) \cap \beta(t) \subseteq P_i$. Note that for any valid ordered partition $\mathcal{P} = (P_1, P_2, \ldots, P_k)$, for all $i \in [k]$, we have $P_i \neq \emptyset$. This follows from the fact that $S^\star \subseteq \beta(t)$.

For a valid ordered partition $\mathcal{P} = (P_1, P_2, \ldots P_k)$ of $\beta(t)$, let $\mathcal{H} = \{(H_i, U_i) \mid H_i \uplus U_i = P_i \text{ and } i \in [k]\}$ be a set comprising of ordered pairs, which are partitions of the sets P_is into two sets. A tuple $\tau = (t, \mathcal{P}, \mathcal{H})$ is a *valid tuple* if $\mathcal{P}$ is a valid ordered partition. For a valid tuple $\tau = (t, \mathcal{P}, \mathcal{H})$, a coloring $c_\tau : V(G_t) \to [k]$ is called a *τ-good coloring* if all the following conditions are satisfied.

1. For all $i \in [k]$, we have $P_i \subseteq c_\tau^{-1}(i)$;
2. For all $i \in [k]$, all the vertices in H_i are happy in G_t with respect to c_τ;
3. $c_\tau|_{S \cap V(G_t)} = c|_{S \cap V(G_t)}$.

For every valid tuple $\tau = (t, \mathcal{P}, \mathcal{H})$, we have a table entry denoted by $\Pi(\tau)$ which is set to an element $z \in [|V(G_t)|] \cup \{-\infty\}$. Intuitively, $\Pi(\tau)$ is set to an element $z \in [|V(G_t)|] \cup \{-\infty\}$ which corresponds to the maximum of the number of happy vertices in G_t over all τ-good colorings (if it exists). Formally, the value of $\Pi(\tau)$ is determined as follows.

1. If there is no τ-good coloring of G_t then $\Pi(\tau) = -\infty$.

2. Otherwise, over all τ-good colorings of G_t, $\Pi(\tau)$ is set to the maximum of the number of happy vertices in $V(G_t) \setminus (\cup_{i\in[k]} U_i)$ of G_t over all such colorings.

Let $\mathbb{H}^\star$ be the set comprising of all $\mathcal{H} = \{(H_i, U_i) \mid H_i \uplus U_i = \{v_i^\star\}$ and $i \in [k]\}$. Observe that $\max_{\mathcal{H}\in\mathbb{H}^\star} \Pi(r, \mathcal{S}^\star, \mathcal{H})$ is exactly the number of happy vertices in G maximized over all colorings that extends c to a coloring of $V(G)$. We now move to the description on how the values of $\Pi(\cdot)$ are computed. Since we have a structured form of tree decomposition we compute the value of each of the entries at node $t \in V(\mathcal{T})$ based on the entries of its children, which will be given by the recursive formula. For leaf nodes, we compute the values directly, which corresponds to the base case for the recursive formula. Therefore, by computing the formula in a bottom-up fashion we compute the value of $\Pi(r, \mathcal{S}^\star, \mathcal{H})$, for each $\mathcal{H} \in \mathbb{H}^\star$, and hence the value of $\max_{\mathcal{H}\in\mathbb{H}^\star} \Pi(r, \mathcal{S}^\star, \mathcal{H})$. We now move to the description of computing $\Pi(\tau)$, where $\tau = (t, \mathcal{P} = (P_1, \ldots, P_k), \mathcal{H} = \{(H_i, U_i) \mid H_i \uplus U_i = P_i$ and $i \in [k]\})$ is a valid tuple.

Leaf node. Suppose t is a leaf node. In this case, we have $\beta(t) = S^\star$, and $\mathcal{P} = \mathcal{S}^\star$. Note that in this case there is exactly one τ-good coloring of G_t namely, $c|_{S^\star}$. Moreover, we can find the set of happy vertices H, in G_t with respect to $c|_{S^\star}$ by looking at the adjacencies between the vertices in $S^\star$. If there exist $i \in [k]$ such that $H_i \setminus H \neq \emptyset$ then we set $\Pi(\tau) = -\infty$. Otherwise, we set $\Pi(\tau) = |H \setminus (\cup_{i\in[k]} U_i)|$. The correctness of setting the values as described is justified by the uniqueness of τ-good coloring in G_t.

Introduce node. Suppose t is an introduce node. Let t' be the unique child of t in $\mathcal{T}$, and $\beta(t) = \beta(t') \cup \{\tilde{v}\}$, where $\tilde{v} \notin \beta(t')$. Furthermore, let P_i be the set containing $\tilde{v}$, where $i \in [k]$. Recall that by the properties of tree decomposition, there is no $u \in N_{G_t}(\tilde{v}) \setminus \beta(t)$, *i.e.* all the neighbors of $\tilde{v}$ in G_t are in $\beta(t)$. Let $\mathcal{P}' = (P_1, P_2, \ldots, P_i \setminus \{\tilde{v}\}, \ldots, P_k)$, and $\mathcal{H}' = (\mathcal{H} \setminus \{(H_i, U_i)\}) \cup \{(H_i \setminus \{\tilde{v}\}, U_i \setminus \{\tilde{v}\})\}$. Finally, let $\tau' = (t', \mathcal{P}', \mathcal{H}')$. Note that τ' is a valid tuple. We start by considering the following simple cases where we can immediately set the value of $\Pi(\tau)$.

Case 1 If $\tilde{v} \in H_i$ and there is $j \in [k] \setminus \{i\}$ such that $P_j \cap N_{G_t}(\tilde{v}) \neq \emptyset$ then we set $\Pi(\tau) = -\infty$ since for any τ-good coloring of G_t, $\tilde{v}$ is not a happy vertex.

Case 2 If there is $j \in [k] \setminus \{i\}$ such that $H_j \cap N_{G_t}(\tilde{v}) \neq \emptyset$ then set $\Pi(\tau) = -\infty$. The correctness of this step is justified by the fact that for any τ-good coloring c_τ, a vertex in $H_j \cap N_{G_t}(\tilde{v})$ cannot be happy in G_t with respect to c_τ.

If none of the above cases are applicable then we (recursively) set the value of $\Pi(\tau)$ as follows.

$$\Pi(\tau) = \begin{cases} 1 + \Pi(\tau') & \text{if } \tilde{v} \in H_i; \\ \Pi(\tau') & \text{if } \tilde{v} \in U_i. \end{cases} \tag{3.2}$$

Correctness of Equation 3.2. We prove that Equation 3.2 correctly computes the value of $\Pi(\tau)$ when Case 1 and 2 are not applicable. Consider a τ-good coloring c_τ of G_t that maximizes the number of happy vertices in $V(G_t) \setminus (\cup_{i\in[k]} U_i)$, and let $H \subseteq V(G_t) \setminus (\cup_{i\in[k]} U_i)$ be the set of happy vertices in G_t with respect to c_τ. Also, let

$c_{\tau'} = c_\tau|_{V(G_{t'})}$. Notice that $c_{\tau'}$ is a τ'-good coloring of $G_{t'}$. Since $G_{t'} = G_t - \{\tilde{v}\}$ therefore, all the vertices in $H \setminus \{\tilde{v}\}$ are happy in $G_{t'}$ with respect to $c_{\tau'}$. This implies that $\Pi(\tau') \geq |H \setminus \{\tilde{v}\}|$. If $\tilde{v} \in H_i$ then it must hold that $\tilde{v} \in H$ since c_τ is a τ-good coloring, and hence, we have $\Pi(\tau') \geq \Pi(\tau) - 1$. Otherwise, $\tilde{v} \in U_i$, and therefore $\tilde{v} \notin H$. This implies that $\Pi(\tau') \geq \Pi(\tau)$.

For the other direction consider a τ'-good coloring of $G_{t'}$ that maximizes the number of happy vertices in $V(G_{t'}) \setminus (\cup_{i\in[k]} U_i)$, and let $H \subseteq V(G_{t'}) \setminus (\cup_{i\in[k]} U_i)$ be the set of happy vertices in $G_{t'}$ with respect to $c_{\tau'}$. Let c_τ be a coloring of G_t such that $c_\tau|_{V(G_t \setminus \{v^\star\})} = c_{\tau'}$ and $c_\tau(\tilde{v}) = i$. Since Case 2 is not applicable therefore, all the vertices in $N_{G_t}(\tilde{v}) \cap H$ must belong to P_i. This implies that all the vertices in H are happy in G_t with respect to c_τ. Consider the case when $\tilde{v} \in H_i$. Since Case 1 is not applicable therefore, $N_{G_t}(\tilde{v}) \subseteq P_i$. This implies that for all $u \in N_{G_t}(\tilde{v})$, we have $c_\tau(u) = c_\tau(\tilde{v})$. Therefore, all the vertices in $H \cup \{\tilde{v}\}$ are happy in G_t with respect to c_τ. Moreover, c_τ is a τ-good coloring of G_t. Therefore, in this case we have that $\Pi(\tau) \geq \Pi(\tau') + 1$. Next, consider the case when $\tilde{v} \in U_i$. Observe that c_τ is a τ-good coloring of G_t. This together with the fact that all the vertices in H are happy in G_t with respect to c_τ, implies that $\Pi(\tau) \geq \Pi(\tau')$.

Forget node. Suppose t is a forget node. Let t' be the unique child of t in $\mathcal{T}$ such that $\beta(t) = \beta(t') \setminus \{\tilde{v}\}$, where $\tilde{v} \in \beta(t')$. For $i \in [k]$, let $\mathcal{P}_i = (P_1, P_2, \ldots, P_i \cup \{\tilde{v}\}, \ldots, P_k)$, *i.e.* the ordered partition of $\beta(t') = \beta(t) \cup \{\tilde{v}\}$ obtained from $\mathcal{P}$ by adding $\tilde{v}$ to the set P_i. Furthermore, for the partition (H_i, U_i) of P_i in $\mathcal{H}$ let $\mathcal{H}_{i1} = (\mathcal{H} \setminus \{(H_i, U_i)\}) \cup \{(H_i \cup \{\tilde{v}\}, U_i)\}$ and $\mathcal{H}_{i2} = (\mathcal{H} \setminus \{(H_i, U_i)\}) \cup \{(H_i, U_i \cup \{\tilde{v}\})\}$, *i.e.* $\mathcal{H}_{i1}$ and $\mathcal{H}_{i2}$ are obtained from $\mathcal{H}$ by adding $\tilde{v}$ to the set of happy and unhappy vertices, respectively. If for some $\tilde{i} \in [k]$, we have $\tilde{v} \in c^{-1}(\tilde{i})$ then we let $\mathbb{P} = \{\mathcal{P}_{\tilde{i}}\}$, otherwise we let $\mathbb{P} = \{\mathcal{P}_i \mid i \in [k]\}$. We set the value of $\Pi(\tau)$ as follows.

$$\Pi(\tau) = \max_{\mathcal{P}_i \in \mathbb{P}, j \in [2]} \{\Pi(t', \mathcal{P}_i, \mathcal{H}_{ij})\}. \tag{3.3}$$

Correctness of Equation 3.3. We proof that Equation 3.3 correctly computes the value of $\Pi(\tau)$. Consider a τ-good coloring c_τ of G_t that maximizes the number of happy vertices in $V(G_t) \setminus (\cup_{i\in[k]} U_i)$, and let $H \subseteq V(G_t) \setminus (\cup_{i\in[k]} U_i)$ be the set of happy vertices in G_t with respect to c_τ. Let $\tilde{i} = c_\tau(\tilde{v})$. Furthermore, let $\tilde{j} = 1$ if $\tilde{v} \in H$ and $\tilde{j} = 2$, otherwise. Consider the tuple $\tau' = (t', \mathcal{P}_{\tilde{i}}, \mathcal{H}_{\tilde{i}\tilde{j}})$. By definition of τ' it holds that c_τ is a τ'-good coloring of $G_{t'} = G_t$. Furthermore, all the vertices in H are happy in $G_{t'}$ with respect to c_τ. This implies that $\Pi(\tau) \leq \max_{\mathcal{P}_i \in \mathbb{P}, j \in [2]} \{\Pi(t', \mathcal{P}_i, \mathcal{H}_{ij})\}$.

For the other direction consider $\mathcal{P}_i \in \mathbb{P}$, $j \in [2]$, and let the corresponding tuple be $\tau' = (t', \mathcal{P}_i, \mathcal{H}_{ij})$. Let $c_{\tau'}$ be a τ'-good coloring of $G_{t'}$ that maximizes the number of happy vertices in $V(G_{t'}) \setminus (\cup_{(H'_i, U'_i) \in \mathcal{H}_{ij}} U'_i)$, and let $H \subseteq V(G_{t'}) \setminus (\cup_{(H'_i, U'_i) \in \mathcal{H}_{ij}} U'_i)$ be the set of happy vertices in $G_{t'}$ with respect to $c_{\tau'}$. Since $G_t = G_{t'}$, therefore all the vertices in H are happy in G_t with respect to $c_{\tau'}$. Furthermore, by the definition of sets $\mathbb{P}$, $\mathcal{P}_i$, and $\mathcal{H}_{ij}$ it follows that $c_{\tau'}$ is a τ-good coloring of G_t. Therefore, $\Pi(\tau) \geq \max_{\mathcal{P}_i \in \mathbb{P}, j \in [2]} \{\Pi(t', \mathcal{P}_i, \mathcal{H}_{ij})\}$.

Join node. Suppose t is a join node. Let t_1, t_2 be the two children of t in $\mathcal{T}$. Recall that by the definition of nice tree decomposition we have $\beta(t) = \beta(t_1) = \beta(t_2)$. We set $\Pi(t, \tilde{w}, \mathcal{P}, \mathcal{H})$ as follows.

$$\Pi(t, \tilde{w}, \mathcal{P}, \mathcal{H}) = \Pi(t_1, \tilde{w}, \mathcal{P}, \mathcal{H}) + \Pi(t_2, \tilde{w}, \mathcal{P}, \mathcal{H}) - |\cup_{i\in[k]} H_i| \tag{3.4}$$

Correctness of Equation 3.4. The proof that Equation 3.4 correctly computes $\Pi(\tau)$ follows from the fact that G_{t_1} and G_{t_2} are subgraphs of G_t, and in G_t there is no edge between a vertex in $G_{t_1} - \beta(t)$ and a vertex in $G_{t_2} - \beta(t)$.

This concludes the description and the correctness proof for the recursive formulas for computing the values $\Pi(\cdot)$. We now move to the runtime analysis of the algorithm.

Runtime Analysis. Let $(G, k, S, c : S \to [k])$ be an instance of Maximum Happy Vertices. In time $\mathcal{O}(2^{\mathcal{O}(\mathsf{tw}(G))}n)$, we compute a nice tree decomposition $(\mathcal{T}', \beta')$ of G, with r as the root node, and of width at most $w \leq 6 \cdot \mathsf{tw}(G)$. Furthermore, the number of nodes in $\mathcal{T}$ is bounded by $\mathcal{O}(wn)$. We then obtain a more structured tree decomposition $(\mathcal{T}, \beta)$, by adding $S^\star$ to each bag of β'. For each node in $\mathcal{T}$ we have at most $k^{w+1}2^{k+w+1}$ many table entries. Here, we get a factor of k^{w+1} in the number of table entries instead of k^{k+w+1} because for a node $t \in \mathcal{T}$, we only consider valid ordered partition of X_t, and therefore, we do not guess the set for vertices in $X_t \cap S$. Using the recursive formula we can compute each value of $\Pi(\cdot)$ in time $\mathcal{O}(2^{\mathcal{O}(k+w\log k)}n^{\mathcal{O}(1)})$. At this point of time, we cannot guarantee the runtime which linearly depends on n because we need to check the adjacency among vertices for setting the value of certain entries of the table, which using the straightforward implementation will require quadratic dependence on n. Nonetheless, we can start by computing a data structure for the graph G of treewidth at most w in time $w^{\mathcal{O}(1)}n$ that allows performing adjacency queries in time $\mathcal{O}(w)$ (for instance using [BBL13] or Exercise 7.16 in [CFK+15]). Thus using this data structure we can compute all the entries of the table in time $\mathcal{O}(2^{\mathcal{O}(k+w\log k)}n) \in \mathcal{O}(2^{\mathcal{O}(k+\mathsf{tw}(G)\log k)}n)$, which gives us the desired running time with linear dependence on n.

Theorem 3.3. *Let $(G, k, S, c : S \to [k])$ be an instance of* Maximum Happy Vertices. *Then in time $\mathcal{O}(2^{\mathcal{O}(k+\mathsf{tw}(G)\log k)}n)$ we can find the maximum of the number of happy vertices over all colorings that extent c to a coloring of $V(G)$. Here, n is the number of vertices in G.*

We note here that using the standard backtracking technique together with the fact that we have a partition of vertices into at most k parts which extends c, we can construct a coloring which achieves the maximum number of happy vertices.

3.2.3 FPT algorithm for Subset Rainbow k-Coloring

In this section, we design an FPT algorithm running in time $\mathcal{O}(2^{|S|}n^{\mathcal{O}(1)})$ for Subset Rainbow k-Coloring, when parameterized by $|S|$. The problem Subset Rainbow k-Coloring is formally defined below.

Subset Rainbow k-Coloring	**Parameter:** $\lvert S\rvert$
Input: A graph G and a subset $S \subseteq V(G) \times V(G)$.	
Output: An edge-coloring $c_R : E(G) \to [k]$ such that for every $(u, v) \in S$, there is a rainbow path between u and v in G, if it exists. Otherwise, return *no*.	

Recall that a path P in a graph G with an edge coloring ϕ is said to be a rainbow path if for all $e, e' \in E(P)$ with $e \neq e'$ we have $\phi(e) \neq \phi(e')$. The algorithm we design is based

on the technique of color coding, which was first introduced by Alon et al. [AYZ95]. We first describe a randomized algorithm for Subset Rainbow k-Coloring, which we derandomize using splitters.

The intuition behind the algorithm is as follows. Let (G, S) be an instance of Subset Rainbow k-Coloring on n vertices and m edges. For a solution $c_R : E(G) \to [k]$, to Subset Rainbow k-Coloring in (G, S) the following holds. For each $(u, v) \in S$, there exist a path P from u to v in G with at most k edges such that for all $e, e' \in E(P)$, where $e \neq e'$ we have $c_R(e) \neq c_R(e')$. Therefore, at most $k|S|$ edges in G seem to be "important" for us, *i.e.* if we color at most $k|S|$ edges "nicely" then we would obtain the desired soultion. To capture this, we start by randomly coloring edges in G, hoping that with sufficiently high probability we obtain a coloring that colors the desired set of edges "nicely". Once we have obtained such a "nice" coloring, we employ the algorithm of Kowalik and Lauri [KL16] to check if there is a rainbow path for each $(u, v) \in S$. We note that we use the algorithm given by [KL16] instead of the one by Uchizawa et al. in [UAI+13] because the latter requires exponential space.

Algorithm Rand-SRC**.** Let $c : E(G) \to [k]$ be a coloring of $E(G)$, where each edge is colored with one of the colors in $[k]$ uniformly and independently at random. If for each $(u, v) \in S$, there is rainbow path between u and v in G' with edge coloring c then the algorithm return c as a solution to Subset Rainbow k-Coloring in (G, S). Otherwise, it returns *no*. We note that for a given graph G with edge coloring c, and vertices u and v, in time $2^k n^{\mathcal{O}(1)}$ time we can check if there is a rainbow path between u and v in G' by using the algorithm given by Corollary 5 in [KL16]. This completes the description of the algorithm.

We now proceed to show how we can obtain an algorithm with constant success probability.

Theorem 3.4. *There is an algorithm that, given an instance (G, S) of* Subset Rainbow k-Coloring*, in time $2^{\mathcal{O}(|S|k \log k)} n^{\mathcal{O}(1)}$ either returns no or outputs a solution to* Subset Rainbow k-Coloring *in (G, S). Moreover, if the input is a yes instance of* Subset Rainbow k-Coloring*, then it returns a solution with constant probability.*

Proof. We start by showing that Rand-SRC runs in time $2^k n^{\mathcal{O}(1)}$, and given a *yes* instance of Subset Rainbow k-Coloring, outputs a solution with probability at least $2^{-\mathcal{O}(|S|k \log k)}$. Clearly, by repeating Rand-SRC $2^{\mathcal{O}(|S|k \log k)}$ times, we obtain the desired success probability and running time.

The algorithm Rand-SRC starts by coloring edges in G' uniformly and independently at random to obtain a coloring $c : E(G') \to [k]$. This step can be executed in time $\mathcal{O}(m)$. Then, for each pair $(u, v) \in S$, in time $2^k n^{\mathcal{O}(1)}$ it checks if there is a rainbow path between u and v in G for the edge coloring c. If for every pair in S it find a rainbow path between them, it correctly outputs a solution. The correctness and the running time bound of this step relies on the correctness of Corollary 5 of [KL16]. Otherwise, Rand-SRC outputs *no*. Therefore, we have the desired running time bound.

Towards proving the desired success probability, assume that (G, S) is a *yes* instance of Subset Rainbow k-Coloring, and c_R be one of its solution. Moreover, for a pair $(u, v) \in S$ let P_{uv} be a rainbow path in G. Here, if there are many such paths then we arbitrarily choose one of them. Note that for each $(u, v) \in S$ we have $|E(P)| \leq k$.

Consider the set $E_R = \cup_{(u,v)\in S} E(P_{uv})$. We now show that the probability with which $c|_{E_R} = c_R|_{E_R}$ is at least $2^{-\mathcal{O}(|S|k\log k)}$. Notice that there are $k^{|E(G)|}$ many distinct colourings of edges in G. Moreover, at least $k^{|E(G)|-k|S|}$ of these colorings satisfy the desired property (agree with c_R on edges in E_R). Thus, we obtain the desired success probability bound. □

We start by defining some terminologies which will be useful in derandomization of our algorithm (see [CFK+15, NSS95]). An (n, p, ℓ)-*splitter* $\mathcal{F}$, is a family of functions from $[n]$ to ℓ such that for every $S \subseteq [n]$ of size at most p there is a function $f \in \mathcal{F}$ such that f splits S evenly. That is, for all $i, j \in [\ell]$, $|f^{-1}(i)|$ and $|f^{-1}(j)|$ differs by at most 1. Observe that when $\ell \geq p$ then for any $S \subseteq [n]$ of size at most p and a function $f \in \mathcal{F}$ that splits S, we have $|f^{-1}(i) \cap S| \leq 1$, for all $i \in [\ell]$. An (n, ℓ, ℓ)-splitter is called as an (n, ℓ)-*perfect hash family*. Moreover, for any $\ell \geq 1$, we can construct an (n, ℓ)-perfect hash family of size $e^{\ell}\ell^{\mathcal{O}(\log \ell)} \log n$ in time $e^{\ell}\ell^{\mathcal{O}(\log \ell)} n \log n$ [NSS95].

We next move to the description of derandomization of the algorithm presented in Theorem 3.4. For the sake of simplicity in explanation, we associate each $e \in E(G)$ with a unique integer, say i_e in $[m]$, and whenever we refer to e as an integer, we actually refer to the integer i_e. We start by computing an $(m, k|S|)$-perfect hash family $\mathcal{F}$ of size $e^{k|S|}(k|S|)^{\mathcal{O}(\log k|S|)} \log m$ in time $e^{k|S|}(k|S|)^{\mathcal{O}(\log k|S|)} m \log m$ using the algorithm of Naor et al. in [NSS95]. We will create a family of function $\mathcal{F}'$ from $[m]$ to $[k]$ of size $e^{k|S|}(k|S|)^{\mathcal{O}(\log k|S|)} k^{k|S|} \log m$. Towards this, consider an $f \in \mathcal{F}$ and a partition $\mathcal{P} = \{P_1, P_2, \ldots P_{k'}\}$ of $[k|S|]$ into k' sets, where $k' \leq k$. We let $f_{\mathcal{P}}$ to be the function obtained from f as follows. For each $i \in [k']$ we have $f_{\mathcal{P}}^{-1}(i) = \cup_{x\in P_i} f^{-1}(x)$. For every such pair f and $\mathcal{P}$, we add the function $f_{\mathcal{P}}$ to the set $\mathcal{F}'$. We will call such an $\mathcal{F}'$ as $(m, k|S|, k)$-*unified perfect hash* family. Observe that $\mathcal{F}'$ has size at most $e^{k|S|}(k|S|)^{\mathcal{O}(\log k|S|)} k^{k|S|} \log m$. We now describe the derandomized algorithm SRC, which is a result of derandomization of Rand-SRC.

Algorithm SRC. Given an instance (G, S) of Subset Rainbow k-Coloring, the algorithm start by computing an $(m, k|S|, k)$-unified perfect hash family $\mathcal{F}'$. If there exists $c : E(G) \to [k]$, where $c \in \mathcal{F}'$ such that for each $(u, v) \in S$, there is rainbow path between u and v in G' with the edge coloring c then we return c as a solution to Subset Rainbow k-Coloring in (G, S). Otherwise, we return that (G, S) is a *no* instance of Subset Rainbow k-Coloring. We note that for a given graph G with edge coloring c, and vertices u and v, in time $2^k n^{\mathcal{O}(1)}$ time we can check if there is a rainbow path between u and v in G' by using the algorithm given by Corollary 5 in [KL16]. This completes the description of the algorithm.

Lemma 3.7. *Given an instance (G, k) of* Subset Rainbow k-Coloring*, the algorithm* SRC *either correctly reports that (G, k) is a no instance of* Subset Rainbow k-Coloring *or returns a solution to* Subset Rainbow k-Coloring *in (G, S). Moreover,* SRC *runs in time $2^{\mathcal{O}(|S|)} n^{\mathcal{O}(1)}$, for every fixed k. Here, $n = |V(G)|$.*

Proof. Suppose (G, k) is a *yes* instance of Subset Rainbow k-Coloring, and let $c_F : E(G) \to [k]$ be one of its solution. For $(u, v) \in S$, let P_{uv} be a rainbow path in G'. Furthermore, let $E_R = \cup_{(u,v)\in S} E(P_{uv})$. If $|E_R| < k|S|$, we arbitrarily add edges in G to E_R to make its size exactly $k|S|$. Since $|E_R| \leq k|S|$, there exists $f \in \mathcal{F}$ that

splits E_F. Moreover, for each $i \in [k|S|]$, we have $|f^{-1}(i) \cap E_F| \leq 1$. For $i \in [k]$, let $P_i = \{f(e) \mid e \in E_F \text{ and } c_F(e) = i\}$, and $\mathcal{P}' = \{P_i \mid i \in [k]\}$. Notice that $\mathcal{P} = \mathcal{P}' \setminus \{\emptyset\}$ is a partition of $[k|S|]$ into at most k parts. Therefore, the function $f_{\mathcal{P}} \in \mathcal{F}'$. Moreover, $f_{\mathcal{P}}|_{E_F} = c_F|_{E_F}$. The algorithm SRC checks for each $c \in \mathcal{F}'$ whether c is a solution to Subset Rainbow k-Coloring in (G, S). In particular, it checks if $f_{\mathcal{P}}$ is a solution to Subset Rainbow k-Coloring in (G, S). The correctness of this checking is given by Corollary 5 of [KL16]. Therefore, SRC correctly concludes that (G, S) is a *yes* instance of Subset Rainbow k-Coloring, and outputs a correct solution.

Given an instance (G, k) of Subset Rainbow k-Coloring, whenever it returns a solution then indeed (G, k) is a *yes* instance of Subset Rainbow k-Coloring. This is implied from Corollary 5 of [KL16].

Next, we move to the runtime analysis. The algorithm starts by computing an $(m, k|S|, k)$-unified perfect hash family $\mathcal{F}'$ of size $e^{k|S|}(k|S|)^{\mathcal{O}(\log k|S|)}k^{k|S|}\log m$ in time $e^{k|S|}(k|S|)^{\mathcal{O}(\log k|S|)}\ k^{k|S|}\ m \log m$. Then, for each $c \in \mathcal{F}'$ it checks if for all $(u, v) \in S$, there is a rainbow path between then in G with edge coloring c in time $2^k n^{\mathcal{O}(1)}$. If it finds such a c then returns it as a solution. Otherwise, correctly reports *no*. Therefore, the running time of the algorithm is bounded by $2^{\mathcal{O}(S)}n^{\mathcal{O}(1)}$, for every fixed k. Here, we rely on the fact that $\log |S| \in o(\sqrt{|S|})$. □

Theorem 3.5. Steiner Rainbow k-Coloring *admits an algorithm running in time* $2^{\mathcal{O}(|S|^2)}n^{\mathcal{O}(1)}$.

Proof. Follows from Lemma 3.7. □

3.3 Slice-wise Polynomial

Some parameterized problems do not belong to the complexity class FPT under reasonable Complexity Theoretic assumptions. An important class of parameterized problems, which is strictly larger than FPT is the complexity class XP, which is defined below.

Definition 3.4. A parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is called *slice-wise polynomial* if there is an algorithm $\mathcal{A}$ and two computable (non-decreasing) functions $f, g : \mathbb{N} \to \mathbb{N}$ such that, given an instance $I = (x, k)$ of Π, the algorithm $\mathcal{A}$ correctly decides whether or not $(x, k) \in \Pi$ in time bounded by $f(k) \cdot |I|^{g(k)}$. The complexity class containing all slice-wise polynomial problems is called XP.

Next, we consider the example of Happy Vertex Coloring, when parameterized by the number of happy vertices, ℓ, and show that it is slice-wise polynomial.

3.3.1 XP algorithm for Happy Vertex Coloring

In this section, we consider the problem Happy Vertex Coloring. Consider a graph G and a coloring $c : V(G) \to [k]$. Recall that a vertex $u \in V(G)$ is happy in G with respect to c if for all $v \in N(u)$, we have $c(u) = c(v)$, *i.e.* all the neighbors of u have color same as that of u. Furthermore, a vertex which is not happy in G with respect to c is unhappy. Next, we formally define the problem Happy Vertex Coloring.

Happy Vertex Coloring	**Parameter:** k, ℓ
Input: A graph G, integers k and ℓ, a vertex subset $S \subseteq V(G)$, and a (partial) coloring $c : S \to [k]$.	
Question: Does there exist a coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}\|_S = c$ and the number of happy vertices in G with respect to $\tilde{c}$ is at least ℓ?	

We show that Happy Vertex Coloring is slice-wise polynomial, when parameterized by ℓ. Let $(G, k, \ell, S, c : S \to [k])$ be an instance of Happy Vertex Coloring, where G is a graph on n vertices. We start by guessing the set of happy vertices in the coloring that we are looking for. Note that there are $n^{\mathcal{O}(\ell)}$ subsets of $V(G)$, which are of size at least ℓ. For each $H \subseteq V(G)$ of size at least ℓ, we try find a coloring $\tilde{c}$ such that $\tilde{c}|_S = c$ and each $h \in H$ is happy with respect to $\tilde{c}$. Towards this, for each $H \subseteq V(G)$ of size at least ℓ, we solve the following problem, which we call Revealed Happy Vertex Coloring (RHVC).

Revealed Happy Vertex Coloring (RHVC)
Input: A graph G, an integer k, vertex subsets $S, H \subseteq V(G)$, and a (partial) coloring $c : S \to [k]$.
Question: Does there exist a coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}\|_S = c$ and each $h \in H$ is happy with respect to $\tilde{c}$?

We design a polynomial time algorithm for RHVC. Using this algorithm, we obtain an algorithm forHappy Vertex Coloring, running in time $n^{\mathcal{O}(\ell)}$ as follows. For each $H \subseteq V(G)$ of size at least ℓ we check whether or not (G, k, S, H, c) is a *yes* instance of RHVC. If for any such H we have that (G, k, S, H, c) is a *yes* of RHVC, then clearly (G, k, ℓ, S, c) is a *yes* instance of Happy Vertex Coloring. Also, if for all $H \subseteq V(G)$ of size at least ℓ, (G, k, S, H, c) is a *no* instance of RHVC, then (G, k, ℓ, S, c) is a *no* instance of Happy Vertex Coloring.

Next, we focus on designing a polynomial time algorithm for RHVC. We give the algorithm Alg-RHVC (Algorithm 1) for RHVC.

Correctness and Runtime Analysis. Let $I = (G, k, S, H, c)$ be an instance of RHVC. We prove the correctness of Alg-RHVC by induction on $|H|$. The base case of is when $|H| = 0$. In this case, using Step 1, we correctly conclude that I is a *yes* instance of RHVC. Next, for the induction hypothesis we assume that the algorithm correctly decides an instance whenever $|H| \leq t$, where $t \in \mathbb{N}$. We now show that the algorithm correctly decides an instance whenever $|H| = t + 1$. If Step 1 or 3 are applicable, then we correctly decide whether or not I is a *yes* instance. Hereafter, we assume Step 1 and 3 are not applicable. This implies that $N[H] \cap S \neq \emptyset$ and for each $h \in H$, and $u, v \in N[h] \cap S$ we have $c(u) = c(v)$. Let $h \in H$ such that $N[h] \cap S \neq \emptyset$. Since $N[h] \cap S \neq \emptyset$, therefore one of Step 5 or Step 8 is applicable. In either case, we obtain a coloring $\hat{c}$ such that any coloring $\tilde{c}$ that extends $\hat{c}$ to a coloring of $V(G)$, the vertex h is happy with respect to $\tilde{c}$. From the above discussion it is safe to remove v from H, and solve the instance $I' = (G, k, S \cup N[h], H \setminus \{h\}, \hat{c})$, which has strictly smaller H. But then, by induction hypothesis we correctly decide whether or not I' is a *yes* instance of RHVC. This concludes the proof of correctness.

Next, we move to the runtime analysis of the algorithm. The measure that we use to analyze the running time is $|H|$. Observe that at each step of the algorithm either we completely resolve the instance in polynomial time, or spend polynomial time and

Algorithm 1: Alg-RHVC

Input: A graph G, an integer k, vertex subsets $S, H \subseteq V(G)$, and a (partial) coloring $c : S \to [k]$.

Output: *yes* or *no*.

1 **if** $N[H] \cap S = \emptyset$ **then**
2 | **return** *yes*
3 **else if** *for some* $h \in H$ *we have* $u, v \in N[h] \cap S$ *such that* $c(u) \neq c(v)$ **then**
4 | **return** *no*
// Hereafter, we assume that $N[H] \cap S \neq \emptyset$ and for each $h \in H$, and $u, v \in N[h] \cap S$ we have $c(u) = c(v)$.
5 **else if** *there is* $h \in H \cap S$ **then**
6 | Let $\tilde{c} : S \cup N(h) \to [k]$ such that $\tilde{c}|_S = c$ and for all $u \in N(h)$ we have $\tilde{c}(u) = c(h)$;
7 | **return** $(G, k, S \cup N(h), H \setminus \{h\}, \tilde{c})$
8 **else**
9 | Let $h \in H$ such that there is $v \in N(h) \cap S$. Furthermore, let $\tilde{c} : S \cup N[h] \to [k]$ such that $\tilde{c}|_S = c$ and for all $u \in N[h]$ we have $\tilde{c}(u) = c(v)$;
10 | **return** $(G, k, S \cup N[h], H \setminus \{h\}, \tilde{c})$

make a recursive call to an instance which has strictly smaller $|H|$. Also, $|H| \leq n$, and if $H = \emptyset$ then we (correctly) conclude that it is a *yes* instance of RHVC (Step 1). The above discussion implies that the algorithm runs in time polynomial in the size of the input graph.

3.4 Kernelization

Another central notion in Parameterized Complexity is *kernelization*, which mathematically captures the efficiency of a pre-processing/ data reduction algorithm. Towards formally defining the notion of kernelization, we introduce the following definitions.

Let $\Pi \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that instances I, I' of Π are *equivalent* if $I \in \Pi$ if and only if $I' \in \Pi$. A *data reduction rule*, or simply, *reduction rule*, for the parameterized problem Π is a function $\tau : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$, which maps an instance I of Π to an equivalent instance I' of Π. Here, $\tau(\cdot)$ is a function that is computable in time polynomial in the size of the input instance. The property of a reduction rule of translating an instance of Π to another equivalent instance of Π is referred to as *reduction rule being safe* or *safeness of the reduction rule*.

A *pre-processing* or *data reduction* algorithm consecutively (and exhaustively) applies a set of reduction rules (in some order) to output a "smaller" instance. Formally, a *pre-processing* algorithm for a parameterized problem Π takes as an input an instance (x, k) of Π, and in time polynomial in the size of the input outputs an equivalent instance (x', k') of Π. To mathematically capture the notion of "smallness" of the output instance, we employ the parameter associated with the input instance of the parameterized problem. Towards this, we need the following definition. The *output size* of a pre-processing algorithm $\mathcal{A}$ is a function $\mathsf{size}_{\mathcal{A}} : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$, which is defined as follows.

$$\mathsf{size}_{\mathcal{A}}(k) = \mathsf{sup}\{|x'| + k' \mid (x', k') = \mathcal{A}(x, k), x \in \Sigma^*\}$$

In other words, we look at all possible instances of Π with parameter k, and take the supremum of the sizes of the output given by $\mathcal{A}$ on them. We note that this supremum can be infinity, which is the case when the output size is not bounded by a function of the parameter alone. Now we are ready to state the definition of *kernelization algorithm* or *kernels*.

Definition 3.5. A *kernelization algorithm* or a *kernel* for a parameterized problem Π is an algorithm $\mathcal{K}$, which takes as an input an instance (x, k) of Π, and in time polynomial in $|x| + k$ outputs an equivalent instance (x', k') of Π. Moreover, there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $\mathsf{size}_{\mathcal{K}}(k) \leq g(k)$.

Depending on whether $g(\cdot)$ is a polynomial, linear or exponential function of k, the problem is said to admit a polynomial, linear or exponential kernel. It is known that a (decidable) parameterized problem is FPT if and only if it admits a kernel, which we prove in Lemma 3.8.

Lemma 3.8. *A decidable parameterized problem Π is in FPT if and only if it admits a kernel.*

Proof. In the forward direction, let $\mathcal{A}$ be an FPT algorithm for Π, which runs in time $\mathcal{O}(f(k)|x|^c)$ for an instance (x, k) of Π. Here, $f(\cdot)$ is some computable function and c is a constant. Next, we show that Π admits a kernelization algorithm $\mathcal{K}$. Let (x, k) be an instance of Π. The kernelization algorithm $\mathcal{K}$ runs the algorithm $\mathcal{A}$ on the input (x, k) for $|x|^{c+1}$ steps. If within these many steps $\mathcal{A}$ return whether or not $(x, k) \in \Pi$ then we output a trivial *yes* or *no* kernel, which is of constant size. Otherwise, the algorithm $\mathcal{K}$ outputs (x, k). Observe that since $\mathcal{A}$ was unable to decide the instance in time $|x|^{c+1}$, therefore we have $|x|^{c+1} < f(k) \cdot |x|^c$. This implies that $|x| + k \leq f(k) + k$. This concludes the proof of forward direction.

In the reverse direction, let $\mathcal{K}$ be a kernelization algorithm for Π of size $g(k)$, and $\mathcal{D}$ be a (finite time) algorithm for the problem Π. For a given instance (x, k) of Π, we use the algorithm $\mathcal{K}$ on it to obtain an instance (x', k') such that $|x'| + k' \leq g(k)$. Then, we run the algorithm $\mathcal{D}$ with input (x', k'). Since $\mathcal{D}$ is a finite time algorithm, therefore the time it requires to decide (x', k') is bounded by $f(|x'| + k')$, where $f(\cdot)$ is some computable function. □

Due to Lemma 3.8, it is more interesting to talk about polynomial (or linear) kernels. Hereafter, whenever we talk about kernels, we refer only to the polynomial (or linear) kernels. Next, we look at some examples of designing kernelization algorithm. In Section 3.4.1, we design a kernel for the problem of Happy Vertex Coloring, when parameterized by $k + \ell$. In Section 3.4.2, we design a kernel for the problem Cluster Vertex Deletion, which illustrates the use of Expansion Lemma in designing kernelization algorithms.

3.4.1 Kernel for Happy Vertex Coloring

We give a polynomial kernel for the problem Happy Vertex Coloring, when parameterized by the number of happy vertices and the number of colors used in the coloring.

In fact, we give a kernel for an annotated version of Happy Vertex Coloring, which we call Annotated Happy Vertex Coloring (AHVC). The problem is formally defined below.

Annotated Happy Vertex Coloring (AHVC) **Parameter:** $k + \ell$
Input: A graph G, integers k and ℓ, a vertex subsets $S, U \subseteq V(G)$, a (partial) coloring $c : S \to [k]$.
Question: Does there exist a coloring $\tilde{c} : V(G) \to [k]$ such that $\tilde{c}\|_S = c$ and $\|H \setminus U\| \geq \ell$, where H is the set of happy vertices in G with respect to $\tilde{c}$?

Observe that Happy Vertex Coloring is a special case of AHVC, where $U = \emptyset$. Moreover, given an instance $(G, k, \ell, S, U, c : S \to [k])$ of AHVC, in polynomial time we can construct an instance $(G', k', \ell, S', c' : S' \to [k'])$ of Happy Vertex Coloring such that $|V(G')| \in \mathcal{O}(|V(G)|)$, $k' \in \mathcal{O}(k)$, and $|S'| \in \mathcal{O}(|S|)$ as follows. Initially, we have $G' = G$ and $c' = c$. We add two (new) vertices $u^\star, v^\star$ to $V(G')$, add the edge $(u^\star, v^\star)$ to $E(G')$, add $u^\star, v^\star$ to S, and set $c'(u^\star) = k + 1$ and $c'(v^\star) = k + 2$. Furthermore, we add the edges $\{(u, u^\star), (u, v^\star) \mid u \in U\}$ to $E(G')$ and set $k' = k + 2$. It is easy to see that $(G, k, \ell, S, U, c : S \to [k])$ is a *yes* instance of AHVC if and only if $(G', k', \ell, S', c' : S' \to [k'])$ is a *yes* instance of Happy Vertex Coloring. Therefore, to design a kernel for Happy Vertex Coloring with $\mathcal{O}(k^2\ell^2)$ vertices it is enough to design a kernel for AHVC with $\mathcal{O}(k^2\ell^2)$ vertices. Hereafter, the focus will be to design a kernel with $\mathcal{O}(k^2\ell^2)$ vertices for AHVC.

Let $(G, k, \ell, S, U, c : S \to [k])$ be an instance of AHVC. The kernelization algorithm applies the following reduction rules in the order in which it is stated. Furthermore, at each step we assume that none of the preceding reduction rules are applicable. When none of the reduction rules are applicable we argue that we get a kernel of the desired size.

Reduction Rule 3.7. *If $\ell \leq 0$, then return that $(G, k, \ell, S, U, c : S \to [k])$ is a yes instance of* AHVC.

Observe that if $\ell \leq 0$, then any coloring that extends c to a coloring of $V(G)$ is a valid solution to the instance $(G, k, \ell, S, U, c : S \to [k])$ of AHVC, which implies that Reduction Rule 3.7 is safe.

Reduction Rule 3.8. *Let $v \in V(G) \setminus U$ be a vertex such that $N(v) \subseteq S$, for all $u, u' \in N(v)$ we have $c(u) = c(u')$, and one of the following conditions is satisfied. i) $v \notin S$; or ii) $c(v) = c(u)$, where $u \in N(v)$. Then delete v from G and decrease ℓ by one. The resulting instance is $(G - \{v\}, k, \ell - 1, S \setminus \{v\}, U, c|_{S \setminus \{v\}} : S \setminus \{v\} \to [k])$.*

Lemma 3.9. *Reduction Rule 3.8 is safe.*

Proof. Let $(G, k, \ell, S, U, c : S \to [k])$ be an instance of AHVC and $v \in V(G) \setminus U$ be a vertex such that $N(v) \subseteq S$, and for all $u, u' \in N(v)$ we have $c(u) = c(u')$. Furthermore, let $(G', k, \ell - 1, S', U, c' : S' \to [k], \ell - 1)$ be the instance resulting after application of the Reduction Rule 3.8, where $G' = G - \{v\}$, $S' = S \setminus \{v\}$, and $c' = c|_{S'}$. The proof of forward direction follows from the fact that $G' = G[V(G) \setminus \{v\}]$. In the reverse direction let $\tilde{c}'$ be a coloring that extends $c|_{S'}$ to a coloring of $V(G')$ such that the number of happy vertices in $V(G') \setminus U$ is at least $\ell - 1$. If $v \in S$, then the coloring $\tilde{c}$ obtained by extending $\tilde{c}'$ to a coloring of $V(G)$ with $\tilde{c}(v) = c(v)$ is a coloring that extends c to a

coloring of $V(G)$ with at least ℓ happy vertices in $V(G) \setminus U$ in G. Otherwise, we have $v \notin S$. In this case, let $\tilde{c}$ be the coloring obtained by extending $\tilde{c}'$ to a coloring of $V(G)$ with $\tilde{c}(v) = c(u)$, where $u \in N_G(v)$. Observe that $\tilde{c}$ is a coloring that extends c to a coloring of $V(G)$, and there are at least ℓ happy vertices in $V(G) \setminus U$ in G with respect to $\tilde{c}$. This concludes the proof. □

Reduction Rule 3.9. *Let $v \in S \setminus U$ be a vertex such that there exists $u \in N(v) \cap S$ with $c(v) \neq c(u)$. Then add v to the set U. The resulting instance is $(G, k, \ell, S, U \cup \{v\}, c : S \to [k])$.*

The safeness of Reduction Rule 3.9 follows from the fact that a vertex $v \in S \setminus U$ with $u \in N(v) \cap S$ such that $c(v) \neq c(u)$ can never be a happy vertex in any coloring of G that extends c to a coloring of $V(G)$.

Reduction Rule 3.10. *Let $v \in V(G) \setminus U$ be a vertex such that there exists $u, u' \in N(v) \cap S$ with $c(u) \neq c(u')$. Then add v to the set U. The resulting instance is $(G, k, \ell, S, U \cup \{v\}, c : S \to [k])$.*

The safeness of Reduction Rule 3.10 follows from the fact that a vertex v with $u, u' \in N(v) \cap S$ such that $c(u) \neq c(u')$ can never be a happy vertex in any coloring of G that extends c to a coloring of $V(G)$.

Next we consider the following sets. For $i \in [k]$, let $U_i = \{v \in U \cap S \mid c(v) = i\}$, and $U_R = U \setminus (\cup_{i \in [k]} U_i)$. We proceed with the following reduction rules.

Reduction Rule 3.11. *If there exists $i \in [k]$ such that there are distinct $u, v \in U_i$ then unify u, v in G to obtain the graph G' with $u^\star$ being the vertex resulting after unification. Furthermore, let $c' : (S \setminus \{u, v\}) \cup \{u^\star\} \to [k]$ be the coloring obtained from c with $c'|_{S \setminus \{u,v\}} = c$ and $c'(u^\star) = c(u)$. The resulting instance is $(G', k, \ell, (S \setminus \{u, v\}) \cup \{u^\star\}, (U \setminus \{u, v\}) \cup \{u^\star\}, c' : (S \setminus \{u, v\}) \cup \{u^\star\} \to [k])$.*

Lemma 3.10. *Reduction Rule 3.11 is safe.*

Proof. Let $i \in [k]$ such that there are distinct $u, v \in U_i$, and G' be the graph obtained from G after unifying u and v with $u^\star$ being the resulting vertex after unification. Furthermore, let $U' = (U \setminus \{u, v\}) \cup \{u^\star\}$, $S' = (S \setminus \{u, v\}) \cup \{u^\star\}$, and $c' : S' \to [k]$ be the coloring obtained from c with $c'|_{S' \setminus \{u^\star\}} = c|_{S \setminus \{u,v\}}$ and $c'(u^\star) = c(u)$. We will show that $(G, k, \ell, S, U, c : S \to [k])$ is a *yes* instance of AHVC if and only if $(G', k, \ell, S', U', c' : S' \to [k])$ is a *yes* instance of AHVC.

In the forward direction let $(G, k, \ell, S, U, c : S \to [k])$ be a *yes* instance of AHVC, $\tilde{c}$ be one of its solutions, and $H \subseteq V(G) \setminus U$ be the set of happy vertices in G with respect to $\tilde{c}$. Notice that we have $|H| \geq \ell$. Let $\tilde{c}' : V(G') \to [k]$ be the coloring obtained from $\tilde{c}$ with $\tilde{c}'(u^\star) = \tilde{c}(u)$ and $\tilde{c}'|_{V(G') \setminus \{u^\star\}} = \tilde{c}|_{V(G) \setminus \{u,v\}}$. Recall that $V(G') \setminus \{u^\star\} = V(G) \setminus \{u, v\}$, hence $\tilde{c}'$ is a coloring of G'. Furthermore, we have $\tilde{c}'|_{S'} = c'$. Hence, we only need to show that with respect to $\tilde{c}'$ in G' we have at least ℓ vertices in $V(G') \setminus U'$ that are happy. Observe that $H \subseteq V(G') \setminus U'$. We claim that all the vertices in H are happy in G' with respect to $\tilde{c}'$, which is enough to show that $(G', k, \ell, S', U', c' : S' \to [k])$ is a *yes* instance of AHVC. Consider a vertex $h \in H$. Recall that $N_{G'}(h) \setminus U' = N_G(h) \setminus U$. This together with the construction of $\tilde{c}'$ implies that for all $w, w' \in N_{G'}(h) \setminus U'$ we have $\tilde{c}'(w) = \tilde{c}'(w') = \tilde{c}'(h)$. For $w \in N_{G'}(h) \cap U'$ if $\tilde{c}'(w) \neq \tilde{c}'(h)$ then consider the

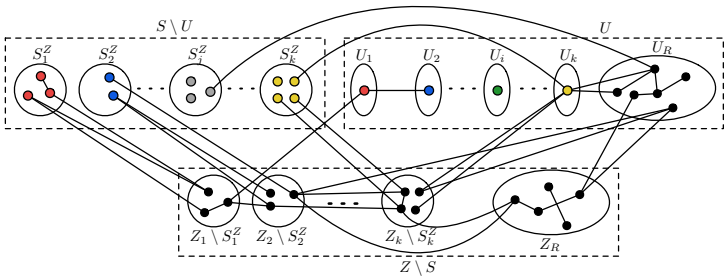

Figure 3.1: An illustration of partition of $V(G)$ into various sets.

following cases. If $w = u^\star$ then replacing w by u (or v) in G we have that $\tilde{c}(h) \neq \tilde{c}(w)$, contradicting that h is a happy vertex in G with respect to $\tilde{c}$. On the other hand if $w \neq u^\star$ then $w \in U$ and in G we have that $\tilde{c}(h) \neq \tilde{c}(w)$, a contradiction. This concludes the proof in the forward direction.

In the reverse direction let $(G', k, \ell, S', U', c' : S' \to [k])$ be a *yes* instance of AHVC, $\tilde{c}'$ be one of its solutions, and $H \subseteq V(G') \setminus U'$ be the set of happy vertices in G' with respect to $\tilde{c}'$. Notice that $|H| \geq \ell$. Let $\tilde{c} : V(G) \to [k]$ be the coloring obtained from $\tilde{c}'$ with $\tilde{c}(u) = \tilde{c}(v) = \tilde{c}'(u^\star)$ and $\tilde{c}|_{V(G)\setminus\{u,v\}} = \tilde{c}'|_{V(G')\setminus\{u^\star\}}$. Observe that we have $\tilde{c}|_S = c$. Hence, we only need to show that with respect to $\tilde{c}$ in G we have at least ℓ vertices in $V(G)\setminus U$ that are happy. Observe that $H \subseteq V(G)\setminus U$. We claim that all vertices in H are happy in G with respect to $\tilde{c}$, which is enough to show that $(G, k, \ell, S, U, c : S \to [k])$ is a *yes* instance of AHVC. Consider a vertex $h \in H$. Recall that $N_G(h) \setminus U = N_{G'}(h) \setminus U'$. This together with the construction of $\tilde{c}$ implies that for all $w \in N_G(h) \setminus U$ we have $\tilde{c}(w) = \tilde{c}(h)$. For $w \in N_G(h) \cap U$ if $\tilde{c}(w) \neq \tilde{c}(h)$ then consider the following cases. If $w \in \{u, v\}$ then replacing w by $u^\star$ in G' we have that $\tilde{c}'(h) \neq \tilde{c}'(w)$, contradicting that h is a happy vertex in G' with respect to $\tilde{c}'$. On the other hand if $w \notin \{u, v\}$ then $w \in U'$ and in G' we have that $\tilde{c}'(h) \neq \tilde{c}'(w)$, a contradiction. □

Hereafter, we assume that Reduction Rule 3.11 is not applicable and hence for each $i \in [k]$, we have $|U_i| \leq 1$. Let $Z = V(G) \setminus U$. For $i \in [k]$, let $S_i^Z = \{v \in S \cap Z \mid c(v) = i\}$ and $Z_i = (Z \cap N(U_i \cup S_i^Z)) \cup S_i^Z$ (see Figure 3.1). Furthermore, we let $Z_R = Z \setminus (\cup_{i\in[k]} Z_i)$. Observe that for $i, j \in [k]$, $i \neq j$ we have $Z_i \cap Z_j = \emptyset$ since Reduction Rule 3.9 and 3.10 are not applicable. Also, for each $v \in Z_R$, we have $N(v) \subseteq V(G) \setminus S$. We proceed with the following reduction rules.

Reduction Rule 3.12. *If there exists $i \in [k]$ such that $|Z_i| \geq \ell$ then return that $(G, k, \ell, S, U, c : S \to [k])$ is a yes instance of* AHVC.

Lemma 3.11. *Reduction Rule 3.12 is safe.*

Proof. Let $(G, k, \ell, S, U, c : S \to [k])$ be an instance of AHVC, and $i \in [k]$ such that $|Z_i| \geq \ell$. Consider the coloring $\tilde{c} : V(G) \to [k]$ with $\tilde{c}|_S = c$, and for all $v \in V(G) \setminus S$, $\tilde{c}(u) = i$. We claim that $\tilde{c}$ is a solution to AHVC in $(G, k, \ell, S, U, c : S \to [k])$, where all the vertices in Z_i are happy with respect to $\tilde{c}$. Note that by definition $\tilde{c}$ extends c to a coloring of G therefore, we only need to show that vertices in Z_i are happy in G with respect to $\tilde{c}$. Consider a vertex $z \in Z_i$ and a vertex $v \in N(z)$. By the definition of $\tilde{c}$ and Z_i, we have $\tilde{c}(z) = i$. If $v \in S$ then we have $\tilde{c}(v) = c(v) = i$, since Reduction Rule 3.9

and 3.10 are not applicable. If $v \in V(G) \setminus S$ then by definition of $\tilde{c}$ we have $\tilde{c}(v) = i$. Therefore, z is a happy vertex in G with respect to $\tilde{c}$. This concludes the proof. □

Reduction Rule 3.13. *If $|Z_R| \geq \ell$ then return that $(G, k, \ell, S, U, c : S \to [k])$ is a yes instance of* AHVC*.*

Lemma 3.12. *Reduction Rule 3.13 is safe.*

Proof. Let $(G, k, \ell, S, U, c : S \to [k])$ be an instance of AHVC such that $|Z_R| \geq \ell$. Consider the coloring $\tilde{c} : V(G) \to [k]$ with $\tilde{c}|_S = c$, and for all $v \in V(G) \setminus S$, $\tilde{c}(u) = 1$. We claim that $\tilde{c}$ is solution to AHVC in $(G, k, \ell, S, U, c : S \to [k])$, where all the vertices in Z_R are happy with respect to $\tilde{c}$. Note that by definition $\tilde{c}$ extends c to a coloring of G therefore, we only need to show that vertices in Z_R are happy in G with respect to $\tilde{c}$. Consider a vertex $z \in Z_R$ and a vertex $v \in N(z)$. By the definition of $\tilde{c}$ we have $\tilde{c}(z) = 1$. By the construction of sets Z_R, we have $N(z) \subseteq V(G) \setminus S$ and by definition of $\tilde{c}$ it follows that $\tilde{c}(v) = 1$. Therefore, z is a happy vertex in G with respect to $\tilde{c}$. This concludes the proof. □

Notice that since Reduction Rule 3.10 is not applicable we have for each $i \in [k]$, $|U_i| = 1$. Furthermore, since Reduction Rule 3.12 is not applicable we have for each $i \in [k]$, $|Z_i| < \ell$, and since Reduction Rule 3.13 is not applicable we have $|Z_R| < \ell$. Therefore, we have $|Z \cup (\cup_{i \in [k]} U_i)| \leq k\ell + \ell - 1$. We now move to bounding the size of U_R, which will give us the desired kernel. To bound the size of U_R we employ the following marking scheme and argue that all the unmarked vertices can be deleted.

Marking Scheme for bounding $|U_R|$. We will denote the set of marked vertices by $M^\star \subseteq U_R$. For all $u, v \in V(G) \setminus U_R$ (not necessarily distinct) such that $N(u) \cap N(v) \cap U_R \neq \emptyset$, choose an arbitrary vertex in $w_{uv} \in N(u) \cap N(v) \cap U_R$ and add it to $M^\star$. That is we add a vertex in U_R to the marked set of vertices which is a common neighbor to vertices u and v.

We call a vertex in $U_R \setminus M^\star$ as an unmarked vertex. We now move to the reduction rule which deletes an unmarked vertex.

Reduction Rule 3.14. *If there exists $u \in U_R \setminus M^\star$ then delete u from G. The resulting instance is $(G - \{u\}, k, \ell, S, U \setminus \{u\}, c : S \to [k])$.*

Lemma 3.13. *Reduction Rule 3.14 is safe.*

Proof. Let $(G, k, \ell, S, U, c : S \to [k])$ be an instance of AHVC, $u \in U_R \setminus M^\star$, $G' = G - \{u\}$, and $U' = U \setminus \{u\}$. Recall that $U_R \cap S = \emptyset$ and therefore, $u \notin S$. The resulting instance after deletion of u in G is $(G', k, \ell, S, U', c : S \to [k])$. We will show that $(G, k, \ell, S, U, c : S \to [k])$ is a *yes* instance of AHVC if and only if $(G', k, \ell, S, U', c : S \to [k])$ is a *yes* instance of AHVC.

In the forward direction let $(G, k, \ell, S, U, c : S \to [k])$ be a *yes* instance of AHVC, $\tilde{c}$ be one of its solutions, and $H \subseteq V(G) \setminus U$ be the set of happy vertices in G with $|H| \geq \ell$. Let $\tilde{c}' : V(G') \to [k]$ be the coloring obtained from $\tilde{c}$ with $\tilde{c}' = \tilde{c}|_{V(G')}$. Notice that since $G' = G[V(G) \setminus \{u\}]$, therefore it follows that all the vertices in H are happy in G' with respect to the coloring $\tilde{c}'$. This concludes the proof in the forward direction.

In the reverse direction let $(G', k, \ell, S, U', c : S \to [k])$ be a *yes* instance of AHVC, $\tilde{c}'$ be one of its solutions, and $H \subseteq V(G') \setminus U'$ be the set of happy vertices in G' with

$|H| \geq \ell$. Let $\tilde{c} : V(G) \to [k]$ be the coloring obtained from $\tilde{c}'$ with $\tilde{c}|_{V(G')} = \tilde{c}'$ and $\tilde{c}(u)$ to be determined shortly. Since $u \notin M^\star$, for all $v, v' \in N(u) \cap Z$, we have a vertex $w_{vv'} \in M^\star \cap N(v) \cap N(v')$. Consider the set $H_u = H \cap N(u)$. If $H_u = \emptyset$, then we set $\tilde{c}(u) = 1$ (or any $i \in [k]$). Otherwise, we have $H_u \neq \emptyset$. Observe that for all $h, h' \in H_u$ we have $\tilde{c}'(h) = \tilde{c}'(h')$. Therefore, we set $\tilde{c}(u) = \tilde{c}'(h)$, where $h \in H_u$. The construction of $\tilde{c}$ implies that all the vertices in H are happy in G with respect to $\tilde{c}$. This concludes the proof. □

Once Reduction Rule 3.14 is not applicable we have $|U_R| \leq \binom{|Z|}{2} + |Z|$. Therefore, when none of the Reduction Rules 3.7 to 3.14 are applicable, we get the desired kernel. Hence, we obtain the following theorem.

Theorem 3.6. AHVC *admits a kernel with $\mathcal{O}(k^2\ell^2)$ vertices, where k is the number of colors in the coloring function and ℓ is the desired number of happy vertices.*

As a corollary to Theorem 3.6 we obtain the following.

Corollary 3.1. Happy Vertex Coloring *admits a kernel of size $\mathcal{O}(k^2\ell^2)$, where k is the number of colors in the coloring function and ℓ is the desired number of happy vertices.*

3.4.2 Kernel for Cluster Vertex Deletion

In the section, we give an $\mathcal{O}(k^2)$ vertex kernel for Cluster Vertex Deletion. A cluster graph is a graph which is disjoint union of cliques (or a P_3-free graph). The problem Cluster Vertex Deletion is formally defined below.

Cluster Vertex Deletion **Parameter:** k
Input: A graph G and an integer k.
Question: Is there a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a cluster graph?

Using this problem, we illustrate an application of using Expansion Lemma in designing kernels. We will be using Expansion Lemma in obtaining kernel for other problems as well. First, we start by computing an approximate solution (in polynomial time) for Cluster Vertex Deletion. Then we apply some reduction rules, and when none of them are applicable, then we argue that we have obtained a kernel of the desired size.

Theorem 3.7. Cluster Vertex Deletion *admits a factor three approximation algorithm.*

Proof. Let G be a graph, and OPT be the size of a minimum sized set X such that $G - X$ is a cluster graph. Furthermore, let $\mathcal{S}$ be a maximal family of sets of vertices which form induced P_3s in G such that any two members of $\mathcal{S}$ are pairwise (vertex) disjoint. One can easily construct such a family $\mathcal{S}$ greedily in polynomial time. Let S be the set of vertices contained in any set in $\mathcal{S}$. That is, $S = \bigcup_{O \in \mathcal{S}} O$. Since any solution to Cluster Vertex Deletion in G must contain a vertex from each set in $\mathcal{S}$ and any two members of $\mathcal{S}$ are pairwise disjoint, we have that $|S_{\mathsf{OPT}} \cap S| \geq |\mathcal{S}|$. It is evident that $G - S$ is a cluster graph, and $|S| \leq 3\mathsf{OPT}$. This completes the proof. □

We use Theorem 3.7 to compute a set S such that $G-S$ is a cluster graph. If $|S| > 3k$, then we can safely return that (G, k) is a *no* instance of Cluster Vertex Deletion. Therefore, we assume that $|S| \leq 3k$. Let $\mathcal{C}$ be the set of connected components in $G-S$. Next, we apply the following reduction rules (in the stated order, if applicable).

Reduction Rule 3.15. *If $k < 0$ then return that (G, k) is a no instance of* Cluster Vertex Deletion.

Reduction Rule 3.16. *If there is $C \in \mathcal{C}$ such that $N(C) \cap S = \emptyset$ then remove C from G. That is, the resulting instance is $(G - V(C), k)$.*

The safeness of Reduction Rule 3.16 follows from the fact that no minimal solution to Cluster Vertex Deletion in (G, k) can contain a vertex from C as it is clique component in the graph. Hereafter, we assume that Reduction Rule 3.16 is not applicbale. This implies that for each $C \in \mathcal{C}$, we have $N(C) \cap S \neq \emptyset$. Next, we construct a bipartite graph, using which we will design a reduction rule that employs Expansion Lemma for bounding the number of components in $G - S$.

Consider the bipartite graph $\mathcal{B}$ with vertex bipartition (S, Q), where Q is the set containing a vertex q_C for each $C \in \mathcal{C}$. For $s \in S$ and $q_C \in Q$, we add the edge (s, q_C) to $E(\mathcal{B})$ if and only if $C \cap N(s) \neq \emptyset$. The non-applicability of Reduction Rule 3.16 implies that no vertex in Q is an isolated vertex in $\mathcal{B}$. We are now ready to describe our next reduction rule.

Reduction Rule 3.17. *If $|\mathcal{C}| \geq 2|S|$ (or equivalently $|Q| \geq 2|S|$) then let $S' \subseteq S$ and $Q' \subseteq Q$ be the sets obtained by applying Expansion Lemma (Lemma 2.1) with $q = 2$ such that S' has a 2-expansion into Q' and $N(Q') \subseteq S'$. Delete vertices in S' from G and decrease k by $|S'|$. That is, the resulting instance is $(G - S', k - |S'|)$.*

Lemma 3.14. *Reduction Rule 3.17 is safe.*

Proof. Consider the case when $|\mathcal{C}| \geq 2|S|$, and let $S' \subseteq S$ and $Q' \subseteq Q$ be the sets obtained by applying Expansion Lemma (Lemma 2.1) with $q = 2$ such that S' has a 2-expansion into Q' and $N(Q') \subseteq S'$. Furthermore, let $G' = G - S$ and $k' = k - |S'|$. We show that (G, k) is a *yes* instance of Cluster Vertex Deletion if and only if (G', k') is a *yes* instance of Cluster Vertex Deletion.

In the forward direction, let (G, k) be a *yes* instance of Cluster Vertex Deletion, and let X be one of its solution. Let $A_C = \cup_{q_C \in Q'} V(C)$ and $\hat{X} = (X \setminus A_C) \cup S'$. We will argue that $|\hat{X}| \leq |X|$ and $G - \hat{X}$ is a cluster graph. Towards this, we start by arguing that $|X \cap (A_C \cup S')| \geq |S'|$. Let M be a 2-expansion from S' to Q', which exists by their construction. For each $s \in S'$ and its two neighbors, say $q_C, q_{C'}$ in $G[M]$, either $s \in X$ or $(V(C) \cup V(C')) \cap X \neq \emptyset$. This follows from the fact that $G[\{s\} \cup V(C) \cup V(C')]$ contains an induced P_3. This together with the disjointness of neighbors of vertices in S' in $G[M]$ implies that $|X \cap (A_C \cup S')| \geq |S'|$. But then we have that $|\hat{X}| \leq |X|$. Next, we argue that $G - \hat{X}$ is a cluster graph. Observe that the components in $\mathcal{C}' = \{C \mid q_C \in Q'\}$ are clique components in $G - \hat{X}$. Moreover, for each $C \in \mathcal{C}'$, we have $N(C) \subseteq S'$. Thus, in $G - \hat{X}$ there is no induced P_3 that contains a vertex from a component in C, where $C \in \mathcal{C}'$. This together with the construction of $\hat{X}$ and $G - X$ being a cluster graph implies that $G - \hat{X}$ is a cluster graph. Next, consider the set $X' = \hat{X} \setminus S'$, which by construction is of size at most $k - |S'|$. Clearly, $G' - X'$ is a cluster graph as $G' = G - S'$, $G - \hat{X}$ is a cluster graph, and $X' = \hat{X} \setminus S'$. This concludes the proof of forward direction.

In the reverse direction, let (G', k') be a *yes* instance of Cluster Vertex Deletion, and let X' be one of its solution. Let $X = X \cup S'$. Observe that $|X| = |S'| + |X'| \leq k$, and X is a solution to Cluster Vertex Deletion in (G, k) as $G' = G - S'$. □

If Reduction Rule 3.17 is not applicable then $|\mathcal{C}| < 2|S|$. Next, we bound the size of a component in $\mathcal{C}$, for which we create a set of marked vertices $\mathcal{M}$ in $G - S$, and delete the unmarked vertices in $G - S$. Towards this, we first construct a new graph H_s, for each $s \in S$. Then we compute a maximum matching M_s in H_s, and either (correctly) conclude that s belongs to every solution to Cluster Vertex Deletion in (G, k) or add the vertices in $H_s[M_s]$ to the marked set of vertices in G. Finally, we mark some more vertices from each components, and then delete the remaining vertices.

Construction of H_s, where $s \in S$. Consider a vertex $s \in S$. We construct the graph H_s as follows. The vertex set of H_s is $V(G - S)$, and for $u, v \in V(G - S)$ we add the edge (u, v) to $E(H_s)$ if and only if $(u, v) \in E(G - S)$ and $|N_G(s) \cap \{u, v\}| = 1$. Observe that for each $(u, v) \in E(H_s)$, $G[\{s, u, v\}]$ contains an induced P_3. Next, we compute (in polynomial time) a maximum matching M_s in H_s.

We are now ready to describe our next reduction rule.

Reduction Rule 3.18. *If there is $s \in S$ such that $|M_s| \geq k + 1$, then delete s from G and decrease k by 1. That is, the resulting instance is $(G - \{s\}, k - 1)$.*

Consider a vertex $s \in S$ such that $|M_s| \geq k + 1$. Notice that by construction of H_s (and M_s), G contains at least $k + 1$ induced P_3s whose pairwise (vertex) intersection is exactly $\{s\}$. Thus, any set X of size at most k such that $G - X$ is a cluster graph must contain the vertex s. This implies that Reduction Rule 3.18 is safe.

If Reduction Rule 3.18 is not applicable then for each $s \in S$, we have $|M_s| \leq k$. For each $s \in S$, we add all the vertices in $V(M_s) \subseteq V(G - S)$ to the set $\mathcal{M}$ of marked vertices. Observe that till now we have added at most $2k|S|$ vertices to $\mathcal{M}$. We let $\mathcal{M}^* = \cup_{s \in S} V(M_s)$. Next, for each $C \in \mathcal{C}$ we add to $\mathcal{M}$, as large as $k + 2$ vertices from $V(C) \setminus \mathcal{M}^*$. Note that $|\mathcal{M}| \in \mathcal{O}(k^2)$. We now state our final reduction rule.

Reduction Rule 3.19. *If there is $C \in \mathcal{C}$ with a vertex $v \in V(C) \setminus \mathcal{M}$, then delete v from G, That is, the resulting instance is $(G - \{v\}, k)$.*

Lemma 3.15. *Reduction Rule 3.19 is safe.*

Proof. Let $C \in \mathcal{C}$ with a vertex $v \in V(C) \setminus \mathcal{M}$, and $G' = G - \{v\}$. We show that (G, k) is a *yes* instance of Cluster Vertex Deletion if and only if (G', k) is a *yes* instance of Cluster Vertex Deletion.

In the forward direction, let (G, k) be a *yes* instance of Cluster Vertex Deletion, and let X be one of its solution. The clearly X is a solution to Cluster Vertex Deletion in (G', k) as $G' = G - \{v\}$. In the reverse direction, let (G', k) be a *yes* instance of Cluster Vertex Deletion, and let X be one of its solution. We show that $G - X$ is a cluster graph. Suppose not then there is an induced P_3, say P in $G - X$. Moreover, P must contain v as $G' - X$ is a cluster graph. Since $v \notin \mathcal{M}$, P is an induced P_3 containing v, and C is a clique, therefore there is at least one vertex say v^* in $(V(C) \setminus \mathcal{M}^*) \setminus (X \cup V(P))$. As $v, v^* \notin \mathcal{M}^*$ and C is a clique, we have $N[v^*] = N[v]$. But then $G - X[(V(P) \setminus \{v\}) \cup \{v^*\}]$ contains an induced P_3, which contradicts that $G - X$ is a cluster graph. □

Note that Reduction Rules 3.15 to 3.19 are safe and can be applied in polynomial time. When none of these reduction rules are applicable then all the vertices in $G - S$ are in $\mathcal{M}$, which is of size at most $\mathcal{O}(k^2)$. Also, $|S| \leq 3k$. Thus, we obtain the following theorem.

Theorem 3.8. Cluster Vertex Deletion *admits a vertex kernel of size* $\mathcal{O}(k^2)$*, where* k *is the size of the solution.*

Chapter 4

Lower Bounds

Until now, we looked at parameterized problems which belonged to the class FPT, and designed FPT algorithms for them with running time $f(k) \cdot n^{\mathcal{O}(1)}$, where $f(\cdot)$ is some computable function, k is the parameter, and n is the size of the input instance. In this chapter, we look at the complementary aspect of this notion. Firstly, we look at the complexity class beyond FPT. Not all parameterized problems admit an FPT algorithm under reasonable Complexity Theoretic assumptions. This is similar to the notion of NP-hardness in the classical Complexity Theory. However, unlike the case of NP-hard problems, for which there are many natural problems that are equally hard i.e. reducible (in polynomial time) to each other, the scenario for the case of parameterized problems is slightly different. This is captured by the notion of W-hierarchy. We discuss the notion of fixed-parameter intractability and W-hierarchy in more details in Section 4.1

Secondly, we look at "optimality" of an FPT algorithm. Consider a parameterized problem Π, and an FPT algorithm for it that runs in time $f(k)n^{\mathcal{O}(1)}$. Ideally, we would want the function $f(\cdot)$ to be as small as possible. For NP-hard problems we do not expect $f(\cdot)$ to be a polynomial function, otherwise it would imply P = NP, which is very unlikely. For many parameterized problems we devised algorithms running in time bounded by $c^{k^d}n^{\mathcal{O}(1)}$, where $c, d \in \mathbb{N}$. In the second part of the chapter, we look at a framework for proving that certain problems do not admit algorithms with running times $c^{o(k^d)}n^{\mathcal{O}(1)}$, under reasonable Complexity Theoretic assumptions.

Towards giving an introduction to a lower bound theory, rather than dwelling into the enigma of efficient computations, we follow a pragmatic algorithm designer's approach. The focus will be towards giving evidences on why some problems do not admit algorithms of certain running times. We prove statements of the form "If problem A does not have an algorithm of a certain type of running time then problem B does not have an algorithm of a certain type of running time as well". If we have a hypothesis that the problem B does not admit an algorithm of certain running time, then this provides an evidence that A can not have an algorithm of certain running time.

4.1 Fixed-Parameter Intractability

The notion of *fixed-parameter intractibility* is the non-existence of an FPT algorithm for a parameterized problem, under reasonable complexity theoretic assumptions. Consider two parameterized problems Π and Γ. Suppose we (somehow) proved that if Π is not in FPT then Γ is also not in FPT. Furthermore, we have a hypothesis that Π does not

admit an FPT algorithm. Then this will imply that Γ cannot be in FPT unless the assumed hypothesis fails. Towards proving statements like: if Π is not in FPT then Γ is also not in FPT, we employ some kind of reductions, which we call *parameterized reductions* (defined in Section 4.1.1). This is similar to the notion of polynomial time reductions defined for NP-hard problems. However, there are few crucial differences. Firstly, for showing the non-existence of an FPT algorithm we do not need the reduction to run in polynomial time. The next (very crucial) difference is that the parameterized reduction must somehow correlated the parameters of the instance, since we allow exponential dependence in the running time on the parameter. Another difference between parameterized reduction and polynomial time reductions is as follows. The NP-complete problems are reducible to each other in polynomial time, which implies that all these problems are "equally hard". The scenario seems to be different in the case of parameterized problems: there seems to be different levels of hardness, and basic problems like Clique and Hitting Set seem to occupy different levels in this hierarchy. To capture these hardness levels, Downey and Fellows introduced the notion of W-hierarchy, a brief introduction of which is given in Section 4.1.2.

4.1.1 Parameterized Reductions

Let us recall the notion of polynomial time reductions that are used in the NP-hardness proofs. A *polynomial time reduction* (many-one) from problem A to problem B is an algorithm that in polynomial time, given an instance x of A outputs an instance x' of B such that x is a *yes* instance of A if and only if x' is a *yes* instance of B. If there is a polynomial time reduction from problem A to B, and B is solvable in polynomial time, then given an instance x of A we can create an equivalent instance x' of B, and then use the polynomial time algorithm for B to solve x'. This gives us a polynomial time algorithm for the problem A. Next, we define an analogous notion for parameterized problems.

Definition 4.1. Let $\Pi, \Gamma \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *parameterized reduction* from Π to Γ is an algorithm, that given an instance (x, k) of Π, outputs an instance (x', k') of Γ such that the following conditions are satisfied.

1. (x, k) is a *yes* instance of Π if and only if (x', k') is a *yes* instance of Γ,
2. $k' \leq g(k)$, where $g(\cdot)$ is some (non-decreasing) computable function, and
3. the running time of the algorithm is bounded by $f(k)|x|^{\mathcal{O}(1)}$, where $f(\cdot)$ is some (non-decreasing) computable function.

We say that an instance (x, k) of Π and (x', k') of Γ are equivalent if (x, k) is a *yes* instance of Π if and only if (x', k') is a *yes* instance of Γ. In the following lemma, we show that parameterized reductions work as intended.

Lemma 4.1. *If there is a parameterized reduction from a parameterized problem Π to a parameterized problem Γ and Γ is in* FPT*, then Π is in* FPT *as well.*

Proof. Let $\mathcal{A}$ be a parameterized reduction (algorithm) from Π to Γ, which given an instance $(\tilde{x}, \tilde{k})$ of Π, runs in time $f(\tilde{k})|\tilde{x}|^c$, and outputs an equivalent $(\hat{x}, \hat{k})$ of Γ, such that $\hat{k} \leq g(k)$. Here, $c \in \mathbb{N}$ and $f(\cdot), g(\cdot)$ are (non-decreasing) computable functions.

Also, let $\mathcal{B}$ be an FPT algorithm for Γ, which given an instance $(\hat{x}, \hat{k})$, runs in time $h(\hat{k})|\hat{x}|^d$, where $d \in \mathbb{N}$ and $h(\cdot)$ is a (non-decreasing) computable function.

Next, consider an instance (x, k) of Π. Using $\mathcal{A}$ with (x, k) as input, in time $f(k)|x|^c$ we obtain an equivalent instance (x', k') of Γ with $k' \leq g(k)$ and $|x'| \leq f(k)|x|^c$. Next, we use the algorithm $\mathcal{B}$ with input (x', k'), and return the same output for the instance (x, k) of Π. The correctness of the output follows from the equivalence of the two instances. Moreover, since the functions $f(\cdot), g(\cdot)$, and $h(\cdot)$ are non-decreasing it follows that the total running time, including the time for the parameterized reduction is bounded by $f(k)|x|^c + h(g(k))(f(k)|x|^c)^d$. Thus we obtained an FPT algorithm for Π. □

Next, we look at few examples of problems that are as hard as Clique.

1. Independent Set. There is an easy parameterized reduction from Clique to Independent Set as follows. Given an instance (G, k) of Clique, the instance $(\bar{G}, k)$ is an equivalent instance of Independent Set, where $\bar{G}$ is the graph with $V(\bar{G}) = V(G)$ and $E(\bar{G}) = \{(u, v) \mid (u, v) \notin E(G), u \neq v\}$. Moreover, $(\bar{G}, k)$ can be constructed in polynomial time, and satisfies all the conditions of Definition 4.1.

2. Multi-Colored Clique. Given an instance (G, k) of Clique, we construct (in polynomial time) an instance $(G', V_1, V_2, \ldots, V_k)$ of Multi-Colored Clique as follows. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$. For $i \in [k]$, we let $V_i = \{v_j^i \mid j \in [n]\}$. Next, for $v_\ell^i \in V_i$ and $v_t^j \in V_j$, we add the edge (v_ℓ^i, v_t^j) to $E(G')$ if and only if $(v_\ell, v_t) \in E(G)$ (and $\ell \neq t$). It is easy to see that indeed all the conditions of Definition 4.1 are satisfied.

3. Multi-Colored Independent Set. Given an instance (G, k) of Clique, we construct (in polynomial time) an instance $(G', V_1, V_2, \ldots, V_k)$ of Multi-Colored Independent Set as follows. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$. For $i \in [k]$, we let $V_i = \{v_j^i \mid j \in [n]\}$. Next, for $v_\ell^i \in V_i$ and $v_t^j \in V_j$, we add the edge (v_ℓ^i, v_t^j) to $E(G')$ if and only if $(v_\ell, v_t) \notin E(G)$ or $\ell = t$. It is easy to see that indeed all the conditions of Definition 4.1 are satisfied.

4.1.2 The W-hierarchy

As mentioned earlier, NP-complete problems are equivalent to each other with respect to polynomial time reduction, while this does not seem to be the case for parameterized reductions. It is known that Multi-Colored Independent Set can be reduced to Dominating Set (see, e.g. Theorem 13.9 [CFK+15]), but a reduction in the other direction is not known. This indicates that unlike the case of NP-complete problems, there is a hierarchy of hard parameterized problems. Towards capturing these hardness levels, Downey and Fellows introduced the notion of W-hierarchy [DF92]. We would not dwell into much technicalities of the W-hierarchy, since from algorithm's designer perspective if there is a reduction from Clique then the problem is unlikely to admit an FPT algorithm. Towards giving a brief (informal) introduction we define *Boolean circuits*.

Definition 4.2. A *Boolean circuit* is a directed acyclic graph where the nodes are labelled as follows.

- Every node with no in-neighbors is an *input node*,
- Every node with exactly one in-neighbor is a *negation node*,
- Every node with more than one in-neighbors is either an *and-node* or an *or-node*.

Moreover, there is exactly one node with no out-neighbors, which is the *output node* (in addition to being other node). The *depth* of a circuit is the maximum length of a path from the input to the output node.

Observe that assigning 0/1 values to the input gate, determines the values at each node, in an obvious way. If the output node is assigned 1 then we say that the input assignment *satisfies* the circuit. By *weight* of an input assignment we denote the number of input gates that are assigned 1. Similar to the notion of Circuit Satisfiability for NP-completeness theory, here we define a parameterized version of it, which is called Weighted Circuit Satisfiability. This takes as an input a circuit C and an integer k, and the objective is to decide if there is a satisfying assignment of weight at most k. Weighted Circuit Satisfiability does not seem to be in FPT, as Clique (for example) can be easily reduced to it.

The W-hierarchy is defined by restricting Weighted Circuit Satisfiability to various classes of circuits. For a family of circuits $\mathcal{C}$, we let WCS[$\mathcal{C}$] to be the problem Weighted Circuit Satisfiability where the input circuit belongs to $\mathcal{C}$. A node in a circuit is *small* if it has at most 2 in-neighbors, and is *large* otherwise. The *weft* of a circuit is the maximum number of large nodes on a path from an input node to the output node. We denote by $\mathcal{C}_{t,d}$ the family of circuits that are of weft at most t and depth at most d.

Definition 4.3. For $t \in \mathbb{N}$, where $t \geq 1$, a parameterized problem Π belongs to the class $W[t]$ if there is a parameterized reduction from WCS[$\mathcal{C}_{t,d}$] for some $d \geq 1$.

A parameterized problem Π is *complete* for a class W[t] if $\Pi \in$ W[t] and there is a parameterized reduction from every problem in W[t] to Π. It is known that the problem Clique if W[1]-complete. This also implies that problems like Independent Set, Multi-Colored Clique, and Multi-Colored Independent Set are W[1]-hard from their respective parameterized reductions from Clique.

4.1.3 W[1]-hardness of Happy Vertex Coloring

We show that Happy Vertex Coloring, when parameterized by the number of happy vertices is W[1]-hard. We give a parameterized reduction from Multi-Colored Independent Set (MIS), which is known to be W[1]-hard [FHRV09].

Intuitively, given an instance $(G, V_1, V_2, \ldots, V_t)$ of MIS, for each V_i we create a vertex selection gadget, W_i which ensures that exactly one vertex from V_i can be happy in any valid coloring. Furthermore, the selected set of vertices from V_is form a set of happy vertices in the instance of Happy Vertex Coloring created. We now move to the formal description of the reduction.

Let $(G, V_1, V_2, \ldots, V_t)$ be an instance of MIS. We create an instance $(G', k, \ell, S, c : S \to [k])$ of Happy Vertex Coloring as follows. Let $n = |V(G)|$ and $V(G) = \{v_i \mid i \in [n]\}$. Initially, we have $V(G') = V(G)$ and $E(G') = \{(u, v) \in E(G) \mid u \in V_i, v \in$

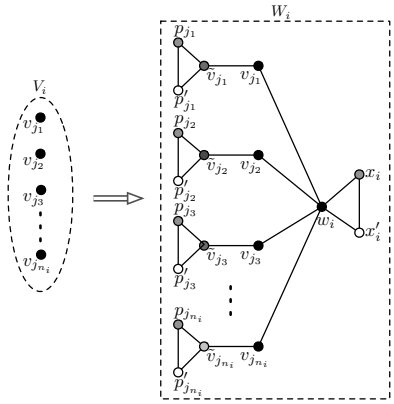

Figure 4.1: Construction of vertex selection gadget.

$V_j, i, j \in [t]$ and $i \neq j\}$. We now describe the vertex selection gadget W_i, for $i \in [t]$ (see Figure 4.1). For each $v_i \in V(G)$, we add three vertices $\tilde{v}_i, p_i, p'_i$ to $V(G')$, and add the edges $(v_i, \tilde{v}_i), (\tilde{v}_i, p_i)$, $(\tilde{v}_i, p'_i)$, and (p_i, p'_i) to $E(G')$. Furthermore, we add $\tilde{v}_i, p_i$, and p'_i to S, and set $c(\tilde{v}_i) = i$, $c(p_i) = n + 1$, and $c(p'_i) = n + 2$. For $i \in [t]$, we add three vertices w_i, x_i, x'_i to $V(G')$ and add all the edges in $\{(u, w_i) \mid u \in V_i\} \cup \{(w_i, x_i), (w_i, x'_i), (x_i, x'_i)\}$ to $E(G')$. Furthermore, we add x_i and x'_i to S, and set $c(x_i) = n + 1$ and $c(x'_i) = n + 2$. Here, the vertices x_i and x'_i are added to ensure that w_i can never be a happy vertex in any coloring of G', and w_i is added to ensure that at most one vertex from V_i can be happy in any coloring of $V(G')$. We have $W_i = G'[V_i \cup \{\tilde{v}_j, p_j, p'_j \mid v_j \in V_i\} \cup \{w_i, x_i, x'_i\}]$. Notice that we have $S = \{\tilde{v}_j, p_j, p'_j \mid v_j \in V(G)\} \cup \{x_i, x'_i \mid i \in [t]\}$. Note that for each $u \in S$, we have described the value of $c(u)$, and we have $k = n + 2$. Finally, we set $\ell = t$, and the resulting instance of Happy Vertex Coloring is $(G', k, \ell, S, c : S \to [k])$.

We state some lemmata which establish certain properties of the instance $(G', k, \ell, S, c : S \to [k])$ of Happy Vertex Coloring that we created.

Lemma 4.2. *Let $\tilde{c}$ be a coloring that extends c to a coloring of G', and H be the set of happy vertices in G' with respect $\tilde{c}$. Then for all $u \in \{w_i \mid i \in [t]\} \cup S$ we have $u \notin H$.*

Proof. Consider $w_{i^*} \in \{w_i \mid i \in [t]\}$. Recall that by construction, w_{i^*} has two neighbors x_{i^*} and x'_{i^*} with $\tilde{c}(x_{i^*}) = c(x'_{i^*}) = n + 1$ and $\tilde{c}(x'_{i^*}) = c(x'_{i^*}) = n + 2$. Therefore, $w_i \notin H$. For each $u \in S$, by construction we have a vertex $v \in N_{G'}(u) \cap S$ such that $\tilde{c}(u) = c(u) \neq c(v) = \tilde{c}(v)$, therefore $u \notin H$. □

Lemma 4.3. *Let $\tilde{c}$ be a coloring that extends c to a coloring of G', and H be the set of happy vertices in G' with respect $\tilde{c}$. Then for all $i \in [t]$, we have $|V_i \cap H| \leq 1$.*

Proof. Consider $i \in [t]$, such that $|V_i \cap H| > 1$. Let $v_j, v_{j'}$ be two distinct ($j \neq j'$) vertices in $V_i \cap H$. Since $v_j \in H$, and $\tilde{c}(\tilde{v}_j) = c(\tilde{v}_j) = j$, we have $\tilde{c}(v_j) = j$. Since $w_i \in N_{G'}(v_j)$ therefore, we have $\tilde{c}(w_i) = j$. By similar arguments we have $\tilde{c}(v_{j'}) = j'$ and $\tilde{c}(w_i) = j'$. This implies that $j = j'$, a contradicting. □

We now state the main lemma of this section.

Lemma 4.4. *$(G, V_1, V_2, \ldots, V_t)$ is a yes instance of* MIS *if and only if $(G', k, \ell, S, c : S \to [k])$ is a yes instance of* Happy Vertex Coloring.

Proof. In the forward direction, let $(G, V_1, V_2, \ldots, V_t)$ be a *yes* instance of MIS and $X = \{v_{i^\star} \mid i \in [t]\}$ be one of its solutions. We construct a solution $\tilde{c}$ to Happy Vertex Coloring in $(G', k, \ell, S, c : S \to [k])$ as follows. For each $u \in S$, we let $\tilde{c}(u) = c(u)$. For $i \in [t]$, for each $v \in V_i$, we let $\tilde{c}(v) = i^\star$ and $\tilde{c}(w_i) = i^\star$. This completes the description of $\tilde{c}$. Notice that for each $v_{i^\star} \in X$, we have that for all $v \in N_{G'}(v_{i^\star})$, $\tilde{c}(v) = i^\star$. This implies that $X \subseteq H$, where H is the set of happy vertices in G' with respect to $\tilde{c}$. Moreover, we have $|X| = t = \ell$. This concludes the proof in the forward direction.

In the reverse direction, let $(G', k, \ell, S, c : S \to [k])$ be a *yes* instance of Happy Vertex Coloring, $\tilde{c}$ be one of its solutions, and H be the set of happy vertices in G' with respect to $\tilde{c}$. We show that H is a solution to MIS in $(G, V_1, V_2, \ldots, V_t)$. By construction we have $V(G') = \cup_{i \in [t]} V(W_i)$. Lemma 4.2 implies that $H \cap (\{w_i \mid i \in [t]\} \cup S) = \emptyset$, and Lemma 4.3 implies that for each $i \in [t]$, we have $|H \cap V_i| \leq 1$. This together with the fact that $|H| \geq \ell$ implies that for each $i \in [t]$, we have $|H \cap V_i| = 1$. Therefore, we only need to show that H is an independent set in G. For $i \in [t]$, let $h_{i^\star}$ be the unique vertex in $H \cap V_i$. Consider $h_{i^\star}, h_{j^\star} \in H$, where $i, j \in [t]$ and $i \neq j$. Recall that $h_{i^\star}$ has a neighbor $\tilde{h}_{i^\star}$ in G' such that $\tilde{c}(\tilde{h}_{i^\star}) = i^\star$. Similarly, $h_{j^\star}$ has a neighbor $\tilde{h}_{j^\star}$ in G' such that $\tilde{c}(\tilde{h}_{i^\star}) = j^\star$. Therefore, we have $\tilde{c}(h_{i^\star}) = i^\star$ and $\tilde{c}(h_{j^\star}) = j^\star$, where $i^\star \neq j^\star$. This implies that $(h_{i^\star}, h_{j^\star}) \notin E(G')$, and hence by construction we have $(h_{i^\star}, h_{j^\star}) \notin E(G)$. Therefore, H is an independent set in G. □

Theorem 4.1. Happy Vertex Coloring *when parameterized by the number of happy vertices is* W[1]-hard.

Proof. Follows from the construction of the instance $(G', k, \ell, S, c : S \to [k])$ of Happy Vertex Coloring for the given instance $(G, V_1, V_2, \ldots, V_t)$ of MIS, Lemma 4.4, and W[1]-hardness of MIS. □

4.2 ETH Based Lower Bounds

In the previous section, we looked at framework for distinguishing between parameterized problems that admit FPT algorithms and the ones that do not. However, FPT algorithms have a wide range of running times. Typically we would want the function $f(\cdot)$ in the running time of FPT algorithm to be as small as possible. The framework that we want to look at in this section is to show that certain types of functions are unavoidable for certain types of parameterized problems. The framework here, will be conditioned on some reasonable Complexity Theoretic assumptions. Unfortunately, the assumptions that we need are stronger than FPT $\neq$ W[1], and therefore we introduce stronger complexity theoretic assumptions.

The Exponential Time Hypothesis (ETH) states that, roughly speaking, 3-SAT does not admit an algorithm than runs in time subexponential in the number of variables. The ETH implies that FPT $\neq$ W[1]. Moreover, ETH can be used to prove that a problem cannot a solved in time say, $2^{o(n)}n^{\mathcal{O}(1)}$, or a parameterized problem does not admit an algorithm running in time say, $2^{o(k)}n^{\mathcal{O}(1)}$, or say, $f(k)n^{o(k)}$. In many cases, the lower bound we obtain in the above way match (up to small factor) to the running time of the best known algorithm for the problem. This provides a tight understanding of the complexity of the problem. There is also a variant of ETH called SETH, which can be used to provide more refined lower bound results. Roughly speaking, SETH states that

CNF-SAT cannot be solved in time $\mathcal{O}^\star((2-\epsilon)^n)$, for any $\epsilon > 0$. In this thesis, we will not be looking at SETH based lower bounds, therefore we only introduce the ETH. Moreover, rather than dwelling into much technicalities, we just introduce the working definition for ETH.

The starting point of these assumptions is the hardness of CNF-SAT. For $q \geq 3$, q-SAT is NP-complete, therefore we do not expect it to be solvable in polynomial time. However, a brute force algorithm which tries all possible assignments solves the problem, and it runs in time $\mathcal{O}^\star(2^n)$. Unfortunately, we do not know any algorithm that solves the problem in substantially better running time. However, when the maximum number of literals in a clause bounded by some constant, then for every $q \geq 3$ there is an algorithm that solves the problem (q-SAT) in time $\mathcal{O}^\star(2^{\gamma(q)n})$. Here, $\gamma(q) > 0$ is a constant, for every $q \geq 3$. The current research status suggests that the exponential barrier is hard to break even for $q = 3$, i.e. obtaining a subexponential algorithm for 3-SAT. Next, we state the ETH.

Conjecture 4.1 (Exponential Time Hypothesis (ETH) [IPZ01])**.** *Let δ_3 be the infimum of set of constants c such that there is an algorithm for* 3-SAT *that runs in time $\mathcal{O}^\star(2^{cn})$. Then $\delta_3 > 0$.*

From the viewpoint of proving better lower bounds, the *Sparsification Lemma* [IPZ01] gives the following theorem.

Theorem 4.2 (Sparsification Lemma [IPZ01])**.** *Unless* ETH *fails, there is a constant $c > 0$ such that no algorithm for* 3-SAT *can achieve a running time which is bounded by $\mathcal{O}^\star(2^{c(n+m)})$, where n is the number of variables and m is the number of clauses in the input instance. In particular,* 3-SAT *cannot be solved in time $\mathcal{O}^\star(2^{o(n+m)})$.*

In the following section, we look at an example of proving lower bound assuming ETH. In Section 3.2.3 we designed an algorithm for Steiner Rainbow k-Coloring which runs in time $2^{\mathcal{O}(|S|^2)}n^{\mathcal{O}(1)}$. In Section 4.2.1 we see that indeed the exponential dependency on $|S|$ in the running time of the algorithm is optimal. In this example, we will use the trick of using harmonious coloring to obtain the result. In Chapter 9 (Section 9.1) we will see another application of using harmonious coloring for proving ETH based lower bound.

4.2.1 Lower Bound for Steiner Rainbow k-Coloring

In this section, we show that for every $k \geq 3$, Steiner Rainbow k-Coloring does not admit an algorithm running in time $2^{o(|S|^2)}n^{\mathcal{O}(1)}$, unless ETH fails. Towards this we give an appropriate reduction from k-Coloring on graphs of maximum degree $2(k-1)$. We note that k-Coloring does not admit an algorithm running in time $2^{o(n)}n^{\mathcal{O}(1)}$ unless ETH fails [IPZ01]. Moreover, assuming ETH, 3-Coloring does not admit an algorithm running in time $2^{o(n)}n^{\mathcal{O}(1)}$ on graph of maximum degree 4 [Kom15, CFG+17]. This follows from the fact that 3-Coloring does not admit such an algorithm, and a reduction from an instance G of 3-Coloring to an equivalent instance G' of 3-Coloring, where G' is a graph with maximum degree 4 with $|V(G')| \in \mathcal{O}(|V(G)|)$ (see Theorem 4.1 [GJ79]). In fact, we can show that k-Coloring does not admit an algorithm running in time $2^{o(n)}n^{\mathcal{O}(1)}$ on graph of maximum degree $2(k-1)$ (folklore). This result can be obtained (inductively) by giving a reduction from an instance G of

$(k-1)$-Coloring on graphs of degree at most $2(k-2)$ to an instance of k-Coloring on a graphs of bounded average degree (by adding global vertex), and then using an approach similar to that in Theorem 4.1 in [GJ79] we can obtain an (equivalent) instance of k-Coloring where the maximum degree of a vertex in the graph is bounded by $2(k-1)$.

Given an instance G of k-Coloring on n vertices and degree bounded by $2(k-1)$, we start by computing a harmonious coloring φ of G with $t \in \mathcal{O}(\sqrt{n})$ color classes such that each color class contains at most $\mathcal{O}(\sqrt{n})$ vertices. A harmonious coloring can be computed in polynomial time on bounded degree graphs using $\mathcal{O}(\sqrt{n})$ colors with each color class having at most $\mathcal{O}(\sqrt{n})$ vertices [CFG+17, Edw97, LM87, MX91]. Let $C_1, C_2, \cdots, C_t$ be the color classes of φ. Recall that for $i, j \in [t]$ with $i \neq j$ there is at most one edge between C_i and C_j in G. Moreover, C_i is an independent set in G, where $i \in [t]$. We create an instance G' of k-Coloring which has a harmonious coloring φ' with color classes $C'_1, C'_2, \cdots, C'_t$ such that for all $i, j \in [t]$, $i \neq j$ we have exactly one edge between C_i and C_j. Initially, we have $G = G'$ and $C'_i = C_i$, for all $i \in [t]$. For each $i, j \in [t]$, $i \neq j$ such that there is no edge between C_i and C_j in G we add two new vertices a_{ij} and a_{ji} to $V(G')$ and add the edge (a_{ij}, a_{ji}) to $E(G')$. Furthermore, we add a_{ij} to C'_i and a_{ji} to C'_{ji}. Observe that $|V(G')| \in \mathcal{O}(n)$, $|E(G')| \in \mathcal{O}(n)$, and for each $i \in [t]$, $|C'_i| \in \mathcal{O}(\sqrt{n})$. Also, for each $i, j \in [t]$, $i \neq j$ there is exactly one edge between C'_i and C'_j in G'. It is easy to see that G is a *yes* instance of k-Coloring if and only if G' is a *yes* instance of k-Coloring.

Hereafter, we will be working with the instance G' of k-Coloring, together with its harmonious coloring φ' with color classes $C'_1, C'_2, \cdots, C'_t$. Moreover, for $i, j \in [t]$, $i \neq j$ there is exactly one edge between C'_i and C'_j in G'.

We now move to the description of creating an equivalent instance $(\tilde{G}, S)$ of Steiner Rainbow k-Coloring, where $k \geq 3$. Initially, we have $V(\tilde{G}) = V(G')$. For $(u, v) \in E(G')$ we add $k-3$ new vertices $x^{uv}_1, x^{uv}_2, \cdots, x^{uv}_{k-3}$ to $\tilde{G}$ and add all the edges in the path $(u, x^{uv}_1, \cdots, x^{uv}_{k-3}, v)$ to $E(\tilde{G})$. Note that for $k = 3$ we do not any new vertex and directly add the edge (u, v) to $\tilde{G}$. For each $i \in [t]$ we add a vertex c_i to $\tilde{G}$ and add all the edges in $\{(c_i, v) \mid v \in C'_i\}$ to $E(\tilde{G})$. Finally, we set $S = \{c_i \mid i \in [t]\}$. Notice that $|S| \in \mathcal{O}(\sqrt{n})$. In the following lemma we establish that G' is a *yes* instance of k-Coloring if and only if $(\tilde{G}, S)$ is a *yes* instance of Steiner Rainbow k-Coloring.

Lemma 4.5. *G' is a yes instance of k-Coloring if and only if $(\tilde{G}, S)$ is a yes instance of* Steiner Rainbow k-Coloring.

Proof. In the forward direction, let G' be a *yes* instance of k-Coloring, and $c : V(G') \to [k]$ be one of its solution. We create a coloring $c_R : E(\tilde{G}) \to [k]$ as follows. For $i \in [t]$ and $v \in C'_i$ we set $c_R(c_i, v) = c(v)$. For $i, j \in [t]$, $i \neq j$ let u, v be the (unique) vertices in C'_i and C'_j such that $(u, v) \in E(G')$. We now describe the value of c_R for edges in the path $P = (u, x^{uv}_1, \cdots, x^{uv}_{k-3}, v)$. Notice that $|E(P)| = k-2$ therefore, we arbitrarily assign distinct integers in $[k] \setminus \{c_R(c_i, u), c_R(c_j, v)\}$ to $c_R(e)$, where $e \in E(P)$. Since c is a proper coloring of G' therefore, $c_R(c_i, u) = c(u) \neq c(v) = c_R(c_j, v)$. This together with the definition of c_R for edges in P implies that there is a rainbow path, namely $(c_i, u, x^{uv}_1, \cdots, x^{uv}_{k-3}, v, c_j)$ in $\tilde{G}$ between c_i and c_j. This concludes the proof in the forward direction.

In the reverse direction, let $(\tilde{G}, S)$ be a *yes* instance of Steiner Rainbow k-Coloring, and $c_R : E(\tilde{G}) \to [k]$ be one of its solution. We create coloring $c : V(G') \to$

$[k]$ as follows. For $i \in [t]$ and $v \in C'_i$, we let $c(v) = c_R(c_i, v)$. We show that c is a solution to k-Coloring in G'. Consider $(u, v) \in E(G')$, and let $u \in C'_i$ and $v \in C'_j$. Note that we have $i \neq j$. Let P be a rainbow path between c_i and c_j in $\tilde{G}$. By the construction of $\tilde{G}$, we have $N_{\tilde{G}}[c_i] \cap N_{\tilde{G}}[c_j] = \emptyset$. Moreover, since P is a rainbow path therefore, it can contain at most k edges. Since $N_{\tilde{G}}(c_i) = C'_i$ and $N_{\tilde{G}}(c_j) = C'_j$, and there is a exactly one path with at most $k-2$ edges between a vertex in C'_i and a vertex in C'_j, namely $(c_i, u, x_1^{uv}, \cdots, x_{k-3}^{uv}, v, c_j)$. Therefore, by construction of c together with the fact that P is a rainbow path we have $c(u) \neq c(v)$. This concludes the proof. □

Theorem 4.3. Steiner Rainbow k-Coloring *does not admit an algorithm running in time* $2^{o(|S|^2)}n^{\mathcal{O}(1)}$*, unless* ETH *fails. Here,* n *is the number of vertices in the input graph.*

Proof. Follows from construction of an instance $(\tilde{G}, S)$ with $|S| \in \mathcal{O}(\sqrt{n})$ of Steiner Rainbow k-Coloring for a given instance G of k-Coloring with maximum degree at most $2(k-2)$, Lemma 4.5, and existence of no algorithm running in time $2^{o(n)}n^{\mathcal{O}(1)}$ for k-Coloring on graphs of maximum degree $2(k-2)$ (assuming ETH). □

Part II

New Results on $\mathcal{F}$-Deletion Problems

Chapter 5

Block Graphs

A graph G is known as a *block graph* if every maximal 2-connected component in G is a clique. Equivalently, we can see a block graph as a graph obtained by replacing each edge in a forest by a clique. A *chordal graph* is a graph which has no induced cycles of length at least four. An equivalent characterisation of a block graph is a chordal graph with no induced $K_4 - e$ [BLS99, How81]. Here, $K_4 - e$ is the graph obtained by removing an edge from a clique on 4 vertices. The class of block graphs is the intersection of the chordal and distance-hereditary graphs [How81].

In this chapter, we consider the problem which we call Block Graph Vertex Deletion (BGVD). Here, as an input we are given a graph G and an integer k, and the objective is to decide if there is a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a block graph. The NP-hardness of BGVD follows from [LY80].

Block Graph Vertex Deletion (BGVD) **Parameter:** k
Input: A graph G and an integer k.
Question: Is there a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a block graph?

Kim and Kwon [KK15, KK17] gave an FPT algorithm with running time $\mathcal{O}(10^k|V(G)|^{\mathcal{O}(1)})$ for the problem. Also, they designed a kernel with $\mathcal{O}(k^9)$ vertices [KK15], which they improved to a kernel with $\mathcal{O}(k^6)$ vertices [KK17]. We improve both these results via a novel connection to the Feedback Vertex Set problem.

Our Results and Methods. We start by stating the results we obtain in this chapter and then we explain how we obtain these results. Our two main results are:

Theorem 5.1. BGVD *has an FPT algorithm running in time* $\mathcal{O}(4^k|V(G)|^{\mathcal{O}(1)})$.

Theorem 5.2. BGVD *has a kernel of with* $\mathcal{O}(k^4)$ *vertices.*

Our two theorems improve both the results in [KK17]. That is, the running time of our FPT algorithm improves over the previous best algorithm for the problem which runs in time $10^k n^{\mathcal{O}(1)}$ and the size of our kernel reduces over the previously known kernel of size $\mathcal{O}(k^6)$.

Our results are based on a connection between the Weighted-FVS and BGVD problems. In particular, we show that if the input graph does not contain induced cycles on four vertices or diamonds ($K_4 - e$), then we can construct an auxiliary bipartite graph and solve Weighted-FVS on it. This results in a faster FPT algorithm for BGVD. In

the algorithm that we give for the BGVD problem, as a sub-routine we use the algorithm for the Weighted-FVS problem, which we designed in Section 3.2.1. For obtaining a better polynomial kernel for BGVD, most of our Reduction Rules are same as those used in [KK17]. On the way to our result we also design a factor four approximation algorithm for BGVD.

5.1 FPT Algorithm for Block Graph Vertex Deletion

In this section, we design an FPT algorithm for the BGVD problem. First, we look at the special case, when the input graph does not have any small obstructions. Here, by small obstruction we mean either a $D_4 = K_4 - e$ or a C_4 (cycle on 4 vertices). We show that, in this case, BGVD reduces to Weighted-FVS. Later, we solve the general problem, using the algorithm of the special case.

5.1.1 Restricted BGVD

In this section, we solve the following special case of BGVD in FPT time.

Restricted BGVD	**Parameter:** k
Input: A graph G, which is $\{D_4, C_4\}$-free and an integer k.	
Question: Is there a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a block graph?	

Let (G, k) be an instance of Restricted BGVD, and $\mathcal{C}$ be the set of maximal cliques in G. We start with the following simple observation about graphs which is $\{D_4, C_4\}$-free.

Lemma 5.1. *For a $\{D_4, C_4\}$-free graph G on n vertices the following conditions hold.*

- *Any two maximal cliques intersect on at most one vertex.*
- *The number of maximal cliques in G is at most n^2.*

Proof. Let C_1 and C_2 be two distinct maximal cliques in $\mathcal{C}$. Since G is D_4-free, $V(C_1) \cap V(C_2)$ can have at most one vertex. Thus, each edge of G belongs to exactly one maximal clique. This gives a bound of n^2 on the number of maximal cliques. □

Next, we construct an auxiliary weighted bipartite graph $\hat{G}$ as follows. The graph $\hat{G}$ is a bipartite graph with vertex set bipartition $V(G) \cup V_{\mathcal{C}}$, where $V_{\mathcal{C}}$ is the set where we add a vertex v_C corresponding to each maximal clique $C \in \mathcal{C}$. Note that there is a bijective correspondence between the vertices of $V_{\mathcal{C}}$ and the maximal cliques in $\mathcal{C}$. A vertex $v \in C$, where $C \in \mathcal{C}$ is called a *shared vertex* if it is part of at least two (distinct) maximal cliques in $\mathcal{C}$. We add an edge between a vertex $v \in V(G)$ and a vertex $v_C \in V_{\mathcal{C}}$ in $E(\hat{G})$ if and only if v is a shared vertex of C.

Lemma 5.2. *Let G be a $\{D_4, C_4\}$-free graph, and $S \subseteq V(G)$. The set S is a block vertex deletion set of G if and only if $\hat{G} - S$ is acyclic.*

Proof. In the forward direction, let S be a block vertex deletion set for G. Suppose that $\hat{G} - S$ has a cycle C. From Lemma 5.1 if follows that C is not a C_4, as this corresponds to two maximal cliques that share 2 vertices. Thus, C is an even cycle of length at least 6. Suppose C has length 6. This corresponds to maximal cliques C_1, C_2, C_3 such that $u = C_1 \cap C_2$, $v = C_2 \cap C_3$ and $w = C_1 \cap C_3$. Since C_1, C_2, C_3 are distinct maximal cliques, at least one of them must have a vertex other than u, v or w. Without loss of generality, let C_1 have a vertex $x \notin \{u, v, w\}$. Then, the set $\{x, u, v, w\}$ forms a D_4 in G. However, this is not possible, as G did not have a D_4 to start with. Hence, C must be an even cycle of length at least 8. However, this corresponds to a set of maximal cliques and external vertices such that the external vertices form an induced cycle of length at least four. This contradicts that S was a block vertex deletion set for G. Thus, $\hat{G} - S$ must be acyclic.

In the reverse direction, let $\hat{G} - S$ be acyclic. Suppose $G - S$ has an induced cycle C, of length at least four. As C is an induced cycle of length at least four, no two edges in C can belong to the same maximal clique. For an edge (u, v) of C, let $C_{(u,v)}$ be the maximal clique containing it. Also, let $c_{(u,v)}$ be the vertex corresponding to $C_{(u,v)}$ in $\hat{G}$. We replace the edge (u, v) in C by two edges $(u, c_{(u,v)})$ and $(v, c_{(u,v)})$. In this way, we obtain an (induced) cycle C' in $\hat{G} - S$, which is a contradiction. Thus, S must be a block vertex deletion set for G.

□

If the input graph G is without induced C_4 and D_4 then Lemma 5.2 tells us that to find block vertex deletion set of G of size at most k one can check whether there is a feedback vertex set of size at most k for $\hat{G}$ contained in $V(G)$. To enforce that we find feedback vertex set for $\hat{G}$ which is completely contained in $V(G)$ we solve an appropriate instance of Weighted-FVS. In particular, we set the weight function $w : V(\hat{G}) \to \mathbb{Q}$ as follows. For $v \in V(G)$, $w(v) = 1$ and for $v_C \in V_{\mathcal{C}}$, $w(v_C) = n^4$. Clearly, $V(G)$ is a feedback vertex set of $\hat{G}$, and thus the weight of a minimum sized feedback vertex set in $\hat{G}$ is at most n. This implies that using an algorithm for Weighted-FVS on the instance $(\hat{G}, w, k)$ either returns a feedback vertex set contained inside $V(G)$ or returns that the given instance is a *no* instance.

Theorem 5.3. Restricted BGVD *can be solved in* $\mathcal{O}^\star(3.618^k)$.

Proof. Let (G, k) be an instance of Restricted BGVD. We construct the instance $(\hat{G}, w, k)$ of Weighted-FVS as described earlier. Next, we solve Weighted-FVS on $(\hat{G}, w, k)$ using the algorithm given by Theorem 3.1 (Section 3.2.1). By Lemma 5.2 if $(\hat{G}, w, k)$ is a *no* instance of Weighted-FVS, then we correctly conclude that (G, k) is a *no* instance of Restricted BGVD. Otherwise, the algorithm return that $(\hat{G}, w, k)$ is a *yes* instance of Restricted BGVD. Recall that by the construction of $(\hat{G}, w, k)$, for any solution S to the weighted-fvs in it we have $|S| \leq k$ and $w(S) \leq k$. This together with Lemma 5.2 implies that we correctly concluded that (G, k) is a *yes* instance of Restricted BGVD. The running time of the algorithm is dominated by the running time of Weighted-FVS, and thus it is $\mathcal{O}^\star(3.618^k)$. This completes the proof.

□

5.1.2 Block Graph Vertex Deletion

We are now ready to describe the FPT algorithm for BGVD, and hence prove Theorem 5.1. We design the algorithm for the general case with the help of the algorithm for Restricted BGVD.

Proof of Theorem 5.1. Let (G, k) be an instance of BGVD, where G is a graph on n vertices. Furthermore, let O be an (induced) D_4 or C_4 (if any) in the input graph G. For any solution to BGVD in (G, k) must contain at least one of the vertices in O. Therefore, we branch on the choice of these vertices, and for every vertex $v \in O$, we recursively apply the algorithm to solve BGVD instance $(G - \{v\}, k - 1)$. Observe that (G, k) is a *yes* instance of BGVD if and only if for some $v \in V(O)$ we have $(G - \{v\}, k - 1)$ is a *yes* instance of BGVD. On the other hand, if G is $\{D_4, C_4\}$-free, then we do not make any further recursive calls. Instead, we run the algorithm for Restricted BGVD given by Theorem 5.3 on (G, k), and return the same output. Thus, the running time of the algorithm is upper bounded by the following recurrence.

$$T(n,k) = \begin{cases} 3.168^k & \text{if G is } \{D_4, C_4\}\text{-free} \\ 4T(n-1,k-1) + n^{\mathcal{O}(1)} & \text{otherwise} \end{cases}$$

Thus, the running time of the algorithm is upper bounded by $\mathcal{O}^\star(4^k)$. □

5.2 An Approximation Algorithm for BGVD

In this section, we present a (simple) approximation algorithm for BGVD. Given a graph G, we give a block vertex deletion set S of size at most $4 \cdot \mathsf{OPT}$, where OPT is the size of a minimum sized block vertex deletion set for G.

Theorem 5.4. BGVD *admits a factor four approximation algorithm.*

Proof. Let G be an instance of BGVD and OPT be the size of a minimum sized block vertex deletion set for G and S_{OPT} be a minimum sized block vertex deletion set for G. Furthermore, let Approx-WFVS be the 2-approximation algorithm given in [BBF99], which takes as an instance $(G, w : V(G) \to \mathbb{Q})$, where G is a graph, and outputs a weighted-fvs whose weight is at most twice the weight of the minimum weighted-fvs.

Let $\mathcal{S}$ be a maximal family of D_4 and C_4 such that any two members of $\mathcal{S}$ are pairwise disjoint. One can easily construct such a family $\mathcal{S}$ greedily in polynomial time. Let S_1 be the set of vertices contained in any obstruction in $\mathcal{S}$. That is, $S_1 = \bigcup_{O \in \mathcal{S}} O$. Since any block vertex deletion set must contain a vertex from each obstruction in $\mathcal{S}$ and any two members of $\mathcal{S}$ are pairwise disjoint, we have that $|S_{\mathsf{OPT}} \cap S_1| \geq |\mathcal{S}|$.

Let $G' = G - S_1$. Observe that G' does not contain either D_4 or C_4 as an induced subgraph. Next, we construct a graph $\hat{G}'$ and a (vertex) weight function w, as described in Section 5.1.1, where we want to solve the problem of finding a weighted-fvs. We now employ the factor two approximation algorithm Approx-WFVS on the instance $(\hat{G}', w)$. This returns an weighted-fvs S_2 of $\hat{G}'$ such that $w(S_2)$ is at most twice the weight of a minimum weighted-fvs. By our construction $S_2 \subseteq V(G')$. Lemma 5.2 implies that S_2 is a factor two approximation for BGVD on G'. We return the set $S = S_1 \cup S_2$ as our solution. Since $S_{\mathsf{OPT}} - S_1$ is also an optimum solution for G' we have that $|S_2| \leq 2|S_{\mathsf{OPT}} - S_1|$.

It is evident that S is block vertex deletion set of G. To conclude the proof of the theorem we will show that $|S| \leq 4\mathsf{OPT}$. Towards this observe the following.

$$\begin{aligned} |S| = |S_1| + |S_2| &\leq 4|\mathcal{S}| + 2|S_{\mathsf{OPT}} - S_1| \\ &\leq 4|S_{\mathsf{OPT}} \cap S_1| + 2|S_{\mathsf{OPT}} - S_1| \\ &\leq 4|S_{\mathsf{OPT}}| = 4\mathsf{OPT}. \end{aligned}$$

This completes the proof.

□

5.3 Improved Kernel for Block Graph Vertex Deletion

In this section, we give a kernel of $\mathcal{O}(k^4)$ vertices for BGVD. Let (G, k) be an instance of BGVD, where G is a graph on n vertices. We start by applying some of the reduction rules from [KK17].

Reduction Rule 5.1. *If G has a component H, where H is a block graph, then remove H from G.*

Reduction Rule 5.2. *If there is a vertex $v \in V(G)$ such that $G - \{v\}$ has a component H, where $G[\{v\} \cup V(H)]$ is a connected block graph then, remove all the vertices in H from G.*

Reduction Rule 5.3. *Let $S \subseteq V(G)$, where each $u, v \in S$ are true-twins in G. If $|S| > k + 1$, then remove all the vertices from S except $k + 1$ vertices.*

Reduction Rule 5.4. *Let (t_1, t_2, t_3, t_4) be an induced path in G. For each $i \in [3]$, let $S_i \subseteq V(G) \setminus \{t_1, t_2, t_3, t_4\}$ be a clique in G such that the following holds.*

- *For $i \in [3]$, and $v \in S_i$ we have $N_G(v) \setminus S_i = \{t_i, t_{i+1}\}$, and*
- *For each $i \in \{2, 3\}$, we have $N_G(t_i) = \{t_{i-1}, t_{i+1}\} \cup S_{i-1} \cup S_i$.*

Then remove S_2 from G and contract the edge (t_2, t_3).

Proposition 5.1 (Proposition 3.1 [KK17])**.** *Let G be a graph and k be an integer. For a vertex $v \in V(G)$, in $\mathcal{O}(kn^3)$ time, we can find one of the following.*

i) $k + 1$ pairwise vertex disjoint obstructions,

ii) $k + 1$ obstructions whose pairwise intersection is exactly v, or

iii) $S'_v \subseteq V(G)$ such that $|S'_v| \leq 7k$ and $G - S'_v$ has no block graph obstruction containing v.

Reduction Rule 5.5. *Let $v \in V(G)$ and $G' = G - \{v\}$. We remove the edges between $N_G(v)$ from G', i.e. $E(G') = E(G') \setminus \{(u, w) \mid u, w \in N_G(v)\}$. In G' if there are at least $2k + 1$ vertex-disjoint $N_G(v)$-paths in G' then we do one of the following.*

- *If G contains $k+1$ vertex disjoint obstructions, then return that (G,k) is a no-instance.*
- *Otherwise, delete v from G and decrease k by 1, i.e., the resulting instance is $(G-\{v\},k-1)$.*

The Reduction rules 5.1 to 5.5 are safe and can be applied in polynomial time [KK17]. For sake of clarity we denote the reduced instance at each step by (G,k). We always apply the lowest numbered Reduction Rule, in the order that they have been stated, that is applicable at any point of time. Hereafter, we assume that Reduction rules 5.1 to 5.5 are not applicable.

For a vertex $v \in V(G)$, by Proposition 5.1, we may find $k+1$ pairwise vertex-disjoint obstructions, and we can safely conclude that the graph is a *no* instance. Secondly, if we find $k+1$ obstructions whose pairwise intersection is exactly v then the Reduction rule 5.5 will be applicable. Thus, we assume that for each vertex $v \in V(G)$, the third condition of Proposition 5.1 holds. In other words, we have a set S'_v of size at most $7k$ such that $G - S'_v$ does not contain any obstruction that contains v. In fact, for each $v \in V(G)$, we can find a block vertex deletion set $S_v \subseteq V(G) \setminus \{v\}$ of bounded size, which does not contain v. Let A be an approximate solution of size at most $4k$ (if it exists) to BGVD obtained by using the approximation algorithm for BGVD given by Theorem 5.4. If the approximation algorithm returns a solution whose size is more than $4k$, then we can trivially return that (G,k) is a *no* instance of BGVD. Therefore, in the following we assume that set A exists.

Observation 2. *For every vertex $v \in V(G)$, we can find in polynomial time, a set $S_v \subseteq V(G) \setminus \{v\}$ such that $|S_v| \leq 11k$ and $G - S_v$ is a block graph.*

Proof. If $v \notin A$, then $S_v = A$. Otherwise, $S_v = (A \setminus \{v\}) \cup S'_v$, where S'_v is the set given by Proposition 5.1. Here, we rely on the fact that Reduction Rule 5.5 is not applicable. Note that for each $v \in V(G)$, we have $|S_v| \leq 11k$ and $G - S_v$ is a block graph. □

For a vertex $v \in V(G)$, *component degree* of v is the number of connected components in $\mathcal{C}$, where $\mathcal{C}$ is the set of connected components in $G - (S_v \cup \{v\})$ that have a vertex adjacent to v. We give a reduction rule that bounds the *component degree* of a vertex $v \in V(G)$, using Expansion Lemma (Lemma 2.1).

For a vertex $v \in V(G)$, let $\mathcal{C}_v$ be the set of connected components in $G - (S_v \cup \{v\})$ that have a vertex adjacent to v. Consider a connected component $C \in \mathcal{C}_v$ such that no vertex in $V(C)$ is adjacent to a vertex in S_v. But then, $G - \{v\}$ has a component which is a block graph (namely, C) therefore, Reduction rule 5.2 is applicable, a contradiction to the assumption that none of the previous reduction rules are applicable. Therefore, for each $C \in \mathcal{C}$ there is a vertex $u \in V(C)$ and $s \in S_v$ such that $(u,s) \in E(G)$. Let $\mathcal{D}$ be a vertex set which contains a vertex d corresponding to each component $D \in \mathcal{C}$. Consider the bipartite graph $\mathcal{B}_v$ with the vertex set bipartition as $(\mathcal{D}, S_v)$. There is an edge between $d \in \mathcal{D}$ and $s \in S_v$ if and only if the component D corresponding to which the vertex d was added to $\mathcal{D}$ has a vertex u_d such that $(u_d, s) \in E(G)$. Next, we state our final reduction rule.

Reduction Rule 5.6. *For a vertex $v \in V(G)$ if $|\mathcal{C}_v| > 33k$, then we do the following.*

- *Let $\mathcal{D}' \subseteq \mathcal{D}$ and $S \subseteq S_v$ be the sets obtained after applying Lemma 2.1 with $q = 3$, $X = S_v$ and $Y = \mathcal{D}$;*
- *For each $d \in \mathcal{D}'$, let the component corresponding to d be $D \in \mathcal{C}_v$. Delete all the edges between (u, v), where $u \in V(D)$;*
- *For each $s \in S$, add two vertex disjoint paths between v and s.*

Safeness of the Reduction rule 5.6 follows from the safeness of Reduction rule 6 in [KK17].

5.3.1 Bounding the number of blocks in $G - A$

First, we bound the number of leaf blocks in $G-A$, when none of the reduction rules are applicable. Note that $G-A$ is a block graph as A is an approximate solution to BGVD. For $v \in A$, let S'_v be the set obtained from Proposition 5.1 and S_v be the set obtained from Observation 2. Let $\mathcal{C}_v$ be the set of connected components in $G-(S_v \cup \{v\})$ which have a vertex adjacent to v. All the connected components in $G-A$ which do not have a vertex that is adjacent to v, must be adjacent to some $v' \in A$ otherwise, Reduction rule 5.1 will be applicable. Also, all the leaf blocks in $G-A$ must have an internal vertex that is adjacent to some vertex in A, since the Reduction rules 5.1 and 5.2 are not applicable. The number of leaf blocks in $G-A$, whose set of internal vertices have a non-empty intersection with S'_v, is at most $7k$. Therefore, it is enough to count, for each $v \in A$, the number of leaf blocks in $\mathcal{C}_v$. In Observation 3, we give a bound on the number of leaf blocks in $G-A$, not containing any vertex from S'_v.

Observation 3. *For $v \in A$, the number of leaf blocks in $G-A$ not containing any vertex from S'_v is at most the number of leaf blocks in $G-(S_v \cup \{v\})$.*

Proof. Note that for $v \in A$, $S_v = (A \setminus \{v\}) \cup S'_v$. By deleting a vertex $u \in S'_v$ from $G-A$ one of the following can happen. If u was a cut vertex in $G-A$, then we increase the number of components after deleting u from $G-A$. By increasing the number of components, the number of leaves cannot decrease. If u was not a cut vertex in $G-A$ then by deleting u the number of cut vertices can only increase. Therefore, the number of leaf blocks after deletion of u from $G-A$ can only increase. Hence, the claim follows. □

Therefore, for each $v \in A$ we count those leaf blocks in $\mathcal{C}_v$ which do not contain any vertex from S'_v.

Lemma 5.3. *Consider a vertex $v \in V(G)$ and its corresponding set S_v. Let $\mathcal{C}$ be the set of connected components in $G-(S_v \cup \{v\})$. For each $C \in \mathcal{C}$, there is a block $\tilde{B}$ in C such that $N_C(v) \subseteq V(\tilde{B})$.*

Proof. Consider a vertex $v \in V(G)$, and let $\mathcal{C}$ be the set of connected components in $G-(S_v \cup \{v\})$. By the definition of S_v, for each $C \in \mathcal{C}$, $G[V(C) \cup \{v\}]$ is a block graph.

If for some $C \in \mathcal{C}$, $N_C(v) = \emptyset$, then the condition is trivially satisfied for that connected component C. Let $C \in \mathcal{C}$ be a connected component such that $N_C(v) \neq \emptyset$. Let t be a vertex in $N_B(v)$, where B is a block in C. Let B' be a block in C, where $B' \neq B$ and B' has a vertex $t' \in V(B') \setminus V(B)$ that is adjacent to v. Note that B, B'

are in the same connected component C. Let P be a path with shortest path length between t and t'.

We first argue for the case when $(t, t') \notin E(G)$. In this case, the path P has at least 2 edges. We prove that we can find an obstruction, by induction on the length of the path (number of edges). If length of path P is 2, say $P = (t, u, t')$. If $(u, v) \in E(G)$, then $G[\{t, t', u, v\}]$ is a D_4, otherwise it is a C_4, contradicting that $G[V(C) \cup \{v\}]$ is a block graph.

Let us assume that we can find an obstruction if the path length is ℓ. Next, we prove it for paths of length $\ell + 1$. Let $P = (t, x_1, x_2, \ldots, x_{\ell-1}, t')$, and y be the first vertex other than t in P such that $(y, v) \in E(G)$. If $y = t'$, then P along with v forms an induced cycle of length at least 5, contradicting that $G[V(C) \cup \{v\}]$ is a block graph. If $y = x_1$, then $G[\{t, x_1, x_2, v\}]$ either is a D_4, which is the case when $(x_2, v) \in E(G)$, or $\hat{P} = (x_1, x_2, \ldots, t')$ is a path of shorter length with at least 2 edges, and thus by induction hypothesis has an obstruction along with v. Otherwise, $y \notin \{x_1, t'\}$, and the path $P' = (t, x_1, \ldots, y)$ is a path of length less than ℓ, which has at least 2 edges with $(y, v) \in E(G)$. Therefore, by induction hypothesis there is an obstruction along with the vertex v, contradicting that $G[V(C) \cup \{v\}]$ is a block graph.

From the above arguments it follows that if v has a neighbour t in block B in C, then v cannot have a neighbour t' in block B', if the shortest path between t, t' has at least 2 edges.

If $(t, t') \in E(G)$, then t, t' are contained in some block $\hat{B}$. If v is adjacent to any other vertex u not in $V(\hat{B})$ then at most one of (t, u) or (t', u) can be an edge in G, since t, t' and u are in different blocks (and $G - (S_v \cup \{v\})$ is a block graph). If there is an edge, say (t, u), then $G[\{t, t', u, v\}]$ is a D_4, contradicting that $G[V(C) \cup \{v\}]$ is a block graph. Otherwise, there is a path with at least two edges between u and t. Therefore, by the previous arguments we can find an obstruction along with the vertex v. Therefore, $N_C(v) \subseteq V(\hat{B})$ when $(t, t') \in E(G)$.

Hence, it follows that there is a block $\tilde{B}$ in C such that $N_C(v) \subseteq V(\tilde{B})$.

□

Lemma 5.4. *For every $v \in A$, the number of leaf blocks in $\mathcal{C}_v$ is at most $\mathcal{O}(k)$.*

Proof. Every leaf block must have at least one internal vertex. By Lemma 5.3 we know that neighbours of v are contained in a block of C, where $C \in \mathcal{C}_v$. Therefore, v cannot be adjacent to internal vertices of two leaf blocks in $\mathcal{C}_v$. In other words, v can be adjacent to vertices in at most one leaf block from $\mathcal{C}_v$. But $|\mathcal{C}_v| \leq 33k$, since the Reduction rule 5.6 is not applicable. Therefore, the number of leaf blocks in $G - (S_v \cup \{v\})$ in which v has a neighbour is at most $\mathcal{O}(k)$.

□

Observe that in $G - A$, a vertex $v \in A$ can be adjacent to at most $\mathcal{O}(k)$ leaf blocks by Observation 3 and Lemma 5.4. Also, for a leaf block B in $G - A$, there must be an internal vertex $b \in V(B)$ such that b is adjacent to some vertex in S_v, since the Reduction rule 5.2 is not applicable. Therefore, the number of leaf blocks in $G - A$ is bounded by $\mathcal{O}(k^2)$.

Lemma 5.5. *The number of blocks B in $G - A$ such that the vertex set of B intersects with the vertex set of at least three other block in $G - A$ is $\mathcal{O}(k^2)$.*

Proof. Consider the block forest F_A of the block graph $G - A$. The number of blocks in $G - A$ is at most the number of vertices in F_A. Note that the set of leaves in F_A corresponds to the set of blocks in $G - A$ with at most one cut vertex. The number of leaf blocks is $G - A$ is bounded by $\mathcal{O}(k^2)$, and therefore the number of leaves in F_A is $\mathcal{O}(k^2)$. For a forest the number of vertices of degree at least 3 is bounded by the number of leaves (minus 2). Therefore, the number of degree three vertices in F_A is bounded by $\mathcal{O}(k^2)$. For a block B in $G - A$ which has at least 3 cut vertices, the vertex b corresponding to block B in F_A will be of degree at least 3. Therefore, the number of blocks in $G - A$ with at least three cut vertices is bounded by $\mathcal{O}(k^2)$.

Consider a block B in $G-A$ such that B has exactly 2 cut vertex, but $V(B)$ intersects with at least three blocks in $G - A$. Let b be the vertex corresponding to B in F_A, and v_B, u_B the cut-vertices in B. At least one of v_B, u_B is a cut-vertex in at least two blocks other than B, say v_B is such a cut vertex. If we contract the edge (b, v_B) in F_A to obtain F'_A, then number of leaves and degree 3 vertices in F'_A remains the same, which is bounded by $\mathcal{O}(k^2)$. Furthermore, the contracted vertex b^* is of degree at least 3. There is a bijection f between the vertices corresponding to blocks in F_A and F'_A, where $f(b) = b^*$ and $f(b') = b'$, for all $b' \in V_B(F_A) \setminus \{b\}$, where $V_B(F_A)$ is the set of vertices corresponding to blocks in $G-A$. Observe that this together with the previous argument implies that the number of blocks whose vertex set intersects with vertex set of at least 3 other blocks is bounded by $\mathcal{O}(k^2)$.

□

Let $\mathcal{L}$ be the set of leaf blocks in $G-A$ and $\mathcal{T}$ be the set of blocks in $G-A$ such that each block in $\mathcal{T}$ intersects with at least three other blocks in $G - A$. By Lemmas 5.4 and 5.5, we have that $|\mathcal{L}| \in \mathcal{O}(k^2)$ and $|\mathcal{T}| \in \mathcal{O}(k^2)$.

Let B be a block in $G^* = G - (S_v \cup \{v\})$ such that the vertex set of B has exactly two cut vertices, intersects with exactly two blocks of G^*, and it does not intersect with blocks in $\mathcal{L} \cup \mathcal{T}$. Also, B has a vertex that is a neighbor of v. We call such blocks as *nice degree two blocks* of v. If a block satisfies the above conditions for some vertex $w \in A$, the block is called a *nice degree two block*. We denote the set of nice degree two blocks by $\mathcal{T}_1$.

Lemma 5.6. *Let $G^* = G - (S_v \cup \{v\})$. Then G^* has at most $\mathcal{O}(k)$ nice degree two blocks of v.*

Proof. Recall that $\mathcal{C}_v$ is the set of connected components in $G - (S_v \cup \{v\})$ which have a vertex adjacent to v. From Lemma 5.3, for each of the connected component $C \in \mathcal{C}_v$, there is a block $\tilde{B}$ in C such that $N_C(v) \subseteq V(\tilde{B})$. Consider two nice degree two blocks B and B' in C of v. Let $b \in V(B)$ and $b' \in V(B')$ be the vertices such that $(v, b), (v, b') \in E(G)$. Next, we consider the following cases.

- Consider the case when $b \neq b'$. In this case, $b, b' \in V(\tilde{B})$, where $N_C(v) \subseteq V(\tilde{B})$ and $\tilde{B}$ is a block with exactly 2 cut vertices. Consider a block $\hat{B}$ in C other than B, B', and $\tilde{B}$. Notice that $V(\hat{B}) \cap V(\tilde{B}) = \emptyset$, and therefore, v cannot be adjacent to any vertex in $\hat{B}$. Hence, it follows that the number of blocks in C that contain a neighbor of v is $\mathcal{O}(1)$.

- Next, consider the case when $b = b'$. In this case, b is a cut vertex in both B and B'. Recall that B, B' both have exactly two cut vertices and intersect exactly two

blocks in C. Therefore, one of B, B' must be same as $\tilde{B}$, say $B = \tilde{B}$. But B shares cut vertices with exactly two blocks. Therefore, v can have neighbors in at most $\mathcal{O}(1)$ blocks in C.

But recall that $|\mathcal{C}_v| = \mathcal{O}(k)$. Hence, the claim follows.

□

What remains is to bound the number of blocks which have exactly two cut vertices which are not nice degree two blocks.

Lemma 5.7. *The number of blocks in $G - A$ with exactly two cut vertices is bounded by $\mathcal{O}(k^2)$.*

Proof. From Lemma 5.4, the number of leaf blocks in $G - A$ is bounded by $\mathcal{O}(k^2)$. Let F_A be the block forest of the block graph $G - A$. From Lemmata 5.4 and 5.5, we know that $|\mathcal{L} \cup \mathcal{T}| \in \mathcal{O}(k^2)$. Also, from Lemma 5.6 the number of blocks in $\mathcal{T}_1$ is bounded by $\mathcal{O}(k^2)$.

Let $\mathcal{P}$ be the set of paths in F_A such that the endpoints are vertices corresponding to blocks in $\mathcal{L} \cup \mathcal{T} \cup \mathcal{T}_1$ and all internal block vertices do not correspond to blocks in $\mathcal{L} \cup \mathcal{T} \cup \mathcal{T}_1$. Note that, all internal vertices of such paths have degree exactly two in F_A. Since F_A is a tree, the number of paths in $\mathcal{P}$ is bounded by $\mathcal{O}(k^2)$.

Let $\mathcal{T}_2$ be the set of blocks in $G - A$ which have exactly two cut vertices and are not in $\mathcal{T}_1$. By definitions of F_A and $\mathcal{P}$, the vertex corresponding to a block $B \in \mathcal{T}_2$ must be an internal vertex in some path of $\mathcal{P}$. Consider a path P_A in $\mathcal{P}$. Note that F_A is a bipartite graph with the vertex bipartition as $(\mathcal{B}, V_C)$, where $\mathcal{B}$ is the set of blocks in $G - A$ and V_C is the set of cut vertices in $G - A$. Therefore, in P no two cut vertices in $V(G-A)$ can be adjacent to each other. Similarly, for $b, b' \in \mathcal{B}$, $(b, b') \notin E(P)$. Let $\mathcal{S}$ be the sequence of blocks in the order they appear in the path P_A. In $\mathcal{S}$, every consecutive blocks share a cut vertex. We remove the starting block and the end block from $\mathcal{S}$. We do this because these blocks belong to $\mathcal{L} \cup \mathcal{T} \cup \mathcal{T}_1$. If $\mathcal{S}$ has more than two blocks then, Reduction rule 5.4 would be applicable. Therefore, the number of blocks in $\mathcal{P}$ can be at most $\mathcal{O}(1)$.

The set of blocks in $G - A$ with exactly two cut vertices is contained in $\mathcal{T} \cup \mathcal{T}_1 \cup \mathcal{T}_2$. Hence, it follows that the number of blocks with exactly 2 cut vertices in $G - A$ is bounded by $\mathcal{O}(k^2)$.

□

Now, we have a bound on the total number of blocks in $G - A$.

Lemma 5.8. *Consider a graph G, a positive integer k, and an approximate block vertex deletion set A of size $\mathcal{O}(k)$. If none of the Reduction rules 5.1 to 5.6 are applicable then the number of blocks in $G - A$ is bounded by $\mathcal{O}(k^2)$.*

Proof. Follows from Lemmas 5.4, 5.5 and 5.7.

□

5.3.2 Bounding the number of internal vertices in a maximal clique of the block graph

We start by bounding the number of internal vertices in a maximal 2-connected component of $G - A$. Consider a block B in $G - A$. We partition the set of internal vertices $V_I(B)$ of block B into three sets namely, $\mathcal{B}, \mathcal{R}$, and $\mathcal{I}$ depending on the neighborhood of A in B. We also partition the vertices in A depending on the number of vertices they are adjacent to in B. In Lemma 5.9 we show that the number of internal vertices in a block in $G - A$ is upper bounded by $\mathcal{O}(k^2)$. We do so by partitioning the vertices into different sets and bounding each of these sets separately.

Lemma 5.9. *Let (G, k) be an instance to* BGVD *and A be an approximate block vertex deletion set of G of size $\mathcal{O}(k)$. If none of the Reduction rules 5.1 to 5.6 are applicable then the number of internal vertices in a block B of $G - A$ is bounded by $\mathcal{O}(k^2)$.*

Proof. Let $A_{\leq 2k+1} = \{v \in A \mid |N_B(v)| \leq 2k + 1\}$ and $A_{>2k+1} = A \setminus A_{\leq 2k+1}$. For a vertex $u \in V_I(B)$, if u is adjacent to at least one of the vertices in $A_{\leq 2k+1}$ then, we add u to the set $\mathcal{B}$. Note that the number of vertices in $\mathcal{B}$ is bounded by $\mathcal{O}(k^2)$, since each $v \in A_{\leq 2k+1}$ is adjacent to at most $2k + 1$ vertices in $V_I(B)$. Also, for a vertex $u \in V_I(B) \setminus \mathcal{B}$, $N(u) \cap A_{\leq 2k+1} = \emptyset$.

For each vertex $u \in V_I(B) \setminus \mathcal{B}$, if u is not adjacent to at least one vertex in $A_{>2k+1}$ then, we add u to the set $\mathcal{R}$. For $v \in A_{>2k+1}$, let Q_v be the set of vertices in $V_I(B) \setminus \mathcal{B}$ which are not adjacent to v. Note that $|Q_v| \leq k$ otherwise, for each pair of vertices $t_1, t_2 \in N_B(v)$ along with one vertex in Q_v we get $k + 1$ vertex disjoint obstruction, namely D_4, intersecting only at v, which contradicts the non-applicability of Reduction rule 5.5. Therefore, the number of vertices in $\mathcal{R}$ is bounded by $\mathcal{O}(k^2)$.

Let $\mathcal{I} = V_I(B) \setminus (\mathcal{B} \cup \mathcal{R})$. Note that the vertices in $\mathcal{I}$ induce a clique. Furthermore, for each $w \in \mathcal{I}$, $N(w) \cap A_{\leq 2k+1} = \emptyset$ and $N(w) \cap A_{>2k+1} = A_{>2k+1}$. Therefore, each $w, w' \in \mathcal{I}$ are twins. In fact, $\mathcal{I}$ is a set of twins. If $|\mathcal{I}| > k + 1$ then Reduction rule 5.3 would be applicable. Therefore, $|\mathcal{I}| \leq k + 1$. But, $|V_I(B)| = |\mathcal{B}| + |\mathcal{R}| + |\mathcal{I}|$, which is bounded by $\mathcal{O}(k^2)$.

□

We wrap up our arguments to show a $\mathcal{O}(k^4)$ sized vertex kernel for BGVD, and hence prove Theorem 5.2.

Proof of Theorem 5.2. Let (G, k) be an instance to BGVD and let A be an approximate block vertex deletion set of G of size $\mathcal{O}(k)$. Also, assume that none of the Reduction rules 5.1 to 5.6 are applicable. By Lemma 5.8, the number of blocks in $G - A$ is bounded by $\mathcal{O}(k^2)$. From Lemma 5.9, the number of internal vertices in a block of $G - A$ is bounded by $\mathcal{O}(k^2)$. Also note that the number of cut-vertices in $G - A$ is bounded by the number of blocks in $G - A$, i.e. $\mathcal{O}(k^2)$. The number of vertices in $G - A$ is sum of the number of internal vertices in $G - A$ and the number of cut vertices in $G - A$. Therefore, $|V(G)| = |V(G - A)| + |A| = (\mathcal{O}(k^2) \cdot \mathcal{O}(k^2) + \mathcal{O}(k^2)) + \mathcal{O}(k) = \mathcal{O}(k^4)$.

□

Chapter 6

Approximation Algorithms for Weighted $\mathcal{F}$-Vertex Deletion

In this chapter, we explore the approximability of Weighted $\mathcal{F}$-Vertex Deletion for several families $\mathcal{F}$ of graphs, and design $\mathcal{O}(\log^{\mathcal{O}(1)} n)$-factor approximation algorithms for these problems. More precisely, we look at the following results.

- Let $\mathscr{F}$ be a finite set of graphs that includes a planar graph. Let $\mathcal{F} = \mathscr{G}(\mathscr{F})$ be the family of graphs such that every graph $H \in \mathscr{G}(\mathscr{F})$ does not contain a graph from $\mathscr{F}$ as a minor. The vertex deletion problem corresponding to $\mathcal{F} = \mathscr{G}(\mathscr{F})$ is known as the Weighted Planar $\mathscr{F}$-Minor-Free Deletion (WP$\mathscr{F}$-MFD). The WP$\mathscr{F}$-MFD problem is a very generic problem and by selecting different sets of forbidden minors $\mathscr{F}$, one can obtain various fundamental problems such as Weighted Vertex Cover, Weighted Feedback Vertex Set or Weighted Treewidth η-Deletion. We look at a randomized $\mathcal{O}(\log^{1.5} n)$-factor (deterministic $\mathcal{O}(\log^2 n)$-factor) approximation algorithm for WP$\mathscr{F}$-MFD, for any finite $\mathscr{F}$ that contains a planar graph.

  We remark that a different approximation algorithm for the same class of problems with a slightly better approximation ratio of $\mathcal{O}(\log n \log\log n)$ follows from recent work of Bansal et al. [BRU17].
- We give an $\mathcal{O}(\log^2 n)$-factor approximation algorithm for Weighted Chordal Vertex Deletion (WCVD), the vertex deletion problem corresponding to the family of chordal graphs. On the way to this algorithm, we also obtain a constant factor approximation algorithm for Weighted Multicut in chordal graphs.
- We give an $\mathcal{O}(\log^3 n)$-factor approximation algorithm for Weighted Distance Hereditary Vertex Deletion (WDHVD). This problem is also known as Weighted Rankwidth-1 Vertex Deletion (WR-1VD). This is the vertex deletion problem corresponding to the family of distance hereditary graphs, or equivalently graphs of rankwidth 1.

All the algorithms follow the same recursive scheme, that find "well structured balanced separators" in the graph by exploiting the properties of the family $\mathcal{F}$. In the following, we first describe the methodology by which we design all these approximation algorithms.

Our Methods. Multicommodity max-flow min-cut theorems are a classical technique in designing approximation algorithms, which was pioneered by Leighton and Rao in their seminal paper [LR99]. This approach can be viewed as using balanced vertex (or edge) separators in a graph to obtain a divide-and-conquer approximation algorithm. In a typical application, the optimum solution S, forms a balanced separator of the graph. Thus, the idea is to find an minimum cost balanced separator W of the graph and add it to the solution, and then recursively solve the problem on each of the connected components. This leads to an $\mathcal{O}(\log^{\mathcal{O}(1)} n)$-factor approximation algorithm for the problem in question.

Our recursive scheme is a strengthening of this approach which exploits the structural properties of the family $\mathcal{F}$. Here, the optimum solution S^* need not be a balanced separator of the graph. Indeed, a balanced separator of the graph could be much larger than S^*. Rather, S^* along with a possibly large but well-structured subset of vertices X, forms a balanced separator of the graph. We then exploit the presence of such a balanced separator in the graph to compute an approximate solution. Consider a family $\mathcal{F}$ for which Weighted $\mathcal{F}$-Vertex Deletion is amenable to this approach, and let G be an instance of this problem. Let S be the approximate solution that we will compute. The approximation algorithm has the following steps:

1. Find a well-structured set X, such that $G - X$ has a balanced separator W which is not too costly.
2. Next, compute the balanced separator W of $G - X$ using the known factor $\mathcal{O}(\sqrt{\log n})$-approximation algorithm (or deterministic $\mathcal{O}(\log n)$-approximation algorithm) for Weighted Vertex Separators [FHL08, LR99]. Then add W into the solution set S and recursively solve the problem on each connected component of $G-(X \cup S)$. Let $S_1, \cdots, S_\ell$ be the solutions returned by the recursive calls. We add $S_1, \cdots, S_\ell$ to the solution S.
3. Finally, we add X back into the graph and consider the instance $(G - S) \cup X$. Observe that, $V(G - S)$ can be partitioned into $V' \uplus X$, where $G[V']$ belongs to $\mathcal{F}$ and X is a well-structured set. We call such instances, the *special case* of Weighted $\mathcal{F}$-Vertex Deletion. We apply an approximation algorithm that exploits the structural properties of the special case to compute a solution.

Now consider the problem of finding the structure X. One way is to enumerate all the candidates for X and then pick the one where $G - X$ has a balanced vertex separator of least cost — this separator plays the role of W. However, the number of candidates for X in a graph could be too many to enumerate in polynomial time. For example, in the case of Weighted Chordal Vertex Deletion, the set X will be a clique in the graph, and the number of maximal cliques in a graph on n vertices could be as many as $3^{\frac{n}{3}}$ [MM65]. Hence, we cannot enumerate and test every candidate structure in polynomial time. However, we can exploit certain structural properties of family $\mathcal{F}$, to reduce the number of candidates for X in the graph. In our problems, we "tidy up" the graph by removing "short obstructions" that forbid the graph from belonging to the family $\mathcal{F}$. Then one can obtain an upper bound on the number of candidate structures. In the above example, recall that a graph G is chordal if and only if there are no induced cycles of length 4 or more. It is known that a graph G without any induced cycle of length 4 has at most $\mathcal{O}(n^2)$ maximal cliques [Far89]. Observe that, we can greedily compute a

set of vertices which intersects all induced cycles of length 4 in the graph. Therefore, at the cost of factor 4 in the approximation ratio, we can ensure that the graph has only polynomially many maximal cliques. Hence, one can enumerate all maximal cliques in the remaining graph [TIAS77] to test for X.

Next consider the task of solving an instance of the special case of the problem. We again apply a recursive scheme, but now with the advantage of a much more structured graph. By a careful modification of an LP solution to the instance, we eventually reduce it to instances of Weighted Multicut. In the above example, for Weighted Chordal Vertex Deletion we obtain instances of Weighted Multicut on a chordal graph. We follow this approach for all three problems that we study in this chapter. We believe our recursive scheme can be applied to obtain $\mathcal{O}(\log^{\mathcal{O}(1)} n)$-approximation algorithms for Weighted $\mathcal{F}$-Vertex (Edge) Deletion corresponding to several other graph families $\mathcal{F}$.

6.1 Approximation Algorithm for WP$\mathscr{F}$-MFD

In this section, we consider the problem Weighted Planar $\mathscr{F}$-Minor-Free Deletion. Let $\mathscr{F}$ be a finite set of graphs containing a planar graph. Formally, Weighted Planar $\mathscr{F}$-Minor-Free Deletion is defined as follows.

Weighted Planar $\mathscr{F}$-Minor-Free Deletion (WP$\mathscr{F}$-MFD)
Input: A graph G and a weight function $w : V(G) \to \mathbb{Q}$.
Output: Find a minimum weight subset $S \subseteq V(G)$ such that $G - S$ does not contain any graph in $\mathscr{F}$ as a minor.

The WP$\mathscr{F}$-MFD problem is a very generic problem that encompasses several known problems. Given a graph family $\mathcal{F}$, by $\mathsf{ForbidMinor}(\mathcal{F})$ we denote the family of graphs such that $G \in \mathcal{F}$ if and only if G does not contain any graph in $\mathsf{ForbidMinor}(\mathcal{F})$ as a minor. By the celebrated Graph Minor Theorem of Robertson and Seymour, every minor closed family $\mathcal{F}$ is characterized by a finite family of forbidden minors [RS04]. That is, $\mathsf{ForbidMinor}(\mathcal{F})$ has finite size. Indeed, the size of $\mathsf{ForbidMinor}(\mathcal{F})$ depends on the family $\mathcal{F}$. Now for a finite collection of graphs $\mathscr{F}$, as above, we may define the Weighted $\mathscr{F}$-Minor-Free Deletion problem. And observe that, even though the definition of Weighted $\mathscr{F}$-Minor-Free Deletion we only consider finite sized $\mathscr{F}$, this problem actually encompasses deletion to every minor closed family of graphs. Let $\mathscr{G}$ be the set of all finite (undirected) graphs, and let $\mathscr{L}$ be the family of all finite subsets of $\mathscr{G}$. Thus, every element $\mathscr{F} \in \mathscr{L}$ is a finite set of graphs, and throughout the chapter we assume that $\mathscr{F}$ is explicitly given. We show that when $\mathscr{F} \in \mathscr{L}$ contains at least one planar graph, then it is possible to obtain an $\mathcal{O}(\log^{\mathcal{O}(1)} n)$-factor approximation algorithm for WP$\mathscr{F}$-MFD.

The case where $\mathscr{F}$ contains a planar graph, while being considerably more restricted than the general case, already encompasses a number of the well-studied instances of WP$\mathscr{F}$-MFD. For example, when $\mathscr{F} = \{K_2\}$, a complete graph on two vertices, this is the Weighted Vertex Cover problem. When $\mathcal{F} = \{C_3\}$, a cycle on three vertices, this is the Weighted Feedback Vertex Set problem. Another fundamental problem, which is also a special case of WP$\mathscr{F}$-MFD, is Weighted Treewidth-η Vertex Deletion or Weighted η-Transversal. Here the task is to delete a minimum weight vertex subset to obtain a graph of treewidth at most η. Since any graph of

treewidth η excludes a $(\eta+1)\times(\eta+1)$ grid as a minor, we have that the set $\mathscr{F}$ of forbidden minors of treewidth η graphs contains a planar graph. Treewidth-η Vertex Deletion plays an important role in generic efficient polynomial time approximation schemes based on Bidimensionality theory [FLRS11, FLS12]. Among other examples of Planar $\mathscr{F}$-Minor-Free Deletion problems that can be found in the literature on approximation and parameterized algorithms, are the cases of $\mathscr{F}$ being $\{K_{2,3}, K_4\}$, $\{K_4\}$, $\{\theta_c\}$, and $\{K_3, T_2\}$, which correspond to removing vertices to obtain an outerplanar graph, a series-parallel graph, a diamond graph, and a graph of pathwidth 1, respectively.

Apart from the case of Weighted Vertex Cover [BYE81, NJ74] and Weighted Feedback Vertex Set [BBF99, BYGNR98], there was not much progress on approximability/non-approximability of WP$\mathscr{F}$-MFD until the work of Fiorini, Joret, and Pietropaoli [FJP10], which gave a constant factor approximation algorithm for the case of WP$\mathscr{F}$-MFD where $\mathscr{F}$ is a diamond graph, i.e., a graph with two vertices and three parallel edges. In 2011, Fomin et al. [FLM$^+$16] considered Planar $\mathscr{F}$-Minor-Free Deletion (i.e. the unweighted version of WP$\mathscr{F}$-MFD) in full generality and designed a randomized (deterministic) $\mathcal{O}(\log^{1.5} n)$-factor ($\mathcal{O}(\log^2 n)$-factor) approximation algorithm for it. Later, Fomin et al. [FLMS12] gave a randomized constant factor approximation algorithm for Planar $\mathscr{F}$-Minor-Free Deletion. The algorithm presented in this section for WP$\mathscr{F}$-MFD extends this result to the weighted setting, at the cost of increasing the approximation factor to $\log^{\mathcal{O}(1)} n$.

Theorem 6.1. *For every set $\mathscr{F} \in \mathscr{L}$,* WP$\mathscr{F}$-MFD *admits a randomized (deterministic) $\mathcal{O}(\log^{1.5} n)$-factor ($\mathcal{O}(\log^2 n)$-factor) approximation algorithm.*

In this section we prove Theorem 6.1. We can assume that the weight $w(v)$ of each vertex $v \in V(G)$ is positive, else we can insert v into any solution. Below we state a result from [RS86], which will be useful in the algorithm.

Proposition 6.1 ([RS86])**.** *Let $\mathscr{F}$ be a finite set of graphs such that $\mathscr{F}$ contains a planar graph. Then, any graph G that excludes any graph from $\mathscr{F}$ as a minor satisfies $\mathsf{tw}(G) \leq c = c(\mathscr{F})$.*

We let $c = c(\mathscr{F})$ to be the constant returned by Proposition 6.1. The approximation algorithm for WP$\mathscr{F}$-MFD comprises of two components. The first component handles the special case where the vertex set of input graph G can be partitioned into two sets C and X such that $|C| \leq c+1$ and $H = G[X]$ is an $\mathscr{F}$-minor free graph. We note that there can be edges between vertices in C and vertices in H. For these special instances, in polynomial time we can compute the size of the optimum solution and a set realizing it.

The second component is a recursive algorithm that solves general instances of the problem. Here, we gradually disintegrate the general instance until it becomes an instance of the special type where we can resolve it in polynomial time. More precisely, for each guess of $c+1$ sized subgraph M of G, we find a small separator S (using an approximation algorithm) that together with M breaks the input graph into two graphs significantly smaller than their origin. It first removes $M \cup S$, and solves each of the two resulting subinstances by calling itself recursively; then, it inserts M back into the graph, and uses the solutions it obtained from the recursive calls to construct an instance of the special case which is then solved by the first component.

6.1.1 Constant sized graph + $\mathscr{F}$-minor free graph

We first handle the special case where the input graph G consists of a graph M of size at most $c+1$ and an $\mathscr{F}$-minor free graph H. We refer to this algorithm as Special-WP. More precisely, along with the input graph G and the weight function w, we are also given a graph M with at most $c+1$ vertices and an $\mathscr{F}$-minor free graph H such that $V(G) = V(M) \cup V(H)$, where the vertex-sets $V(M)$ and $V(H)$ are disjoint. Note that the edge-set $E(G)$ may contain edges between vertices in M and vertices in H. We will show that such instances may be solved optimally in polynomial time. We start with the following easy observation.

Observation 4. *Let G be a graph with $V(G) = X \uplus Y$, such that $|X| \leq c+1$ and $G[Y]$ is an $\mathscr{F}$-minor free graph. Then, the treewidth of G is at most $2c+1$.*

Lemma 6.1. *Let G be a graph of treewidth t with a non-negative weight function w on the vertices, and let $\mathscr{F}$ be a finite family of graphs. Then, one can compute a minimum weight vertex set S such that $G - S$ is $\mathscr{F}$-minor free, in time $f(q,t) \cdot n$, where n is the number of vertices in G and q is a constant that depends only on $\mathscr{F}$.*

Proof. This follows from the fact that finding such a set S is expressible as an MSO-optimization formula ϕ whose length, q, depends only on the family $\mathscr{F}$ [FLMS12]. Then, by Theorem 7 [BPT92], we can compute an optimal sized set S in time $f(q,t) \cdot n$. □

Now, we apply the above lemma to the graph G and the family $\mathscr{F}$, and obtain a minimum weight set S such that $G - S$ is $\mathscr{F}$-minor free.

6.1.2 General graphs

We proceed to handle general instances by developing a $d \cdot \log^2 n$-factor approximation algorithm for WP$\mathscr{F}$-MFD, Gen-WP-APPROX, thus proving the correctness of Theorem 6.1. The exact value of the constant d will be determined later.

Recursion. We define each call to our algorithm Gen-WP-APPROX to be of the form (G', w'), where (G', w') is an instance of WP$\mathscr{F}$-MFD such that G' is an induced subgraph of G, and we denote $n' = |V(G')|$.

Goal. For each recursive call Gen-WP-APPROX(G', w'), we aim to prove the following.

Lemma 6.2. Gen-WP-APPROX *returns a solution that is at least* opt *and at most $\frac{d}{2} \cdot \log^2 n' \cdot$ opt. Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution.*

At each recursive call, the size of the graph G' becomes smaller. Thus, when we prove that Lemma 6.2 is true for the current call, we assume that the approximation factor is bounded by $\frac{d}{2} \cdot \log^2 \widehat{n} \cdot$ opt for any call where the size $\widehat{n}$ of the vertex-set of its graph is strictly smaller than n'.

Termination. In polynomial time we can test whether G' has a minor $F \in \mathscr{F}$ [RS95]. Furthermore, for each $M \subseteq V(G)$ of size at most $c+1$, we can check if $G - M$ has a minor $F \in \mathscr{F}$. If $G - M$ is $\mathscr{F}$-minor free then we are in a special instance, where $G - M$ is $\mathscr{F}$ minor free and M is a constant sized graph. We optimally resolve this instance

in polynomial time using the algorithm Special-WP. Since we output an optimal sized solution in the base cases, we thus ensure that at the base case of our induction Lemma 6.2 holds.

Recursive Call. For the analysis of a recursive call, let S^* denote a hypothetical set that realizes the optimal solution opt of the current instance (G', w'). Let (F, β) be a forest decomposition of $G' - S^*$ of width at most c, whose existence is guaranteed by Proposition 6.1. Using standard arguments on forests we have the following observation.

Observation 5. *There exists a node $v \in V(F)$ such that $\beta(v)$ is a balanced separator for $G' - S^*$.*

From Observation 5 we know that there exists a node $v \in V(F)$ such that $\beta(v)$ is a balanced separator for $G' - S^*$. This together with the fact that $G' - S^*$ has treewidth at most c results in the following observation.

Observation 6. *There exist a subset $M \subseteq V(G')$ of size at most $c+1$ and a subset $S \subseteq V(G') \setminus M$ of weight at most opt such that $M \cup S$ is a balanced separator for G'.*

This gives us a polynomial time algorithm as stated in the following lemma.

Lemma 6.3. *There is a deterministic (randomized) algorithm which in polynomial-time finds $M \subseteq V(G')$ of size at most $c+1$ and a subset $S \subseteq V(G') \setminus M$ of weight at most $q \cdot \log n' \cdot \mathsf{opt}$ ($q^* \cdot \sqrt{\log n'} \cdot \mathsf{opt}$) for some fixed constant q (q^*) such that $M \cup S$ is a balanced separator for G'.*

Proof. Note that we can enumerate every $M \subseteq V(G')$ of size at most $c+1$ in time $\mathcal{O}(n^c)$. For each such M, we can either run the randomized $q^* \cdot \sqrt{\log n'}$-factor approximation algorithm by Feige et al. [FHL08] or the deterministic $q \cdot \log n'$-factor approximation algorithm by Leighton and Rao [LR99] to find a balanced separator S_M of $G' - M$. Here, q and q^* are fixed constants. By Observation 6, there is a set S in $\{S_M \mid M \subseteq V(G') \text{ and } M \leq c+1\}$ such that $w(S) \leq q \cdot \log n' \cdot \mathsf{opt}$ ($w(S) \leq q^* \cdot \sqrt{\log n'} \cdot \mathsf{opt}$). Thus, the desired output is a pair (M, S) where M is one of the vertex subset of size at most $c+1$ such that $S_M = S$. □

We call the algorithm in Lemma 6.3 to obtain a pair (M, S). Since $M \cup S$ is a balanced separator for G', we can partition the set of connected components of $G' - (M \cup S)$ into two sets, $\mathcal{A}_1$ and $\mathcal{A}_2$, such that for $V_1 = \bigcup_{A \in \mathcal{A}_1} V(A)$ and $V_2 = \bigcup_{A \in \mathcal{A}_2} V(A)$ it holds that $n_1, n_2 \leq \frac{2}{3} n'$ where $n_1 = |V_1|$ and $n_2 = |V_2|$. We remark that we use different algorithms for finding a balanced separator in Lemma 6.3 based on whether we are looking for a randomized algorithm or a deterministic algorithm.

Next, we define two inputs of (the general case of) WP-$\mathscr{F}$-MFD: $I_1 = (G'[V_1], w'|_{V_1})$ and $I_2 = (G'[V_2], w'|_{V_2})$. Let opt_1 and opt_2 denote the optimal solutions to I_1 and I_2, respectively. Observe that since $V_1 \cap V_2 = \emptyset$, it holds that $\mathsf{opt}_1 + \mathsf{opt}_2 \leq \mathsf{opt}$. We solve each of the subinstances by recursively calling algorithm Gen-WP-APPROX. By the inductive hypothesis, we thus obtain two sets, S_1 and S_2, such that $G'[V_1] - S_1$ and $G'[V_2] - S_2$ are $\mathscr{F}$-minor free graphs, and $w'(S_1) \leq \frac{d}{2} \cdot \log^2 n_1 \cdot \mathsf{opt}_1$ and $w'(S_2) \leq \frac{d}{2} \cdot \log^2 n_2 \cdot \mathsf{opt}_2$.

We proceed by defining an input of the special case of WP-$\mathscr{F}$-MFD: $J = (G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)], w'|_{(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)})$. Observe that $G'[V_1 \setminus S_1]$ and $G'[V_2 \setminus S_2]$ are $\mathscr{F}$-minor free graphs and there are no edges between vertices in V_1 and vertices in

V_2 in $G' - M$, and M is of constant size. Therefore, we resolve this instance by calling algorithm Special-WP. We thus obtain a set, $\widehat{S}$, such that $G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2 \cup \widehat{S})]$ is a $\mathscr{F}$-minor graph, and $w'(\widehat{S}) \leq \mathsf{opt}$ (since $|(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)| \leq n'$ and the optimal solution of each of the special subinstances is at most opt).

Observe that any obstruction in $G' - S$ is either completely contained in $G'[V_1]$, or completely contained in $G'[V_2]$, or it contains at least one vertex from M. This observation, along with the fact that $G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2 \cup \widehat{S})]$ is a $\mathscr{F}$-minor free graph, implies that $G' - T$ is a $\mathscr{F}$-minor free graph where $T = S \cup S_1 \cup S_2 \cup \widehat{S}$. Thus, it is now sufficient to show that $w'(T) \leq \frac{d}{2} \cdot (\log n')^2 \cdot \mathsf{opt}$.

By the discussion above, we have that

$$\begin{aligned} w'(T) &\leq w'(S) + w'(S_1) + w'(S_2) + w'(\widehat{S}) \\ &\leq q \cdot \log n' \cdot \mathsf{opt} + \tfrac{d}{2} \cdot ((\log n_1)^2 \cdot \mathsf{opt}_1 + (\log n_2)^2 \cdot \mathsf{opt}_2) + \mathsf{opt} \end{aligned}$$

Recall that $n_1, n_2 \leq \frac{2}{3}n'$ and $\mathsf{opt}_1 + \mathsf{opt}_2 \leq \mathsf{opt}$. Thus, we have that

$$\begin{aligned} w'(T) &< q \cdot \log n' \cdot \mathsf{opt} + \tfrac{d}{2} \cdot (\log \tfrac{2}{3}n')^2 \cdot \mathsf{opt} + \mathsf{opt} \\ &< \tfrac{d}{2} \cdot (\log n')^2 \cdot \mathsf{opt} + \log n' \cdot \mathsf{opt} \cdot (q + 1 + \tfrac{d}{2} \cdot (\log \tfrac{3}{2})^2 - \tfrac{d}{2} \cdot 2 \cdot \log \tfrac{3}{2}). \end{aligned}$$

Overall, we conclude that to ensure that $w'(T) \leq \frac{d}{2} \cdot \log^2 n' \cdot \mathsf{opt}$, it is sufficient to ensure that $q + 1 + \frac{d}{2} \cdot (\log \frac{3}{2})^2 - \frac{d}{2} \cdot 2 \cdot \log \frac{3}{2} \leq 0$, which can be done by fixing $d = \dfrac{2}{2 \log \frac{3}{2} - (\log \frac{3}{2})^2} \cdot (q + 1)$.

If we use the $\mathcal{O}(\sqrt{\log n})$-factor approximation algorithm by Feige et al. [FHL08] for finding a balance separator in Lemma 6.3, then we can do the analysis similar to the deterministic case and obtain a randomized factor-$\mathcal{O}(\log^{1.5} n)$approximation algorithm for WP$\mathscr{F}$-MFD.

6.2 Approximation Algorithm for Weighted Chordal Vertex Deletion

In this section, we look at the problem Weighted Chordal Vertex Deletion, which is formally defined below.

> Weighted Chordal Vertex Deletion (WCVD)
> **Input:** A graph G and a weight function $w : V(G) \to \mathbb{Q}$.
> **Output:** Find a minimum weight subset $S \subseteq V(G)$ such that $G - S$ is a chordal graph.

Prior to the work which is presented here, only two non-trivial approximation algorithms for CVD were known. The first one, by Jansen and Pilipczuk [JP17], is a deterministic $\mathcal{O}(\mathsf{opt}^2 \log \mathsf{opt} \log n)$-factor approximation algorithm, and the second one, by Agrawal et al. [ALM+17a], is a deterministic $\mathcal{O}(\mathsf{opt} \log^2 n)$-factor approximation algorithm. The second result implies that CVD admits an $\mathcal{O}(\sqrt{n} \log n)$-factor approximation algorithm.[1] The focus of this section is the proof of the following theorem.

[1] If $\mathsf{opt} \geq \sqrt{n}/\log n$, we output a greedy solution to the input graph, and otherwise we have that $\mathsf{opt} \log^2 n \leq \sqrt{n} \log n$, hence we call the $\mathcal{O}(\mathsf{opt} \log^2 n)$-factor approximation algorithm.

Theorem 6.2. CVD *admits a deterministic* $\mathcal{O}(\log^2 n)$*-factor approximation algorithm.*

We assume that the weight $w(v)$ of each vertex $v \in V(G)$ is positive, else we can insert v into any solution. Roughly speaking, our approximation algorithm consists of two components. The first component handles the special case where the input graph G consists of a clique C and a chordal graph H. Here, we also assume that the input graph has no "short" chordless cycle. This component is comprised of a recursive algorithm that is based on the method of divide and conquer. The algorithm keeps track of a fractional solution $\mathbf{x}$ of a special form that it carefully manipulated at each recursive call, and which is used to analyze the approximation ratio. In particular, we ensure that $\mathbf{x}$ does not assign high values, and that it assigns 0 to vertices of the clique C as well as vertices of some other cliques. To divide a problem instance into two instances, we find a maximal clique M of the chordal graph H that breaks H into two "simpler" chordal graphs. The clique C remains intact at each recursive call, and the maximal clique M is also a part of both of the resulting instances. Thus, to ensure that we have simplified the problem, we measure the complexity of instances by examining the maximum size of an independent set of their graphs. Since the input graph has no "short" chordless cycle, the maximum depth of the recursion tree is bounded by $\mathcal{O}(\log n)$. Moreover, to guarantee that we obtain instances that are independent, we incorporate multicut constraints while ensuring that we have sufficient "budget" to satisfy them. We ensure that these multicut constraints are associated with chordal graphs, which allows us to utilize the algorithm we design in Section 6.3. We note that the problem Weighted Multicut takes as an input a graph G, a weight function $w : V(G) \to \mathbb{Q}$, and a set $T = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k pairs in vertices of G, and the objective is to output a minimum weight subset $S \subseteq V(G)$ such that for any pair $(s_i, t_i) \in \mathcal{T}$, $G - S$ does not have any path between s_i and t_i.

The second component is a recursive algorithm that solves general instances of the problem. Initially, it easily handles "short" chordless cycles. Then, it gradually disintegrates a general instance until it becomes an instance of the special form that can be solved using the first component. More precisely, given a problem instance, the algorithm divides it by finding a maximal clique M (using an exhaustive search which relies on the guarantee that G has no "short" chordless cycle) and a small separator S (using an approximation algorithm) that together break the input graph into two graphs significantly smaller than their origin. It first removes $M \cup S$ and solves each of the two resulting subinstances by calling itself recursively; then, it inserts M back into the graph, and uses the solutions it obtained from the recursive calls to construct an instance of the special case solved by the first component.

6.2.1 Clique+Chordal graph

In this subsection, we handle the special case where the input graph G consists of a clique C and a chordal graph H. More precisely, along with the input graph G and the weight function w, we are also given a clique C an a chordal graph H such that $V(G) = V(C) \cup V(H)$, where the vertex-sets $V(C)$ and $V(H)$ are disjoint. Here, we also assume that G has no chordless cycle on at most 48 vertices. Note that the edge-set $E(G)$ may contain edges between vertices in C and vertices in H. We call this special case the *Clique+Chordal special case*. Our objective is to prove the following result.

Lemma 6.4. *The Clique+Chordal special case of* WCVD *admits an $\mathcal{O}(\log n)$-factor approximation algorithm.*

We assume that $n \geq 64$,[2] else the input instance can be solve by brute-force. Let c be a fixed constant (to be determined). In the rest of this subsection, we design a $c \cdot \log n$-factor approximation algorithm for the Clique+Chordal special case of WCVD.

Recursion. Our approximation algorithm is a recursive algorithm. We call our algorithm CVD-APPROX, and define each call to be of the form $(G', w', C, H', \mathbf{x})$. Here, G' is an induced subgraph of G such that $V(C) \subseteq V(G')$, and H' is an induced subgraph of H. The argument $\mathbf{x}$ is discussed below. We remark that we continue to use n to refer to the size of the vertex-set of the input graph G rather than the current graph G'.

Arguments. While the execution of our algorithm progresses, we keep track of two arguments: the size of a maximum independent set of the current graph G', denoted by $\alpha(G')$, and a fractional solution $\mathbf{x}$. Due to the special structure of G', the computation of $\alpha(G')$ is simple:

Observation 7. *The measure $\alpha(G')$ can be computed in polynomial time.*

Proof. Any maximum independent set of G' consists of at most one vertex from C and an independent set of H'. It is well known that the computation of the size of a maximum independent set of a chordal graph can be performed in polynomial time [Gol04]. Thus, we can compute $\alpha(H')$ in polynomial time. Next, we iterate over every vertex $v \in V(C)$, and we compute $\alpha_v = \alpha(\widehat{H}) + 1$ for the graph $\widehat{H} = H' \setminus N_{G'}(v)$ in polynomial time (since $\widehat{H}$ is a chordal graph). Overall, we return $\max\{\alpha(H'), \max_{v \in V(C)}\{\alpha_v\}\}$. □

The necessity of tracking $\alpha(G')$ stems from the fact that our recursive algorithm is based on the method of divide-and-conquer, and to ensure that when we divide the current instance into two instances we obtain two "simpler" instances, we need to argue that some aspect of these instances has indeed been simplified. Although this aspect cannot be the size of the instance (since the two instances can share many common vertices), we show that it can be the size of a maximum independent set.

A fractional solution $\mathbf{x}$ is a function $\mathbf{x} : V(G') \to [0, \infty)$ such that for every chordless cycle Q of G' it holds that $\mathbf{x}(V(Q)) \geq 1$. An optimal fractional solution minimizes the weight $w'(\mathbf{x}) = \sum_{v \in V(G')} w'(v) \cdot \mathbf{x}(v)$. Clearly, the solution to the instance (G', w') of WCVD is at least as large as the weight of an optimal fractional solution. Although we initially compute an optimal fractional solution $\mathbf{x}$ (at the initialization phase that is described below), during the execution of our algorithm, we manipulate this solution so it may no longer be optimal. Prior to any call to CVD-APPROX with the exception of the first call, we ensure that $\mathbf{x}$ satisfies the following invariants:

- **Low-Value Invariant**: For any $v \in V(G')$, it holds that $\mathbf{x}(v) < (\frac{c \cdot \log n + 9}{c \cdot \log n})^{\delta} \cdot \frac{1}{c \cdot \log n}$. Here, δ is the depth of the current recursive call in the recursion tree.[3]
- **Zero-Clique Invariant**: For any $v \in V(C)$, it holds that $\mathbf{x}(v) = 0$.

[2] This assumption simplifies some of the calculations ahead.

[3] The depth of the first call is defined to be 1.

Goal. The depth of the recursion tree will be bounded by $q \cdot \log n$ for some fixed constant q. The correctness of this claim is proved when we explain how to perform a recursive call. For each recursive call CVD-APPROX $(G', w', C, H', \mathbf{x})$ with the exception of the first call, we aim to prove the following.

Lemma 6.5. *For any $\delta \in \{1, 2, \ldots, q \cdot \log n\}$, each recursive call to* CVD-APPROX *of depth $\delta \geq 2$ returns a solution that is at least* opt *and at most $(\frac{c \cdot \log n}{c \cdot \log n + 9})^{\delta - 1} \cdot c \cdot \log(\alpha(G')) \cdot w'(\mathbf{x})$. Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution.*

At the initialization phase, we see that in order to prove Lemma 6.4, it is sufficient to prove Lemma 6.5.

Initialization. Initially, the graphs G' and H' are simply set to be the input graphs G and H, and the weight function w' is simply set to be input weight function w. Moreover, we compute an optimal fractional solution $\mathbf{x} = \mathbf{x}_{\text{init}}$ by using the ellipsoid method. Recall that the following claim holds.

Observation 8. *The solution of the instance (G', w') of* WCVD *is lower bounded by $w'(\mathbf{x}_{\text{init}})$.*

Thus, to prove Lemma 6.4, it is sufficient to return a solution that is at least opt and at most $c \cdot \log n \cdot w'(\mathbf{x})$. We would like to proceed by calling our algorithm recursively. For this purpose, we first need to ensure that $\mathbf{x}$ satisfies the low-value and zero-clique invariants, to which end we use the following notation. We let $h(\mathbf{x}) = \{v \in V(G') \mid \mathbf{x}(v) \geq 1/(c \cdot \log n)\}$ denote the set of vertices to which $\mathbf{x}$ assigns high values. Moreover, given a clique M in G', we let $(\mathbf{x} \setminus M) : V(G') \to [0, \infty)$ denote the function that assigns 0 to any vertex in M and $(1 + 3 \cdot \max\limits_{u \in V(G')} \{\mathbf{x}(u)\})\mathbf{x}(v)$ to any other vertex $v \in V(G')$. Now, to adjust $\mathbf{x}$ to be of the desired form both at this phase and at later recursive calls, we rely on the two following lemmata.

Lemma 6.6. *Define $\widehat{G} = G' - h(\mathbf{x})$, $\widehat{w} = w'|_{V(\widehat{G})}$ and $\widehat{\mathbf{x}} = \mathbf{x}|_{V(\widehat{G})}$. Then, $c \cdot \log n \cdot w'(\widehat{\mathbf{x}}) + w'(h(\mathbf{x})) \leq c \cdot \log n \cdot w'(\mathbf{x})$.*

Proof. By the definition of $h(\mathbf{x})$, it holds that $w'(\widehat{\mathbf{x}}) \leq w'(\mathbf{x}) - \frac{1}{c \cdot \log n} \cdot w'(h(\mathbf{x}))$. Thus, $c \cdot \log n \cdot w'(\widehat{\mathbf{x}}) + w'(h(\mathbf{x})) \leq c \cdot \log n \cdot w'(\mathbf{x})$. □

Thus, it is safe to update G' to $G' - h(\mathbf{x})$, w' to $w'|_{V(\widehat{G})}$, H' to $H' - h(\mathbf{x})$ and $\mathbf{x}$ to $\mathbf{x}|_{V(\widehat{G})}$, where we ensure that once we obtain a solution to the new instance, we add $w'(h(\mathbf{x}))$ to this solution and $h(\mathbf{x})$ to the set realizing it.

Lemma 6.7. *Given a clique M in G', the function $(\mathbf{x} \setminus M)$ is a valid fractional solution such that $w'(\mathbf{x} \setminus M) \leq (1 + 3 \cdot \max_{v \in V(G')}\{\mathbf{x}(v)\})w'(\mathbf{x})$.*

Proof. To prove that $(\mathbf{x} \setminus M)$ is a valid fractional solution, let Q be some chordless cycle in G'. We need to show that $(\mathbf{x} \setminus M)(V(Q)) \geq 1$. Since M is a clique, Q can contain at most two vertices from M. Thus, since $\mathbf{x}$ is a valid fractional solution, it holds that $\mathbf{x}(V(Q) \setminus V(M)) \geq 1 - 2 \cdot \max_{u \in V(G')}\{\mathbf{x}(u)\}$. By the definition $(\mathbf{x} \setminus M)$, this fact implies that $(\mathbf{x} \setminus M)(V(Q)) = (\mathbf{x} \setminus M)(V(Q) \setminus V(M)) \geq (1 + 3 \cdot \max_{u \in V(G')}\{\mathbf{x}(u)\})(1 - 2 \cdot \max_{u \in V(G')}\{\mathbf{x}(u)\}) \geq \min\{(1 + \frac{3}{c \cdot \log n})(1 - \frac{2}{c \cdot \log n}), 1\} = \min\{1 + 1/(c \cdot \log n) - 6/((c \cdot \log n)^2), 1\} \geq 1$, where the last inequality relies on the assumption $n \geq 64$.

For the proof of the second part of the claim, note that $w'(\mathbf{x} \setminus M) = (1 + 3 \cdot \max_{v \in V(G')}\{\mathbf{x}(v)\})\ w'(\mathbf{x}|_{V(G')\setminus V(M)}) \leq (1 + 3 \cdot \max_{v \in V(G')}\{\mathbf{x}(v)\})w'(\mathbf{x})$. □

Next, it is possible to call CVD-APPROX recursively with the fractional solution $(\mathbf{x} \setminus C)$. In the context of the low-value invariant, observe that indeed, for any $v \in V(G')$, it now holds that $(\mathbf{x} \setminus C)(v) = (1 + 3 \cdot \max_{u \in V(G')}\{\mathbf{x}(u)\})\mathbf{x}(v) < (1 + \frac{3}{c \cdot \log n}) \cdot \frac{1}{c \cdot \log n} < (\frac{c \cdot \log n + 9}{c \cdot \log n})^{\delta} \cdot \frac{1}{c \cdot \log n}$ for $\delta = 1$. Similarly, by Lemma 6.7, $w'(\mathbf{x} \setminus C) \leq (\frac{c \cdot \log n + 9}{c \cdot \log n})^{\delta} \cdot w'(\mathbf{x})$ for $\delta = 1$. It is also clear that $\alpha(G') \leq n$. Thus, if Lemma 6.5 is true, we return a solution that is at least opt and at most $c \cdot \log n \cdot w(\mathbf{x})$ as desired. In other words, to prove Lemma 6.4, it is sufficient that we next focus only on the proof of Lemma 6.5. The proof of this lemma is done by induction. When we consider some recursive call, we assume that the solutions returned by the additional recursive calls that it performs, which are associated with graphs $\widetilde{G}$ such that $\alpha(\widetilde{G}) \leq \frac{3}{4}\alpha(G')$, comply with the demands of the lemma.

Termination. Once G' becomes a chordal graph, we return 0 as our solution and $\emptyset$ as the set that realizes it. Clearly, we thus satisfy the demands of Lemma 6.5. In fact, we thus also ensure that the execution of our algorithm terminates once $\alpha(G') < 24$:

Lemma 6.8. *If $\alpha(G') < 24$, then G' is a chordal graph.*

Proof. Suppose, by way of contradiction, that G' is not a chordal graph. Then, it contains a chordless cycle Q. Since G' is an induced subgraph of G, where G is assumed to exclude any chordless cycle on at most 48 vertices, we have that $|V(Q)| > 48$. Note that if we traverse Q in some direction, and insert every second vertex on Q into a set, excluding the last vertex in case $|V(Q)|$ is odd, we obtain an independent set. Thus, we have that $\alpha(G) \geq 24$, which is a contradiction. □

Thus, since we will ensure that each recursive calls is associated with a graph whose independence number is at most 3/4 the independence number of the current graph, we have the following observation.

Observation 9. *The maximum depth of the recursion tree is bounded by $q \cdot \log n$ for some fixed constant q.*

Recursive Call. Since H' is a chordal graph, it admits a clique forest (Proposition 2.2). In particular, it contains only $\mathcal{O}(n)$ maximal cliques, and one can find the set of these maximal cliques in polynomial time [Gol04]. By standard arguments on trees, we deduce that H' has a maximal clique M such that after we remove M from G' we obtain two (not necessarily connected) graphs, $\widehat{H}_1$ and $\widehat{H}_2$, such that $\alpha(\widehat{H}_1), \alpha(\widehat{H}_2) \leq \frac{2}{3}\alpha(H')$, and that the clique M can be found in polynomial time. Let $G_1 = G'[V(\widehat{H}_1) \cup V(M) \cup V(C)]$, $H_1 = H'[V(\widehat{H}_1) \cup V(M)]$, $G_2 = G'[V(\widehat{H}_2) \cup V(M) \cup V(C)]$ and $H_2 = H'[V(\widehat{H}_2) \cup V(M)]$, and observe that $\alpha(G_1), \alpha(G_2) \leq \frac{2}{3}\alpha(G') + 2 \leq \frac{3}{4}\alpha(G')$. Here, the last inequality holds because $\alpha(G') \geq 24$, else by Lemma 6.8, the execution should have already terminated.

We proceed by replacing $\mathbf{x}$ by $(\mathbf{x} \setminus M)$. For the sake of clarity, we denote $\mathbf{x}^* = (\mathbf{x} \setminus M)$. By Lemmata 6.6 and 6.7, to prove Lemma 6.5, it is now sufficient to return a solution that is at least opt and at most $(1/(1 + 3 \cdot (\frac{c \cdot \log n + 9}{c \cdot \log n})^{\delta} \cdot \frac{1}{c \cdot \log n})) \cdot (\frac{c \cdot \log n}{c \cdot \log n + 9})^{\delta - 1} \cdot \log \alpha(G') \cdot w(\mathbf{x}^*)$, along with a set that realizes it. Moreover, for any $v \in V(G')$, it

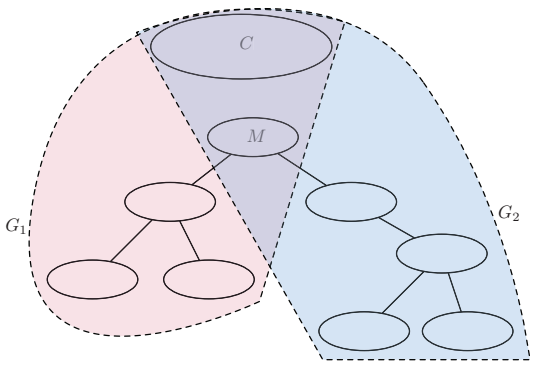

Figure 6.1: Subinstances created by a recursive call

holds that $\mathbf{x}^*(v) < (1 + 3 \cdot (\frac{c \cdot \log n + 9}{c \cdot \log n})^\delta \cdot \frac{1}{c \cdot \log n}) \cdot (\frac{c \cdot \log n + 9}{c \cdot \log n})^\delta \cdot \frac{1}{c \cdot \log n}$. Note that by Observation 9, by setting $c \geq 9q$, we have that $(\frac{c \cdot \log n + 9}{c \cdot \log n})^\delta \leq e < 3$, and therefore $1 + 3 \cdot (\frac{c \cdot \log n + 9}{c \cdot \log n})^\delta \cdot \frac{1}{c \cdot \log n} \leq \frac{c \cdot \log n + 9}{c \cdot \log n}$. In particular, to prove Lemma 6.5, it is sufficient to return a solution that is at least opt and at most $(\frac{c \cdot \log n}{c \cdot \log n + 9})^\delta \cdot \log \alpha(G') \cdot w(\mathbf{x}^*)$.

Next, we define two subinstances, $I_1 = (G_1, w|_{V(G_1)}, C, H_1, \mathbf{x}^*|_{V(G_1)})$ and $I_2 = (G_2, w|_{V(G_2)}, C, H_2, \mathbf{x}^*|_{V(G_2)})$ (see Figure 6.1). We solve each of these subinstances by a recursive call to CVD-APPROX (by the above discussion, these calls are valid — we satisfy the low-value and zero-clique invariants). Thus, we obtain two solutions, s_1 to I_1 and s_2 to I_2, and two sets that realize these solutions, S_1 and S_2. By the inductive hypothesis, we have the following observations.

Observation 10. *$S_1 \cup S_2$ intersects any chordless cycle in G' that lies entirely in either G_1 or G_2.*

Observation 11. *Given $i \in \{1, 2\}$, $s_i \leq (\frac{c \cdot \log n}{c \cdot \log n + 9})^\delta \cdot c \cdot \log(\alpha(G_i)) \cdot w(\mathbf{x}_i^*)$.*

Moreover, since $\mathbf{x}^*(V(C) \cup V(M)) = 0$, we also have the following observation.

Observation 12. $w(\mathbf{x}_1^*) + w(\mathbf{x}_2^*) = w(\mathbf{x}^*)$.

We say that a cycle of G' is *bad* if it is a chordless cycle that belongs entirely to neither G_1 nor G_2 (see Figure 6.2). Next, we show how to intersect bad cycles.

Bad Cycles. For any pair (v, u) of vertices $v \in V(C)$ and $u \in V(M)$, we let $\mathcal{P}_1(v, u)$ denote the set of any (simple) path P_1 between v and u whose internal vertices belong only to G_1 and which does not contain a vertex $v' \in C$ and a vertex $u' \in M$ such that $(v', u') \in E(G')$. Symmetrically, we let $\mathcal{P}_2(v, u)$ denote the set of any path P_2 between v and u whose internal vertices belong only to G_2 and which does not contain a vertex $v' \in C$ and a vertex $u' \in M$ such that $(v', u') \in E(G')$. We note here that when $(v, u) \in E(G')$ then $\mathcal{P}_1(v, u) = \mathcal{P}_2(v, u) = \emptyset$.

We first examine the relation between bad cycles and pairs (v, u) of vertices $v \in V(C)$ and $u \in V(M)$.

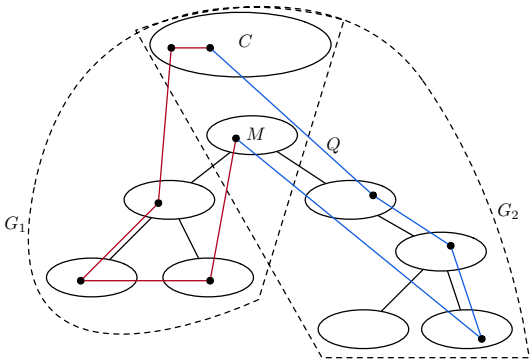

Figure 6.2: An illustration of a bad cycle

Lemma 6.9. *For any bad cycle Q there exist a pair (v, u) of vertices $v \in V(C)$, $u \in V(M)$, a path $P_1 \in \mathcal{P}_1(v, u)$ such that $V(P_1) \subseteq V(Q)$, and a path $P_2 \in \mathcal{P}_2(v, u)$ such that $V(P_2) \subseteq V(Q)$.*

Proof. Let Q be some bad cycle. By the definition of a bad cycle, Q must contain at least one vertex a from $H_1 \setminus V(M)$ and at least one vertex b from $H_2 \setminus V(M)$. Since C and M are cliques, Q can contain at most two vertices from C and at most two vertices from M, and if it contains two vertices from C (resp. M), then these two vertices are neighbors. Moreover, since the set $V(C) \cup V(M)$ contains all vertices common to G_1 and G_2, Q must contain at least one vertex $v \in V(C)$ and at least one vertex $u \in V(M)$ with $(v, u) \notin E(G')$. Overall, we conclude that the subpath of Q between v and u that contains a belongs to $\mathcal{P}_1(v, u)$, while the subpath of Q between v and u that contains b belongs to $\mathcal{P}_2(v, u)$. □

In light Lemma 6.9, to intersect bad cycles, we now examine how the fractional solution $\mathbf{x}^*$ handles pairs (v, u) of vertices $v \in V(C)$ and $u \in V(M)$.

Lemma 6.10. *For each pair (v, u) of vertices $v \in V(C)$ and $u \in V(M)$ with $(v, u) \notin E(G')$, there exists $i \in \{1, 2\}$ such that for any path $P \in \mathcal{P}_i(v, u)$, $\mathbf{x}^*(V(P)) \geq 1/2$.*

Proof. Suppose, by way of contradiction, that the lemma is incorrect. Thus, there exist a pair (v, u) of vertices $v \in V(C)$ and $u \in V(M)$ with $(v, u) \notin E(G')$, a path $P_1 \in \mathcal{P}_1(v, u)$ such that $\mathbf{x}^*(V(P_1)) < 1/2$, and a path $P_2 \in \mathcal{P}_2(v, u)$ such that $\mathbf{x}^*(V(P_2)) < 1/2$. Since $\mathbf{x}^*$ is a valid fractional solution, we deduce that $G'[V(P_1) \cup V(P_2)]$ does not contain any chordless cycle. Consider a shortest subpath $\widehat{P}_1$ of P_1 between a vertex $a_1 \in V(C)$ and a vertex $b_1 \in V(M)$, and a shortest subpath $\widehat{P}_2$ of P_2 between a vertex $a_2 \in V(C)$ and a vertex $b_2 \in V(M)$. Since neither P_1 nor P_2 contains any edge such that one of its endpoints belongs to $V(C)$ while the other endpoint belongs to $V(M)$, we have that $|V(\widehat{P}_1)|, |V(\widehat{P}_2)| \geq 3$. Furthermore, since vertices common in P_1 and P_2 must belong to $V(C) \cup V(M)$, we have that $\widehat{P}_1$ does not contain internal vertices that belong to $\widehat{P}_2$ or adjacent to internal vertices on $\widehat{P}_2$. Overall, since C and M are cliques, we deduce that $G'[V(\widehat{P}_1) \cup V(\widehat{P}_2)]$ contains a chordless cycle. To see this, let a be the vertex closest to b_2 on $\widehat{P}_2$ that is a neighbor of a_1. Observe that a exists as a_1 and a_2 are neighbors, and $a \neq b_2$. Moreover, we assume without loss of generality that if $a = a_2$, then a_2 has

no neighbor on $\widehat{P}_1$ apart from a_1. Now, let b be the vertex closest to a on the subpath of $\widehat{P}_2$ between a and b_2 that is a neighbor of b_1. If $b \neq b_2$, then the vertex-sets of $\widehat{P}_1$ and the subpath of $\widehat{P}_1$ between a and b together induce a chordless cycle. Else, let b' be the vertex closest to a_1 on $\widehat{P}_1$ that is a neighbor of b_2. Then, the vertex-sets of the subpath of $\widehat{P}_1$ between a_1 and b' and the subpath of $\widehat{P}_1$ between a and b_2 together induce a chordless cycle. Since $G'[V(\widehat{P}_1) \cup V(\widehat{P}_2)]$ is an induced subgraph of $G'[V(P_1) \cup V(P_2)]$, we have reached a contradiction. □

Given $i \in \{1, 2\}$, let $2\mathbf{x}_i^*$ denote the fractional solution that assigns to each vertex the value assigned by $\mathbf{x}_i^*$ times 2. Moreover, let $\widehat{G}_1 = G_1 - (V(C) \cup V(M))$ and $\widehat{G}_2 = G_2 - (V(C) \cup V(M))$. Observe that $\widehat{G}_1$ and $\widehat{G}_2$ are chordal graphs. Now, for every pair (v, u) such that $v \in V(C), u \in V(M)$, we perform the following operation. We initialize $\mathcal{T}_1(v, u) = \emptyset$. Next, we consider every pair (v', u') such that $v' \in V(C)$, $u' \in V(M)$, $\{v, v'\} \cap N_{G_1}(u') = \emptyset$ and $\{u, u'\} \cap N_{G_1}(v') = \emptyset$, and insert each pair in $\{(a, b) \mid a \in N_{G_1}(v') \cap V(\widehat{G}_1), b \in N_{G_1}(u') \cap V(\widehat{G}_1), \widehat{G}_1$ has a path between a and $b\}$ into $\mathcal{T}_1(v, u)$. We remark that the vertices in a pair in $\mathcal{T}_1(v, u)$ are not necessarily distinct. The definition of $\mathcal{T}_2(v, u)$ is symmetric to the one of $\mathcal{T}_1(v, u)$.

The following lemma translates Lemma 6.10 into an algorithm.

Lemma 6.11. *For each pair (v, u) of vertices $v \in V(C)$, $u \in V(M)$ and $(v, u) \notin E(G')$, one can compute (in polynomial time) an index $i(v, u) \in \{1, 2\}$ such that for any path $P \in \mathcal{P}_i(v, u)$, $2\mathbf{x}_i^*(V(P)) \geq 1$.*

Proof. Let (v, u) be a pair of vertices such that $v \in V(C)$, $u \in V(M)$ and $(v, u) \notin E(G')$. If there is $i \in \{1, 2\}$ such that $P \in \mathcal{P}_i(v, u) = \emptyset$, then we have trivially obtained the required index which is $i(v, u) = i$. Otherwise, we proceed as follows. For any index $j \in \{1, 2\}$, we perform the following procedure. For each pair $(a, b) \in \mathcal{T}_i(v, u)$, we use Dijkstra's algorithm to compute the minimum weight of a path between a and b in the graph $\widehat{G}_i$ where the weights are given by $2\mathbf{x}_i^*$. In case for every pair (a, b) the minimum weight is at least 1, we have found the desired index $i(v, u)$. Moreover, by Lemma 6.10 and since for all $v' \in V(C) \cup V(M)$ it holds that $\mathbf{x}_1^*(v') = \mathbf{x}_2^*(v') = 0$, for at least one index $j \in \{1, 2\}$, the maximum weight among the minimum weights associated with the pairs (a, b) should be at least 1 (if this value is at least 1 for both indices, we arbitrarily decide to fix $i(v, u) = 1$). □

At this point, we need to rely on approximate solutions to Weighted Multicut in chordal graphs (in this context, we will employ the algorithm given by Theorem 6.5 in Section 6.3). Here, a fractional solution $\mathbf{y}$ is a function $\mathbf{y} : V(G') \to [0, \infty)$ such that for every pair $(s_i, t_i) \in \mathcal{T}$ and any path P between s_i and t_i, it holds that $\mathbf{y}(V(P)) \geq 1$. An optimal fractional solution minimizes the weight $w(\mathbf{y}) = \sum_{v \in V(G')} w(v) \cdot \mathbf{y}(v)$. Let fopt denote the weight of an optimal fractional solution.

By first employing the algorithm given by Lemma 6.11, we next construct two instances of Weighted Multicut. The first instance is $J_1 = (\widehat{G}_1, w_1, \mathcal{T}_1)$ and the second instance is $J_2 = (\widehat{G}_2, w_2, \mathcal{T}_2)$, where the sets $\mathcal{T}_1$ and $\mathcal{T}_2$ are defined as follows. We initialize $\mathcal{T}_1 = \emptyset$. Now, for every pair (v, u) such that $v \in V(C), u \in V(M)$, $i(v, u) = 1$ and $\mathcal{P}_1(v, u) \neq \emptyset$, we insert each pair in $\mathcal{T}_1(v, u)$ into $\mathcal{T}_1$. The definition of $\mathcal{T}_2$ is symmetric to the one of $\mathcal{T}_1$.

By Lemma 6.11 and since for all $v \in V(C) \cup V(M)$ it holds that $\mathbf{x}_1^*(v) = \mathbf{x}_2^*(v) = 0$, we deduce that $2\mathbf{x}_1^*$ and $2\mathbf{x}_2^*$ are valid solutions to J_1 and J_2, respectively. Thus, by calling the algorithm given by Theorem 6.5 with each instance, we obtain a solution r_1 to the first instance, along with a set R_1 that realizes it, such that $r_1 \leq 2d \cdot w(\mathbf{x}_1^*)$, and we also obtain a solution r_2 to the second instance, along with a set R_2 that realizes it, such that $r_2 \leq 2d \cdot w(\mathbf{x}_2^*)$, for some fixed constant d.

By Observation 10 and Lemma 6.9, we obtained a set $S^* = S_1 \cup S_2 \cup R_1 \cup R_2$ for which we have the following observation.

Observation 13. *S^* intersects any chordless cycle in G', and it holds that $w(S^*) \leq s_1 + s_2 + r_1 + r_2$.*

Recall that to prove Lemma 6.5 we need to show that $s_1+s_2+r_1+r_2 \leq (\frac{c\cdot\log n}{c\cdot\log n+9})^{\delta-1} \cdot c\cdot\log(\alpha(G'))\cdot w'(\mathbf{x})$ and we have $\delta \geq 2$. Furthermore, we have $(\frac{c\cdot\log n}{c\cdot\log n+9})^{\delta}\cdot c\cdot\log(\alpha(G'))\cdot w'(\mathbf{x}) \leq (\frac{c\cdot\log n}{c\cdot\log n+9})^{\delta-1}\cdot c\cdot\log(\alpha(G'))\cdot w'(\mathbf{x})$. This together with Lemma 6.7 implies that it is enough to show $s_1 + s_2 + r_1 + r_2 \leq (\frac{c\cdot\log n}{c\cdot\log n+9})^{\delta}\cdot c\cdot\log(\alpha(G'))\cdot w(\mathbf{x}^*)$. Recall that for any $i \in \{1,2\}$, $r_i \leq 2d\cdot w(\mathbf{x}_i^*)$. Thus, by Observation 11 and since for any $i \in \{1,2\}$, $\alpha(G_i) \leq \frac{3}{4}\alpha(G')$, we have that

$$w(S^*) \leq (\frac{c\cdot\log n}{c\cdot\log n+9})^{\delta}\cdot c\cdot\log(\frac{3}{4}\alpha(G'))\cdot(w(\mathbf{x}_1^*)+w(\mathbf{x}_2^*)) + 2d\cdot(w(\mathbf{x}_1^*)+w(\mathbf{x}_2^*)).$$

By Observation 12, we further deduce that

$$w^*(S^*) \leq \left((\frac{c\cdot\log n}{c\cdot\log n+9})^{\delta}\cdot c\cdot\log(\frac{3}{4}\alpha(G')) + 2d\right)\cdot w(\mathbf{x}^*).$$

Now, it only remains to show that $(\frac{c\log n}{c\log n+9})^{\delta}\cdot c\cdot\log(\frac{3}{4}\alpha(G')) + 2d \leq (\frac{c\log n}{c\log n+9})^{\delta}\cdot c\cdot\log\alpha(G')$, which is equivalent to $2d \leq (\frac{c\log n}{c\log n+9})^{\delta}\cdot c\cdot\log(\frac{4}{3})$. Recall that $\delta \leq q\cdot\log n$ (Observation 9). Thus, it is sufficient that we show that $2d \leq (\frac{c\log n}{c\log n+9})^{q\cdot\log n}\cdot c\cdot\log(\frac{4}{3})$. However, the term $(\frac{c\log n}{c\log n+9})^{q\cdot\log n}$ is lower bounded by $1/e^{9q}$. In other words, it is sufficient that we fix $c \geq 2\cdot e^{9q}\cdot d\cdot 1/\log(\frac{4}{3})$.

6.2.2 General graphs

In this subsection, we handle general instances by developing a $d\cdot\log^2 n$-factor approximation algorithm for WCVD, Gen-CVD-APPROX, thus proving the correctness of Theorem 6.2. The exact value of the constant $d \geq \max\{96, 2c\}$ is determined later.[4] This algorithm is based on recursion, and during its execution, we often encounter instances that are of the form of the Clique+Chordal special case of WCVD, which will be dealt with using the algorithm CVD-APPROX of Section 6.2.1.

Recursion. We define each call to our algorithm Gen-CVD-APPROX to be of the form (G', w'), where (G', w') is an instance of WCVD such that G' is an induced subgraph of G, and we denote $n' = |V(G')|$. We ensure that after the initialization phase, the

[4]Recall that c is the constant we fixed to ensure that the approximation ratio of CVD-APPROX is bounded by $c\cdot\log n$.

graph G' never contains chordless cycles on at most 48 vertices. We call this invariant the *C_{48}-free invariant*. In particular, this guarantee ensures that the graph G' always contains only a small number of maximal cliques:

Lemma 6.12 ([Far89, TIAS77])**.** *The number of maximal cliques of a graph G' that has no chordless cycles on four vertices is bounded by $\mathcal{O}(n'^2)$, and they can be enumerated in polynomial time using a polynomial delay algorithm.*

Goal. For each recursive call Gen-CVD-APPROX(G', w'), we aim to prove the following.

Lemma 6.13. Gen-CVD-APPROX *returns a solution that is at least* opt *and at most* $\frac{d}{2} \cdot \log^2 n' \cdot \mathsf{opt}$*. Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution.*

At each recursive call, the size of the graph G' becomes smaller. Thus, when we prove that Lemma 6.13 is true for the current call, we assume that the approximation factor is bounded by $\frac{d}{2} \cdot \log^2 \widehat{n} \cdot \mathsf{opt}$ for any call where the size $\widehat{n}$ of the vertex-set of its graph is strictly smaller than n'.

Initialization. Initially, we set $(G', w') = (G, w)$. However, we need to ensure that the C_{48}-free invariant is satisfied. For this purpose, we update G' as follows. First, we let $\mathcal{C}_{48}$ denote the set of all chordless cycles on at most 48 vertices of G'. Clearly, $\mathcal{C}_{48}$ can be computed in polynomial time and it holds that $|\mathcal{C}_{48}| \leq n^{48}$. Now, we construct an instance of Weighted 48-Hitting Set, where the universe is $V(G')$, the family of 48-sets is $\mathcal{C}_{48}$, and the weight function is w'. Since each chordless cycle must be intersected, it is clear that the optimal solution to our Weighted 48-Hitting Set instance is at most opt. By using the standard c'-approximation algorithm for Weighted c'-Hitting Set [KT05], which is suitable for any fixed constant c', we obtain a set $S \subseteq V(G')$ that intersects all cycles in $\mathcal{C}_{48}$ and whose weight is at most $48 \cdot \mathsf{opt}$. Having the set S, we remove its vertices from G'. Now, the C_{48}-free invariant is satisfied, which implies that we can recursively call our algorithm. To the outputted solution, we add $w(S)$ and S. If Lemma 6.13 is true, we obtain a solution that is at most $\frac{d}{2} \cdot \log^2 n \cdot \mathsf{opt} + 48 \cdot \mathsf{opt} \leq d \cdot \log^2 n \cdot \mathsf{opt}$, which allows us to conclude the correctness of Theorem 6.2. We remark that during the execution of our algorithm, we only update G' by removing vertices from it, and thus it will always be safe to assume that the C_{48}-free invariant is satisfied.

Termination. Observe that due to Lemma 6.12, we can test in polynomial time whether G'consists of a clique and a chordal graph: we examine each maximal clique of G', and check whether after its removal we obtain a chordal graph. Once G' becomes such a graph that consists of a chordal graph and a clique, we solve the instance (G', w') by calling algorithm CVD-APPROX. Since $c \cdot \log n' \leq \frac{d}{2} \cdot \log^2 n'$, we thus ensure that at the base case of our induction, Lemma 6.13 holds.

Recursive Call. For the analysis of a recursive call, let S^* denote a hypothetical set that realizes the optimal solution opt of the current instance (G', w'). Moreover, let (F, β) be a clique forest of $G' - S^*$, whose existence is guaranteed by Proposition 2.2. Using standard arguments on forests, we have the following observation.

Observation 14. *There exist a maximal clique M of G' and a subset $S \subseteq V(G') \setminus M$ of weight at most* opt *such that $M \cup S$ is a balanced separator for G'.*

The following lemma translates this observation into an algorithm.

Lemma 6.14. *There is a polynomial-time algorithm that finds a maximal clique M of G' and a subset $S \subseteq V(G') \setminus M$ of weight at most $q \cdot \log n' \cdot \mathsf{opt}$ for some fixed constant q such that $M \cup S$ is a balanced separator for G'.*

Proof. We examine every maximal clique of G'. By Lemma 6.12, we need only consider $\mathcal{O}(n'^2)$ maximal cliques, and these cliques can be enumerated in polynomial time. For each such clique M, we run the $q \cdot \log n'$-factor approximation algorithm by Leighton and Rao [LR99] to find a balanced separator S_M of $G' - M$. Here, q is some fixed constant. We let S denote some set of minimum weight among the sets in $\{S_M \mid M \text{ is a maximal clique of } G'\}$. By Observation 14, $w(S) \leq q \cdot \log n' \cdot \mathsf{opt}$. Thus, the desired output is a pair (M, S) where M is one of the examined maximal cliques such that $S_M = S$. □

We call the algorithm in Lemma 6.14 to obtain a pair (M, S). Since $M \cup S$ is a balanced separator for G', we can partition the set of connected components of $G' - (M \cup S)$ into two sets, $\mathcal{A}_1$ and $\mathcal{A}_2$, such that for $V_1 = \bigcup_{A \in \mathcal{A}_1} V(A)$ and $V_2 = \bigcup_{A \in \mathcal{A}_2} V(A)$ it holds that $n_1, n_2 \leq \frac{2}{3}n'$ where $n_1 = |V_1|$ and $n_2 = |V_2|$. We remark that we used the $\mathcal{O}(\log n)$-factor approximation algorithm by Leighton and Rao [LR99] in Lemma 6.14 to find the balanced separator instead of the $\mathcal{O}(\sqrt{\log n})$-factor approximation algorithm by Feige et al. [FHL08], as the algorithm by Feige et al. is randomized.

Next, we define two inputs of (the general case of) WCVD: $I_1 = (G'[V_1], w'|_{V_1})$ and $I_2 = (G'[V_2], w'|_{V_2})$. Let opt_1 and opt_2 denote the optimal solutions to I_1 and I_2, respectively. Observe that since $V_1 \cap V_2 = \emptyset$, it holds that $\mathsf{opt}_1 + \mathsf{opt}_2 \leq \mathsf{opt}$. We solve each of the subinstances by recursively calling algorithm Gen-CVD-APPROX. By the inductive hypothesis, we thus obtain two sets, S_1 and S_2, such that $G'[V_1] - S_1$ and $G'[V_2] - S_2$ are chordal graphs, and $w'(S_1) \leq \frac{d}{2} \cdot \log^2 n_1 \cdot \mathsf{opt}_1$ and $w'(S_2) \leq \frac{d}{2} \cdot \log^2 n_2 \cdot \mathsf{opt}_2$.

We proceed by defining an input of the Clique+Chordal special case of WCVD: $J = (G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)], w'|_{(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)})$. Observe that since $G'[V_1] - S_1$ and $G'[V_2] - S_2$ are chordal graphs and M is a clique, this is indeed an instance of the Clique+Chordal special case of WCVD. We solve this instance by calling algorithm CVD-APPROX. We thus obtain a set, $\widehat{S}$, such that $G'[(V_1 \cup V_2 \cup M) - (S_1 \cup S_2 \cup \widehat{S})]$ is a chordal graphs, and $w'(\widehat{S}) \leq c \cdot \log n' \cdot \mathsf{opt}$ (since $|(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)| \leq n'$ and the optimal solution of each of the subinstances is at most opt).

Observe that since M is a clique and there is no edge in $E(G')$ between a vertex in V_1 and a vertex in V_2, any chordless cycle of $G' - (S \cup S_1 \cup S_2)$ entirely belongs to either $G'[(V_1 \cup M) \setminus S_1]$ or $G'[(V_2 \cup M) \setminus S_2]$. This observation, along with the fact that $G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2 \cup \widehat{S})]$ is a chordal graphs, implies that $G' - T$ is a chordal graphs where $T = S \cup S_1 \cup S_2 \cup \widehat{S}$. Thus, it is now sufficient to show that $w'(T) \leq \frac{d}{2} \cdot \log^2 n' \cdot \mathsf{opt}$.

By the discussion above, we have that

$$\begin{aligned} w'(T) &\leq w'(S) + w'(S_1) + w'(S_2) + w'(\widehat{S}) \\ &\leq q \cdot \log n' \cdot \mathsf{opt} + \tfrac{d}{2} \cdot (\log^2 n_1 \cdot \mathsf{opt}_1 + \log^2 n_2 \cdot \mathsf{opt}_2) + c \cdot \log n' \cdot \mathsf{opt}. \end{aligned}$$

Recall that $n_1, n_2 \leq \frac{2}{3}n'$ and $\mathsf{opt}_1 + \mathsf{opt}_2 \leq \mathsf{opt}$. Thus, we have that

$$\begin{aligned} w'(T) &\leq q \cdot \log n' \cdot \mathsf{opt} + \tfrac{d}{2} \cdot (\log^2 \tfrac{2}{3}n') \cdot \mathsf{opt} + c \cdot \log n' \cdot \mathsf{opt} \\ &\leq \tfrac{d}{2} \cdot \log^2 n' \cdot \mathsf{opt} + (q + c - \tfrac{d}{2} \log \tfrac{3}{2}) \cdot \log n' \cdot \mathsf{opt}. \end{aligned}$$

Overall, we conclude that to ensure that $w'(T) \leq \frac{d}{2} \cdot \log^2 n' \cdot \mathsf{opt}$, it is sufficient to ensure that $q + c - \frac{d}{2} \log \frac{3}{2} \leq 0$, which can be done by fixing $d = \dfrac{2}{\log \frac{3}{2}} \cdot (q + c)$.

6.3 Weighted Multicut in Chordal Graphs

In this section, we consider the problem Weighted Multicut on chordal graphs. In the following, we formally define the problem Weighted Multicut.

Weighted Multicut
Input: A graph G, a weight function $w : V(G) \to \mathbb{Q}$ and a set $T = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k pairs of vertices of G.
Output: A minimum weight subset $S \subseteq V(G)$ such that for any pair $(s_i, t_i) \in \mathcal{T}$, $G - S$ does not have any path between s_i and t_i.

For Weighted Multicut on chordal graphs, no constant-factor approximation algorithm was previously known prior to the work that is presented here. We remark that Weighted Multicut is NP-hard on trees [GVY96], and hence it is also NP-hard on chordal graphs. The goal of this section is to proof the following theorem.

Theorem 6.3. Weighted Multicut *admits a constant-factor approximation algorithm on chordal graphs.*

This algorithm is inspired by the work of Garg, Vazirani and Yannakakis on Weighted Multicut on trees [GVY96]. Here, we carefully exploit the well-known characterization of the class of chordal graphs as the class of graphs that admit clique forests. We believe that this result is of independent interest. The algorithm by Garg et al. [GVY96] is a classic primal-dual algorithm. A more recent algorithm, by Golovin et al. [GNS06], uses total modularity to obtain a different algorithm for Multicut on trees.

Let us denote $c = 8$. Recall that for Weighted Multicut, a fractional solution $\mathbf{x}$ is a function $\mathbf{x} : V(G) \to [0, \infty)$ such that for every pair $(s, t) \in \mathcal{T}$ and any path P between s and t, it holds that $\mathbf{x}(V(P)) \geq 1$. An optimal fractional solution minimizes the weight $w(\mathbf{x}) = \sum_{v \in V(G)} w(v) \cdot \mathbf{x}(v)$. Let fopt denote the weight of an optimal fractional solution. Theorem 6.3 follows from the next result, whose proof is the focus of this section.

Lemma 6.15. *Given an instance of* Weighted Multicut *in chordal graphs, one can find (in polynomial time) a solution that is at least* opt *and at most* $4c \cdot \mathsf{fopt}$*, along with a set that realizes it.*

Preprocessing. By using the ellipsoid method, we may next assume that we have optimal fractional solution $\mathbf{x}$ at hand. We say that $\mathbf{x}$ is *nice* if for all $v \in V(G)$, there exists $i \in \{0\} \cup \mathbb{N}$ such that $\mathbf{x}(v) = \frac{i}{n}$. Let $h(\mathbf{x}) = \{v \in V(G) \mid \mathbf{x}(v) \geq 1/c\}$ denote the set of vertices to which $\mathbf{x}$ assigns high values.

Lemma 6.16. *Define a function* $\widehat{\mathbf{x}} : V(G) \to [0, \infty)$ *as follows. For all* $v \in V(G)$*, if* $\mathbf{x}(v) < 1/2n$ *then* $\widehat{\mathbf{x}}(v) = 0$*, and otherwise* $\widehat{\mathbf{x}}(v)$ *is the smallest value of the form* i/n*, for some* $i \in \mathbb{N}$*, that is at least* $2\mathbf{x}(v)$*. Then,* $\widehat{\mathbf{x}}$ *is a fractional solution such that* $w(\widehat{\mathbf{x}}) \leq 4w(\mathbf{x})$*.*

Proof. To show that $\widehat{\mathbf{x}}$ is a fractional solution, consider some path P between s and t such that $(s,t) \in \mathcal{T}$. Let $\ell'(\mathbf{x}) = \{v \in V(G) \mid \mathbf{x}(v) < 1/2n\}$. We have that $\widehat{\mathbf{x}}(V(P)) = \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \widehat{\mathbf{x}}(v) \geq 2 \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v)$. Thus, to show that $\widehat{\mathbf{x}}(V(P)) \geq 1$, it is sufficient to show that $\frac{1}{2} \leq \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v)$. Since $\mathbf{x}$ is a fractional solution, it holds that $\mathbf{x}(V(P)) = \sum_{v \in V(P) \cap \ell'(\mathbf{x})} \mathbf{x}(v) + \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v) \geq 1$. Thus, $1 \leq \frac{1}{2n} |V(P) \cap \ell'(\mathbf{x})| + \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v)$. Since $|V(P) \cap \ell'(\mathbf{x})| \leq n$, we conclude that $\frac{1}{2} \leq \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v)$.

The second part of the claim follows from the observation that for all $v \in V(G)$, $\widehat{\mathbf{x}}(v) \leq 4\mathbf{x}(v)$. □

Accordingly, we update $\mathbf{x}$ to $\widehat{\mathbf{x}}$. Our preprocessing step also relies on the following standard lemma.

Lemma 6.17. *Define* $\widehat{G} = G - h(\mathbf{x})$, $\widehat{w} = w|_{V(\widehat{G})}$ *and* $\widehat{\mathbf{x}} = \mathbf{x}|_{V(\widehat{G})}$. *Then,* $c \cdot w(\widehat{\mathbf{x}}) + w(h(\mathbf{x})) \leq c \cdot w(\mathbf{x})$.

Proof. By the definition of $h(\mathbf{x})$, it holds that $w(\widehat{\mathbf{x}}) \leq w(\mathbf{x}) - \frac{1}{c} w(h(\mathbf{x}))$. Thus, $c \cdot w(\widehat{\mathbf{x}}) + w(h(\mathbf{x})) \leq c \cdot (w(\mathbf{x}) - \frac{1}{c} \cdot w(h(\mathbf{x}))) + w(h(\mathbf{x})) = c \cdot w(\mathbf{x})$. □

We thus further update G to $\widehat{G}$, w to $\widehat{w}$ and $\mathbf{x}$ to $\widehat{\mathbf{x}}$, where we ensure that once we obtain a solution to the new instance, we add $w(h(\mathbf{x}))$ to this solution and $h(\mathbf{x})$ to the set realizing it. Overall, we may next focus only on the proof of the following lemma.

Lemma 6.18. *Let* (G, w) *be an instance of* Weighted Multicut *in chordal graphs, and* $\mathbf{x}$ *be a nice fractional solution such that* $h(\mathbf{x}) = \emptyset$. *Then, one can find (in polynomial time) a solution that is at least* opt *and at most* $c \cdot w(\mathbf{x})$, *along with a set that realizes it.*

The Algorithm. Since G is a chordal graph, we can first construct in polynomial time a clique forest (F, β) of G (Proposition 2.2). Without loss of generality, we may assume that F is a tree, else G is not a connected graph and we can handle each of its connected components separately. Now, we arbitrarily root F at some node r_F, and we arbitrarily choose a vertex $r_G \in \beta(r_F)$. We then use Dijkstra's algorithm to compute (in polynomial time) for each vertex $v \in V(G)$, the value $d(v) = \min_{P \in \mathcal{P}(v)} \mathbf{x}(V(P))$, where $\mathcal{P}(v)$ is the set of paths in G between r_G and v.

We define $n+1$ bins: for all $i \in \{0, 1, \ldots, n\}$, the bin B_i contains every vertex $v \in V(G)$ for which there exists $j \in \{0\} \cup \mathbb{N}$ such that $d(v) - \mathbf{x}(v) < (\frac{i}{n} + 2j)\frac{1}{c} \leq d(v)$ (i.e., $0 \leq d(v) - (\frac{i}{n} + 2j)\frac{1}{c} < \mathbf{x}(v)$). Let B_{i^*}, $i^* \in \{0, 1, \ldots, n\}$, be a bin that minimizes $w(B_{i^*})$. The output consists of $w(B_{i^*})$ and B_{i^*}.

Approximation Factor. Given $r \in [0,1]$, let $\widehat{B}_r$ be the set that contains every vertex $v \in V(G)$ for which there exists $j \in \{0\} \cup \mathbb{N}$ such that $0 \leq d(v) - (r + 2j)\frac{1}{c} < \mathbf{x}(v)$. We start with the following claim.

Lemma 6.19. *There exists* $r^* \in [0,1]$ *such that* $w(\widehat{B}_{r^*}) \leq c \cdot w(\mathbf{x})$.

Proof. For any $d \geq 0$, observe that there exists exactly one $j \in \{0\} \cup \mathbb{N}$ for which there exists $r \in [0,1]$ such that $0 \leq d - (r + 2j)\frac{1}{c} < \frac{1}{c}$, and denote it by $j(d)$. Suppose that we choose $r \in [0,1]$ uniformly at random. Consider some vertex $v \in V(G)$. Then, since

$h(\mathbf{x}) = \emptyset$, the probability that there exists $j \in \{0\} \cup \mathbb{N}$ such that $0 \leq d(v) - (r + 2j)\frac{1}{c} < \mathbf{x}(v)$ is equal to the probability that $0 \leq d(v) - (r + 2j(d(v)))\frac{1}{c} < \mathbf{x}(v)$. Now, the probability that $0 \leq d(v) - (r + 2j(d(v)))\frac{1}{c} < \mathbf{x}(v)$ is equal to $c \cdot \mathbf{x}(v)$. The expected weight $w(\widehat{B}_r)$ is $c \cdot \sum_{v \in V(G)} \mathbf{x}(v) \cdot w(v) = c \cdot w(\mathbf{x})$. Thus, there exists $r^* \in [0, 1]$ such that $w(\widehat{B}_{r^*}) \leq c \cdot w(\mathbf{x})$. □

Now, the proof of the approximation factor follows from the next claim.

Lemma 6.20. *There exists $i \in \{0, 1, \ldots, n\}$ such that $B_i \subseteq \widehat{B}_{r^*}$.*

Proof. Let i be the smallest index in $\{0, 1, \ldots, n\}$ such that $r^* \leq \frac{i}{n}$. Consider some vertex $v \in B_i$. Then, for some $j \in \{0\} \cup \mathbb{N}$, $d(v) - \mathbf{x}(v) < (\frac{i}{n} + 2j)\frac{1}{c} \leq d(v)$. Since $r^* \leq \frac{i}{n}$, we have that $(r^* + 2j)\frac{1}{c} \leq d(v)$. Since $\mathbf{x}$ is nice, it holds that there exists $t \in \{0\} \cup \mathbb{N}$ such that $d(v) - \mathbf{x}(v) = \frac{t}{n}$. Thus, for any $p < \frac{1}{n}$, it holds that $d(v) - \mathbf{x}(v) < (\frac{i}{n} + 2j - p)\frac{1}{c}$. By the choice of i, $\frac{i}{n} - r^* < \frac{1}{n}$, and therefore $d(v) - \mathbf{x}(v) < (r^* + 2j)\frac{1}{c}$, which implies that $v \in \widehat{B}_{r^*}$. □

Feasibility. We need to prove that for any pair $(s, t) \in \mathcal{T}$, $G - B_{i^*}$ does not have any path between s and t. Consider some path $P = (v_1, v_2, \cdots, v_\ell)$ between s and t. Here, $v_1 = s$ and $v_\ell = t$. Suppose, by way of contradiction, that $V(P) \cap B_{i^*} = \emptyset$. Then, for all $v_i \in V(P)$, it holds that there is no $j \in \mathbb{N}$ such that $0 \leq d(v_i) - (\frac{i^*}{n} + 2j)\frac{1}{c} < \mathbf{x}(v_i)$.

Let $s \in V(F)$ be the closest node to r_F that satisfies $\beta(s) \cap V(P) \neq \emptyset$ (since F is a clique tree and P is a path, the node s is uniquely defined). Let $v_{\widehat{i}}$ be some vertex in $\beta(s) \cap V(P) \neq \emptyset$. For the sake of clarity, let us denote the subpath of P between $v_{\widehat{i}}$ and v_ℓ by $Q = (u_1, u_2, \cdots, u_t)$, where $u_1 = v_{\widehat{i}}$ and $u_t = v_\ell$. Let j^* be the smallest value in $\{0\} \cup \mathbb{N}$ that satisfies $d(u_1) - \mathbf{x}(u_1) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$. Note that $d(u_1) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$. It is thus well defined to let p denote the largest index in $[t]$ such that $d(u_p) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$.

First, suppose that $p \in [t-1]$. We then have that $(\frac{i^*}{n} + 2j^*)\frac{1}{c} \leq d(u_{p+1})$. For all $2 \leq i \leq t$, it holds that $d(u_i) \leq d(u_{i-1}) + \mathbf{x}(u_i)$. We thus obtain that $d(u_{p+1}) - \mathbf{x}(u_{p+1}) \leq d(u_p) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$. This statement implies that $u_{p+1} \in B_{i^*}$, which is a contradiction.

Now, we suppose that $p = t$. Note that $(\frac{i^*}{n} + 2j^* - 2)\frac{1}{c} \leq d(u_1) - \mathbf{x}(u_1)$ (by the minimality of j^*), and $d(u_t) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$. We get that $d(u_t) < d(u_1) - \mathbf{x}(u_1) + \frac{2}{c}$. In other words, $d(u_t) - d(u_1) + \mathbf{x}(u_1) < \frac{2}{c}$. Let $\mathrm{des}(s)$ denote the set consisting of s and its descendants in F. Since F is a clique tree, we have that $V(Q) \subseteq \bigcup_{s' \in \mathrm{des}(s)} \beta(s')$. Thus, any path from r_G to u_t that realizes $d(u_t)$ contains a vertex from $\beta(s)$. Since there exists a path from r_G to u_t that realizes $d(u_t)$, we deduce that there exists a path, P_t, from r_G to u_t that realizes $d(u_t)$ and contains a vertex $x \in N_G[u_1]$. Let P_t^* denote the subpath of P_t between x and u_t, and let P^* denote the path that starts at u_1 and then traverses P_t^*. Then, $\mathbf{x}(V(P^*)) \leq \mathbf{x}(u_1) + \mathbf{x}(V(P_t^*)) = \mathbf{x}(u_1) + d(u_t) - d(x) + \mathbf{x}(x)$. Note that $d(u_1) \leq d(x) + \mathbf{x}(u_1)$, and therefore $\mathbf{x}(V(P^*)) \leq \mathbf{x}(u_1) + d(u_t) - (d(u_1) - \mathbf{x}(u_1)) + \mathbf{x}(x) = \mathbf{x}(u_1) + \mathbf{x}(x) + (d(u_t) - d(u_1) + \mathbf{x}(u_1))$. Since $h(\mathbf{x}) = \emptyset$ and $d(u_t) - d(u_1) + \mathbf{x}(u_1) < \frac{2}{c}$, we get that $\mathbf{x}(V(P^*)) < \frac{4}{c}$. The symmetric analysis of the subpath of P between $u_1 = v_{\widehat{i}}$ and v_1 shows that there exists a path P^{**} between u_1 and v_1 such that $\mathbf{x}(V(P^{**})) < \frac{4}{c}$. Overall, we get that there exists a path, P', between $v_1 = s$ and $v_\ell = u_\ell = t$ such that $\mathbf{x}(V(P')) < \frac{8}{c}$. Since $c \geq 8$, we reach a contradiction to the assumption that $\mathbf{x}$ is a fractional solution.

6.4 Approximation Algorithm for Weighted Distance Hereditary Vertex Deletion

We apply the general scheme of designing approximation algorithm that we discussed to another problem. A graph G is a *distance hereditary graph* (also called a completely separable graph [HM90]) if the distances between vertices in every connected induced subgraph of G are the same as in the graph G. In this section, we consider the problem Weighted Distance Hereditary Vertex Deletion, which is formally defined below.

Weighted Distance Hereditary Vertex Deletion (WDHVD)
Input: A graph G and a weight function $w : V(G) \to \mathbb{Q}$.
Output: A minimum weight subset $S \subseteq V(G)$ such that $G - S$ is a distance hereditary graph.

Distance hereditary graphs were named and first studied by Hworka [How77]. However, an equivalent family of graphs was earlier studied by Olaru and Sachs [Sac70] and shown to be perfect. It was later discovered that these graphs are precisely the graphs of rankwidth 1 [Oum05]. Rankwidth is a graph parameter introduced by Oum and Seymour [OS06].

As algorithms for Treewidth-η Vertex Deletion are applied as subroutines to solve many graph problems, we believe that algorithms for Weighted Rankwidth-η Vertex Deletion (WR-ηVD) will be useful in this respect. In particular, Treewidth-η Vertex Deletion has been considered in designing efficient approximation, kernelization and fixed parameter tractable algorithms for WP-$\mathscr{F}$-MFD and its unweighted counterpart Planar $\mathscr{F}$-Minor-Free Deletion [BRU17, FLM+16, FLRS11, FLS12, FLST10]. Along similar lines, we believe that WR-ηVD and its unweighted counterpart will be useful in designing efficient approximation, kernelization and fixed parameter tractable algorithms for Weighted $\mathcal{F}$-Vertex Deletion where $\mathcal{F}$ is characterized by a finite family of forbidden *vertex minors* [Oum05].

Recently, Kim and Kwon [KK16] designed an $\mathcal{O}(\mathsf{opt}^2 \log n)$-factor approximation algorithm for Distance Hereditary Vertex Deletion (DHVD). This result implies that DHVD admits an $\mathcal{O}(n^{2/3} \log n)$-factor approximation algorithm. In this section, we take a step towards obtaining good approximation algorithm for WR-ηVD by designing a $\mathcal{O}(\log^{\mathcal{O}(1)} n)$-factor approximation algorithm for WDHVD. In particular, the focus of this section will be the proof of the following theorem.

Theorem 6.4. WDHVD *or* WR-1VD *admits an* $\mathcal{O}(\log^3 n)$*-factor approximation algorithm.*

Preliminaries. A graph G is *distance hereditary* if every connected induced subgraph H of G, for all $u, v \in V(H)$ the number of vertices in shortest path between u and v in G is same as the number of vertices in shortest path between u and v in H. Another characterization of distance hereditary graphs is the graph not containing an induced sub-graph isomorphic to a house, a gem, a domino or an induced cycle on 5 or more vertices (refer Figure 6.3). We refer to a house, a gem, a domino or an induced cycle on at least 5 vertices as a DH-obstruction. A DH-obstruction on at most 48 vertices is a small DH-obstruction. A *biclique* is a graph G with vertex bipartition X, Y each of

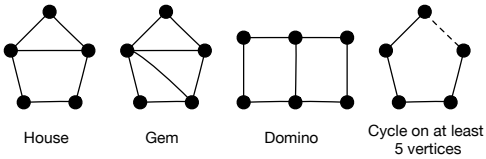

Figure 6.3: Obstruction set for distance hereditary graphs

them being non-empty such that for each $x \in X$ and $y \in Y$ we have $\{x, y\} \in E(G)$. We note here that, X and Y need not be independent sets in a *biclique* G.

Clearly, we can assume that the weight $w(v)$ of each vertex $v \in V(G)$ is positive, else we can insert v into any solution. Our approximation algorithm for WDHVD comprises of two components. The first component handles the special case where the input graph G consists of a biclique C and a distance hereditary H. Here, we also assume that the input graph has no "small" DH-obstruction. We show that when input restricted to these special instances WDHVD admits an $\mathcal{O}(\log^2 n)$-factor approximation algorithm.

The second component is a recursive algorithm that solves general instances of the problem. Initially, it easily handles "small" DH-obstruction. Then, it gradually disintegrates a general instance until it becomes an instance of the special form that can be solved in polynomial time. More precisely, given a problem instance, the algorithm divides it by finding a maximal biclique M (using an exhaustive search which relies on the guarantee that G has no "small" DH-obstruction) and a small separator S (using an approximation algorithm) that together break the input graph into two graphs significantly smaller than their origin.

6.4.1 Biclique+ Distance Hereditary graph

In this subsection, we handle the special case where the input graph G consists of a biclique C and a distance hereditary graph H. More precisely, along with the input graph G and the weight function w, we are also given a biclique C and a distance hereditary graph H such that $V(G) = V(C) \cup V(H)$, where the vertex-sets $V(C)$ and $V(H)$ are disjoint. Here, we also assume that G has no DH-obstruction on at most 48 vertices, which means that every DH-obstructionin G is a chordless cycle of strictly more than 48 vertices. Note that the edge-set $E(G)$ may contain edges between vertices in C and vertices in H. We call this special case the *Biclique + Distance Hereditary special case*. Our objective is to prove the following result.

Lemma 6.21. *The Biclique + Distance Hereditary special case of* WDHVD *admits an* $\mathcal{O}(\log^2 n)$*-factor approximation algorithm.*

We assume that $n \geq 2^{12}$, else the input instance can be solve by brute-force [5]. Let c be a fixed constant (to be determined later). In the rest of this subsection, we design a $c \cdot \log n$-factor approximation algorithm for the Biclique + Distance Hereditary special case of WDHVD.

Recursion. Our approximation algorithm is a recursive algorithm. We call our algorithm DHD-APPROX, and define each call to be of the form $(G', w', C, H', \mathbf{x})$. Here, G'

[5]This assumption simplifies some of the calculations ahead.

is an induced subgraph of G such that $V(C) \subseteq V(G')$, and H' is an induced subgraph of H. The argument $\mathbf{x}$ is discussed below. We remark that we continue to use n to refer to the size of the vertex-set of the input graph G rather than the current graph G'.

Arguments. While the execution of our algorithm progresses, we keep track of two arguments: the number of vertices in the current distance hereditary graph H' that are assigned a non-zero value by $\mathbf{x}$, which we denote by $\alpha(G')$ and the fractional solution $\mathbf{x}$.

Observation 15. *The measure $\alpha(G')$ can be computed in polynomial time.*

A fractional solution $\mathbf{x}$ is a function $\mathbf{x} : V(G') \to [0, \infty)$ such that for every chordless cycle Q of G' on at least 5 vertices it holds that $\mathbf{x}(V(Q)) \geq 1$. An optimal fractional solution minimizes the weight $w'(\mathbf{x}) = \sum_{v \in V(G')} w'(v) \cdot \mathbf{x}(v)$. Clearly, the solution to the instance (G', w') of WDHVD is at least as large as the weight of an optimal fractional solution. Although we initially compute an optimal fractional solution $\mathbf{x}$ (at the initialization phase that is described below), during the execution of our algorithm, we manipulate this solution so it may no longer be optimal. Prior to any call to DHD-APPROX with the exception of the first call, we ensure that $\mathbf{x}$ satisfies the following invariants:

- **Low-Value Invariant**: For any $v \in V(G')$, it holds that $\mathbf{x}(v) < 1/\log n$.
- **Zero-Biclique Invariant**: For any $v \in V(C)$, it holds that $\mathbf{x}(v) = 0$.

We note that the Low-Value Invariant used here is simpler than the one used in Section 6.2.1 since it is enough for the purpose of this section.

Goal. The depth of the recursion tree will be bounded by $\Delta = \mathcal{O}(\log n)$, where the depth of initial call is 1. The correctness of this claim is proved when we explain how to perform a recursive call. For each recursive call to DHD-APPROX$(G', w', C, H', \mathbf{x})$, we aim to prove the following.

Lemma 6.22. *For any $\delta \in \{1, 2, \ldots, \Delta\}$, each recursive call to* DHD-APPROX *of depth $\delta \geq 2$ returns a solution that is at least* opt *and at most $(\frac{\log n}{\log n + 4})^{\delta} \cdot c \cdot \log n \cdot \log(\alpha(G')) \cdot w'(\mathbf{x})$. Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution.*

At the initialization phase, we see that in order to prove Lemma 6.21, it is sufficient to prove Lemma 6.22.

Initialization. Initially, the graphs G' and H' are simply set to be the input graphs G and H, and the weight function w' is simply set to be input weight function w. Moreover, we compute an optimal fractional solution $\mathbf{x} = \mathbf{x}_{\text{init}}$ by using the ellipsoid method. Recall that the following claim holds.

Observation 16. *The solution of the instance (G', w') of* WDHVD *is lower bounded by $w'(\mathbf{x}_{\text{init}})$.*

Moreover, it holds that $\alpha(G') \leq n$, and therefore to prove Lemma 6.21, it is sufficient to return a solution that is at least opt and at most $c \cdot \log n \cdot \log(\alpha(G)) \cdot w(\mathbf{x})$ (along with a subset that realizes the solution). Part of the necessity of the stronger claim given by Lemma 6.22 will become clear at the end of the initialization phase.

We would like to proceed by calling our algorithm recursively. For this purpose, we first need to ensure that $\mathbf{x}$ satisfies the low-value and zero-biclique invariants, to which end we use the following notation. We let $h(\mathbf{x}) = \{v \in V(G') \mid \mathbf{x}(v) \geq 1/\log n\}$ denote the set of vertices to which $\mathbf{x}$ assigns high values. Note that we can assume for each $v \in h(\mathbf{x})$, we have $\mathbf{x}(v) \leq 1$. Moreover, given a biclique M in G', we let $(\mathbf{x} \setminus M) : V(G') \to [0, \infty)$ denote the function that assigns 0 to any vertex in M and $(1 + \frac{4}{\log n})\mathbf{x}(v)$ to any other vertex $v \in V(G')$. Now, to adjust $\mathbf{x}$ to be of the desired form both at this phase and at later recursive calls, we rely on the following lemmata.

Lemma 6.23. *Define $\widehat{G} = G' - h(\mathbf{x})$, $\widehat{w} = w'|_{V(\widehat{G})}$ and $\widehat{\mathbf{x}} = \mathbf{x}|_{V(\widehat{G})}$. Then, $c' \cdot \log n \cdot \log(\alpha(\widehat{G})) \cdot w'(\widehat{\mathbf{x}}) + w'(h(\mathbf{x})) \leq c' \cdot \log n \cdot \log(\alpha(G)) \cdot w'(\mathbf{x})$, where $c' \geq 1$.*

Proof. By the definition of $h(\mathbf{x})$, it holds that $w'(\widehat{\mathbf{x}}) \leq w'(\mathbf{x}) - \frac{1}{\log n} \cdot w'(h(\mathbf{x}))$. Since $\widehat{G}$ is an induced subgraph of G', it also holds that $\alpha(\widehat{G}) \leq \alpha(G')$. Thus, $c' \cdot \log n \cdot \log(\alpha(\widehat{G})) \cdot w'(\widehat{\mathbf{x}}) + w'(h(\mathbf{x})) \leq c' \cdot \log n \cdot \log(\alpha(G')) \cdot (w'(\mathbf{x}) - \frac{1}{\log n} \cdot w'(h(\mathbf{x}))) + w'(h(\mathbf{x})) \leq c' \cdot \log n \cdot \log(\alpha(G)) \cdot w'(\mathbf{x})$. □

Thus, it is safe to update G' to $G' - h(\mathbf{x})$, w' to $w'|_{V(\widehat{G})}$, H' to $H' - h(\mathbf{x})$ and $\mathbf{x}$ to $\mathbf{x}|_{V(\widehat{G})}$, where we ensure that once we obtain a solution to the new instance, we add $w'(h(\mathbf{x}))$ to this solution and $h(\mathbf{x})$ to the set realizing it.

Lemma 6.24. *Let Q be a chordless cycle on at least 5 vertices and M be a biclique in G' with vertex partitions as $V(M) = M_1 \uplus M_2$ such that $V(Q) \cap V(M) \neq \emptyset$. Then there is a chordless cycle Q' on at least 5 vertices that intersects M in at most 3 vertices such that $E(Q' \setminus M) \subseteq E(Q \setminus M)$. Furthermore, Q' is of one of the following three types.*

- *$Q' \cap M$ is a single vertex*
- *$Q' \cap M$ is an edge in $G[M]$*
- *$Q' \cap M$ is an induced path on 3 vertices in M.*

Proof. Observe that no chordless cycle on 5 or more vertices may contain two vertices from each of M_1 and M_2, as that would imply a chord in it. Now, if the chordless cycle Q already satisfies the required conditions we output it as Q'.

First consider the case, when $Q \cap M$ contains exactly two vertices that don't have an edge between them. Then the two vertices, say v_1, v_2, are both either in M_1 or in M_2. Suppose that they are both in M_1 and consider some vertex $u \in M_2$. Let P_1 is the longer of the two path segments of Q between v_1 and v_2, and note that it must length at least 3. Then observe that $G'[P_1 \cup \{u, v_1, v_2\}]$ contains a DH-obstruction, as v_1, v_2 have different distances depending on if u is included in an induced subgraph or not. And further, it is easy to see that this DH-obstruction contains the induced path v_1, u, v_2. However, as all small obstructions have been removed from the graph, we have that Q' is a chordless cycle in G' on at least 5 vertices. Furthermore, $Q' \cap M$ is the induced path (v_1, u, v_2), in G' and $E(Q' \setminus M) \subseteq E(Q \setminus M)$.

Now consider the case when $Q \cap M$ contains exactly three vertices. Observe that it cannot contain two vertices of M_1 and one vertex of M_2, or vice versa, as Q doesn't satisfy the required conditions. Therefore, $Q \cap M$ contains exactly three vertices from M_1 (or from M_2), which again don't form an induced path of length 3. So there is an

independent set of size 2 in $Q \cap M$, and now, as before, we can again obtain the chordless cycle Q' on at least 5 vertices with $E(Q' \setminus M) \subseteq E(Q \setminus M)$. Before we consider the other cases, we have the following claim.

Claim 6.1. *Let M be a biclique in G' with vertex partition as $V(M) = M_1 \uplus M_2$. Then $G'[M]$ has no induced P_4.*

Proof. Let P be any induced path of length 4 in $G'[M]$. Then, either $V(P) \subseteq M_1$ or $V(P) \subseteq M_2$. Now consider any such path P in M_1 and some vertex $u \in M_2$. Then $G'[P \cup \{u\}]$ contains a DH-obstruction of size 5 which is a contradiction to the fact that G' has no small obstructions. □

Next, let $Q \cap M$ contain 4 or more vertices. Note that in this case all these vertices are all either in M_1 or in M_2 since otherwise, Q would not be a chordless cycle in G' on at least 5 vertices. Let us assume these vertices lie in M_1 (other case is symmetric). Let $v_1, v_2, v_3, \cdots, v_\ell \in M_1 \cap Q$ be the sequence of vertices obtained when we traverse Q starting from an arbitrary vertex, where $\ell \geq 4$. By Claim 6.1 they cannot form an induced path on 4 vertices, i.e. $G'[V(Q) \cap M_1]$ consists of at least two connected components. Without loss of generality we may assume that v_1 and v_ℓ are in different components. Observe that the only possible edges between these vertices may be at most two of the edges $(v_1, v_2), (v_2, v_3)$, and (v_3, v_ℓ). Hence, we conclude that either v_1, v_3 or v_2, v_ℓ are a distance of at least 3 in Q. Let us assume that v_2, v_ℓ are at distance 3 or more in Q, and the other case is symmetric and $P_{23}, P_{3\ell}$ be the paths not containing v_1 in Q between v_2 and v_3, and v_3 and v_ℓ, respectively. Notice that for any $u \in M_2$ the graph $G'[\{u\} \cup V(P_{23}) \cup V(P_{3\ell})]$ contains a DH-obstruction. Since the graph is free of all small obstruction, this DH-obstruction, denoted $\hat{Q}$, must be a chordless cycle on at least 5 vertices. Furthermore this obstruction can contain at most 2 vertices from $\{v_2, v_3, v_\ell\}$, as otherwise there would be a chord in it. Hence $\hat{Q} \cap M$ contains strictly fewer vertices than $Q \cap M$. Moreover, we have $E(\hat{Q} \setminus M) \subseteq E(Q \setminus M)$. Now, by a recursive application of this lemma to $\hat{Q}$, we obtain the required Q'. □

A consequence of the above lemma is that, whenever M is a biclique in G', we may safely ignore any DH-obstructionthat intersects M in more than 3 vertices. This leads us to the following lemma.

Lemma 6.25. *Given a biclique M in G', the function $(\mathbf{x} \setminus M)$ is a valid fractional solution such that $w'(\mathbf{x} \setminus M) \leq (1 + \frac{4}{\log n}) w'(\mathbf{x})$.*

Proof. To prove that $(\mathbf{x} \setminus M)$ is a valid fractional solution, let Q be some chordless cycle (not on 4 vertices) in G'. We need to show that $(\mathbf{x} \setminus M)(V(Q)) \geq 1$. By our assumption Q can contain at most 3 vertices from M. Thus, since $\mathbf{x}$ is a valid fractional solution, it holds that $\mathbf{x}(V(Q) \setminus V(M)) \geq 1 - \frac{3}{\log n}$. By the definition of $(\mathbf{x} \setminus M)$, this fact implies that $(\mathbf{x} \setminus M)(V(Q)) = (\mathbf{x} \setminus M)(V(Q) \setminus V(M)) \geq (1 + \frac{4}{\log n})(1 - \frac{3}{\log n}) = 1 + \frac{1}{\log n} - \frac{12}{(\log n)^2} \geq 1$, where the last inequality relies on the assumption $n \geq 2^{12}$.

For the proof of the second part of the claim, note that $w'(\mathbf{x} \setminus M) = (1 + \frac{4}{\log n})$ $w'(\mathbf{x}|_{V(G') \setminus V(M)}) \leq (1 + \frac{4}{\log n}) w'(\mathbf{x})$. □

We call DHD-APPROX recursively with the fractional solution $(\mathbf{x} \setminus C)$, and by Lemma 6.25, $w'(\mathbf{x} \setminus C) \leq (1 + \frac{4}{\log n}) w'(\mathbf{x})$. If Lemma 6.22 were true, we return a solution that is

at least opt and at most $(\frac{\log n}{\log n+4})\cdot c\cdot\log n\cdot\log(\alpha(G))\cdot w(\mathbf{x}\setminus M) \leq c\cdot\log n\cdot\log(\alpha(G))\cdot w(\mathbf{x})$ as desired. In other words, to prove Lemma 6.21, it is sufficient that we next focus only on the proof of Lemma 6.22. The proof of this lemma is done by induction. When we consider some recursive call, we assume that the solutions returned by the additional recursive calls that it performs, which are associated with graphs $\widetilde{G}$ such that $\alpha(\widetilde{G}) \leq \frac{3}{4}\alpha(G')$, complies with the conclusion of the lemma.

Termination. Once G' becomes a distance hereditary graph, we return 0 as our solution and $\emptyset$ as the set that realizes it. Clearly, we thus satisfy the demands of Lemma 6.22. In fact, we thus also ensure that the execution of our algorithm terminates once $\alpha(G') < \log n$.

Lemma 6.26. *If $\alpha(G') < \log n$, then G' is a distance hereditary graph.*

Proof. Suppose that G' is not a distance hereditary graph. Then, it contains an obstruction Q. Since $\mathbf{x}$ is a valid fractional solution, it holds that $\mathbf{x}(V(Q)) \geq 1$. But $\mathbf{x}$ satisfies the low-value invariant therefore, it holds that $\mathbf{x}(V(Q)) < |V(Q)|/\log n$. These two observations imply that $|V(Q)| > \log n$. Furthermore, at least $\log n$ of these vertices are assigned a non-zero value by $\mathbf{x}$, i.e. $\alpha(G') \geq \log n$. Therefore, if $\alpha(G') < \log n$, then G' must be a distance hereditary graph. □

The fact that, the recursive calls are made onto graphs where the distance hereditary subgraph contains at most 3/4 the number of vertices in the current distance hereditary subgraph, we observe the following.

Observation 17. *The maximum depth of the recursion tree is bounded by $q \cdot \log n$ for some fixed constant q.*

Recursive Call. Since H' is a distance hereditary graph, it has a rank-width-one decomposition $(\mathcal{T}, \phi)$, where $\mathcal{T}$ is a binary tree and ϕ is a bijection from $V(G')$ to the leaves of $\mathcal{T}$. Furthermore, rank-width of $\mathcal{T}$ is 1, which means that for any edge of the tree, by deleting it, we obtain a partition of the leaves in $\mathcal{T}$. This partition induces a cut of the graph, where the set of edges crossing this cut forms a biclique M, with vertex partition as $V(M) = M_1 \uplus M_2$ in the graph. By standard arguments on trees, we deduce that $\mathcal{T}$ has an edge that defines a partition such that after we remove the biclique edges between M_1 and M_2 from G' we obtain two (not necessarily connected) graphs, H_1 and H_2, such that $|V(H_1)|, |V(H_2)| \leq \frac{3}{4}|V(H')|$ and $M_1 \subseteq H_1$, $M_2 \subseteq H_2$. Note that the bicliques M and C are vertex disjoint. We proceed by replacing the fractional solution $\mathbf{x}$ by $(\mathbf{x} \setminus M)$. For the sake of clarity, we denote $\mathbf{x}^* = (\mathbf{x} \setminus M)$. Let $G_1 = G'[V(H_1) \cup V(C) \cup V(M)]$, $G_2 = G'[V(H_2) \cup V(C) \cup V(M)]$.

We adjust the current instance by relying on Lemma 6.23 so that $\mathbf{x}^*$ satisfies the low-value invariant (in the same manner as it is adjusted in the initialization phase). In particular, we remove $h(\mathbf{x}^*)$ from G',H', G_1, H_1, G_2 and H_2, and we let $(G^*, w^*, C, H^*, \mathbf{x}^*)$, G_1^*, H_1^* G_2^* and H_2^* denote the resulting instance and graphs. Observe that, now we have $\alpha(G_1^*), \alpha(G_2^*) \leq \frac{3}{4}\alpha(G')$. We will return a solution that is at least opt and at most $(\frac{\log n}{\log n+4})^{\delta+1}\cdot c\cdot\log n\cdot\log(\alpha(G'))\cdot w^*(\mathbf{x}^*)$, along with a set that realizes it.[6] In the analysis we will argue this it is enough for our purposes.

[6]Here, the coefficient $(\frac{\log n}{\log n+4})^{\delta}$ has been replaced by the smaller coefficient $(\frac{\log n}{\log n+4})^{\delta+1}$.

Next, we define two subinstances, $I_1^* = (G_1^*, w^*|_{V(G_1^*)}, C, H_1^*, \mathbf{x}^*|_{V(G_1^*)})$ and $I_2^* = (G_2^*, w^*|_{V(G_2^*)}, C, H_2^*, \mathbf{x}^*|_{V(G_2^*)})$. We solve each of these subinstances by a recursive call to DHD-APPROX, and thus we obtain two solutions of sizes, s_1^* to I_1^* and s_2^* to I_2^*, and two sets that realize these solutions, S_1^* and S_2^*. By the inductive hypothesis, we have the following observations.

Observation 18. *$S_1^* \cup S_2^*$ intersects any chordless cycle on at least 6 vertices in G^* that lies entirely in either G_1^* or G_2^*.*

Observation 19. *Given $i \in \{1,2\}$, $s_i^* \leq (\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log n \cdot \log(\alpha(G_i^*)) \cdot w(\mathbf{x}_i^*)$.*

Moreover, since $\mathbf{x}^*(V(C) \cup V(M)) = 0$, we also have the following observation.

Observation 20. $w^*(\mathbf{x}_1^*) + w^*(\mathbf{x}_2^*) = w^*(\mathbf{x}^*)$.

We say that a cycle in G^* is *bad* if it is a chordless cycle not on four vertices that belongs entirely to neither G_1^* nor G_2^*. Next, we show how to intersect bad cycles.

Bad Cycles. Let us recall the current state of the graph G'. G' is partitioned into a biclique C and a distance hereditary graph H'. Furthermore, there is a biclique M with vertex bipartition as M_1 and M_2 so that deleting the edges between M_1 and M_2, gives a balanced partition of H' into H_1 and H_2. Now, by Lemma 6.24, we may ignore any chordless cycle that intersects either of the two bicliques C and M in more than three vertices each, and this allows us to update our fractional feasible solution to $\mathbf{x}^* = (\mathbf{x}/M)$. Then we recursively solve the instances G_1^* and G_2^* and remove the returned solution. Now consider the remaining graph, and any obstructions that are left. As the graph no longer contains small obstructions, it is clear that any remaining obstruction is a chordless cycle on at least 6 vertices and is a *bad cycle*. We first examine the relation between bad cycles and pairs (v,u) of vertices $v \in V(C)$ and $u \in V(M)$.

Lemma 6.27. *If a bad cycle exists, then there must also be a bad cycle Q such that $Q-(M\cup C)$ is a union of two internally vertex disjoint and non-adjacent path-segments, P_1 and P_2 such that, $P_1 \subseteq G_1$ and $P_2 \subseteq G_2$, and each of them connect a pair of vertices in $M \times C$.*

Proof. Let Q' be a bad cycle. Let us recall that the input graph G' can be partitioned into the biclique C and a distance hereditary graph H'. Hence $Q' \cap C \neq \emptyset$. Furthermore, if $Q' \cap M = \emptyset$, then Q' is preserved in $G' - M$. This means that Q' is either present in G_1^*, or in G_2^*, and hence it cannot be a bad cycle, which is a contradiction. Hence $Q' \cap M \neq \emptyset$ as well. Finally, Q' contains vertices from both H_1 and H_2, which implies $Q' \cap G_1$ and $Q' \cap G_2$ are both non-empty as well.

Now, by applying Lemma 6.24 to Q' and C, we obtain a bad cycle $\hat{Q}$ such that $\hat{Q} \cap C$ is either a single vertex, or an edge or an induced path of length three. Since, $M \cap C = \emptyset$, we can again apply Lemma 6.24 to $\hat{Q}$ and M, and obtain a bad cycle Q such that each of $Q \cap C$ and $Q \cap M$ is either a single vertex, or, an edge or an induced path of length three. Hence, $Q - (V(M) \cup V(C))$ is a pair of internally disjoint paths, whose endpoints are in $M \times C$. Furthermore, one of these paths, denoted P_1, is contained in G_1, and the other, denoted P_2, is contained in G_2. □

The above lemma (Lemma 6.27) implies that it is safe to ignore all the bad cycles that don't satisfy the conclusion of this lemma. We proceed to enumerate some helpful properties of those bad cycles that satisfy the above lemma. We call P_1, P_2 the *path segments* of the bad cycle Q.

Lemma 6.28. *Suppose P_1, P_2 are path segments of a bad cycle Q where $P_1 \subseteq G_1 - S_1^*$ and $P_2 \subseteq G_2 - S_2^*$, where S_1^* and S_2^* are a solution to G_1^* and G_2^*, respectively. Then for any P_1' which is an induced path in $G_1 - S_1^*$ with the same endpoints as P_1 we have that $Q' = G'[(Q \cap (M \cup C)) \cup V(P_1') \cup V(P_2)]$ is also a bad cycle.*

Proof. Observe that, P_1 and P_1' are paths between the same endpoints in $G_1 - S_1^*$, which is a distance hereditary graph. Therefore, P_1' is an induced path of the same length as P_1. Furthermore, no vertex in P_1' is adjacent to a vertex in $Q - P_1$. Hence Q' is also a bad cycle. □

The above lemma allows us to reduce the problem of computing a solution that intersects all bad-cycles, to computing a solution for an instance of WEIGHTED MULTICUT. More formally, let Q be a bad cycle with path segments P_1 and P_2 , the feasible fractional solution $\mathbf{x}^*$ assigns a total value of at least 1 to the vertices in Q. As $\mathbf{x}^*$ assigns 0 to every vertex in $M \cup C$, we have that at least one of P_1 or P_2 is assigned a total value of at least $1/2$. Suppose that it were P_1 then $2\mathbf{x}^*$ assigns a total value 1 to P_1 in G_1. This fractional solution is a solution to the WEIGHTED MULTICUT problem defined on the pairs of vertices in $C \times M$, which are separated by $2\mathbf{x}^*$ in G' (whose description is given below).

Given $i \in \{1, 2\}$, let $2\mathbf{x}_i^*$ denote the fractional solution that assigns to each vertex the value assigned by $\mathbf{x}_i^*$ times 2. For a pair (v, u) of vertices such that $v \in V(C)$ and $u \in V(M)$ we call (v, u) an *important pair* if there is a bad cycle Q with path segments P_1 and P_2 that connects v and u. Let S_1^* and S_2^* be a solution to G_1^* and G_2^*, respectively (obtained recursively). For an important pair (v, u) we let $\mathcal{P}_1(v, u)$ denote the set of any (simple) path P_1 between v and u whose internal vertices belong only to $G_1 - S_1^*$ and which does not contain any edge such that one of its endpoints belongs to $V(C)$ while the other endpoint belongs to $V(M)$. Symmetrically, we let $\mathcal{P}_2(v, u)$ denote the set of any path P_2 between v and u whose internal vertices belong only to $G_2 - S_2^*$ and which does not contain any edge such that one of its endpoints belongs to $V(C)$ while the other endpoint belongs to $V(M)$.

Lemma 6.29. *For an important pair (v, u) of vertices where $v \in V(C)$ and $u \in V(M)$, in polynomial time we can compute an index $i(v, u) \in \{1, 2\}$ such that for any path $P \in \mathcal{P}_i(v, u)$, $2\mathbf{x}_i^*(V(P)) \geq 1$.*

Proof. Let (v, u) be an important pair of vertices with $v \in V(C)$ and $u \in V(M)$. We start by arguing that such an index exists. Assuming a contradiction, suppose there exists $P_1 \in \mathcal{P}_1(v, u)$ and $P_2 \in \mathcal{P}_2(v, u)$ such that $2\mathbf{x}_1^*(V(P_1)) < 1$ and $2\mathbf{x}_2^*(V(P_2)) < 1$. Recall that we have a bad cycle bad cycle Q in $G' - (S_1^* \cup S_2^*)$ with paths segments as P_1 and P_2 which connects v and u. But this implies that $2x^*(Q) < 1$, contradicting that x^* was a feasible solution to $G' - (S_1^* \cup S_2^*)$. Therefore, such an index always exists.

For any index $j \in \{1, 2\}$, we use Dijkstra's algorithm to compute the minimum weight of a path between v and u in the graph $\widehat{G}_i^*$ where the weights are given by $2\mathbf{x}_i^*$. In case the minimum weight is at least 1, we have found the desired index $i(v, u)$. Moreover,

we know that for at least one index $j \in \{1,2\}$, the minimum weight should be at least 1 (if the minimum weight is at least 1 for both induces, we arbitrarily decide to fix $i(v,u)=1$). □

We say that an important pair (u,v) is *separated* in G_i, if the index assigned by Lemma 6.29 assigns i to $\mathcal{P}_i(u,v)$. Now, for every important pair (v,u) such that $v \in V(C), u \in V(M)$ and $(v,u) \notin E(G')$, we perform the following operation. We check if this pair is separated in G_1, and if so, then we initialize $\mathcal{T}_1(v,u)=\emptyset$. Then for each pair of neighbors of x of v and y of u, we add the pair (x,y) to $\mathcal{T}_1(u,v)$. The set $\mathcal{T}_2(u,v)$ is similarly defined. At this point, we need to rely on approximate solutions to the Weighted Multicut problem which is given by theorem below (Theorem 6.5).

Theorem 6.5 ([GVY96]). *Given an instance of* Weighted Multicut*, one can find (in polynomial time) a solution that is at least* opt *and at most* $d \cdot \log n \cdot$ fopt *for some fixed constant* $d > 0$*, along with a set that realizes it.*

Here, a fractional solution $\mathbf{y}$ is a function $\mathbf{y} : V(G) \to [0,\infty)$ such that for every pair $(s_i,t_i) \in \mathcal{T}$ and any path P between s_i and t_i, it holds that $\mathbf{y}(V(P)) \geq 1$. An optimal fractional solution minimizes the weight $w(\mathbf{y}) = \sum_{v \in V(G)} w(v) \cdot \mathbf{y}(v)$. Let fopt denote the weight of an optimal fractional solution.

By employing the algorithm given by Lemma 6.29, we next construct two instances of Weighted Multicut. The first instance is $J_1 = (\widehat{G}_1^*, w_1^*, \mathcal{T}_1 = \{\mathcal{T}_1(v,u) \mid v \in V(C), u \in V(M), i(v,u) = 1, \text{ and } (v,u) \text{ is an important pair}\})$ and the second instance is $J_2 = (\widehat{G}_2^*, w_2^*, \mathcal{T}_2 = \{\mathcal{T}_2(v,u) \mid v \in V(C), u \in V(M), i(v,u) = 2, \text{ and } (v,u) \text{ is an important pair}\})$. By Lemma 6.29, $2\mathbf{x}_1^*$ and $2\mathbf{x}_2^*$ are valid solutions to J_1 and J_2, respectively. Thus, by calling the algorithm given by Theorem 6.5 with each instance, we obtain a solution r_1 to the first instance, along with a set R_1 that realizes it, such that $r_1 \leq 2d \cdot \log|V(G_1^*)| \cdot w^*(\mathbf{x}_1^*)$, and we also obtain a solution r_2 to the second instance, along with a set R_2 that realizes it, such that $r_2 \leq 2d \cdot \log|V(G_2^*)| \cdot w^*(\mathbf{x}_2^*)$. Now by Observation 18 and Lemma 6.27, we have obtained a set $S^* = S_1^* \cup S_2^* \cup R_1 \cup R_2$ for which we have the following observation.

Observation 21. *S^* intersects any chordless cycle in G^*, and it holds that $w^*(S^*) \leq s_1^* + s_2^* + r_1 + r_2$.*

We start by showing that $s_1^* + s_2^* + r_1 + r_2 + w(h(x)) \leq (\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log n \cdot \log(\alpha(G')) \cdot w^*(\mathbf{x}^*)$. Recall that for any $i \in \{1,2\}$, $r_i \leq 2d \cdot \log|V(G_i^*)| \cdot w^*(\mathbf{x}_i^*)$. Thus, by Observation 19 and since for any $i \in \{1,2\}$, $|V(G_i^*)| \leq n$ and $\alpha(G_i^*) \leq \frac{3}{4}\alpha(G')$, we have that

$$w^*(S^*) \leq (\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log n \cdot \log(\frac{3}{4}\alpha(G')) \cdot (w^*(\mathbf{x}_1^*) + w^*(\mathbf{x}_2^*)) + 2d \cdot \log n \cdot (w^*(\mathbf{x}_1^*) + w^*(\mathbf{x}_2^*)).$$

By Observation 20, we further deduce that

$$w^*(S^*) \leq \left((\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log(\frac{3}{4}\alpha(G')) + 2d \right) \cdot \log n \cdot w^*(\mathbf{x}^*).$$

Now, it only remains to show that $(\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log(\frac{3}{4}\alpha(G')) + 2d \leq (\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log \alpha(G')$, which is equivalent to $2d \leq (\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log(\frac{4}{3})$. Observe that $\delta \leq q \cdot \log n - 1$

for some fixed constant q. Indeed, it initially holds that $\alpha(G) \leq n$, at each recursive call, the number of vertices assigned a non-zero value by $\mathbf{x}^*$ decreases to at most a factor of $3/4$ of its previous value, and the execution terminates once this value drops below $(\log n)/2$. Thus, it is sufficient to choose the constant c so that $2d \leq (\frac{\log n}{\log n+4})^{q\cdot\log n} \cdot c \cdot \log(\frac{4}{3})$. As the term $(\frac{\log n}{\log n+4})^{q\cdot\log n}$ is lower bounded by $1/(e^{4q})$, it is sufficient that we fix $c = 2 \cdot e^{4q} \cdot d \cdot 1/\log(\frac{4}{3})$.

Note that $2d \leq (\frac{\log n}{\log n+4})^{q\cdot\log n}\cdot c\cdot\log(\frac{4}{3})$, where $d \geq 1$. Therefore, $(\frac{\log n}{\log n+4})^{q\cdot\log n}\cdot c \geq 1$. This together with Lemma 6.23 and 6.25 implies that $w'(S^*) + w'(h(x)) \leq (\frac{\log n}{\log n+4})^{\delta} \cdot c \cdot \log(\alpha(G'))w'(x)$, which proves Lemma 6.22.

6.4.2 General graphs

In this subsection, we handle general instances by developing a $d \cdot \log^2 n$-factor approximation algorithm for WDHVD, Gen-DHD-APPROX, thus proving the correctness of Theorem 6.4.

The Recursive Algorithm. We define each call to our algorithm Gen-DHD-APPROX to be of the form (G', w'), where (G', w') is an instance of WDHVD such that G' is an induced subgraph of G, and we denote $n' = |V(G')|$. We ensure that after the initialization phase, the graph G' never contains a DH-obstruction on at most 50 vertices. We call this invariant the $\mathcal{O}_{50}$-*free invariant*. In particular, this guarantee ensures that the graph G' always contains only a small number of maximal bicliques, as stated in the following lemma.

Lemma 6.30 (Lemma 3.5 [KK16])**.** *Let G be a graph on n vertices with no DH-obstruction on at most* 6 *vertices. Then G contains at most $(n^3+5n)/6$ maximal bicliques, and they can be enumerated in polynomial time.*

Goal. For each recursive call Gen-DHD-APPROX(G', w'), we aim to prove the following.

Lemma 6.31. Gen-DHD-APPROX *returns a solution that is at least* opt *and at most $\frac{d}{2} \cdot \log^3 n' \cdot$* opt*. Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution. Here d is a constant, which will be determined later.*

At each recursive call, the size of the graph G' becomes smaller. Thus, when we prove that Lemma 6.31 is true for the current call, we assume that the approximation factor is bounded by $\frac{d}{2} \cdot \log^3 \widehat{n} \cdot$ opt for any call where the size $\widehat{n}$ of the vertex-set of its graph is strictly smaller than n'.

Initialization. We are given (G, w) as input, and first we need to ensure that the $\mathcal{O}_{50}$-free invariant is satisfied. For this purpose, we update G as follows. First, we let $\mathcal{O}_{50}$ denote the set of all DH-obstruction on at most 50 vertices of G. Clearly, $\mathcal{O}_{50}$ can be computed in polynomial time and it holds that $|\mathcal{O}_{50}| \leq n^{\mathcal{O}(1)}$. Now, we construct an instance of Weighted 50-Hitting Set, where the universe is $V(G)$, the family of all setsof size at most 50 in $\mathcal{O}_{50}$, and the weight function is w'. Since each DH-obstruction must be intersected, therefore, the optimal solution to our Weighted 50-Hitting Set instance is at most opt. By using the standard c'-approximation algorithm for

Weighted c'-Hitting Set [KT05], which is suitable for any fixed constant c', we obtain a set $S \subseteq V(G)$ that intersects all the DH-obstruction in $\mathcal{O}_{50}$ and whose weight is at most $50 \cdot \mathsf{opt}$. Having the set S, we remove its vertices from G to obtain the graph G', and $w' = w|_{G'}$. Now that the $\mathcal{O}_{50}$-free invariant is satisfied, we can call Gen-DHD-APPROX on (G', w') and to the outputted solution, we add $w(S)$ and S.

We note that during the execution of the algorithm, we update G' only by removing vertices from it, and thus it will always be safe to assume that the $\mathcal{O}_{50}$-free invariant is satisfied. Now, by Lemma 6.31, we obtain a solution of weight at most $\frac{d}{2} \cdot \log^3 n \cdot \mathsf{opt} + 50 \cdot \mathsf{opt} \leq d \cdot \log^3 n \cdot \mathsf{opt}$, then combined with S, it allows us to conclude the correctness of Theorem 6.4.

Termination. Observe that due to Lemma 6.30, we can test in polynomial time, if our current graph G' is of the special kind that can be partitioned into a biclique and a distance hereditary graph: we examine each maximal biclique of G', and check whether after its removal we obtain a distance hereditary graph. Once G' becomes such a graph that consists of a biclique and a distance hereditary graph, we solve the instance (G', w') by calling algorithm DHD-APPROX. Observe that this returns a solution of value $\mathcal{O}(\log^2 n \cdot \mathsf{opt})$ which is also $\mathcal{O}(\log^3 n \cdot \mathsf{opt})$.

Recursive Call. Similar to the case for WCVD, instead computing a balanced separators with a maximal clique and some additional vertices, here we find a balanced separator that comprises of a biclique and some additional, but small number of vertices. Existence of such a separator is guaranteed by Lemma 6.32. From Lemma 6.30, it follows that the graph with no DH-obstruction of size at most 50 contains at most $\mathcal{O}(n^3)$ maximal bicliques and they can enumerated in polynomial time. We use the weighted variant of Lemma 3.8 from [KK16] in Lemma 6.32. The proof of Lemma 6.32 remains exactly the same as that in Lemma 3.8 of [KK16].

Lemma 6.32 (Lemma 3.8 [KK16])**.** *Let G' be a connected graph on n' vertices not containing any DH-obstruction of size at most 50 and $w : V(G) \to \mathbb{Q}$ be a weight function. Then in polynomial time we can find a balanced vertex separator $K \uplus X$ such that the following conditions are satisfied.*

- *K is a biclique in G or an empty set;*
- *$w(X) \leq q \cdot \log n' \cdot \mathsf{opt}$, where q is some fixed constant.*

Here, opt *is the weight of the optimum solution to* WDHVD *of G.*

We note that we used the $\mathcal{O}(\log n')$-factor approximation algorithm by Leighton and Rao [LR99] in Lemma 6.32 to find the balanced separator, instead of the $\mathcal{O}(\sqrt{\log n'})$-factor approximation algorithm by Feige et al. [FHL08], as the algorithm by Feige et al. is randomized. Let us also remark that if K is a biclique, then there is a bipartition of the vertices in K into $A \uplus B$, where both A and B are non-empty, which will be crucially required in later arguments.

Next, we apply in Lemma 6.32 to (G', w') to obtain a pair (K, X). Since $K \cup X$ is a balanced separator for G', we can partition the set of connected components of $G' - (M \cup S)$ into two sets, $\mathcal{A}_1$ and $\mathcal{A}_2$, such that for $V_1 = \bigcup_{A \in \mathcal{A}_1} V(A)$ and $V_2 = \bigcup_{A \in \mathcal{A}_2} V(A)$ it holds that $n_1, n_2 \leq \frac{2}{3} n'$ where $n_1 = |V_1|$ and $n_2 = |V_2|$. We then define

two inputs of (the general case) WDHVD: $I_1 = (G'[V_1], w'_{V_1})$ and $I_2 = (G'[V_2], w'_{V_2})$. Let opt_1 and opt_2 denote the optimal solutions to I_1 and I_2, respectively. Observe that since $V_1 \cap V_2 = \emptyset$, it holds that $\mathsf{opt}_1 + \mathsf{opt}_2 \leq \mathsf{opt}$. We solve each of the two sub-instances by recursively calling algorithm Gen-DH-APPROX. By the inductive hypothesis, we obtain two sets, S_1 and S_2, such that $G'[V_1] - S_1$ and $G'[V_2] - S_2$ are both distance hereditary graphs, and $w'(S_1) \leq \frac{d}{2} \cdot \log^3 n_1 \cdot \mathsf{opt}_1$ and $w'(S_2) \leq \frac{d}{2} \cdot \log^3 n_2 \cdot \mathsf{opt}_2$. Now, if K were an empty set then it is easy to see that $X \cup S_1 \cup S_2$ is a feasible solution to the instance (G', w'). Now let us bound the total weight of this subset.

$$\begin{aligned} w'(X \cup S_1 \cup S_2) &\leq w'(X) + w'(S_1) + w'(S_2) \\ &\leq q \cdot \log n' \cdot \mathsf{opt} + \tfrac{d}{2} \cdot (\log^3 n_1 \cdot \mathsf{opt}_1 + \log^3 n_2 \cdot \mathsf{opt}_2) \\ &\quad \text{Recall that } n_1, n_2 \leq \tfrac{2}{3}n' \text{ and } \mathsf{opt}_1 + \mathsf{opt}_2 \leq \mathsf{opt}. \\ &< q \cdot \log n' \cdot \mathsf{opt} + \tfrac{d}{2} \cdot \log^3 \tfrac{2}{3}n' \cdot \mathsf{opt} \\ &< \tfrac{d}{2} \cdot \log^3 n' \cdot \mathsf{opt} \end{aligned}$$

The more interesting case is when K is a biclique. Then, we first remove $X \cup S_1 \cup S_2$ from the graph, and note that the above bound also holds for this subset of vertices. Now observe that the graph $G'' = G' - (X \cup S_1 \cup S_2)$ can be partitioned into a biclique K and a distance hereditary graph $H = G[(V_1 \cup V_2) \setminus (S_1 \cup S_2)]$, along with the weight function $w'' = w'_{V(G'')}$. Thus we have an instance of the Biclique + Distance Hereditary Graph spacial case of WDHVD. Furthermore, note that we retained a fractional feasible solution $\mathbf{x}$ to the LP of the initial input G', w', which upperbounds the value of a fractional feasible solution $\mathbf{x}''$ to the LP of the instance G'', w''. We apply the algorithm DHD-APPROX on $(G'', w'', K, H, \mathbf{x}'')$ which outputs a solution S such that $w''(S) = w'(S) = \mathcal{O}(\log^2 n \cdot \mathsf{opt})$.

Observe that, any obstruction in $G' \setminus S$ is either completely contained in $G'[V_1 \setminus S]$, or completely contained in $G'[V_2 \setminus S]$ or it contains at least one vertex from K. This observation, along with the fact that $G'[(V_1 \cup V_2 \cup K) \setminus (S_1 \cup S_2 \cup \widehat{S})]$ is a distance hereditary graph, implies that $G' - T$ is a distance hereditary graph where $T = X \cup S_1 \cup S_2 \cup \widehat{S}$. Thus, it is now sufficient to show that $w'(T) \leq \frac{d}{2} \cdot (\log n')^3 \cdot \mathsf{opt}$. By the discussion above, we have that DHD-APPROX returns a solution of value $c \log^2 n \cdot \mathsf{opt}$, where c is some constant.

$$\begin{aligned} w'(T) &\leq w'(S) + w'(S_1) + w'(S_2) + w'(\widehat{S}_1) + w'(\widehat{S}_2) \\ &\leq q \cdot \log n' \cdot \mathsf{opt} + \tfrac{d}{2} \cdot (\log^3 n_1 \cdot \mathsf{opt}_1 + \log^3 n_2 \cdot \mathsf{opt}_2) + c \cdot \log^2 n' \cdot \mathsf{opt}. \end{aligned}$$

Recall that $n_1, n_2 \leq \frac{2}{3}n'$ and $\mathsf{opt}_1 + \mathsf{opt}_2 \leq \mathsf{opt}$. Thus, we have that

$$\begin{aligned} w'(T) &\leq q \cdot \log n' \cdot \mathsf{opt} + \tfrac{d}{2} \cdot (\log^3 \tfrac{2}{3}n') \cdot \mathsf{opt} + c \cdot \log^2 n' \cdot \mathsf{opt} \\ &\leq \tfrac{d}{2} \cdot \log^3 n' \cdot \mathsf{opt} + (c - d \log \tfrac{3}{2}) \cdot \log^2 n' \cdot \mathsf{opt}. \end{aligned}$$

Overall, we conclude that to ensure that $w'(T) \leq \frac{d}{2} \cdot \log^3 n' \cdot \mathsf{opt}$, it is sufficient to ensure that $c - d \log \frac{3}{2} \leq 0$, which can be done by fixing $d = \dfrac{c}{\log \frac{3}{2}}$.

Chapter 7

Kernel for Chordal Vertex Deletion

In this chapter, we look at the problem Chordal Vertex Deletion, which is formally defined below.

Chordal Vertex Deletion (CVD)	**Parameter:** k
Input: A graph G and an integer k.	
Question: Does there exist a subset $S \subseteq V(G)$ such that $\lvert S\rvert \leq k$ and $G - S$ is a chordal graph?	

The existence of a polynomial kernel for CVD was a well-known open problem in the field of Parameterized Complexity. Recently, Jansen and Pilipczuk resolved this question affirmatively by designing a polynomial (vertex) kernel for CVD of size $\mathcal{O}(k^{161}\log^{58} k)$, and asked whether one can design a kernel of size $\mathcal{O}(k^{10})$ [JP17]. In this chapter, we look at a kernel for the problem with $\mathcal{O}(k^{12}\log^{10} k)$vertices. In particular, the focus of this chapter will be the proof of the following theorem.

Theorem 7.1. CVD *admits a polynomial kernel of size* $\mathcal{O}(k^{12}\log^{10} k)$.

The kernelization algorithm is inspired by the $\mathcal{O}(k^2)$-size kernel for Feedback Vertex Set (FVS), designed by Thomassé [Tho10]. The kernel for FVS consists of the two following steps.

1. Reduce the maximum degree of the graph by using Expansion Lemma. That is, upper bound the maximum degree Δ of the graph by $\mathcal{O}(k)$.

2. When the graph has maximum degree Δ, one can show that if a graph has minimum degree at least 3 (which can be easily achieved for FVS) then any minimum feedback vertex set has size $\mathcal{O}(n/\Delta)$. This together with an upper bound on Δ implies $\mathcal{O}(k^2)$-size kernel.

Recall that a graph is chordal if it does not contain any induced cycle of length at least 4. That is, every cycle of length at least 4 has a chord. Thus, FVS is about intersecting all cycles and CVD is about intersecting all chordless cycles. Unfortunately, this apparent similarity stops here! Nevertheless, we are still able to exploit ideas used in the $\mathcal{O}(k^2)$-size kernel for FVS. Towards this, we define the notion of *independence degree* of vertices and graphs. Roughly speaking, the independence degree of a vertex v is the size of a maximum independent set in its neighborhood ($G[N(v)]$). The study of this notion is key conceptual contribution of this chapter.

As a first step we bound the independence degree of every vertex by $k^{\mathcal{O}(1)}$ – this is similar to the first step of the kernel for FVS. Once we have bounded the independence degree of a graph, we obtain an approximate solution M (also called modulator) and analyze the graph $G - M$. The bound on the independence degree immediately implies that the number of leaves, vertices of degree at least three and the number of maximal degree two paths in the clique forest of $G - M$ is bounded by $k^{\mathcal{O}(1)}$. Then, using ideas similar to those used by Marx [Mar10] to bound the size of a maximal clique while designing the first algorithm for CVD, we reduce the size of each maximal clique in $G - M$ to $k^{\mathcal{O}(1)}$. Finally, we use structural analysis to bound the size of each maximal degree two path, which includes the design of a reduction rule that computes a family of minimum cuts, and thus we obtain our final kernel. We believe that the notion of independent degree is likely to be useful for designing algorithms for other graph modification problems. Not only does it lead to a significant improvement, once it is bounded, it greatly simplifies the analysis of the resulting instance.

Finally, after we obtain a kernel, we show that by reruning our entire kernelization procedure once, we can actually reduce the size of the kernel. Since we rely on an $\mathcal{O}(\log^2 n)$-approximation algorithm, when we call our kernelization procedure for the first time, we work with an approximate solution of size $\mathcal{O}(k^3 \log^2 k)$; indeed, it can be assumed that $\log n < k \log k$, else the $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$-time algorithm for CVD by Cao and Marx [CM16] solves the input instance in polynomial time. However, once we have our first kernel, it holds that $n = \mathcal{O}(k^{22} \log^{12} k)$. At this point, if we reuse the approximation algorithm, we obtain an approximate solution of size $\mathcal{O}(k \log^2 k)$. The size of the kernel depends on the size of the approximate solution; in particular, having an approximate solution of size $\mathcal{O}(k \log^2 k)$ allows us to obtain a kernel of size $\mathcal{O}(k^{12} \log^{10} k)$.

An overview of the kernelization algorithm. First, in Section 7.1, we briefly state results relating to approximate solutions for CVD, which will be relevant to following subsections. In Section 7.2 we address annotations that will be added to the input instance. Next, in Section 7.3, we introduce the notion of the *independent degree* of a vertex, which lies at the heart of the design of our kernelization algorithm. We carefully examine the independent degrees of vertices in our graphs, and show how these degrees can be bounded by a small polynomial in k. In Section 7.4, we consider the clique forest of the graph obtained by removing (from the input graph) the vertices of an approximate solution. In particular, we demonstrate the usefulness of our notion of an independent degree of a vertex – having bounded the independent degree of each vertex, we show that the number of leaves in the clique forest can be bounded in a simple and elegant manner. We also efficiently bound the size of a maximal clique. Then, in Section 7.5, we turn to bound the length of degree-2 paths in the clique forest. This subsection is quite technical, and its proofs are based on insights into the structure of chordal graphs and their clique forests. In particular, we use a reduction rule which computes a collection of minimum cuts rather than one minimum cut which overall allows us to capture the complexity of a degree-2 path using only few vertices. In Section 7.6, we remove annotations introduced in preceding sections. Next, in Section 7.7, we bound the size of our kernel. Finally, in Section 7.8, we show that an alternating application of approximation and kernelization can improve the performance of our kernelization algorithm.

7.1 Approximation

Observe that it can be assumed that $\log n < k \log k$, else the $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$-time algorithm for CVD by Cao and Marx [CM16] solves the input instance in polynomial time. This together with the fact that CVD admits an $\mathcal{O}(\log^2 n)$-factor approximation algorithm (Chapter 6, Section 6.2, Theorem 6.2), we obtain the following result.

Lemma 7.1. CVD *admits an $\mathcal{O}(k^2 \log^2 k)$-factor approximation algorithm.*

Throughout the chapter, we let APPROX denote a polynomial-time algorithm for CVD that returns approximate solutions of size $f(\mathsf{opt})$ for some function f. Initially, it will denote the algorithm given by Lemma 7.1. Given an instance of CVD, we say that an approximate solution D is *redundant* if for every vertex $v \in D$, $D \setminus \{v\}$ is also an approximate solution, that is, $G - (D \setminus \{v\})$ is a chordal graph. Jansen and Pilipczuk [JP17] showed that given an approximate solution of size $g(k)$ for some function $g(\cdot)$, one can find (in polynomial time) either a vertex contained in every solution of size at most k or a redundant approximate solution of size $\mathcal{O}(k \cdot g(k))$. Thus, we have the following result.

Corollary 7.1 ([JP17])**.** *Given an instance of* CVD*, one can find (in polynomial time) either a vertex contained in every solution of size at most k or a redundant approximate solution of size $\mathcal{O}(k \cdot f(k))$.*

Next, we fix an instance (G, k) of CVD. By relying on APPROX and Corollary 7.1, we may assume that we have a vertex-set $\widetilde{D} \subseteq V(G)$ that is an approximate solution of size $f(k)$ and a vertex-set $D \subseteq V(G)$ that is a redundant approximate solution of size $c \cdot k \cdot f(k)$ for some constant c independent of the input.

Given a $v \in V(G)$, we say that a vertex set $B \subseteq V(G) \setminus \{v\}$ is a *v-blocker* if B intersects every chordless cycle in G. Observe that the set B must cannot contain the vertex v. In the following section, we will also need to strengthen approximate solutions to be v-blockers for some vertices $v \in V(G)$. To this end, we will rely on the following result.

Lemma 7.2. *Given a vertex $v \in V(G)$, one can find (in polynomial time) either a vertex contained in every solution of size at most k or an approximate solution of size $f(k)$ that is a v-blocker.*

Proof. Fix a vertex $v = v_0 \in V(G)$. We define the graph G' by setting $V(G') = V(G) \cup \{v_1, v_2, \ldots, v_{f(k)}\}$, where $v_1, v_2, \ldots, v_{f(k)}$ are new vertices, and $E(G') = E(G) \cup \{(u, v_i) \mid (u, v) \in E(G), i \in [f(k)]\} \cup \{(v_i, v_j) \mid i \neq j \in \{0\} \cup [f(k)]\}$. In other words, G' is the graph G to which we add $f(k)$ copies of the vertex v, which form a clique amongst themselves and v.

We call the algorithm APPROX to obtain an approximate solution S. Since $G'[\{v_0, v_1, \ldots, v_{f(k)}\}]$ is a clique and for each $i, j \in \{0\} \cup [f(k)]$ we have $N[v_i] = N[v_j]$, any chordless cycle in G' contains at most one vertex from $\{v_0, v_1, \ldots, v_{f(k)}\}$. In particular, for any chordless cycle C in G', by replacing v_i by v_0 (in case v_i belongs to C), we obtain a chordless cycle in G. Therefore, if the instance (G, k) admits a solution of size at most k that does not contain v, then this solution is also a solution for the instance (G', k), which implies that the size of S should be at most $f(k)$. Thus, we can next assume that $|S| \leq f(k)$, else we conclude that the vertex v is contained in every solution

of size at most k. Since the vertices $v_0, v_1, v_2, \ldots, v_{f(k)}$ have the same neighbor-set, if $\{v_0, v_1, v_2, \ldots, v_{f(k)}\} \cap S \neq \emptyset$, we can assume that $\{v_0, v_1, v_2, \ldots, v_{f(k)}\} \subseteq S$, since otherwise we can take $S \setminus \{v_0, v_1, v_2, \ldots, v_{f(k)}\}$ as our approximate solution S. Since we assume that $|S| \leq f(k)$, we deduce that $\{v_0, v_1, v_2, \ldots, v_{f(k)}\} \cap S = \emptyset$. In particular, we have that $v \notin S$ and therefore S is a v-blocker. □

In light of Lemma 7.2, we may next assume that for every vertex $v \in V(G)$, we have a vertex-set B_v that is both a v-blocker and an approximate solution of size $f(k)$.

7.2 Irrelevant and Mandatory Edges

During the execution of our kernelization algorithm, we mark some edges in $E(G)$ as *irrelevant edges*. At the beginning of its execution, all of the edges in $E(G)$ are assumed to be relevant edges. When we mark an edge as an irrelevant edge, we prove that any solution that intersects all of the chordless cycles in G that contain only relevant edges also intersects all of the chordless cycles in G that contain the irrelevant edge. In other words, we prove that we can safely ignore chordless cycles that contain at least one irrelevant edge. Observe that we cannot simply remove irrelevant edges from $E(G)$ since this operation may introduce new chordless cycles in G. Instead we maintain a set E_I, which contains the edges marked as irrelevant.

We also mark some edges in $E(G)$ as *mandatory edges*. We will ensure that at least one endpoint of a mandatory edge is present in any solution of size at most k. We let E_M denote the set of mandatory edges.

In some situations, we identify a pair of non-adjacent vertices such that any solution of size at most k must contain at least one of them. Then, we add an edge between the vertices of the pair, and mark this edge as a mandatory edge. The correctness of this operation follows from the observation that any chordless cycle affected by the addition of this edge contains both vertices of the pair, and since the edge is marked as a mandatory edge, we ensure this chordless cycle will be intersected although it may no longer exist. Formally, we have the following reduction rule.

Reduction Rule 7.1. *Given two non-adjacent vertices in G, v and u, such that at least one of them belongs to any solution of size at most k, insert the edge (v, u) into both $E(G)$ and E_M.*

Hence from now onwards our instance is of the form (G, k, E_I, E_M), and during the execution of our kernelization algorithm, we will update the sets E_I and E_M. In Section 7.6, we show that we can unmark the edges in $E_I \cup E_M$, obtaining an ordinary instance of CVD. For the sake of simplicity, when E_I and E_M are clear from context, we omit them.

The Number of Mandatory Edges. If a vertex v is incident to at least $k + 1$ mandatory edges, it must belong any solution of size at most k. Therefore, we may safely apply the following reduction rule.

Reduction Rule 7.2. *If there exists a vertex v incident to at least $k + 1$ mandatory edges, remove v from G and decrement k by 1.*

After exhaustively applying the above reduction rule we have the following lemma.

Lemma 7.3. *If $|E_M| > k^2$ then the input instance is a no-instance.*

Proof. After the exhaustive application of Reduction Rule 7.2, any vertex incident to a mandatory edge is incident to at most k such edges. Therefore, any set of at most k vertices from $V(G)$ may intersect at most k^2 mandatory edge. Since every solution of size at most k must intersect every mandatory edge, we deduce that if $|E_M| > k^2$, the input instance is a *no*-instance. □

Thus, we will next assume that $|E_M| \leq k^2$ (else Reduction Rule 7.2 applies). Moreover, we let D' denote the set $D \cup \widetilde{D}$ to which we add every vertex that is an endpoint of a vertex in E_M (recall that $\widetilde{D}$ is our approximate solution of size at most $f(k)$ and D is our redundant approximate solution of size $\mathcal{O}(k \cdot f(k))$). Observe that by adding vertices to a redundant approximate solution, it remains a redundant approximate solution, and therefore D' or any other superset of D is such a solution.

7.3 Independent Degree

Given a vertex $v \in V(G)$, we use the notation $N^R_G(v)$ to refer to the set containing each vertex $u \in N_G(v)$ such that (v, u) does not belong to $E_I \cup E_M$. We remark that in this section we identify and mark some edges as irrelevant edges.

Independent Degrees. We start by introducing the notion of the independent degree of vertices and graphs.

Definition 7.1. Given a vertex $v \in V(G)$, the *independent degree of* v, denoted by $d^I_G(v)$, is the size of a maximum independent set in the graph $G[N^R_G(v)]$. The *independent degree of* G, denoted by Δ^I_G, is the maximum independent degree of a vertex in $V(G)$.

Fix $\Delta = (k+3)f(k)$. The objective of this subsection is to investigate the notion of an independent degree, ultimately proving the following result.

Lemma 7.4. *One can construct (in polynomial time) an instance (G', k', E'_I, E'_R) of* CVD *that is equivalent to the input instance (G, k, E_I, E_R) and such that both $k' \leq k$ and $\Delta^I_{G'} \leq \Delta$.*

To this end, we may assume that we are given a vertex $v \in V(G)$ such that $d^I_G(v) > \Delta$. We say that an instance (G', k', E'_I, E'_M) of CVD is *better* than the input instance (G, k, E_I, E_M) if $k' \leq k$, $V(G') = V(G)$, $E_I \subseteq E'_I$, $E_M \subseteq E'_M$ $d^I_{G'}(v) \leq \Delta$ and for all $u \in V(G')$, $d^I_{G'}(u) \leq d^I_G(u)$. To prove the correctness of Lemma 7.4, it is sufficient to prove the correctness of the following lemma.

Lemma 7.5. *We can construct (in polynomial time) an instance (G', k', E'_I, E'_M) of* CVD *that is better than the input instance (G, k, E_I, E_M).*

Indeed, to prove Lemma 7.4, one can repeatedly apply the operation given by Lemma 7.5 in the context of every vertex $u \in V(G)$ such that $d^I_G(u) > \Delta$. We start with a simple result concerning independent degrees.

Lemma 7.6. *Let $u \in V(G)$ be a vertex such that $d^I_G(u) \geq |B_u|$. Then, one can find (in polynomial time) an independent set in $G[N^R_G(u) \setminus B_u]$ of size at least $d^I_G(u) - |B_u|$.*

Proof. Since B_u is a solution, $G - B_u$ is a chordal graph. In particular, $G[N_G^R(u) \setminus B_u]$ is a chordal graph. Since chordal graphs are perfect, Maximum Independent Set in chrodal graphs is solvable in polynomial time [Gol04]. This means that we can find a maximum independent set in $G[N_G^R(u) \setminus B_u]$ in polynomial time. Since the size of a maximum independent set in $G[N_G^R(u)]$ is at least $d_G^I(u)$, the size of a maximum independent set in $G[N_G^R(u) \setminus B_u]$ is at least $d_G^I(u) - |B_u|$, which concludes the correctness of the lemma. □

Recall that for any vertex $u \in V(G)$, $|B_u| \leq f(k)$. Thus, we have the following corollary.

Corollary 7.2. *One can find (in polynomial time) an independent set in $G[N_G^R(v) \setminus B_v]$ of size at least $\Delta - f(k)$.*

We let I denote the independent set given by Corollary 7.2.

Independent Components. Let $X = N_G(v) \setminus (B_v \cup I)$ denote the neighbor-set of v from which we remove the vertices of the v-blocker B_v and of the independent set I. We also let $H = G - (\{v\} \cup B_v \cup X)$ denote the graph obtained by removing (from G) the vertex v, the v-blocker B_v and any neighbor of v that does not belong to the independent set I. We define the set independent components of v as the set of each connected component of H that contains at least one vertex from I, and denote this set by $\mathcal{A}$.

For the set $\mathcal{A}$, we prove the following lemmata.

Lemma 7.7. *Each connected component $A \in \mathcal{A}$ contains exactly one vertex from I and no other vertex from $N_G(v)$.*

Proof. The graph H does not contain any vertex from $N_G(v) \setminus I$, and therefore we only need to prove the first part of the statement of the lemma. Fix a connected component $A \in \mathcal{A}$. By the definition of $\mathcal{A}$, it is only necessary to prove that A cannot contain (at least) two vertices from I. Suppose, by way of contradiction, that it contains two such vertices, u and w. Let P denote the shortest path in A that connects u and w. Since I is an independent set, this path contains at least two edges. Therefore, together with the vertex v, the path P forms a chordless cycle. However, this chordless cycle contain no vertex from B_v, which contradicts the fact that B_v is an approximate solution. □

By Lemma 7.7, for each connected component $A \in \mathcal{A}$ we can let $z(A) \in I$ denote the unique neighbor of v in A. In fact, Corollary 7.2 and Lemma 7.7 imply that $\Delta - f(k) \leq |\mathcal{A}|$.

Lemma 7.8. *Every vertex $x \in X$ that is adjacent (in G) to some vertex $y \in V(A)$, where $A \in \mathcal{A}$, is also adjacent (in G) to the vertex $z(A)$.*

Proof. Let $x \in X$ be a vertex that is adjacent (in G) to some vertex $y \in V(A)$. Suppose, by way of contradiction, that $(x, z(A)) \notin E(G)$. Let P denote the shortest path in A that connects y and $z(A)$. This path contains at least two edges. Therefore, together with the vertex v, the path P forms a chordless cycle. However, this chordless cycle contain no vertex from B_v, which contradicts the fact that B_v is an approximate solution. □

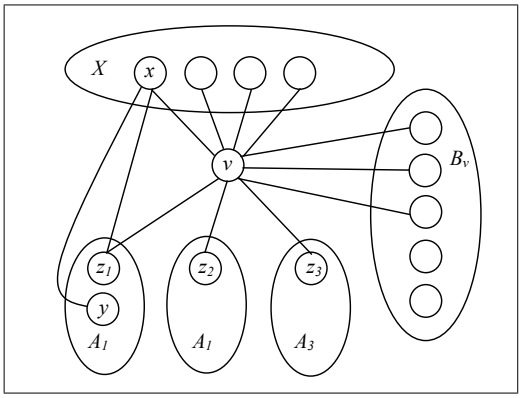

Figure 7.1: The relations between v, B_v, X and $\mathcal{A}$.

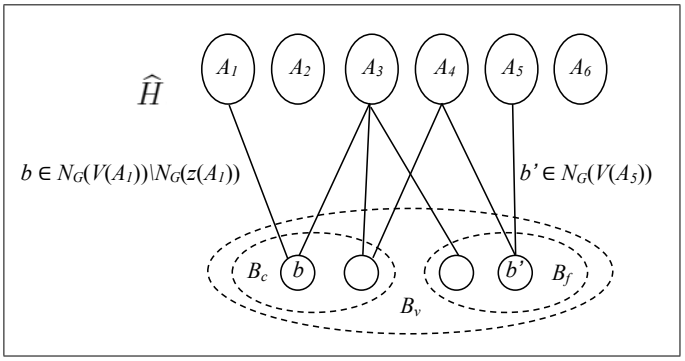

Figure 7.2: The bipartite graph $\widehat{H}$.

An illustration of the relations between v, B_v X and $\mathcal{A}$ is given in Figure 7.1.

The Bipartite Graph $\widehat{H}$. To decrease $d^I_G(u)$, we will consider the bipartite graph $\widehat{H}$, which is defined as follows. We define vertex-set of $\widehat{H}$ by $V(\widehat{H}) = \mathcal{A} \cup B_v$. In this context, we mean that each connected component $A \in \mathcal{A}$ is represented by a vertex in $V(\widehat{H})$, and for the sake of simplicity, we use the symbol A also to refer to this vertex. We partition B_v into two sets, B_c and B_f, where B_c contains the vertices in B_v that are adjacent (in G) to v, while B_f contains the remaining vertices in B_v. Here, the letters c and f stand for "close" and "far". Having this partition, we define the edge-set of $\widehat{H}$ as follows. For every vertex $b \in B_c$ and connected component $A \in \mathcal{A}$ such that $b \in N_G(V(A)) \setminus N_G(z(A))$ (i.e., b is a neighbor of some vertex in A but not of the vertex $z(A)$), insert the edge (b, A) into $E(\widehat{H})$. Moreover, for every vertex $b \in B_f$ and connected component $A \in \mathcal{A}$ such that $b \in N_G(V(A))$, insert the edge (b, A) into $E(\widehat{H})$. An illustration of the bipartite graph $\widehat{H}$ is given in Figure 7.2. The motivation behind its definition lies at the following lemma.

Lemma 7.9. *The bipartite graph $\widehat{H}$ satisfies the following properties.*

1. *Suppose that we are given an edge $(b, A) \in E(\widehat{H})$ such that $b \in B_c$ and $A \in \mathcal{A}$.*

Then, the graph G has a chordless cycle defined by the edges (v, b) and $(v, z(A))$ and a path in A.

2. *Suppose that we are given edges $(b, A), (b, A') \in E(\widehat{H})$ such that $b \in B_f$ and $A, A' \in \mathcal{A}$. Then, the graph G has a chordless cycle defined by the edges $(v, z(A)), (v, z(A'))$, the vertex b and paths in A and A'.*

Proof. To prove the first item, let y be a vertex in $N_G(b) \cap V(A)$, whose existence is guaranteed by the assumption that $(b, A) \in E(\widehat{H})$. The desired chordless cycle can be defined by the edges $(v, b), (v, z(A)), (b, y)$ and the edge-set of a shortest path between $z(A)$ and y in A – in particular, by the definition of $\mathcal{A}$ and Lemma 7.7, v is not adjacent to any vertex on this path, excluding $z(A)$, and since $(b, A) \in E(\widehat{H})$, b is not adjacent to $z(A)$.

For the second item, choose $y \in N_G(b) \cap V(A)$ and $y' \in N_G(b) \cap V(A')$, whose existence is guaranteed by the assumption that $(b, A), (b, A') \in E(\widehat{H})$, such that the length of a shortest path between $z(A)$ ($z(A')$) and y (resp. y') in A (resp. A') is minimum. Observe that the cycle defined the edges $(v, z(A)), (v, z(A')), (y, b), (y', b)$ and the shortest paths between $z(A)$ and y in A and $z(A')$ and y' in A', respectively, is a chordless cycle – in particular, by the definition of $\mathcal{A}$, Lemma 7.7 and since $b, b' \in B_f$, the cycle cannot have a chord containing v. □

Isolated Vertices in $\widehat{H}$. We start investigating the bipartite graph $\widehat{H}$ by examining the isolated vertices in $\mathcal{A}$ that it contains. In this context, we need the two following lemmata.

Lemma 7.10. *Let $A \in \mathcal{A}$ be an isolated vertex in $\widehat{H}$, and denote $z = z(A)$. Then, $N_G(V(A)) = N_G(z) \setminus V(A)$.*

Proof. Since $z \in A$, it is clear that $N_G(z) \setminus V(A) \subseteq N_G(V(A))$. Next, we show that $N_G(V(A)) \subseteq N_G(z) \setminus V(A)$. To this end, consider some vertices $y \in V(A)$ and $u \in N_G(y) \setminus V(A)$, and suppose, by way of contradiction, that $u \notin N_G(z)$. Because A is a connected component of H, it holds that $u \in \{v\} \cup B_v \cup X$. Moreover, because A is isolated in $\widehat{H}$ and yet $u \in N_G(V(A)) \setminus N_G(z)$, it further holds that $u \in X$. However, this results in a contradiction to the statement of Lemma 7.8. □

Lemma 7.11. *Let $A \in \mathcal{A}$ be an isolated vertex in $\widehat{H}$, and denote $z = z(A)$. Then, G does not have a chordless cycle that contains the edge (v, z).*

Proof. Suppose, by way of contradiction, that G has a chordless cycle C that contains the edge (v, z). The cycle C must contain an edge (y, u) such that $y \in A$ and $u \notin A$. By Lemma 7.10, $N_G(V(A)) = N_G(z) \setminus A$, and therefore $(z, u) \in E(G)$. Hence, since the cycle C is chordless, it holds that $z = y$, which, in turn, implies that $u \notin N_G(v)$. We deduce that u must belong to B_f. However, this implies that $(A, u) \in E(\widehat{H})$, contradicting the assumption that A is an isolated vertex in $\widehat{H}$. □

Lemma 7.11 leads us to the design of the following reduction rule.

Reduction Rule 7.3. *If the graph $\widehat{H}$ contains an isolated vertex $A \in \mathcal{A}$, mark the edge (v, z) as irrelevant.*

After an exhaustive application of this rule, we can assume that $\widehat{H}$ does not contain an isolated vertex $A \in \mathcal{A}$.

Applying the Expansion Lemma. Next, we would like to apply Lemma 2.1 in the context of the bipartite graph $\widehat{H}$. Since $|\mathcal{A}| \geq \Delta - f(k) \geq (k+2) \cdot |B_v|$ and we have already ensured that $\widehat{H}$ does not contain any isolated vertex $A \in \mathcal{A}$, this lemma implies that we can find (in polynomial time) subsets $\mathcal{A}^* \subseteq \mathcal{A}$ and $B^* \subseteq B_v$ such that there exists a $(k+2)$-expansion from B^* into $\mathcal{A}^*$.

The usefulness of B^* is stated in the following lemma.

Lemma 7.12. *Any solution of size at most k to the input instance that does not contain v contains all of the vertices in B^*.*

Proof. Suppose that the input instance is a yes-instance, and suppose it has a solution S of size at most k that does not contain v. Consider some vertex $b \in B^*$. Let $A_1, A_2, \ldots, A_{k+2}$ be the neighbors of b in $\widehat{H}$ that correspond to our $(k+2)$-expansion. For any choice of A_i and A_j, Lemma 7.9 implies that if there is no chordless cycle defined by v, b and a path in A_i, then there is a chordless cycle defined by v, b, a path in A_i and a path in A_j. Therefore, if S contains neither v nor b, it must contain at least one vertex from each connected component A_i excluding at most one such component. However, there are $k+2$ such components, and since S does not contain v, we deduce that it contains b. The choice of $b \in B^*$ was arbitrary, and therefore we conclude that S contains all of the vertices in B^*. □

Decreasing the Independent Degree of v. Armed with Lemma 7.12, we can apply the following reduction rule.

Reduction Rule 7.4. *For each vertex $b \in B^*$, insert the edge (b, v) into $E(G)$ (if it is not already present), and mark (b, v) as a mandatory edge. Moreover, mark each edge $(v, z(A))$ such that $A \in \mathcal{A}^*$ as an irrelevant edge.*

Lemma 7.13. *Reduction Rule 7.4 is safe.*

Proof. First we claim that adding a mandatory edge between v and some $b \in B^*$ is safe. Let G' denote the graph obtained after applying this operation. By Lemma 7.12, at least one vertex among v and b must be present in any solution of size at most k. Therefore, we may mark the newly added edge (v, b) as a mandatory edge. Since this edge is a mandatory edge, any solution of size at most k to (G', k) will intersect any new chordless cycle created by adding this edge as well as any chordless cycle C that exists in G even if in $G'[V(C)]$ is not a chordless cycle (since if $G'[V(C)]$ is not a chordless cycle, $\{v, b\} \subseteq V(C)$). By adding the edges one-by-one, arguing that each operation is safe given that the preceding operation was safe, we derive the safeness of the insertion of all of the edges in $\{(b, v) \mid b \in B^*\}$ as well as the marking of these edges as mandatory.

Now, let G' denote the graph obtained after the application of Reduction Rule 7.4. Let us argue that it is indeed safe to ignore (in G') any chordless cycle containing an edge $(v, z(A))$ such that $A \in \mathcal{A}^*$. To this end, consider such a chordless cycle C and a solution S of size at most k to (G', k). If S contains v, it clearly intersects C. Otherwise, we have that S contains all of the vertices in B^*. Recall that by Lemma 7.7, $z(A)$ is the only neighbor of v in A, and by Lemma 7.8, $N_G(z(A)) \cap X = N_G(V(A)) \cap X$. Therefore,

since the cycle C is chordless, it must contain a vertex from B_v, but then, by Lemma 2.1, we further deduce that it must contain a vertex from B^*. We thus get that C is intersected by S. □

Reduction Rule 7.4 decreases $|N_G^R(v)|$. Moreover, as long as $d_G^R(v) > \Delta$, we can apply this rule. Thus, after an exhaustive application of this rule, it should hold that $d_G^R(v) \leq \Delta$. Furthermore, this rule neither inserts vertices into $V(G)$ nor unmarks edges, and therefore we conclude that Lemma 7.5 is correct. Denote $\Delta' = \Delta + k$. Thus, by Reduction Rule 7.2, the size of a maximum independent set in the neighborhood of each vertex in the graph G from which we remove irrelevant edges is bounded by Δ'.

7.4 The Clique Forest

Let F denote the clique forest associated with the chordal graphs $G - D'$. Towards bounding the number of leaves in F, we need the following lemma.

Lemma 7.14. *Let I be an independent set in the graph $G - D'$. Then, there are most $|D'| \cdot \Delta'$ relevant edges between vertices in D' and vertices in I.*

Proof. The claim follows from the observation that since $\Delta_G^I \leq \Delta$, $|E_M| \leq k^2$ and I is an independent set, for every vertex $v \in D'$, there are at most Δ' relevant edges between v and vertices in I. □

We will also need the following reduction rule.

Reduction Rule 7.5. *If there exists a vertex v in $G - D'$ such that the vertices in $N_G(v)$ which are connected to v via relevant edges form a clique, remove the vertex v from G.*

Lemma 7.15. *Reduction Rule 7.5 is safe.*

Proof. The special choice of the vertex v implies that every chordless cycle containing v must contain at least one irrelevant edge, and can therefore be ignored. We thus conclude that it is safe to remove the vertex v from G. □

The bound on the number of leaves will follow from a bound on the number of bags containing *private vertices*, which are defined as follows.

Definition 7.2. A vertex v in $G - D'$ is a *private vertex* if there exists only one bag in F that contains it.

Lemma 7.16. *The number of bags in F containing private vertices is bounded by $|D'| \cdot \Delta'$.*

Proof. Let ℓ denote the number of bags in F containing private vertices. By the definition of a clique forest, if we take exactly one private vertex from each bag in F, we obtain an independent set. Therefore, the graph G contain an independent set of size at least ℓ. Let I denote this independent set. Observe that since each vertex in I is a private vertex, its neighborhood in $G - D'$ forms a clique. Therefore, after an exhaustive application of Reduction Rule 7.5, each vertex in I must be connected by at least one relevant edge to a vertex in D'. By Lemma 7.14, we conclude that $\ell \leq |D'| \cdot \Delta'$. □

We are now ready to bound the number of leaves and nodes of degree at least 3 in the clique forest F.

Lemma 7.17. *Both the number of leaves in F and the number of nodes of degree at least 3 in F are bounded by $|D'| \cdot \Delta'$.*

Proof. Observe that since every leaf in F corresponds to a maximal clique in $G - D'$, it contains a private vertex. Thus, by Lemma 7.16, the number of leaves is bounded by $|D'| \cdot \Delta'$. Since in a forest, the number of nodes of degree at least 3 is bounded by the number of leaves, we conclude that the lemma is correct. □

Next, we turn to bound the size of a bag of F, to which end we prove the correctness of the following lemma.

Lemma 7.18. *In polynomial time we can produce an instance (G', k') equivalent to (G, k) such that $k' \leq k$ and the size of any maximal clique in G' is bounded by $\kappa = c \cdot (|\widetilde{D}|^3 \cdot k + |\widetilde{D}| \cdot \Delta' \cdot (k+2)^3)$. (Here c is some constant independent of the input.)*

In the following we proof Lemma 7.19, which immediately implies Lemma 7.18.

Lemma 7.19. *Let (G, k) be an instance of* CVD *whose independent degree is bounded Δ, and let D be a solution to this instance. Then in polynomial time we can produce an equivalent instance (G', k') such that $k' \leq k$, and the size of any maximal clique in G' is bounded by $c \cdot (|D|^3 \cdot k + |D| \cdot \Delta \cdot (k+2)^3)$. (Here c is some constant independent of the input.)*

The proof of the above lemma is an adaptation of the work of Marx [Mar10] with a few modifications. Specifically, the lemmata in [Mar10] construct a so called "necessary set" of vertices with the property that one of the vertices in this set must be part of any solution of size k. We modify these lemmas to ensure that a necessary set output by them always has at most two vertices. Observe that such a necessary set is either a vertex that must be part of any solution of size at most k, or a mandatory edge. We note that Jansen and Pilipczuk [JP17] also give similar result, inspired by the results of [Mar10]. Given a redundant approximate solution $\widehat{D}$, the size of a maximal clique in $G - \widehat{D}$ can be bounded by $\mathcal{O}(|\widehat{D}|^3 k)$. However the present method for computing a redundant approximate solution implies that $|\widehat{D}| \geq |D| \cdot k$, and hence Lemma 7.19 gives a better upper-bound.

Lemma 7.20 ([JP17])**.** *The size of each bag in the clique forest F is bounded by $c \cdot |D'|^3 k$, where c is some constant independent of the input.*

For the rest of this subsection, we fix a maximal clique K of $G - D$, which contains more than $c \cdot (|D|^3 \cdot k + |D| \cdot \Delta \cdot (k+2)^3)$ vertices. We will show that we can mark a bounded number of vertices of K so that the following holds. Let X be any set of at most k vertices such that $G - X$ has a chordless cycle H that contains a vertex u of K. If u is an unmarked vertex then there is another chordless cycle H' in $G - X$ that avoids u and contains strictly fewer unmarked vertex in K. This condition implies that we can safely ignore any chordless cycle that includes an unmarked vertex, which further implies that it is safe to delete these vertices from the graph. We shall closely follow the notations and proofs of [Mar10], but in light of the bound on (relevant) independent-degree of

vertices in D, we shall modify them appropriately. For a vertex $v \in V(G) \setminus D$, we say that the vertex has the *label* $t \in D$ if v is a neighbor of t. Note that the edge (t, v) could be relevant or irrelevant, and a vertex may have several labels depending on its set of neighbors in D.

7.4.1 Dangerous vertices and their witnesses in K

Let us begin with the notion of dangerous vertices for the clique K in the graph.

Definition 7.3. Let $v \in V(G) \setminus (D \cup K)$ be a relevant neighbor of $t \in D$ such that there is a path P from v to $u \in K$ whose internal vertices do not have the label t.

(i) The vertex v is a *t-dangerous vertex* for K if the vertex u does not have the label t.

(ii) The vertex v is a *t^*-dangerous vertex* for K if the vertex u is also a relevant neighbor of t and u, v are not neighbors in G.

The vertex $u \in K$ is called a *t-witness (t^*-witness)* of v, and the path P is called a *t-witness (t^*-witness) path* of v.

In [Mar10], it is shown that there are many vertices in the clique K whose deletion does not affect any of the dangerous vertices in the following sense. We mark a bounded number of vertices in K such that, for any subset X of vertices of size at most k, if there is chordless cycle in $G-X$ that passes through t, a dangerous vertex v and through some unmarked vertex u in K, then there is another chordless cycle in $G - X$ which contains a marked vertex u' and avoids u. This implies that we may ignore any chordless cycles in G that includes an unmarked vertex of K. Note that the definition above differs from [Mar10] in the requirement that v (and u) must be a relevant neighbor of t, since if the edge (t, v) (or the edge (t, u)) were irrelevant then any chordless cycles that contain both t and v (or t and u) can be safely ignored. We have the following bounds on the size of an independent set of dangerous vertices in G.

Lemma 7.21 (Lemma 11, [Mar10])**.** *If I is any collection of independent t-dangerous vertices, then either $|I| \leq 6k^2$ or we can find a new mandatory edge in the graph, or we can find a vertex that must be part of any solution of size at most k.*

The following lemma improves upon Lemma 12 of [Mar10] by using the bound on the independent degree of vertices in D.

Lemma 7.22 (Lemma 12, [Mar10])**.** *If I is any collection of independent t^*-dangerous vertices, then $|I| \leq \Delta$.*

The following lemma shows that if Q is a clique of t-dangerous vertices we require only $k + 1$ vertices as witnesses for all the vertices of Q.

Lemma 7.23 (Lemma 13 [Mar10])**.** *Let Q be a clique of t-dangerous vertices. Then we can mark $k + 1$ vertices in K such that for any set X of k vertices, if $v \in Q$ has an unmarked t-witness in $K - X$ then it has a marked t-witness in $K - X$.*

In the context of the following lemma, we remark that there is a similar lemma for cliques of t^*-dangerous vertices in [Mar10], but we give a version of it that is more suitable for our purposes.

Lemma 7.24 (Lemma 14, [Mar10])**.** *Let Q be a clique of t^* dangerous vertices. Then either we can find a vertex that must be part of any solution of size at most k, or we can find a new mandatory edge, or else we can mark $(k+2)^3$ vertices of K such that for any set X of k vertices, if $v \in Q$ has a unmarked t^* witness in $K - X$ then it also has a marked witness $K - X$.*

Proof. This proof is a minor modification of the proof of Lemma 14, [Mar10]. Let T be a clique tree of $G - D$. Following [Mar10], we say that a vertex v covers a bag x of T if v is contained in the bag x, and then T_v denotes the sub-clique tree of all the bags which are covered by v. Now, since Q and K are cliques in $G - D$, there are two bags x and y which contain Q and K, respectively. Consider the unique path connecting x and y in T, and suppose that the bags on this path are numbered as $x = 1, 2, \ldots s = y$. Let $u_1, u_2, \ldots$ be vertices of K with label t, and let a_i denote the smallest numbered bag on this path that occurs in T_{u_i}, i.e. the smallest numbered bag containing u_i. Similarly, let $v_1, v_2, \ldots$ be the vertices of Q and let b_i denote the largest numbered bag which is contained in T_{v_i}. It follows that T_{v_i} and T_{u_j} intersect if and only if $a_j \leq b_i$, in which case there is an edge (v_i, u_j) in the graph. We also assume that the collection of a_i and b_i are distinct, which is easily achieved by adding additional bags along this path in T. Further we assume that the vertices $u_1, u_2, \ldots$ and $v_1, v_2, \ldots$ are ordered so that the sequences of a_i and b_j are strictly increasing.

We define a subsequence of b_i and a_j as follows. Let $\beta_1 = 1$, and for ever $j \geq 1$ let α_j be the smallest value such that $a_{\alpha_j} > b_{\beta_j}$. For every $i \geq 2$, let β_i be the smallest value such that $b_{\beta_i} > a_{\alpha_{i-1}}$. Observe that we obtain a strictly increasing sequence, $b_{\beta_1} < a_{\alpha_1} < b_{\beta_2} < a_{\alpha_2} \ldots$. Let β_ℓ be the last element of the above sequence which corresponds to a vertex in Q.

Now, let u_s be a witness of a t^*-dangerous vertex v_{β_j}. We will show that a_{α_j} is also a witness for v_{β_j}. Clearly, $a_s > b_{\beta_j}$, which implies $a_s \geq a_{\alpha_j}$. Hence, the t^*-witness path from v_{β_j} to u_s passes through the bag a_{α_j} which contains u_{α_j} (see the proof of Lemma 14, [Mar10]) . Further, $a_{\alpha_j} > b_{\beta_j}$ implies that v_{β_j} and a_{α_j} are not neighbors in G. Therefore, a_{α_j} is also a witness for v_{β_j}.

Now, suppose that $\ell \leq (k+2)^2$. Then for each $i = 1, 2, \ldots, \ell$ we mark the $k+2$ vertices $u_{\alpha_i}, u_{\alpha_i+1}, \ldots, u_{\alpha_i+k+1}$ (if they exist), and we will show that this set of marked vertices contains a sufficient number of witnesses for every vertex in Q. Note that we have marked at most $(k+2)^3$ vertices of K. Consider any set X of k vertices such that a vertex $v_x \in Q$ has a witness path in $G - X$ to some unmarked $u_y \in K$. In other words there is a chordless cycle in $G - X$ which includes the vertex t, a t^*-dangerous vertex v_x and its t^*-witness $u_y \in K$. Since v_x and u_y are non-neighbors, we have that $b_x < a_y$, which implies that there is some j for which $b_x < a_{\alpha_j} \leq a_y$. If a_y is not marked then $y > \alpha_j + k + 1$, and hence $G - X$ contains some $b_x < a_{\alpha_j+r} < a_y$. Now the witness path from v_x to a_y must contain a neighbor of a_{α_j+r}, which implies that a_{α_j+r} is also a witness for v_x, in $G - X$.

Finally, suppose that $\ell > (k+2)^2$. We will show that either the vertex t must be part of any solution of size k, or we can find a new mandatory edge in the graph. Let P_i be a witness path from v_{β_i} to u_{α_i} for every $1 \leq i \leq \ell$. Consider the collection of chordless

cycles $H_{(k+2)\cdot j}$ for $1 \leq j \leq k+1$, where $H_{(k+2)\cdot j} = (t\, v_{\beta_{(k+2)\cdot j}}\, P_{(k+2)\cdot j}\, u_{\alpha_{(k+2)\cdot j}}\, t)$. If all these cycles are pairwise vertex disjoint except for the vertex t, then we have obtained a collection of $k+1$ chordless cycles with t as the only common vertex. Hence t must be part of any solution of size at most k.

Otherwise, let $H_{(k+2)\cdot j}$ and $H_{(k+2)\cdot j'}$ have a common vertex w, for some $j < j'$. Then observe that w can only be an internal vertex of the paths $P_{(k+2)\cdot j}$ and $P_{(k+2)\cdot j'}$, and therefore it is not a neighbor of t. Since T is a tree-decomposition, this implies that w occurs in every bag numbered between $a_{\alpha_{(k+2)\cdot j}}$ and $b_{\beta_{(k+2)\cdot j'}}$. Since $a_{\alpha_{(k+2)\cdot j}} < b_{\beta_{(k+2)\cdot j+1}} < a_{\alpha_{(k+2)\cdot j+1}} < \ldots\ldots < b_{\beta_{(k+2)\cdot j+k+1}} < a_{\alpha_{(k+2)\cdot j+k+1}} < b_{\beta_{(k+2)\cdot j'}}$, we have that w occurs in every one of these bags. Observe that we obtain a collection of $k+1$ chordless cycles of length 4, namely $(t\, v_{\beta_{(k+2)\cdot j+i}}\, w\, u_{\alpha_{(k+2)\cdot j+i}}\, t)$ for $i = 1, 2, \ldots k+1$, such that the vertices t and w are the only common vertices. Since t and w are non-adjacent in G, we obtain a new mandatory edge (t, w) in G. □

Since $G - D$ is a chordal graph which contains all the dangerous vertices for K, it follows that the graph induced by the set of all t-dangerous (t^*-dangerous) vertices forms a chordal graph as well. Since a chordal graph is perfect, it has a clique cover of size α, where α is the cardinality of a maximum independent set in the graph (see [Gol04]). This gives us the following lemma, using the bound on the size of an independent set of dangerous vertices.

Lemma 7.25 (Lemmas 15 & 16, [Mar10]). *(i) Either we can find a vertex that must be part of any solution of size at most k, or we can find a new mandatory edge, or we can mark $6k^2(k+1)$ vertices in K such that for any set X of k vertices, if a t-dangerous vertex v has a unmarked witness in $K - X$, then it also has a marked witness in $K - X$.*

(ii) Either we can find a vertex that must be part of any solution of size at most k, or we can find a new mandatory edge, or we can mark $\Delta \cdot (k+2)^3$ vertices in K such that for any set X of k vertices, if a t^*-dangerous vertex has a unmarked witness in $K - X$, then it also has a marked witness in $K - X$.

From the above lemma we conclude that, for any set X of at most k vertices, to test if X intersects all those chordless cycles that pass through t, K and some dangerous vertex v, it is sufficient to consider only the marked vertices of K as witnesses. However the above lemma considers only a single vertex $t \in D$, and we must mark additional witness vertices for each $t \in D$.

Lemma 7.26 (Lemmas 15 & 16, [Mar10]). *(i) Either we can find a vertex that must be part of any solution of size at most k, or we can find a new mandatory edge, or we can mark $|D| \cdot 6k^2(k+1)$ vertices in K such that for any set X of k vertices, if a t-dangerous vertex v has a unmarked witness in $K - X$, then it also has a marked witness in $K - X$.*

(ii) Either we can find a vertex that must be part of any solution of size at most k, or we can find a new mandatory edge, or we can mark $|D| \cdot \Delta \cdot (k+2)^3$ vertices in K such that for any set X of k vertices, if a t^*-dangerous vertex has a unmarked witness in $K - X$, then it also has a marked witness in $K - X$.

7.4.2 Fragments of chordless cycles intersecting K

Now we shall mark vertices for chordless cycles in G that intersect K. If H is a chordless cycle in G, then consider $F, P_1, P_2, \ldots, P_s$ where $F = H \cap D$ and $P_1, P_2 \ldots, P_s$ are the paths in $H - D$. It follows that each P_i has exactly two labels from F on its endpoints, and the internal vertices have no labels from F, and further these paths are pairwise independent (i.e. there is no edge between two vertices that are in two different paths). We call $F, P_1, \ldots, P_s$ the *fragments* of the chordless cycle H. Since K is a clique, at most one of these paths intersects K, which we assume to be the path P_1, and further it contains at most two vertices from K. We will show that we can mark a bounded number of vertices in K such that for any chordless cycle H that includes an unmarked vertex (that lies in P_1), there is another chordless cycle H' that avoids this unmarked vertex. Let us first consider the case when P_1 just a single vertex (in K). The following lemma is a close variant of Lemma 18, [Mar10] for this case.

Lemma 7.27 (Lemma 18, [Mar10])**.** *Let $F, P_1, \ldots, P_s$ be the fragments of H where P_1 is just a single vertex that lies in K. Then either we can find a new mandatory edge, or we can mark $|D|^3 \cdot (k+2)$ vertices in K such that the following holds. For any set X of k vertices such that $G - X$ has a chordless cycle which intersects K in an unmarked vertex, there is another chordless cycle in $G - X$ which does not use any unmarked vertices of K.*

Proof. Our proof of this lemma is obtained by modifying proof of Lemma 18, [Mar10]. And note that, as P_1 is a single vertex, F must have at least two vertices. For every $l_1, l_2, l_3 \in D$, mark $k+1$ vertices of K which have labels l_1, l_2 and not l_3. Hence, in total we have marked $|D|^3 \cdot (k+1)$ vertices of K. Marx [Mar10] shows that this is sufficient for the case when F has 3 or more vertices.

When F contains only two vertices l_1, l_2, we need to mark some additional vertices. Let us recall that P_1 is only a single vertex of K and has the labels l_1, l_2, and so it follows that l_1, l_2 are non-neighbors. Let x be the bag in the clique tree of $G \setminus D$, which corresponds to the maximal clique K, and further assume that x is the root of this tree-decomposition. For any chordless cycle H^i such that $H^i - D = P_1^i, P_2^i$ where P_1^i is a single vertex, let w_i be the bag closest to x in the tree decomposition that contains a vertex of P_2^i. Note that $w_i \neq x$, as vertices of P_1^i and P_2^i have no edges between them, and it follows that the vertices of P_2^i are contained in the sub-clique tree rooted at w_i, and furthermore none of them are present in any bag outside this sub-clique tree. We can generate a list of these bags $w_1, w_2, \ldots$, by considering every choice of P_1^i, l_1 and l_2 and selecting those bags of the tree decomposition that have a path from l_1 to l_2 with at least one internal vertex in the non-neighborhood of the choice of P_1^i. Among these bags, select $w_{i_1}, w_{i_2}, \ldots, w_{i_q}$ such that none of its descendants in the clique tree are in the collection $w_1, w_2, \ldots$ computed above. Note that, by definition no bag selected above is an ancestor or a descendant of another selected bag. Let $H_{i_1}, H_{i_2}, \ldots, H_{i_q}$ be a collection of chordless cycles such that $P_2^{i_j}$ is contained in the sub-clique tree of w_{i_j}, for $j = 1, 2 \ldots, q$. Note that the collection of $P_2^{i_j}$ are pairwise vertex disjoint, and further there are no edges between the vertices of $P_2^{i_j}$ and $P_2^{i_{j'}}$ for $j \neq j'$. This follows from the fact that no bag outside the sub-clique tree rooted at w_{i_j} contains any vertex of $P_2^{i_j}$, and that no bag selected in the collection of w_{i_j} is an ancestor or a descendant of another.

Consider the case when $q \leq k+1$. We define the distance of a vertex v from a bag w in the clique tree as the minimum of the distance between w and a bag x that contains v. Then for each w_{i_j}, sort the vertices of of K, which have labels l_1, l_2, according to the distance from w_{i_j}, and mark $k+1$ vertices from w_{i_j}. It follows that we mark $|D|^2 \cdot (k+1)^2$ vertices of K. Let us argue that these marked vertices satisfy the requirements. For any set X of at most k vertices, suppose H is a chordless cycle in $G - X$ with fragments F, P_1, P_2 where P_1 is just a single vertex $u \in K$ with labels $l_1, l_2 \in F$. Consider some bag w_i whose sub-clique tree contains the path P_2, and note that w_i, or some descendent w_{i_j} was selected in the above collection. Now if u was not marked for w_{i_j}, l_1, l_2, then any of the vertices that were marked for this tuple has a greater distance from the bags w_{i_j} (and w_i) than u. Since we marked $k+1$ vertices for this tuple, at least one of them is present in $K \setminus X$ and let u' be that vertex. It follows that replacing u with u' in H gives us a chordless cycle in $G - X$.

Now, consider the case when $q \geq k+2$. The collection of paths P^{i_j}, along with l_1 and l_2, defines a collection of chordless cycles such that any solution of size k must pick at least one of l_1 or l_2. Since l_1, l_2 are non-neighbors, we obtain a new mandatory edge (l_1, l_2).

Further note that the total number of marked vertices is $|D|^3 \cdot (k+1) + |D|^2 \cdot (k+1)^2$. Since $|D| \geq k$, it follows that the total number of marked vertices is upper bounded by $|D|^3 \cdot (k+2)$. □

Now, the only remaining case is when the path P_1 has two or more vertices. For this case, we have the following lemma, which follows from Lemma 19 of [Mar10] where we rely on Lemma 7.26.

Lemma 7.28 (Lemma 19, [Mar10])**.** *Let $F, P_1, \ldots, P_s$ be the fragments of a chordless cycle H where P_1 contains at least two vertices, and further at least one of those vertices are in K. Then we can mark at most $|D| \cdot 6k^2(k+1) + |D| \cdot \Delta \cdot (k+2)^3$ vertices in K such that, for any set X of at most k vertices, if $G - X$ contains a chordless cycle that includes an unmarked vertex $u \in K$, then $G - X$ also contains a chordless cycle that avoids u, and it has strictly fewer unmarked vertices of K.*

Combining all of the above lemmas we obtain the following.

Lemma 7.29. *Let (G, k) be an instance of* CVD *whose independent-degree is bounded by Δ. Let D be solution to this instance and let K be a maximal clique in $G - D$. Then in polynomial time, either we can find a vertex which must be part of any solution of size at most k, or we can find a new mandatory edge, or we can safely remove all but $c \cdot (|D|^3 \cdot k + |D| \cdot \Delta \cdot (k+2)^3)$ vertices of K. (Here c is some constant independent of the input.)*

Proof. We apply the Lemmas 7.26, 7.27 and 7.28 to the given instance. If any of them return a vertex that must be part of any solution of size at most k, or a new mandatory edge then we output that vertex or edge. Otherwise, together they mark a maximum of $c' \cdot (|D|^3 \cdot k + |D| \cdot \Delta \cdot (k+2)^3)$ vertices in K, where c' is some constant independent of the input. In addition, let us also mark any unmarked vertex in K which is an endpoint of a mandatory edge in the instance. Since the total number of mandatory edges is always bounded by k^2, the total number of marked vertices in K does not

exceed $c \cdot (|D|^3 \cdot k + |D| \cdot \Delta \cdot (k+2)^3)$, where c is again a constant independent of the input.

Let u be any unmarked vertex in K. We will show that the instances G and $G - \{u\}$ are equivalent. Let X be any set of at most k vertices such that $(G - \{u\}) - X$ is a chordal graph. If $G - X$ is not chordal, then there is some chordless cycle H in $G - X$. If H does not contain the vertex u, then it is also present in $(G - \{u\}) - X$ which is a contradiction. On the other hand if H contains u, then by the above lemmas, we can argue that there is some chordless cycle H' in $G - X$ which avoids u (see Lemma 19, [Mar10]). Therefore H' is present in $(G - \{u\}) - X$, which is again a contradiction. Now, by induction, we can show that it is safe to remove all the unmarked vertices in K from the graph. □

Now we are ready to prove Lemma 7.19. Consider any maximal clique in the graph $G - D$ with more than $c \cdot (|D|^3 \cdot k + |D| \cdot \Delta \cdot (k+2)^3)$ vertices, and apply Lemma 7.29 to it. If it returns a vertex v that must be part of any solution of size k, we remove it from the graph and decrease k by one. Else if it returns a mandatory edge, we add this edge to the graph and mark it as mandatory. We can argue, as before, that both these operations are safe. Otherwise Lemma 7.29 ensures that it is safe remove all but the $c \cdot (|D|^3 \cdot k + |D| \cdot \Delta \cdot (k+2)^3)$ marked vertices of K. Therefore, we remove these vertices from the graph. Observe that, each application of Lemma 7.29 either reduces k, or adds a new mandatory edge, or reduces the number of vertices in the graph. Further, we may add at most $k^2 + 1$ new mandatory edges before finding a new vertex that must be part of any solution of size k or concluding that the given instance is a No instance. Hence, in polynomial time, either we bound the size of every maximal clique in the graph or conclude that the given instance is a *no*-instance.

Observe that the size of each bag of F is bounded by the size of a maximal clique of $G - D'$. Furthermore, since $G - D'$ is a subgraph of G, the size of a maximal clique of $G - D'$ is bounded by the size of a maximal clique of G. Thus, having applied the procedure given by Lemma 7.18, we have the following result.

Lemma 7.30. *The size of any bag of F is upper bounded by κ.*

7.5 The Length of Degree-2 Paths

The Family of Paths $\mathcal{P}$. Let V_F denote the set of each node of degree at least 3 in the forest F as well as each node whose bag has at least one private vertex. Let $\mathcal{P}$ denote the set of paths whose endpoints belong to V_F and such that all of their internal nodes do not belong to V_F. Clearly, it holds that $|\mathcal{P}| \leq |V_F|$. By Lemmata 7.16 and 7.17, we have the following observation.

Observation 22. $|\mathcal{P}| \leq 2|D'| \cdot \Delta'$.

Thus, in light of Lemma 7.30, by bounding the maximum number of nodes on each path in $\mathcal{P}$, we can bound the total number of vertices in the graph. To this end, we fix some path $P \in \mathcal{P}$. Moreover, we orient the path from left to right, where the choice of the leftmost and rightmost nodes is arbitrary.

Partitioning the Path P. Next, we will partition P into more "manageable paths". To this end, we need the following definition.

Definition 7.4. We say that a subpath Q of P complies with a vertex $d \in D'$ if at least one of the following conditions holds.

1. For every two bags B and B' on Q, both $B \subseteq N_G(d)$ and $B' \subseteq N_G(d)$.
2. Let $B_1, B_2, \ldots, B_t$ denote the bags of Q ordered from left to right. Then, at least one of the following condition holds.
   (a) $B_1 \cap N_G(d) \subseteq B_2 \cap N_G(d) \subseteq \ldots \subseteq B_t \cap N_G(d)$.
   (b) $B_t \cap N_G(d) \subseteq B_{t-1} \cap N_G(d) \subseteq \ldots \subseteq B_1 \cap N_G(d)$.

   In particular, $N_G(d)$ is a subset of at least one of the two bags B_1 and B_t.

We would like to find a set $\mathcal{B}$ of at most $\mathcal{O}(|D'|)$ bags on the path P such that after their removal from P, the following lemma will be true.

Lemma 7.31. *Each subpath resulting from the removal of the bags in $\mathcal{B}$ from P complies with every vertex in D'.*

The rest of this subsubsection concerns the proof of this lemma. To prove it, it is sufficient to show that for each vertex $d \in D'$, we can find $\mathcal{O}(1)$ bags such that after their removal from P, each of the resulting subpaths complies with d. To this end, fix some vertex $d \in D'$.

First, we need the following lemma.

Lemma 7.32. *Let $u, v \in N_G(d) \setminus D'$ be non-adjacent vertices, B_u be a bag containing u such that no bag to its right (on P) contains u, and B_v be a bag containing v such that not bag to its left (on P) contains v. Then, d is adjacent to every vertex in every bag that lies strictly between B_u and B_v.*

Proof. Let z be a vertex in a bag that lies strictly between B_u and B_v. Since this bag is not an endpoint of the path P, z belongs to at least two bags that lie between B_u and B_v. Denote the path between B_u and B_v by $(B_u = B_1, B_2, B_3, \ldots, B_t = B_v)$. Let B_i be the leftmost bag containing z, and let B_j be the rightmost bag containing z. Suppose, by way of contradiction, that $z \notin N_G(d)$.

We claim that there exists a path, P_1, from u to z such that none of its internal vertices belongs to $B_{i+1} \cup B_{i+2} \ldots \cup B_v$. The proof by induction on the number i. When $i = 1$, we have that $B_i = B_u$. Then, since the vertices in B_u form a clique, the claim is correct. Now, suppose that the claim holds for $i - 1 \geq 1$, and let us prove it for i.

Since the vertices of the bag B_i form a maximal clique in the graph $G - D'$, and the bag B_i does not contain private vertices, we have that $B_i \subseteq B_{i-1} \cup B_{i+1}$ and $B_i \setminus B_{i+1} \subset B_{i-1}$ is non-empty. Consider a vertex $w \in B_i \setminus B_{i+1}$ and observe that z and w are adjacent in G (since both belong to B_i), and further the vertex w is not present in $B_{i+1} \cup B_{i+2} \ldots \cup B_v$. By induction, there is a path P from u to w whose internal vertices do not belong to $B_i \cup B_{i+1} \ldots \cup B_v$. Appending the edge (w, z) to P gives us the path P_1.

Similarly, there is a path P_2 from z to v such that none of its internal vertices belongs to $B_u \cup \ldots \cup B_{i-2} \cup B_{i-1}$. Since $i < j$, the paths P_1 and P_2 have no common vertex except z, and there is no edge between an internal vertex of P_1 and an internal vertex of P_2.

Let P'_1 be the subpath of P_1 from u' to z, where u' is the last vertex in P_1 adjacent to d. Similarly, let P'_2 be the subpath of P_2 from z to v', where v' is the first vertex in P_2 adjacent to d. We may assume that P'_1 and P'_2 do not contain chords, else we can replace P'_1 and P'_2 by a chordless subpath of P'_1 and chordless subpath of P'_2, respectively, which will still contain u', z and v'. The cycle (d, P'_1, z, P'_2, d) is a chordless cycle in G, contradicting the fact that $G - (D' \setminus \{d\})$ is a chordal graph. □

We also need the following notation. Let B_ℓ be the leftmost bag on P that contains a neighbor v_ℓ of d such that v_ℓ does not belong to any bag to the right of B_ℓ. Similarly, let B_r be the rightmost bag on P that contains a neighbor v_r of d such that v_r does not belong to any bag to the left of B_ℓ.

Lemma 7.33. *Let B and B' be two bags on P that do not lie on the right of B_ℓ and such that B lies on the left of B'. Then, it holds that $B \cap N_G(d) \subseteq B' \cap N_G(d)$.*

Proof. Suppose, by way of contradiction, that $B \cap N_G(d) \not\subseteq B' \cap N_G(d)$. However, this implies that B contains a neighbor v of d such that v does not belong to any bag to the right of B, which contradicts the choice of B_ℓ. □

Lemma 7.34. *Let B and B' be two bags on P that do not lie on the left of B_r and such that B lies to the right of B'. Then, it holds that $B \cap N_G(d) \subseteq B' \cap N_G(d)$.*

Proof. The proof of this lemma is symmetric to the proof of Lemma 7.33. □

By Lemmas 7.32–7.34, each of the subpaths resulting from the removal of B_ℓ and B_r from P complies with d. Thus, we conclude that Lemma 7.31 is correct.

Handling a Manageable Path. We now examine a subpath of P, denoted by Q, which complies with every vertex $d \in D'$. We will devise reduction rules such that after applying them exhaustively, the number of vertices in the union of the bags of the path Q will be bounded by $\mathcal{O}(\kappa)$.

Let $B_1, B_2, \ldots, B_t$ denote the bags of Q ordered from left to right. Moreover, denote $V(Q) = \bigcup_{i=1}^{t} B_i$ and $A = \bigcap_{i=1}^{t} B_i$. We partition D' into two sets D_a and D_p, where $D_a = \{d \in D' \mid V(Q) \subseteq N_G(d)\}$ and $D_p = D' \setminus D_a$. Here the letters a and p stand for "all" and "partial", respectively. Definition 7.4 directly implies that the following observation is correct.

Observation 23. *For every vertex $d \in D_p$, either* **(i)** $B_1 \cap N_G(d) \subseteq B_2 \cap N_G(d) \subseteq \ldots \subseteq B_t \cap N_G(d)$ *or* **(ii)** $B_t \cap N_G(d) \subseteq B_{t-1} \cap N_G(d) \subseteq \ldots \subseteq B_1 \cap N_G(d)$. *In particular, either* **(i)** $N_G(d) \cap V(Q) \subseteq B_1$ *or* **(ii)** $N_G(d) \cap V(Q) \subseteq B_t$.

We also denote $U = V(Q) \setminus (B_1 \cup B_t)$. It is sufficient to ensure that $|U| = \mathcal{O}(\kappa)$ since then, by Lemma 7.30, $|V(Q)| = \mathcal{O}(\kappa)$. Thus, we can next suppose that $|U| > \delta$ where $\delta = 2(k+1) + 6\kappa$.

For each pair of non-adjacent vertices in D_a, we apply Reduction Rule 7.1. The correctness of this operation is given by the following lemma.

Lemma 7.35. *Let u and v be two distinct non-adjacent vertices in D_a. Then, every solution of size at most k contains at least one of the vertices u and v.*

Proof. Since $|U| > \delta$ and the size of each bag is bounded by κ, standard arguments on weighted paths imply that there exists $i \in [t]$ such that $|\bigcup_{j=1}^{i} B_j| > \delta/2 - \kappa$ and $|\bigcup_{j=i+1}^{t} B_j| > \delta/2 - \kappa$; indeed, we may iteratively increase i from 1 to t until we reach the first time where it holds that $|\bigcup_{j=1}^{i} B_j| > \delta/2 - \kappa$, in which case it will also hold that $|\bigcup_{j=i+1}^{t} B_j| > \delta/2 - \kappa$. Denote $U^1 = U \cap ((\bigcup_{j=1}^{i-1} B_j) \setminus (B_1 \cup B_i))$ and $U^2 = U \cap ((\bigcup_{j=i+2}^{t} B_j) \setminus (B_{i+1} \cup B_t))$. Then, again since the size of each bag is bounded by κ, it holds that $|U^1|, |U^2| > \delta/2 - 3\kappa \geq k + 1$. Moreover, by the definition of a clique forest, $U^1 \cap U^2 = \emptyset$ and there is no vertex in U^1 that is adjacent to a vertex in U^2. Thus, for any pair of vertices $x \in U^1$ and $y \in U^2$, the subgraph induced by $\{u, v, x, y\}$ is a chordless cycle. However, any solution of size at most k can only contain at most k vertices from $U^1 \cup U^2$, and thus, to intersect all of these chordless cycles, it must contain at least one vertex among u and v. □

Thus, from now on we can assume that $G[D_a]$ is a clique. However, by the definition of A, for every vertex in A and every vertex in $V(Q)$, there exists a bag B_i, $i \in [t]$, which contains both of them, and therefore they are adjacent. We thus deduce that the following observation is correct.

Observation 24. *Any two distinct vertices $v \in D_a \cup A$ and $u \in D_a \cup V(Q)$ are adjacent.*

Let us now examine chordless cycles that contain vertices from U.

Lemma 7.36. *Let C be a chordless cycle in G that contains some vertex $u \in U$. Then, no vertex on $V(C)$ belongs to $D_a \cup A$, and both neighbors of u in C do not belong to $D' \cup A$.*

Proof. First, by Observation 23, we have that u does not have neighbors (in G) in D_p, and therefore both neighbors of u in C do not belong to D_p. Thus, it remains to show that no vertex in $V(C)$ belongs to $D_a \cup A$. By Observation 24, if at least one of vertex $v \in V(C)$ belongs to $D_a \cup A$, it is adjacent to u in G and it is either a neighbor of u in C adjacent in G to the other neighbor of u in C or it is adjacent in G to both neighbors of u in C. In any case, if at least one of vertex $v \in V(C)$ belonged to $D_a \cup A$, the cycle C would have contained a chord, contradicting the supposition that C is a chordless cycle. We thus conclude that the lemma is correct. □

Lemma 7.37. *Let C be a chordless cycle in G that contains some vertex $u \in U$. Then, C contains a path between a vertex in $B_1 \setminus A$ and a vertex in $B_t \setminus A$ whose internal vertices belong to U and one of them is u.*

Proof. Since D' is an approximate solution, the cycle C must contain at least one vertex that does not belong to U. Thus, by Lemma 7.36, we deduce that the cycle C contains two subpaths, each between u and a vertex in $(B_1 \cup B_t) \setminus A$, whose only common vertex is u and whose internal vertices belong to U. Moreover, one of these paths must contain an endpoint from $B_1 \setminus A$ and the other from $B_t \setminus A$, else the cycle C contains a chord corresponding to the edge between these two endpoints. Therefore, by concatenating the two paths, we obtain the desired subpath of C. □

We continue our examination of chordless cycles that contain vertices from U in the context of separators.

Lemma 7.38. *Let S be a minimal solution that contains at least one vertex from U. Then, there exists $i \in [t-1]$ such that* **(i)** $(B_i \cap B_{i+1}) \setminus A \subseteq S$, *and* **(ii)** $S \cap U \subseteq B_i \cap B_{i+1}$.

Proof. **Property (i).** Since S is a minimal solution, G contains a chordless cycle C with a vertex $u \in S \cap U$ and no other vertex from S. By Lemma 7.37, C contains a path between a vertex in $B_1 \setminus A$ and a vertex in $B_t \setminus A$ whose set of internal vertices includes u and is a subset of U. In particular, since each bag induces a clique while C is a chordless cycle, C contains a path P between a vertex $x \in B_1 \setminus A$ and a vertex in $y \in B_t \setminus A$ whose set of internal vertices is a non-empty subset of $V(Q) \setminus A$, and such that $(V(C) \setminus V(P)) \cap V(Q) = \emptyset$. Thus, x and y are not adjacent. Moreover, by Lemma 7.36, $V(C) \cap (D_a \cup A) = \emptyset$. Thus, since D' is a redundant approximate solution, C contains a vertex $d \in D_p$.

Suppose, by way of contradiction, that there does not exist $i \in [t-1]$ such that $(B_i \cap B_{i+1}) \setminus A \subseteq S$. Then, G has a path P' between x and y whose internal vertices belong to $V(Q) \setminus (S \cup A)$. Since $(V(C) \setminus V(P)) \cap V(Q) = \emptyset$, it holds that $(V(C) \setminus V(P)) \cap V(P') = \emptyset$, which implies that C contains a path between x and y whose set of internal vertices is disjoint from the one of $V(P')$. Observe that if a graph H contains a vertex a with two non-adjacent neighbors, b and c, such that $H \setminus \{a\}$ has a chordless path between b and c with at least one vertex that is not a neighbor of a, then H has a chordless cycle. On the one hand, by the definition of a clique forest, any vertex in $V(G) \setminus (V(Q) \cup D')$ cannot be adjacent to both a vertex in $B_1 \setminus A$ and a vertex in $B_t \setminus A$, and it is adjacent to no vertex in U. On the other hand, Observation 23 implies that any vertex in D_p also satisfies this property. Thus, $V(P') \cap U = \emptyset$, since any vertex in this set fits the above description of the vertex a where $H = G[(V(C) \setminus V(P)) \cup V(P')]$, which is a subgraph of the chordal graph $G \setminus S$. Without loss of generality, we have that P' contains a subpath $p - q - r$ where $p, q \in B_1 \setminus A$ and $r \in B_t \setminus A$. Let P'' denote a shortest path between p and r in $G[(V(C) \setminus V(P)) \cup (V(P') \setminus \{q\})]$, which contains at least two internal vertices (since it must contain a vertex from $V(C) \setminus V(P)$, else P' would have had a chord, and such a vertex cannot be adjacent to both p and r). Every vertex on P'' should be adjacent to q, else q fits the above description of the vertex a. Since P' is a chordless path and $(V(C) \setminus V(P)) \cap V(Q) = \emptyset$, P'' does not contain any vertex from B_1 that is not p, and the only vertex from B_t that is not r and which P'' can contain is the neighbor of r. Overall, this implies that P'' contains a vertex from $V(C) \setminus V(P)$ adjacent to both q and a vertex in $B_t \setminus A$, which is a contradiction.

Property (ii). It remains to show that $S \cap U \subseteq B_i \cap B_{i+1}$. To this end, we consider some arbitrary vertex $u \in S \cap U$ and show that it belongs to $B_i \cap B_{i+1}$. Since S is a minimal solution, G contains a chordless cycle C' such that $V(C') \cap S = \{u\}$. By Lemma 7.37, C' contains a subpath between a vertex in $B_1 \setminus A$ and a vertex in $B_t \setminus A$ whose internal vertices belong to U. Thus, by the definition of a clique forest, C' must contain a vertex from $(B_i \cap B_{i+1}) \setminus A$. However, we have already shown that $(B_i \cap B_{i+1}) \setminus A \subseteq S$, and therefore it must hold that $u \in B_i \cap B_{i+1}$ (since otherwise we reach a contradiction to the fact that $V(C') \cap S = \{u\}$). □

Let $\mathcal{W}$ be the family of each subset $W \subseteq (B_1 \cup B_t) \setminus A$ of size at most k for which there exists an index $i \in [t-1]$ such that $W = (B_i \cap B_{i+1}) \setminus (A \cup U)$ and $|(B_i \cap B_{i+1}) \cap U| \leq k$. We can easily bound the size of the family $\mathcal{W}$ as follows.

Lemma 7.39. $|\mathcal{W}| \leq 2k + 1$.

Proof. Let i be the smallest index in $[t-1]$ for which there exists $W^i \in \mathcal{W}$ such that $W^i = (B_i \cap B_{i+1}) \setminus (A \cup U)$, and let j be the largest or which there exists $W^j \in \mathcal{W}$ such that $W^j = (B_i \cap B_{i+1}) \setminus (A \cup U)$. Then, by the definition of a clique forest, for every set $W \in \mathcal{W}$ it holds that $W \subseteq W^i \cup W^j$. Furthermore, the sets in $\mathcal{W}$ can be sorted by $W_1, W_2, \ldots, W_{|\mathcal{W}|}$ such that for all $r \in [|\mathcal{W}|-1]$, $W_{r+1} \cap B_1 \subseteq W_r \cap B_1$ and $W_r \cap B_t \subset W_{r+1} \cap B_t$. We thus conclude that $|\mathcal{W}| \leq 2k+1$. □

We proceed by associating a separator with each set $W \in \mathcal{W}$ as follows. First, let $\mathcal{I}_W$ denote the set of all indices $i \in [t-1]$ such that $W = (B_i \cap B_{i+1}) \setminus (A \cup U)$. Now, let i_W denote an index in $\mathcal{I}_W$ that minimizes $|(B_i \cap B_{i+1}) \cap U|$ (if there are several choices, choose one arbitrarily). We further denote $M = \bigcup_{W \in \mathcal{W}} ((B_{i_W} \cap B_{i_W+1}) \cap U)$. Observe that by Lemmata 7.30 and 7.39, $|M| = \mathcal{O}(k^2)$. Thus, it is sufficient to argue that there exists a vertex in $U \setminus M$ that can be removed from G (since as long as $|U| > \delta$, we will be able to find such a vertex). To this end, we will need the following lemma.

Lemma 7.40. *Let $u \in U \setminus M$. If (G,k) is a yes-instance, then it has a solution S of size at most k that does not contain the vertex u.*

Proof. Suppose that (G,k) is a *yes*-instance, and let S be a solution of minimum size. Assume that $u \in S$, else we are done. By Lemma 7.38, there exists $i \in [t-1]$ such that $(B_i \cap B_{i+1}) \setminus A \subseteq S$ and $S \cap U \subseteq B_i \cap B_{i+1}$. Denote $W = (B_i \cap B_{i+1}) \setminus (A \cup U)$. Since $|S| \leq k$ and $W \subseteq S$, we have that $W \in \mathcal{W}$. Denote $R = (B_{i_W} \cap B_{i_W+1}) \cap U$ and $T = (B_i \cap B_{i+1}) \cap U$. Since $u \in U \setminus M$, it holds that $u \notin R$. Moreover, since $S \cap U \subseteq B_i \cap B_{i+1}$, it holds that $u \in T$. By the definition of i_W, we have that $|R| \leq |T|$. Thus, to show that the lemma is correct, it is sufficient to show that $S' = (S \setminus T) \cup R$ is a solution.

Suppose, by way of contradiction, that S' is not a solution. Then, since S is a solution of minimum size, there exist a chordless cycle C and a vertex $v \in T$ such that $v \in V(C)$ and $V(C) \cap S' = \emptyset$. Since $T \subseteq U$, by Lemma 7.37, C contains a path between a vertex in $(B_1 \cap B_2) \setminus A$ and a vertex in $(B_{t-1} \cap B_t) \setminus A$ whose internal vertices belong to U. In particular, by the definition of a clique forest, $V(C) \cap ((B_{i_W} \cap B_{i_W+1}) \setminus A) \neq \emptyset$. However, we have that $(B_{i_W} \cap B_{i_W+1}) \setminus A = R \cup W \subseteq S'$, which contradicts the fact that $V(C) \cap S' = \emptyset$. □

We are now ready to present our reduction rule.

Reduction Rule 7.6. *Let $u \in U \setminus M$. Remove the vertex u from the graph G and add an edge between any two non-adjacent vertices in $N_G(u)$.*

Lemma 7.41. *Reduction Rule 7.6 is safe.*

Proof. Let G' be the graph resulting from the application of this rule. For the forward direction, suppose that (G,k) is a yes-instance, and let S be a solution of minimum size. By Lemma 7.40, we can assume that $u \notin S$, and therefore $S \subseteq V(G')$. Thus, to show that (G',k) is a yes-instance, we need to prove that S intersects every chordless cycle in G'. Let C be a chordless cycle in G'. Suppose that this cycle does not exist in G, else it is clear that S intersects it. Then, C contains an edge between two vertices $v, w \in N_G(u)$ that are non-adjacent in G. Observe that since C is a chordless cycle and $G'[N_G(u)]$ is a clique, $V(C) \cap N_G(u) = \{v, w\}$. Thus, by replacing (v,w) by (v,u) and (u,w), we

obtain a chordless cycle in G. Since S intersects this cycle and $u \notin S$, it holds that S also intersects C.

For the backward direction, suppose that (G', k) is a yes-instance, and let S be a solution of minimum size. To show that (G, k) is a yes-instance, it is sufficient to show that S is also a solution to (G, k). Since $V(G') \subseteq V(G)$, it is clear that $S \subseteq V(G)$. We need to prove that S intersects every chordless cycle in G. Let C be a chordless cycle in G. Suppose that this cycle does not exist in G', else it is clear that S intersects it. Furthermore, suppose that $G'[V(C) \setminus \{u\}]$ does not contain a chordless cycle, else again it is clear that S intersects C. We get that C contains two vertices $v, w \in N_G(u)$ that are not adjacent in G. Since $u \in U$, Observation 23 and the definition of a clique forest imply that $N_G(u) \subseteq V(Q) \cup D_a$. By Observation 24, we deduce that $v, w \notin D_a \cup A$, and that C contains at most one vertex from $D_a \cup A$ (since any two vertices from $D_a \cup A$ are adjacent to each other and to both v and w). Since D' is a redundant approximate solution, C must contain a vertex $p \in D_p$. Since $G'[V(C) \setminus \{u\}]$ does not contain a chordless cycle, the neighbors of p on C belong to $N_G(u)$, and we can assume w.l.o.g that these neighbors are v and w. However, since v and w are not adjacent there cannot be a bag that contains both of them, which results in a contradiction to Observation 23. □

7.6 Unmarking Irrelevant and Mandatory Edges

Recall that our instance includes a set E_I of irrelevant edges and a set E_M of mandatory edges. It is clear that we can unmark each irrelevant edge (these edges were marked only for the sake of clarity of the analysis of our kernel). However, to unmark mandatory edge, we need the following operation.

Reduction Rule 7.7. *For every mandatory edge (x, y) introduce $k + 1$ pairs of new vertices, $\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_{k+1}, y_{k+1}\}$, and for each pair $\{x_i, y_i\}$ add the edges $(x, x_i), (x_i, y_i)$ and (y_i, y). Moreover, unmark the edge (x, y).*

Lemma 7.42. *Reduction Rule 7.7 is safe.*

Proof. Let $(G', k, E_I = \emptyset, E'_M)$ be the instance resulting from the application of this rule. Each edge (x, y) is contained (in G') in $k + 1$ cycles of size 4 which do not share vertices other than x and y. Therefore, and solution of size at most k to (G', k) must contain at least one of the vertices x and y. Thus, since any chordless cycle in G is also present in G' and we have only unmarked the edge (x, y), we conclude that if (G', k, E_I, E'_M) is a yes-instance, so is (G, k, E_I, E_M). On the other hand, any solution to (G, k, E_I, E_M) intersects all of the chordless cycles in G' since each of these chordless cycles is either present in G or contain both of the vertices x and y. Therefore, if (G, k, E_I, E_M) is a yes-instance, so is (G', k, E_I, E'_M). □

Recall that $|E_M| \leq k^2$. Hence, the total number of newly added vertices does not exceed $\mathcal{O}(k^3)$.

7.7 The Number of Vertices in the Kernel

In this section, we obtained an approximate solution $\widetilde{D}$ of size $f(k)$ and a redundant approximate solution D of size $\mathcal{O}(k \cdot f(k))$. Then, we examined the clique forest F associated with the chordal graph $G - D'$ where $|D'| = \mathcal{O}(|D| + k^2)$. To this end, we considered a set $\mathcal{P}$ of degree-2 paths that together cover all of the nodes of the forest, and showed that $|\mathcal{P}| = \mathcal{O}(|D'| \cdot \Delta')$. Recall that $\Delta' = \mathcal{O}(k \cdot f(k))$. We removed $\mathcal{O}(|D'|)$ bags, each of size $\kappa = \mathcal{O}(|\widetilde{D}|^3 \cdot k + |\widetilde{D}| \cdot \Delta' \cdot k^3)$, from each path $P \in \mathcal{P}$, and considered each of the resulting subpaths Q. We showed the number of vertices in the union of the bags of the path Q will be bounded by $\mathcal{O}(\kappa)$. Finally, we added $\mathcal{O}(k^3)$ new vertices to unmark mandatory edges. Thus, we conclude that the number of vertices in our kernel is bounded by $\mathcal{O}(|\mathcal{P}| \cdot |D'| \cdot \kappa) = \mathcal{O}(|D'|^2 \cdot \Delta' \cdot (|\widetilde{D}|^3 \cdot k + |\widetilde{D}| \cdot \Delta' \cdot k^3)) = \mathcal{O}(f(k)^3 k^3 \cdot (f(k)^3 k + f(k)^2 k^4)) = \mathcal{O}(f(k)^5 k^4 \cdot (f(k) + k^3))$. Recall that by Lemma 7.1, we can assume that $f(k) = \mathcal{O}(k^3 \log^2 k)$. Thus, at this point, we obtain a kernel of size $\mathcal{O}(k^{22} \log^{12} k)$.

7.8 A Better Kernelization Algorithm

Finally, we present a bootstrapping trick that will exploit the nature of our approximation algorithm to obtain a kernel of size $\mathcal{O}(k^{12} \log^{10} k)$. Recall that at this point, where we have already run our kernelization algorithm once, it holds that $n = \mathcal{O}(k^{22} \log^{12} k)$. Now, we again recall that CVD admits an $\mathcal{O}(\log^2 n)$-factor approximation algorithm (Chapter 6, Section 6.2, Theorem 6.2). Currently, it holds that $f(k) = \mathcal{O}(k \log^2 k)$ rather than $f(k) = \mathcal{O}(k^3 \log^2 k)$. Thus, if we rerun our kernelization procedure, obtaining a kernel of size $\mathcal{O}(f(k)^5 k^4 \cdot (f(k) + k^3))$ (see Section 7.7), it now holds that this size is upper bounded by $\mathcal{O}(k^{12} \log^{10} k)$. This concludes the proof of correctness of Theorem 7.1.

Chapter 8

Kernel for Interval Vertex Deletion

In this chapter, we look at the problem Interval Vertex Deletion, which is formally defined below.

Interval Vertex Deletion (IVD) **Parameter:** k
Input: A graph G and an integer k.
Question: Does there exist a subset $S \subseteq V(G)$ of size at most k such that $G - S$ is an interval graph?

A graph G is an *interval graph* if it is the intersection graph of intervals on the real line. Due to their intriguing combinatorial properties and many applications from diverse areas, such as industrial engineering and archeology [BBF+01, Ken69], the class of interval graphs is perhaps one of the most studied graph classes [BLS99, Gol04]. Whether IVD admits an FPT algorithm was a long standing open problem in the area until it was resolved by Cao and Marx [CM15a], who gave an algorithm with running time $\mathcal{O}(10^k n^9)$. Subsequently Cao [Cao16] designed an FPT algorithm with slightly better dependence on the parameter k, and linear dependence on the input size. Cao's algorithm has running time $\mathcal{O}(8^k(n+m))$. A natural follow up question to this work, explicitly asked multiple times in the literature [CKP13, Jan13, JP16], is whether IVD admits a polynomial kernel. In this chapter, we look at a polynomial kernel for IVD, and obtain the following theorem.

Theorem 8.1. Interval Vertex Deletion *admits a polynomial kernel.*

An overview of the kernelization algorithm. The first ingredient of the kernelization algorithm is the factor 8 polynomial time approximation algorithm for IVD by Cao [Cao16]. We use this algorithm to obtain an approximate solution of size at most $8k$, or conclude that no solution of size at most k exists. Re-running the approximation algorithm on the graph with some of the vertices marked as "un-deletable" we then grow this solution to a 10-*redundant* solution M of size $\mathcal{O}(k^{11})$. Here, 10-redundant means that for every subset $W \subseteq M$ of size at most 10, either $M \setminus W$ is a solution or every solution S' of size at most $k+2$ has non-empty intersection with W.

The kernelization heavily uses the characterization of interval graphs in terms of their forbidden induced subgraphs, also called obstructions. A graph H is an obstruction to the class of interval graphs if H is not an interval graph, and for every vertex $v \in V(H)$ we have that $H - \{v\}$ is an interval graph. A graph G is an interval graph if and only if

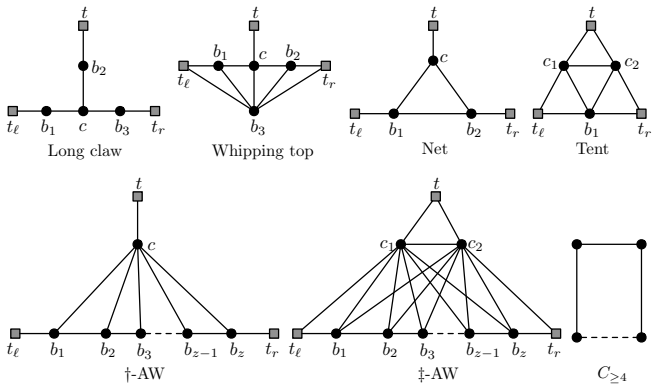

Figure 8.1: The set of obstructions for an interval graph.

it does not contain any obstruction as an induced subgraph. The set of obstructions to interval graphs have been completely characterized by Lekkerkerker and Boland, [LB62]. The set of obstructions consists of the *long claw*, the *whipping top*, the *net*, the *tent*, as well as three infinite families of graphs: the *single-dagger asteroidal witness* (†-AW), the *double-dagger asteroidal witnesses* (‡-AW), and the cycles of length at least 4 (see Figure 8.1).

Having a 10-redundant solution yields the following advantage. In several places we remove a carefully chosen vertex $v \notin M$ from G and claim that $G - \{v\}$ has a solution of size at most k if and only if G does. One direction of the equivalence is trivial. The interesting direction is to show that a solution X of size k to $G - \{v\}$ implies the existence of a solution of size at most k for G. The starting point for such an analysis is to ask why X is not already a solution for G. The only possible reason is that $G - X$ contains an obstruction $\mathbb{O}$, and $\mathbb{O}$ must contain v. We claim that $\mathbb{O}$ contains at least 11 vertices from M. Suppose not, then let W be the intersection of M and $\mathbb{O}$. We know that $(G - (M \setminus W))$ contains $\mathbb{O}$, and therefore it is not an interval graph. Hence, by 10-redundancy of M, X intersects $\mathbb{O}$ which contradicts the choice of $\mathbb{O}$. Thus, in this analysis we only need to care about the *large* obstructions, and furthermore only large obstructions whose intersection with M is large. This is crucial throughout the design and analysis of the kernel.

We then proceed to classify the connected components of $G - M$ based on whether they are *modules* in G (a module is a set X such that all vertices in X have the same neighbors outside X) or not. For each component C that is not a module there is an edge (u, v) in C and a vertex w in M such that w us adjacent to u and not v. If there are at least $2|M| + 1$ non-module components then some vertex w will be the starting point of at least 3 different paths w, u, v on this form into three different compoenents. But this is a long claw whose intersection with M is at most 1 vertex w. From 10-redundancy of M it follows that w must be a part of every solution of size at most $k + 5$, so we might just as well remove w and decrease the budget k by 1. Hence, the number of non-module components is at most $2|M| + 1$, which is polynomial in k.

Since none of the obstructions have any modules on more than one vertex, and the components of $G - M$ are interval graphs, we have that every obstruction will intersect

every module component in at most one vertex. From this we can deduce that every module component can be made into a clique by adding all possible edges between the vertices in that component. Further, there is no point in keeping more than $k+1$ copies of any vertex, so we can reduce the module components to cliques of size $k+1$.

We are left with the following picture: we have a 10-reduntant solution M of size $\mathcal{O}(k^{11})$. At most $\mathcal{O}(|M|)$ components of $G-M$ are not modules. These components could be arbitrarily large. The remaining components are all modules and cliques of size at most $k+1$. Thus the module components are structured and small, but there could be arbitrarily many of them. We are left with two tasks: reduce the *number* of module components, and reduce the *size* of the non-module components. The two tasks can be approached separately, and both turn out to be non-trivial. Since both tasks are quite technically involved we only give a few highlights in this overview.

Bounding the Number of Module Components. Consider first the case when there are no non-module components at all, and every module component is a single vertex. In this case $G-M$ is edgeless, so M is a *vertex cover* of G. The kernelization complexity of even this very special case was asked as an open problem by Fomin et al. [FJP14].

A key ingredient in the solution to this special case is a new bound for the setting considered in the classic two families theorem of Bollobás [Bol65]. Suppose there are two families $A_1, \ldots, A_m$ and $B_1, \ldots, B_m$ over a universe U, such that every set A_i has size p, every set B_j has size q, for every i the sets A_i and B_i are disjoint, while for every $i \neq j$ the sets A_i and B_j intersect. The two families theorem gives an upper bound of $\binom{p+q}{p}$ for the size m of the family. The upper bound on m is *independent of the universe size*, and this has been extensively used in the design of parameterized algorithms [FLPS16, Mar09]. Further, when p or q is a *constant* the bound is *polynomial* in $p+q$, and this has been extensively used in kernelization [KW12]. In our setting neither of the sets $A_1, \ldots, A_m$ nor the sets $B_1, \ldots, B_m$ have constant cardinality. On the other hand we know that for every $i \neq j$, $|A_i \cap B_j| \in \{1, 2\}$. We prove that in this case, m is bounded by $\mathcal{O}(|U|^2)$.

In the setting of kernelizing IVD parameterized by the size of a vertex cover M, the size of the kernel is intimately linked to m for the case when $A_1, \ldots, A_m$ is a collection of cliques in $G[M]$ while $B_1, \ldots, B_m$ is a collection of induced paths. Since a clique can only intersect an induced path in at most 2 vertices we can apply the above stated result to obtain an $\mathcal{O}(|M|^2)$ bound for m and (after a significant amount of additional effort, which we skip in this overview) a polynomial bound on the kernel size.

The kernel for IVD parameterized by vertex cover quite simply translates into a procedure that bounds the number, and therefore the total size, of module components of $G-M$. We remark that, because the *number* of non-module components is bounded by $\mathcal{O}(|M|)$, by bounding the number of module components we also bound the total number of components of $G-M$.

Bounding the Size of non-Module Components. Suppose now that the number of module components has been bounded by $k^{\mathcal{O}(1)}$. We can now include all of the module components in M, and proceed under the assumption that there are no module components at all.

The size-reduction for non-Module components proceeds in three phases. In the first

phase we bound the maximum clique size in the component. Our clique-reduction procedure builds on the clique-reduction procedure of Marx [Mar10] used in the kernelizations for Chordal Vertex Deletion (Chapter 7 and [JP17]). Both the procedure of Marx and one in this chapter are based on an "irrelevant vertex rule". However, the procedure here is much more involved, because the irrelevant vertex rule needs to preserve not only long induced cycles, but also large †-AWs and ‡-AWs.

Having reduced the maximum clique size in the component we proceed to the second phase, where we reduce the set of vertices which appear in at least two maximal cliques in the component. In this phase we partition the component into $k^{O(1)}$ "long" and "thin" parts, and prove that an optimal solution will either not touch the part at all, or it will cut it in two pieces using a minimum size separator. Then, provided that a part is sufficiently large we identify an edge such that contracting this edge will not decrease the size of a minimal separator inside this part. Contracting an edge can not *increase* the size of an optimal solution, and since we did not decrease the size of the minimal separators inside the part, contracting this edge does not decrease the size of an optimal solution either.

After the second phase the number of vertices appearing in at least two maximal cliques of the component is upper bounded by $k^{\mathcal{O}(1)}$. In the third phase we bound the number of remaining vertices – the ones that are private to some maximal clique of the component. At this point we can take the set of vertices appearing in at least two components and add them to M. This makes M grow by $k^{\mathcal{O}(1)}$ vertices, but now the large component breaks up into components whose size is no larger than that of a maximal clique, that is $k^{\mathcal{O}(1)}$. We can now re-apply the procedure for bounding the number of components and this bounds the total number of vertices in G by $k^{\mathcal{O}(1)}$. *We remark that, for technical reasons, in the actual proof phase* 2 *and* 3 *as described here are interleaved.*

Some notations. An equivalent definition of interval graphs given by Lekkerkerker and Boland [LB62], which is used in this chapter is as follows. An *interval graph* is a graph G that does not contain any of the following graphs, called *obstructions*, as an induced subgraph (see Figure 8.1).

- **Long Claw.** A graph $\mathbb{O}$ such that $V(\mathbb{O}) = \{t_\ell, t_r, t, c, b_1, b_2, b_3\}$ and $E(\mathbb{O}) = \{(t_\ell, b_1), (t_r, b_3), (t, b_2), (c, b_1), (c, b_2), (c, b_3)\}$.
- **Whipping Top.** A graph $\mathbb{O}$ such that $V(\mathbb{O}) = \{t_\ell, t_r, t, c, b_1, b_2, b_3\}$ and $E(\mathbb{O}) = \{(t_\ell, b_1), (t_r, b_2), (c, t), (c, b_1), (c, b_2), (b_3, t_\ell), (b_3, b_1), (b_3, c), (b_3, b_2), (b_3, t_r)\}$.
- **†-AW.** A graph $\mathbb{O}$ such that $V(\mathbb{O}) = \{t_\ell, t_r, t, c\} \cup \{b_1, b_2, \ldots, b_z\}$, where $t_\ell = b_0$ and $t_r = b_{z+1}$, $E(\mathbb{O}) = \{(t, c), (t_\ell, b_1), (t_r, b_z)\} \cup \{(c, b_i) \mid i \in [z]\} \cup \{(b_i, b_{i+1}) \mid i \in [z-1]\}$, and $z \geq 2$. A †-AW where $z = 2$ will be called a *net*.
- **‡-AW.** A graph $\mathbb{O}$ such that $V(\mathbb{O}) = \{t_\ell, t_r, t, c_1, c_2\} \cup \{b_1, b_2, \ldots, b_z\}$, where $t_\ell = b_0$ and $t_r = b_{z+1}$, $E(\mathbb{O}) = \{(t, c_1), (t, c_2), (c_1, c_2), (t_\ell, b_1), (t_r, b_z), (t_\ell, c_1), (t_r, c_2)\} \cup \{(c, b_i) \mid i \in [z]\} \cup \{(b_i, b_{i+1}) \mid i \in [z-1]\}$, and $z \geq 1$. A ‡-AW where $z = 1$ will be called a *tent*.
- **Hole.** A chordless cycles on at least four vertices.

An obstruction $\mathbb{O}$ is *minimal* if there does not exist an obstruction $\mathbb{O}'$ such that $V(\mathbb{O}') \subset V(\mathbb{O})$. In each of the first four obstructions, the vertices t_ℓ, t_r, and t are called *terminals*, the verices c, c_1, and c_2 are called *centers*, and the other vertices are called *base vertices*. Furthermore, the vertex t is called the *shallow terminal*. In the cases of AWs, the induced path on the set of base vertices is called the *base* of the AW and is denoted by $\mathsf{base}(\mathbb{O})$. Moreover, we say that the induced path on the set of base vertices, t_ℓ and t_r is the *extended base* of the AW, which is denoted by $P(\mathbb{O})$.

8.1 Computing a Redundant Solution

Let (G, k) be an instance of IVD. A subset $S \subseteq V(G)$ such that $G - S$ is an interval graph is called a *solution*, and a solution of size at most t is also called a *t-solution*. Given a family $\mathcal{W} \subseteq 2^{V(G)}$, we say that a subset $S \subseteq V(G)$ *hits* $\mathcal{W}$ if for all $W \in \mathcal{W}$, we have $S \cap W \neq \emptyset$. A family $\mathcal{W} \subseteq 2^{V(G)}$ is *t-necessary* if every solution of size at most t hits $\mathcal{W}$. We say that an obstruction $\mathbb{O}$ is *covered by* $\mathcal{W}$ if there exists $W \in \mathcal{W}$ such that $W \subseteq V(\mathbb{O})$.

Definition 8.1. Given a family $\mathcal{W} \subseteq 2^{V(G)}$ and $t \in \mathbb{N}$, a subset $M \subseteq V(G)$ is *t-redundant with respect to $\mathcal{W}$* if for every obstruction $\mathbb{O}$ that is not covered by $\mathcal{W}$, it holds that $|M \cap V(\mathbb{O})| > t$.

Proposition 8.1 ([Cao16])**.** IVD *admits a polynomial-time 6-approximation algorithm, called* `ApproxIVD`*.*

The purpose of this section is to prove the following lemma. We remark that in this statement we use the letter ℓ rather than k to avoid confusion, as we will use this result with $\ell = k + 2$.

Lemma 8.1. *Let $r \in \mathbb{N}$ be a fixed constant, and (G, ℓ) be an instance of* IVD*. In polynomial time, it is possible to either conclude that (G, ℓ) is a no-instance, or compute an ℓ-necessary family $\mathcal{W} \subseteq 2^{V(G)}$ along with an $(r+1)(6\ell)^{r+1}$-solution M that is r-redundant with respect to $\mathcal{W}$.*

Towards the proof of Lemma 8.1, let us first give a simple definition of a graph using which we will determine whether a set of vertices should be added to $\mathcal{W}$.

Definition 8.2. Let G be a graph, $U \subseteq V(G)$ and $t \in \mathbb{N}$. Then, $\mathsf{copy}(G, U, t)$ is defined as the graph G' on the vertex set $V(G) \cup \{v^i \mid v \in U, i \in [t]\}$ and the edge set $E(G) \cup \{(u^i, v) \mid (u, v) \in E(G), u \in U, i \in [t]\} \cup \{(u^i, v^j) \mid (u, v) \in E(G), u, v \in U, i, j \in [t]\} \cup \{(v, v^i) \mid v \in U, i \in [t]\} \cup \{(v^i, v^j) \mid v \in U, i, j \in [t], i \neq j\}$.

Informally, $\mathsf{copy}(G, U, t)$ is the graph G where for every vertex $u \in U$, we add t twins that (together with u) form a clique. Before we describe our algorithm to compute a redundant solution, we prove three simple claims.

Lemma 8.2. *Let G be a graph, $U \subseteq V(G)$ and $t \in \mathbb{N}$. If G is an interval graph, then $G' = \mathsf{copy}(G, U, t)$ is an interval graph as well.*

Proof. Suppose that G is an interval graph. Then, by Proposition 2.3, G admits a clique path (P, β). Now, we define (P', β') as follows: $P' = P$, and for all $x \in V(P')$, $\beta'(x) = \beta(x) \cup \{v^i \mid v \in \beta(x) \cap U, i \in [t]\}$. We claim that (P', β') is a clique path for G'. By using the fact that (P, β) is a path decomposition of G, we directly have the following properties. First, it is clear that $\bigcup_{x \in V(P')} \beta'(x) = V(G')$. Second, for any edge $e = (u, v) \in E(G')$ such that $u, v \in V(G)$, there exists $x_e \in V(P')$ such that $u, v \in \beta'(x_e)$. Then, since for all $v \in U$ and $i \in [t]$, it holds that $\beta'^{-1}(v) = \beta'^{-1}(v^i)$, we derive that for any edge $(u', v') \in E(G')$ there is a node $x \in V(P')$ such that $u', v' \in \beta'(x)$. Third, for any $v \in V(G)$, the collection of nodes $P'_v = \{x \in V(P') \mid v \in \beta'(u)\}$ is a subpath of P', and since for any $v \in U$ and $i \in [t]$, it holds that $\beta'^{-1}(v) = \beta'^{-1}(v^i)$, we derive that for any $v' \in V(G')$, the collection of nodes $P'_{v'} = \{x \in V(P') \mid v' \in \beta'(x)\}$ is a subpath of P'. Now, note that for all $x \in V(P')$, $\beta(x)$ is a clique, and for all $\{u, v\} \subseteq \beta(x)$ (possibly $u = x$) and $i, j \in [t]$, u^i is adjacent to u, u^j (if $i \neq j$), v and v^j, which implies that $\beta'(x)$ is also a clique. Hence, (P', β') is indeed clique path for G'. By Proposition 2.3, we derive that G' is an interval graph. □

Lemma 8.3. *Let G be a graph, $U \subseteq V(G)$ and $\ell \in \mathbb{N}$. If the algorithm* `ApproxIVD` *returns a set A of size larger than 6ℓ when called with $G' = \mathsf{copy}(G, U, 6\ell + 1)$ as input, then $\{U\}$ is ℓ-necessary.*

Proof. Suppose that `ApproxIVD` returns a set A of size larger than 6ℓ when called with G' as input. Then, (G', ℓ) is a *no*-instance. Suppose, by way of contradiction, that $\{U\}$ is not ℓ-necessary. Then, G has an ℓ-solution S such that $S \cap U = \emptyset$. In particular, $\widehat{G} = G - S$ is an interval graph such that $U \subseteq V(\widehat{G})$. However, this means that $\mathsf{copy}(\widehat{G}, U, 6\ell+1) = G' - S$, which by Lemma 8.2 implies that $G' - S$ is an interval graph. Thus, S is an ℓ-solution for G', which is a contradiction (as (G', ℓ) is a *no*-instance). □

Lemma 8.4. *Let G be a graph, $U \subseteq V(G)$ and $\ell \in \mathbb{N}$. If the algorithm* `ApproxIVD` *returns a set A of size at most 6ℓ when called with $G' = \mathsf{copy}(G, U, 6\ell + 1)$ as input, then for every obstruction $\mathbb{O}$ of G, $|V(\mathbb{O}) \cap U| + 1 \leq |V(\mathbb{O}) \cap (U \cup (A \cap V(G)))|$.*

Proof. Suppose that `ApproxIVD` returned a set A of size at most 6ℓ when called with G' as input. Let $\mathbb{O}$ be some obstruction of G, and denote $B = V(\mathbb{O}) \cap U$. Since $|A| \leq 6\ell$, for every vertex $v \in B$, we have that $v \in V(G') \setminus A$ or there exists $i(v) = i \in [6\ell]$ such that $v^i \in V(G') \setminus A$. Moreover, we have that the graph obtained from $\mathbb{O}$ by replacing each vertex $v \in B \cap A$ by $v^{i(v)}$ is an obstruction (as v and $v^{i(v)}$ are twins). Thus, as A is a solution for G', there exists $v \in V(G) \setminus B$ such that $v \in A \cap V(\mathbb{O})$. Hence, we have that $|V(\mathbb{O}) \cap U| + 1 \leq |V(\mathbb{O}) \cap (U \cup (A \cap V(G)))|$. □

Now, let us describe our algorithm, `RedundantIVD`, to compute a redundant solution. First, `RedundantIVD` initializes M_0 to be the output obtained by calling the algorithm `ApproxIVD` with G as input, $\mathcal{W}_0 := \emptyset$ and $\mathcal{T}_0 := \{(v) \mid v \in M_0\}$. If $|M_0| > 6\ell$, then `RedundantIVD` concludes that (G, ℓ) is a *no*-instance. Otherwise, for $i = 1, 2, \ldots, r$ (in this order), the algorithm executes the following steps:

1. Initialize $M_i := M_{i-1}$, $\mathcal{W}_i := \mathcal{W}_{i-1}$ and $\mathcal{T}_i := \emptyset$.

2. For every tuple $(v_0, v_1, \ldots, v_{i-1}) \in \mathcal{T}_{i-1}$:

(a) Let A be the output obtained by calling the algorithm `ApproxIVD` with $\mathsf{copy}(G, \{v_0, v_1, \ldots, v_{i-1}\}, 6\ell + 1)$ as input.

(b) If $|A| > 6\ell$, then insert $\{v_0, v_1, \ldots, v_{i-1}\}$ into $\mathcal{W}_i$.

(c) Otherwise, insert every vertex in $(A \cap V(G)) \setminus \{v_0, v_1, \ldots, v_{i-1}\}$ into M_i, and for all $u \in (A \cap V(G)) \setminus \{v_0, v_1, \ldots, v_{i-1}\}$, insert $(v_0, v_1, \ldots, v_{i-1}, u)$ into $\mathcal{T}_i$.

Eventually, the algorithm outputs the pair $(M_r, \mathcal{W}_r)$.

The properties of the algorithm `RedundantIVD` that are relevant to us are summarized in the two following claims, which are proved by induction and Lemmata 8.2, 8.3 and 8.4.

Lemma 8.5. *Consider a call to* `RedundantIVD` *with (G, ℓ, r) as input that did not conclude that (G, ℓ) is a* no*-instance. For all $i \in [r]_0$, the following conditions hold:*

1. *For any set $W \in \mathcal{W}_i$, every solution S of size at most ℓ satisfies $W \cap S \neq \emptyset$.*
2. *For any obstruction $\mathbb{O}$ of G that is not covered by $\mathcal{W}_i$, there exists $(v_0, v_1, \ldots, v_i) \in \mathcal{T}_i$ such that $\{v_0, v_1, \ldots, v_i\} \subseteq V(\mathbb{O})$.*

Proof. The proof is by induction on i. In the base case, where $i = 0$, Condition 1 trivially holds as $\mathcal{W}_0 = \emptyset$, and Condition 2 holds as M_0 is a solution and $\mathcal{T}_0$ simply contains a 1-vertex tuple for every vertex in M_0. Now, suppose that the claim is true for $i - 1 \geq 0$, and let us prove it for i.

To prove Condition 1, consider some set $W \in \mathcal{W}_i$. If $W \in \mathcal{W}_{i-1}$, then by the inductive hypothesis, every solution of size at most ℓ satisfies $W \cap S \neq \emptyset$. Thus, we next suppose that $W \in \mathcal{W}_i \setminus \mathcal{W}_{i-1}$. Then, there exists a tuple $(v_0, v_1, \ldots, v_{i-1}) \in \mathcal{T}_{i-1}$ in whose iteration `RedundantIVD` inserted $W = \{v_0, v_1, \ldots, v_{i-1}\}$ into $\mathcal{W}_i$. In that iteration, `ApproxIVD` was called with $\mathsf{copy}(G, W, 6\ell+1)$ as input, and returned a set A of size larger than 6ℓ. Thus, by Lemma 8.3, every solution S of size at most ℓ satisfies $W \cap S \neq \emptyset$.

To prove Condition 2, consider some obstruction $\mathbb{O}$ of G that is not covered by $\mathcal{W}_i$. By the inductive hypothesis and since $\mathcal{W}_{i-1} \subseteq \mathcal{W}_i$, there exists a tuple $(v_0, v_1, \ldots, v_{i-1}) \in \mathcal{T}_{i-1}$ such that $\{v_0, v_1, \ldots, v_{i-1}\} \subseteq V(\mathbb{O})$. Consider the iteration of `RedundantIVD` corresponding to this tuple, and denote $U = \{v_0, v_1, \ldots, v_{i-1}\}$. In that iteration, `ApproxIVD` was called with $\mathsf{copy}(G, U, 6\ell + 1)$ as input, and returned a set A of size at most 6ℓ. By Lemma 8.4, $|V(\mathbb{O}) \cap U| + 1 \leq |V(\mathbb{O}) \cap (U \cup (A \cap V(G)))|$. Thus, there exists $v_i \in (A \cap V(G)) \setminus U$ such that $U \cup \{v_i\} \subseteq V(\mathbb{O})$. However, by the specification of `ApproxIVD`, this means that there exists $(v_0, v_1, \ldots, v_i) \in \mathcal{T}_i$ such that $\{v_0, v_1, \ldots, v_i\} \subseteq V(\mathbb{O})$. □

Observation 25. *Consider a call to* `RedundantIVD` *with (G, ℓ, r) as input that did not conclude that (G, ℓ) is a* no*-instance. For all $i \in [r]_0$, $|M_i| \leq \sum_{j=0}^{i} (6\ell)^{j+1}$, $|\mathcal{T}_i| \leq (6\ell)^{i+1}$ and every tuple in $\mathcal{T}_i$ consists of distinct vertices.*

Proof. The proof is by induction on i. In the base case, where $i = 0$, the correctness follows as `ApproxIVD` returned a set of size at most 6ℓ. Now, suppose that the claim is true for $i - 1 \geq 0$, and let us prove it for i. By the specification of the algorithm and inductive hypothesis, we have that $|M_i| \leq |M_{i-1}| + 6\ell|\mathcal{T}_{i-1}| \leq \sum_{j=1}^{i+1} (6\ell)^j$ and $|\mathcal{T}_i| \leq 6\ell|\mathcal{T}_{i-1}| \leq (6\ell)^{i+1}$. Moreover, by the inductive hypothesis, for every tuple in $\mathcal{T}_i$, the first i vertices are distinct, and by the specification of `ApproxIVD`, the last vertex is not equal to any of them. □

By the specification of `RedundantIVD`, as a corollary to Lemma 8.5 and Observation 25, we directly obtain the following result.

Corollary 8.1. *Consider a call to* `RedundantIVD` *with* (G, ℓ, r) *as input that did not conclude that* (G, ℓ) *is a* no*-instance. For all* $i \in [r]_0$, $\mathcal{W}_i$ *is an* ℓ*-necessary and* M_i *is a* $\sum_{j=0}^{i}(6\ell)^{j+1}$*-solution that is* i*-redundant with respect to* $\mathcal{W}_i$.

Clearly, `RedundantIVD` runs in polynomial time (as r is a fixed constant), and by the correctness of `ApproxIVD`, if it concludes that (G, ℓ) is a *no*-instance then this decision is correct. Thus, since $\sum_{i=0}^{r}(6\ell)^{r+1} \leq (r+1)(6\ell)^{r+1}$, the correctness of Lemma 8.1 now directly follows as a special case of Corollary 8.1. In light of Lemma 8.1, from now on, we suppose that we have a $(k+2)$-necessary family $\mathcal{W} \subseteq 2^{V(G)}$ along with a $(r+1)(6(k+2))^{r+1}$-solution M that is r-redundant with respect to $\mathcal{W}$ for $r = 9$. Let us note that, any obstruction in G that is not covered by $\mathcal{W}$ intersects M in at least 10-vertices. We have the following reduction rule that follows immediately from Lemma 8.5.

Reduction Rule 8.1. *Let* v *be a vertex such that* $\{v\} \in \mathcal{W}$. *Then output the instance* $(G - \{v\}, k-1)$.

Going forward, we will assume that the above reduction rule is not applicable. Then observe that any set in $\mathcal{W}$ has at least 2 vertices. Let $E^+ = \{(m_1, m_2) \mid \{m_1, m_2\} \in \mathcal{W}, (m_1, m_2) \notin E(G)\}$. Let G^+ be the graph obtained by adding the edges of E^+ to the graph G. Observe that $V(G^+) = V(G)$ and $G^+ - M = G - M$. Furthermore, if S is any solution in G of cardinality $k+2$, then $G^+ - S = G - S$ i.e. S is a solution in G^+. And conversely, if S is a solution in G^+ such that S hits all sets in $\mathcal{W}$, then $G^+ - S = G - S$ i.e. S is a solution in G. The graph G^+ helpful in obtaining certain structural properties of the graph $G - M$.

8.2 Handling Module Components

Identification of Module Components. Let $\mathcal{C}$ denote the set of connected components of $G - M$. Moreover, we let $\mathcal{D}$ denote the set of connected components in $\mathcal{C}$ that are modules, and $\overline{\mathcal{D}} = \mathcal{C} \setminus \mathcal{D}$.

Reduction Rule 8.2. *Suppose that there exist* $v \in M$ *and a set* $\mathcal{A} \subseteq \overline{\mathcal{D}}$ *of size* $k+3$ *such that for each* $D \in \mathcal{A}$, *there exist* $u, w \in V(D)$ *such that* $u \in N_G(v)$ *and* $w \notin N_G(v)$. *Then, output the instance* $(G - \{v\}, k-1)$.

Lemma 8.6. *Reduction Rule 8.2 is safe.*

Proof. In one direction, suppose that (G, k) is a *yes*-instance, and let S be a k-solution for G. Since $|\mathcal{A}| \geq k+3$, there exist three components $D_1, D_2, D_2 \in \overline{\mathcal{D}}$ such that $S \cap (V(D_1) \cup V(D_2) \cup V(D_3)) = \emptyset$. However, the subgraph of G induced by the vertex set consisting of v, together with a neighbor and non-neighbor of v in D_i for all $i \in [3]$, is a long claw. Thus, as $G - S$ is an interval graph, we derive that $v \in S$, and therefore $S \setminus \{v\}$ is a $(k-1)$-solution for $G - \{v\}$.

In the other direction, it is clear that if $(G - \{v\}, k-1)$ is a *yes*-instance, then (G, k) is a *yes*-instance. □

Observation 26. *After the exhaustive application of Reduction Rule 8.2,* $|\overline{\mathcal{D}}| \leq (k+2)|M|$.

Proof. After the exhaustive application of Reduction Rule 8.2, every vertex in M has at most $k+2$ connected components in $\mathcal{C}$ where it has both a neighbor and a non-neighbor. Since for a connected component in $\overline{\mathcal{D}}$ that is not a module, there must exist a vertex in M that has both a neighbor and a non-neighbor in that component, we conclude that the observation is correct. □

The purpose of the rest of this section would be to bound the total number of vertices in all module components. The arguments used to derive this bound will also be necessary at a later stage of our kernelization algorithm, and hence we present our goal in the form of a more general statement:

Lemma 8.7. *Let* $\widehat{M} \subseteq V(G)$*, and* $\widehat{\mathcal{C}}$ *be some set of connected components of* $G-(M\cup\widehat{M})$ *that are modules. In polynomial time, it is possible to either output an instance* (G',k) *equivalent to* (G,k) *where* G' *is a strict subgraph of* G*, or to compute a subset* $B \subseteq V(\widehat{\mathcal{C}})$ *of size at most* $4(k+1)^2|M\cup\widehat{M}|^6$*, such that for any subset* $S \subseteq V(G)$ *of size at most* k*, the following property holds: If there exists an obstruction* $\mathbb{O}$ *for* G *that is not covered by* $\mathcal{W}$ *and such that* $V(\mathbb{O}) \cap S = \emptyset$*, there exists an obstruction* $\mathbb{O}'$ *for* G *such that* $V(\mathbb{O}') \cap S = \emptyset$ *and* $V(\mathbb{O}') \cap (V(\widehat{\mathcal{C}}) \setminus B) = \emptyset$.

Let us now show that having Lemma 8.7 at hand, we can indeed bound the total number of vertices in all module components.

Reduction Rule 8.3. *Let* X *be the output of the algorithm in Lemma 8.7 when called with* $\widehat{M} = \emptyset$ *and* $\widehat{\mathcal{C}} = \mathcal{D}$*. If* X *is an instance* (G',k)*, then output* X*. Otherwise,* X *is a set* $B \subseteq V(\mathcal{D})$*, and we output the instance* $(G-\{v\},k)$*, where* v *is an arbitrarily chosen vertex from* $V(\mathcal{D}) \setminus B$.

By using Lemma 8.7, we derive the safeness of Reduction Rule 8.3.

Lemma 8.8. *Reduction Rule 8.7 is safe.*

Proof. If X is an instance (G',k), then Lemma 8.7 directly implies that the rule is safe. Thus, we next suppose that $X = B$. In one direction, it is clear that if (G,k) is a *yes*-instance, then $(G-\{v\},k)$ is a *yes*-instance as well.

In the other direction, suppose that $(G-\{v\},k)$ is a *yes*-instance. Let S be a k-solution for $G-\{v\}$. We claim that S is also a k-solution for G. Suppose, by way of contradiction, that this claim is false. Then, there exists an obstruction $\mathbb{O}$ for $G-S$. As $S\cup\{v\}$ is a $(k+1)$-solution for G and $\mathcal{W}$ is $(k+2)$-necessary, we have that $S\cup\{v\}$ hits $\mathcal{W}$. Since $v \notin M$ and $\mathcal{W} \subseteq 2^M$, we derive that S hits $\mathcal{W}$. Thus, since $\mathbb{O}$ is an obstruction for $G-S$, we deduce that $\mathbb{O}$ is not covered by $\mathcal{W}$. Hence, by Lemma 8.7, there exists an obstruction $\mathbb{O}'$ for G such that $V(\mathbb{O}') \cap S = \emptyset$ and $V(\mathbb{O}') \cap (V(\widehat{\mathcal{D}}) \setminus B) = \emptyset$. However, as $v \in V(\mathcal{D}) \setminus B$, this implies that $\mathbb{O}'$ is also an obstruction for $(G-\{v\})-S$, which is a contradiction as S is a k-solution for $G-\{v\}$. □

Due to Reduction Rule 8.3, we have the following result.

Observation 27. *After the exhaustive application of Reduction Rule 8.3,* $|V(\mathcal{D})| \leq 4(k+1)^2|M|^6$.

We now turn to prove Lemma 8.7. In what follows, $\widehat{M}$ and $\widehat{\mathcal{C}}$ are as stated in this lemma. We denote $M' = M \cup \widehat{M}$. Note that since M is 9-redundant with respect to $\mathcal{W}$, we have that M' is also 9-redundant with respect to $\mathcal{W}$.

Structure of Obstructions Intersecting Module Components. We first consider the neighborhoods of non-adjacent vertices in M'.

Lemma 8.9. *Let $u, v \in V(G)$ such that $(u, v) \notin E(G)$ and no (non-empty) subset of $\{u, v\}$ belongs to $\mathcal{W}$. Then, $G[(N_G(u) \cap N_G(v)) \setminus M']$ is a clique.*

Proof. Suppose, by way of contradiction, that $G[N_G(u) \cap N_G(v) \setminus M']$ is not a clique. Then, there exist two vertices $x, y \in N_G(u) \cap N_G(v) \setminus M'$ that are not neighbors in G. Note that $\mathbb{O} = G[\{u, v, x, y\}]$ is a hole, and that $M \cap V(\mathbb{O}) \subseteq \{u, v\}$. Moreover, $\mathbb{O}$ is not covered by $\mathcal{W}$ (as no subset of $\{u, v\}$ belongs to $\mathcal{W}$). Since M is 9-redundant, this means that $|M \cap V(\mathbb{O})| > 9$, which is a contradiction. □

Let us now state a proposition by Cao and Marx [CM15b].

Proposition 8.2 ([CM15b])**.** *Let C be a module in G and $\mathbb{O}$ be a minimal obstruction. If $|V(\mathbb{O})| > 4$, then either $V(\mathbb{O}) \subseteq V(C)$ or $|V(\mathbb{O}) \cap V(C)| \leq 1$.*

By Proposition 8.2, we directly obtain the following lemma.

Lemma 8.10. *Let C be a module such that $V(C) \cap M' = \emptyset$, and let $\mathbb{O}$ be a minimal obstruction that is not covered by $\mathcal{W}$. Then, $|V(\mathbb{O}) \cap V(C)| \leq 1$.*

Proof. Since $\mathbb{O}$ be an obstruction that is not covered by $\mathcal{W}$, it holds that $|M' \cap V(\mathbb{O})| > 9$. In particular, as $V(C) \cap M' = \emptyset$, we have that $|V(\mathbb{O})| > 4$ and $V(\mathbb{O}) \setminus V(C) \neq \emptyset$. Then, as C is a module and $\mathbb{O}$ is minimal, by Proposition 8.2, we have that $|V(\mathbb{O}) \cap V(C)| \leq 1$. □

Reducing the Size of Module Components. To ensure we have only small module components, we apply the following rule.

Reduction Rule 8.4. *Suppose that there exists $C \in \widehat{\mathcal{C}}$ such that $|V(C)| > k+1$. Then, output the instance $(G - \{v\}, k)$, where v is an arbitrarily chosen vertex of C.*

Lemma 8.11. *Reduction Rule 8.4 is safe.*

Proof. In one direction, it is clear that if (G, k) is a *yes*-instance, then $(G - \{v\}, k)$ is a *yes*-instance as well.

In the other direction, suppose that $(G - \{v\}, k)$ is a *yes*-instance. Let S be a k-solution for $G - \{v\}$. We claim that S is also a k-solution for G. Suppose, by way of contradiction, that this claim is false. Then, there exists a minimal obstruction $\mathbb{O}$ for $G - S$. As $S \cup \{v\}$ is a $(k+1)$-solution for G and $\mathcal{W}$ is $(k+2)$-necessary, we have that $S \cup \{v\}$ hits $\mathcal{W}$. Since $v \notin M$ and $\mathcal{W} \subseteq 2^M$, we derive that S hits $\mathcal{W}$. Thus, since $\mathbb{O}$ is an obstruction for $G - S$, we deduce that $\mathbb{O}$ is not covered by $\mathcal{W}$. Hence, by Lemma 8.10, $|V(\mathbb{O}) \cap V(C)| \leq 1$. Thus, $V(\mathbb{O}) \cap V(C) = \{v\}$. Then, as C is a module, for any vertex $u \in V(C)$, it holds that $G[(V(\mathbb{O}) \setminus \{v\}) \cup \{u\}]$ is an obstruction. Since $|V(C)| > k+1$, we have that $V(C) \setminus (S \cup \{v\}) \neq \emptyset$. However, this implies that there exists an obstruction $\mathbb{O}'$ for $(G - \{v\}) - S$, which is a contradiction as S is a k-solution for $G - \{v\}$. □

Preliminary Marking Scheme. By Lemma 8.9, for all $u, v \in M'$ such that $(u, v) \notin E(G)$ and no subset of $\{u, v\}$ belongs to $\mathcal{W}$, there exists at most one $C \in \widehat{\mathcal{C}}$, denoted by C_{uv}, such that $N_G(u) \cap N_G(v) \cap V(C) \neq \emptyset$. Accordingly, denote $\mathcal{C}^\star = \{C_{uv} \in \widehat{\mathcal{C}} \mid u, v \in M', (u, v) \notin E(G),$ no subset of $\{u, v\}$ belongs to $\mathcal{W}\}$. Moreover, denote $A^\star = V(\mathcal{C}^\star)$. Clearly, we have the following observation.

Let us recall the graph G^+ that is obtained from G by adding a new edge for each set of cardinality 2 in $\mathcal{W}$.

Observation 28. *For any vertex $v \in V(\widehat{\mathcal{C}} \setminus \mathcal{C}^\star)$, $N_G(v) \cap M'$ is a clique in G^+. Further, $N_G(v)$ is a clique in G^+.*

Furthermore, due to Reduction Rule 8.4, we have the following observation.

Observation 29. *The size of $A^\star$ is upper bounded by $(k+1)|M'|^2$.*

Lemma 8.12. *Let $C \in \widehat{\mathcal{C}} \setminus \mathcal{C}^\star$, and $\mathbb{O}$ be a minimal obstruction that is not covered by $\mathcal{W}$ such that $V(\mathbb{O}) \cap V(C) \neq \emptyset$. Then, $|V(\mathbb{O}) \cap V(C)| = 1$ and $\mathbb{O}$ is an AW where the vertex in $V(\mathbb{O}) \cap V(C)$ is a terminal.*

Proof. Suppose not, and consider a vertex v for which the lemma doesn't hold. First, as C is a module, from Lemma 8.10 we deduce that $|V(\mathbb{O}) \cap V(C)| = 1$. Furthermore, as $\mathbb{O}$ is not covered by $\mathcal{W}$, we have that $|V(\mathbb{O})| > 9$. In particular, this means that $\mathbb{O}$ is neither a long claw nor a whipping top. Now, observe that for every vertex v on a chordless cycle, it holds that the neighborhood of v on that chordless cycles is not a clique. Moreover, for every vertex v on an AW that is not a terminal of that AW, it holds that the neighborhood of v on that AW is not a clique. Therefore, $N(v) \cap \mathbb{O}$ contains a pair of non-adjacent vertices x, y. Now, by Observation 28, we have that $N_G(v)$ is a clique in G^+, i.e. $(x, y) \in E(G^+)$. Hence, it follows that $x, y \in \mathbb{O} \cap M$ and $\{x, y\} \in \mathcal{W}$. But then $\mathbb{O}$ is covered by $\mathcal{W}$, which is a contradiction. Hence the lemma holds. □

Marking Scheme to Handle Non-Shallow Terminals. For every two subsets $X, Y \subseteq M'$ such that $|X| \leq 2$ and $|Y| \leq 2$, denote $A_{X,Y} = \{v \in V(\widehat{\mathcal{C}} \setminus \mathcal{C}^\star) \mid X \subseteq N_G(v), Y \cap N_G(v) = \emptyset\}$. Now, if $|A_{X,Y}| \leq k+1$, then define $A'_{X,Y} = A_{X,Y}$, and otherwise let $A'_{X,Y}$ be an arbitrarily chosen subset of size $k+1$ of $A_{X,Y}$. Let us denote $A' = \bigcup_{X,Y} A'_{X,Y}$, where X, Y range over all subsets $X, Y \subseteq M'$ such that $|X| = 2$ and $|Y| \leq 2$.

Let us first observe that $|A'|$ is small.

Observation 30. *The size of A' is upper bounded by $(k+1)|M'|^4$.*

Now, let us verify that we have a set of vertices that is sufficient to "handle" non-shallow terminals.

Lemma 8.13. *Let $C \in \widehat{\mathcal{C}} \setminus \mathcal{C}^\star$, $v \in V(C) \setminus A'$, and $\mathbb{O}$ be a minimal obstruction that is not covered by $\mathcal{W}$ such that $v \in V(\mathbb{O})$. If $\mathbb{O}$ is not an AW where v is the shallow terminal, then there exists a set $\hat{A} \subseteq A'$ of size $k+1$ such that for each $u \in \hat{A}$, $G[(V(\mathbb{O}) \setminus \{v\}) \cup \{u\}]$ is an obstruction.*

Proof. First, by Lemma 8.12, we have that $\mathbb{O}$ is an AW such that $V(\mathbb{O}) \cap V(C) = \{v\}$ and v is a terminal of $\mathbb{O}$. Let us also note that $N_G(v) \subseteq M' \cup C$ and therefore $N_G(v) \cap V(\mathbb{O}) \subseteq M'$. Suppose that v is not the shallow terminal of $\mathbb{O}$. Then, we have that v is either t_ℓ or t_r, where we denote the vertex set of $\mathbb{O}$ as in the definition of an AW. Without loss of generality, suppose that $v = t_\ell$. Let us consider two cases, depending on whether $\mathbb{O}$ is a †-AW or a ‡-AW.

- Suppose that $\mathbb{O}$ is a †-AW. By the construction of A', there exists a subset $\hat{A} \subseteq A'$ of $k+1$ vertices such that for each $u \in \hat{A}$, u is adjacent to b_1, and u is not adjacent to b_2 and c. Notice that $b_1 \in M'$ as $(b_1, v) \in E(G)$. Furthermore b_1 is not adjacent to any vertex on $\mathbb{O}$ besides v, c and b_2. For all $u \in Q$, by Observation 28 and since $N_G(u) \subseteq V(C) \cup M'$, we have that u is not adjacent to any vertex on $\mathbb{O} - \{v\}$ besides b_1. Hence, for any vertex $u \in Q$, $G[(V(\mathbb{O}) \setminus \{v\}) \cup \{u\}]$ is also a †-AW.
- Suppose that $\mathbb{O}$ is a ‡-AW. By the construction of A', there exists a subset $Q \subseteq A'$ of $(k+1)$ vertices $u \in A'$ such that u is adjacent to both c_1 and b_1, and u is adjacent to neither c_2 nor b_2. Notice that $c_1, b_1 \in M'$ and that b_1 is not adjacent to any vertex on $\mathbb{O}$ besides v, c_1, c_2 and b_2. For all $u \in Q$, by Observation 28 and since $N_G(u) \subseteq V(C) \cup M'$, we have that u is not adjacent to any vertex on $\mathbb{O} - \{v\}$ besides c_1 and b_1. Hence, for any vertex $u \in Q$, $G[(V(\mathbb{O}) \setminus \{v\}) \cup \{u\}]$ is also a ‡-AW.

In both cases, we derived the desired claim, and thus the proof is complete. □

Marking Scheme to Handle Shallow Terminals. Before we present our marking scheme, let us explicitly state the following observation, which follows from Observation 28 in the same manner as Lemma 8.12. We require the following notation. We say that the a path P is covered by $\mathcal{W}$ if there is a set $W \in \mathcal{W}$ such that $W \subseteq V(P)$.

Observation 31. *Let P be an induced path in $G[V(G) \setminus V(C)]$ for some $C \in \widehat{\mathcal{C}} \setminus \mathcal{C}^\star$ such that P is not covered by $\mathcal{W}$. For all $v \in V(C)$, $|N_G(v) \cap V(P)| \leq 2$, and if $|N_G(v) \cap V(P)| = 2$, then the two vertices in $N_G(v) \cap V(P)$ are adjacent on P.*

Proof. Since P is an induced path in G that is not covered by $\mathcal{W}$, it follows from the definition of G^+ that P is an induced path in G^+ as well. And clearly, the statement holds in G^+ by Observation 28. Now the claim follows by the observation that $G[P \cup v] = G^+[P \cup v]$. □

Denote $N = M' \cup A^\star \cup A'$. For all (not necessarily distinct) vertices $c_1, c_2 \in M'$, denote $A_{\{c_1,c_2\}} = \{v \in V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A') \mid \{c_1, c_2\} \subseteq N_G(v)\}$. Moreover, let us arbitrarily order N and $E(G[N])$ as follows: $N = \{v_1, v_2, \ldots, v_{|N|}\}$ and $E(G[N]) = \{e_1, e_2, \ldots, e_{|E(G[N])|}\}$. Next, with every vertex $u \in V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A')$, we associate two binary vectors that capture the relations between u and the vertices in N: **(i)** $\mathsf{vinc}(u) = (b_1, b_2, \ldots, b_{|N|})$, where for all $i \in [|N|]$, $b_i = 1$ if and only if $v_i \in N_G(u)$; **(ii)** $\mathsf{einc}(u) = (b_1, b_2, \ldots, b_{|E(G[N])|})$, where for all $i \in [|E(G[N])|]$, $b_i = 1$ if and only if u is adjacent to both endpoints of e_i. In addition, we define $\mathsf{inc}(u)$ as the vector that is the concatenation of $\mathsf{vinc}(u)$ and $\mathsf{einc}(u)$, to which we add 1 at the end. Formally, $\mathsf{inc}(u)$ is a binary vector with $|N| + |E(G[N])| + 1$ entries, where for all $i \in [|N|]$, the i^{th} entry of $\mathsf{inc}(u)$ equals the i^{th} entry of $\mathsf{vinc}(u)$, for all $i \in [|E(G[N])| + |N|] \setminus [|N|]$, the i^{th} entry of $\mathsf{inc}(u)$ equals the $(i - |N|)^{th}$ entry of

$\mathsf{einc}(u)$, and the last entry of $\mathsf{inc}(u)$ is 1. These incidence vectors are associated with the vector space $\mathbb{F}_2^q$ for $q = |N| + |E(G[N])| + 1$, and all calculations related to these vector are performed accordingly.

Next, for all (not necessarily distinct) vertices $c_1, c_2 \in M'$, we have the following marking scheme. First, we define the *multiset* $\mathbf{V}_{\{c_1,c_2\}}$ as $\{\mathsf{inc}(u) \mid u \in A_{\{c_1,c_2\}}\}$ (more precisely, the number of occurrences of a vector in $\mathbf{V}_{\{c_1,c_2\}}$ equals the number of vertices $u \in A_{\{c_1,c_2\}}$ such that $\mathsf{inc}(u)$ equals that vector). Initialize $\widehat{\mathbf{V}}^0_{\{c_1,c_2\}} = \emptyset$. Now, for $i = 1, 2, \ldots, k+1$, compute some basis $\mathbf{B}^i_{\{c_1,c_2\}}$ for the vector subspace $\mathbf{V}_{\{c_1,c_2\}} \setminus \widehat{\mathbf{V}}^{i-1}_{\{c_1,c_2\}}$ (with respect to $\mathbb{F}_2^q$),[1] and denote $\widehat{\mathbf{V}}^i_{\{c_1,c_2\}} = \widehat{\mathbf{V}}^{i-1}_{\{c_1,c_2\}} \cup \mathbf{B}^i_{\{c_1,c_2\}}$. For every occurrence of a vector $\mathbf{v} \in \widehat{\mathbf{V}}^{k+1}_{\{c_1,c_2\}}$, we arbitrarily choose a unique vertex $u \in A_{\{c_1,c_2\}}$ such that $\mathsf{inc}(u) = \mathbf{v}$ and denote it by $u_{\mathbf{v}}$ (the existence of sufficiently many such distinct vertices directly follows from the definition of $\mathbf{V}_{\{c_1,c_2\}}$). Denote $\widehat{A}_{\{c_1,c_2\}} = \{u_{\mathbf{v}} \mid \mathbf{v} \in \widehat{\mathbf{V}}^{k+1}_{\{c_1,c_2\}}\}$, and note that $\widehat{A}_{\{c_1,c_2\}}$ is a set (rather than a multiset). Finally, we denote $\widehat{A} = \bigcup_{c_1,c_2 \in M'} \widehat{A}_{\{c_1,c_2\}}$ (here, union refers to sets, that is, every vertex occurs in $\widehat{A}$ once even if it belongs to more than one set of the form $\widehat{A}_{\{c_1,c_2\}}$). Let us first observe that $|\widehat{A}|$ is small.

Lemma 8.14. *The size of $\widehat{A}$ is upper bounded by $(k+1)|M'|^2|N|^2 \leq 2(k+1)^2|M'|^6$.*

Proof. To show that $|\widehat{A}| \leq (k+1)|M'|^2|N|^2$, it is sufficient to show that for all $c_1, c_2 \in M'$, $|\widehat{A}_{\{c_1,c_2\}}| \leq (k+1)|N|^2$. To this end, consider some $c_1, c_2 \in M'$. Now, observe that the number of entries of the vectors in $\mathbf{V}_{\{c_1,c_2\}}$ is $q = |N| + |E(G[N])| + 1 \leq |N| + \frac{|N|(|N|-1)}{2} + 1 \leq |N|^2$ (assuming $|N| > 1$). Hence, every basis of $\mathbf{V}_{\{c_1,c_2\}}$ (or a subset of $\mathbf{V}_{\{c_1,c_2\}}$) is of size at most $|N|^2$. As $\mathbf{V}_{\{c_1,c_2\}}$ is a multiset that is the union of $(k+1)$ bases of $\mathbf{V}_{\{c_1,c_2\}}$ (or subsets of $\mathbf{V}_{\{c_1,c_2\}}$), we have that $|\mathbf{V}_{\{c_1,c_2\}}| \leq (k+1)|N|^2$. Since $|\mathbf{V}_{\{c_1,c_2\}}| = |\widehat{A}_{\{c_1,c_2\}}|$, the proof is complete. □

Now, let us verify that we have a set of vertices that is sufficient to "handle" shallow terminals. For this purpose, we need the following notation. Given a pair (X, Y), where $X \subseteq N$ and $Y \subseteq E(G[N])$, and a vertex $u \in V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A')$, we define $\mathsf{inc}^{X,Y}(u)$ to be the vector $\mathsf{inc}(u)$ where all entries associated with vertices and edges that do not belong to $X \cup Y$ are changed to 0. Formally, $\mathsf{inc}^{X,Y}(u)$ is a binary vector with $|N| + |E(G[N])| + 1$ entries, where for all $i \in [|N|]$, the i^{th} entry of $\mathsf{inc}(u)$ equals the i^{th} entry of $\mathsf{vinc}(u)$ if $v_i \in X$ and to 0 otherwise, for all $i \in [|E(G[N])| + |N|] \setminus [|N|]$, the i^{th} entry of $\mathsf{inc}^{X,Y}(u)$ equals the $(i - |N|)^{th}$ entry of $\mathsf{einc}(u)$ if $e_{i-|N|} \in Y$ and to 0 otherwise, and the last entry of $\mathsf{inc}^{X,Y}(u)$ is 1. Furthermore, for an induced path P in $G - (V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A'))$ and a vertex $u \in V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A')$, we denote $\mathsf{inc}^P(u) = \mathsf{inc}^{X,Y}(u)$ where $X = V(P) \cap N$ and $Y = E(P) \cap E(G[N])$. Moreover, recall that given a vector $\mathbf{v}$ and an entry index i, $\mathbf{v}[i]$ denotes the i^{th} entry of $\mathbf{v}$.

Lemma 8.15. *Let P be an induced path in $G[V(G) \setminus V(C)]$ for some $C \in \widehat{\mathcal{C}} \setminus \mathcal{C}^\star$ such that P is not covered by $\mathcal{W}$. For all $u \in V(C)$, $\sum_{i=1}^q \mathsf{inc}^P(u)[i] = 1 \mod 2$ if and only if $N_G(u) \cap V(P) = \emptyset$.*

[1] Here, note that the subtraction concerns multisets. In particular, if an element occurs x times in a multiset X, and y times in a multiset $Y \subseteq X$, then it occurs $x - y$ times in $X \setminus Y$.

Proof. Consider some vertex $u \in V(C)$. If $N_G(u) \cap V(P) = \emptyset$, then all of the entries of $\mathsf{inc}^P(u)$ equal 0, except for the last entry which equals 1. Then, $\sum_{i=1}^{q} \mathsf{inc}^P(u)[i] = 1 \mod 2$. Now, suppose that $N_G(u) \cap V(P) \neq \emptyset$. Then, by Observation, 31, $|N_G(u) \cap V(P)|$ is either 1 or 2, and if it is 2, then the two vertices in $N_G(u) \cap V(P)$ are adjacent on P. Furthermore, observe that that as $V(P) \cap V(C) = \emptyset$, we have that $N_G(u) \cap V(P) \subseteq M'$. Thus, in case $|N_G(u) \cap V(P)| = 1$, then there exists exactly one entry in $\mathsf{inc}^P(u)$ that equals 1 apart from the last entry, which is the entry corresponding to the vertex in $N_G(u) \cap V(P)$. Moreover, in case $|N_G(u) \cap V(P)| = 2$, then there exist exactly three entries in $\mathsf{inc}^P(u)$ that equal 1 apart from the last entry, which are two entries corresponding to the two vertices in $N_G(u) \cap V(P)$ and the entry corresponding to the edge between these two vertices. In both cases, we derive that $\sum_{i=1}^{q} \mathsf{inc}^P(u)[i] = 0 \mod 2$ as desired. □

Lemma 8.16. *Let $w \in V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A' \cup \widehat{A})$, and $\mathbb{O}$ be an AW that is not covered by $\mathcal{W}$ such that $V(\mathbb{O}) \cap (V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A' \cup \widehat{A})) = \{w\}$. Let $\{c_1, c_2\}$ be the set of centers of $\mathbb{O}$, where if $\mathbb{O}$ is a †-AW then $c_1 = c_2$. If w is the shallow terminal of $\mathbb{O}$, then for all $i \in [k+1]$, there exists $\mathbf{v} \in \mathbf{B}^i_{\{c_1,c_2\}}$ such that $G[(V(\mathbb{O}) \setminus \{w\}) \cup \{u_{\mathbf{v}}\}]$ is an obstruction.*

Proof. Suppose that w is the shallow terminal of $\mathbb{O}$, and consider some $i \in [k+1]$. Let us first argue that there exists an occurrence of $\mathsf{inc}(w)$ in $\mathbf{V}_{\{c_1,c_2\}} \setminus \widehat{\mathbf{V}}^{i-1}_{\{c_1,c_2\}}$. To this end, note that as w is the shallow terminal of $\mathbb{O}$, it is adjacent to c_1 and c_2, and therefore $w \in A_{\{c_1,c_2\}}$. Moreover, because $w \notin \widehat{A}$, there exists an occurrence of $\mathsf{inc}(w)$ that does not belong to $\mathbf{V}^{k+1}_{\{c_1,c_2\}}$, which overall implies that there indeed exists an occurrence of $\mathsf{inc}(w)$ in $\mathbf{V}_{\{c_1,c_2\}} \setminus \widehat{\mathbf{V}}^{i-1}_{\{c_1,c_2\}}$. Thus, since $\mathbf{B}^i_{\{c_1,c_2\}}$ is a basis for $\mathbf{V}_{\{c_1,c_2\}} \setminus \widehat{\mathbf{V}}^{i-1}_{\{c_1,c_2\}}$, there exist vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t$ for some $t \in \mathbb{N}$ (in particular, $t \geq 1$) and coefficients $\lambda_1 = \lambda_2 = \ldots = \lambda_t = 1$ (as the coefficient are from field $\mathbb{F}_2$, they are necessarily 1) such that $\lambda_1\mathbf{v}_1 + \lambda_2\mathbf{v}_2 + \cdots + \lambda_t\mathbf{v}_t = \mathsf{inc}(w)$ over $\mathbb{F}_2^q$, that is, $\mathbf{v}_1 + \mathbf{v}_2 + \cdots + \mathbf{v}_t = \mathsf{inc}(w)$ over $\mathbb{F}_2^q$.

Denote $u_i = u_{\mathbf{v}_i}$ for all $i \in [t]$. Then, $\mathsf{inc}(u_1) + \mathsf{inc}(u_2) + \cdots + \mathsf{inc}(u_t) = \mathsf{inc}(w)$ over $\mathbb{F}_2^q$. In particular, $\mathsf{inc}^P(u_1) + \mathsf{inc}^P(u_2) + \cdots + \mathsf{inc}^P(u_t) = \mathsf{inc}^P(w)$ over $\mathbb{F}_2^q$, where P is the extended base of $\mathbb{O}$. (Note that since $V(\mathbb{O}) \cap (V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A' \cup \widehat{A})) = \{w\}$, the extended base is completely contained in $G[V(G) \setminus (V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A' \cup \widehat{A}))]$, and furthermore P is not covered by $\mathcal{W}$ by the premise of the lemma.) This implies that $\sum_{i=1}^{t} \sum_{j=1}^{q} \mathsf{inc}^P(u_i)[j] = \sum_{j=1}^{q} \mathsf{inc}^P(w)[j] \mod 2$. By Lemma 8.15 and because $N_G(w) \cap V(P) = \emptyset$ (as w is the shallow terminal of $\mathbb{O}$), we have that $\sum_{j=1}^{q} \mathsf{inc}^P(w)[j] = 1 \mod 2$. Thus, $\sum_{i=1}^{t} \sum_{j=1}^{q} \mathsf{inc}^P(u_i)[j] = 1 \mod 2$. This implies that there exists $i \in [t]$ such that $\sum_{j=1}^{q} \mathsf{inc}^P(u_i)[j] = 1 \mod 2$. However, by Lemma 8.15, this means that $N_G(u_i) \cap V(P) = \emptyset$. Moreover, we have that $u_i \in A_{\{c_1,c_2\}}$, else u_i could not have been associated with $\mathbf{v}_i \in \mathbf{B}^i_{\{c_1,c_2\}}$ (note that here $\mathbf{v}_i$ refers to a specific occurrence of a vector). Hence, $G[(V(\mathbb{O}) \setminus \{w\}) \cup \{u_i\}]$ is an AW. This completes the proof. □

As a direct corollary to Lemma 8.14 we have the following result.

Corollary 8.2. *Let $w \in V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A' \cup \widehat{A})$, and $\mathbb{O}$ be an AW that is not covered by $\mathcal{W}$ such that $V(\mathbb{O}) \cap (V(\widehat{\mathcal{C}}) \setminus (A^\star \cup A' \cup \widehat{A})) = \{w\}$. If w is the shallow terminal of $\mathbb{O}$, then there exists a set $\tilde{A} \subseteq \widehat{A}$ of size $k+1$ such that for each $u \in \tilde{A}$, $G[(V(\mathbb{O}) \setminus \{w\}) \cup \{u\}]$ is an obstruction.*

Proof of Lemma 8.7. We are now ready to conclude the proof of Lemma 8.7. For this purpose, first note that if the condition of Reduction Rule 8.4 applies, then we are clearly done, as in the case we output an instance (G', k) equivalent to (G, k) where G' is a strict subgraph of G. Thus, we next suppose that this rule has been applied exhaustively. Then, our output is the set $B = A^\star \cup A' \cup \widehat{A}$. By Observations 30 and 30, and by Lemma 8.14, we have that $|B| \leq |A^\star| + |A'| + |\widehat{A}| \leq (k+1)|M'|^2 + (k+1)|M'|^4 + 2(k+1)^2|M'|^6 \leq 4(k+1)^2|M'|^6$ as desired.

Let $S \subseteq V(G)$ be some arbitrary set of size at most k. We claim that the following property holds: If there exists an obstruction $\mathbb{O}$ for G that is not covered by $\mathcal{W}$ and such that $V(\mathbb{O}) \cap S = \emptyset$, there exists an obstruction $\mathbb{O}'$ for G such that $V(\mathbb{O}') \cap S = \emptyset$ and $V(\mathbb{O}') \cap (V(\widehat{\mathcal{C}}) \setminus B) = \emptyset$. Clearly, if there does not exist any obstruction $\mathbb{O}$ for G that is not covered by $\mathcal{W}$ and such that $V(\mathbb{O}) \cap S = \emptyset$, then our proof is complete. Hence, we next suppose that such an obstruction exists, and we let $\mathbb{O}'$ be such a minimal obstruction that minimizes $V(\mathbb{O}') \cap (V(\widehat{\mathcal{C}}) \setminus B)$. We claim that for this obstruction $\mathbb{O}'$, it holds that $V(\mathbb{O}') \cap (V(\widehat{\mathcal{C}}) \setminus B) = \emptyset$, which would complete the proof. Suppose, by way of contradiction, that this claim is false. Then, as $V(\mathcal{C}^\star) \subseteq B$, there exists $C \in \widehat{\mathcal{C}} \setminus \mathcal{C}^\star$ and $v \in V(C)$ such that $v \in V(\mathbb{O}')$. By Lemma 8.12, $|V(\mathbb{O}) \cap V(C)| = 1$ and $\mathbb{O}'$ is an AW where v is a terminal.

Let us first suppose that v is not the shallow terminal of $\mathbb{O}'$. Then, by Lemma 8.13, there exist $(k+1)$ vertices $u \in A'$ such that $G[(V(\mathbb{O}') \setminus \{v\}) \cup \{u\}]$ is an obstruction. However, as $|S| \leq k$, this means that there exists $u \in A' \setminus S$ such that $G[(V(\mathbb{O}') \setminus \{v\}) \cup \{u\}]$ is an obstruction. As $A' \subseteq B$ and $G[(V(\mathbb{O}') \setminus \{v\}) \cup \{u\}]$ has fewer vertices from $V(\widehat{\mathcal{C}}) \setminus B$ than $\mathbb{O}'$, we have reached a contradiction to the choice of $\mathbb{O}$.

As the choice of v was arbitrary, we derive that $V(\mathbb{O}') \cap (V(\widehat{\mathcal{C}}) \setminus B)$ contains exactly one vertex, which we denote by w, that is the shallow terminal of $\mathbb{O}'$. In this case, by Corollary 8.2, there exist $(k+1)$ vertices $u \in \widehat{A}$ such that $G[(V(\mathbb{O}) \setminus \{w\}) \cup \{u\}]$ is an obstruction. However, as $|S| \leq k$, this means that there exists $u \in \widehat{A} \setminus S$ such that $G[(V(\mathbb{O}') \setminus \{w\}) \cup \{u\}]$ is an obstruction. As $\widehat{A} \subseteq B$ and $G[(V(\mathbb{O}') \setminus \{w\}) \cup \{u\}]$ has no vertices from $V(\widehat{\mathcal{C}}) \setminus B$, we have again reached a contradiction to the choice of $\mathbb{O}$. This completes the proof. □

8.3 Bounding the Maximum Size of a Clique

Let $\eta = 2^{10} \cdot 4(k+5)\binom{|M|}{10}$. Let $(\mathbb{P}, \beta)$ be a clique-path of $G[V(\overline{\mathcal{D}})]$, $V(\mathbb{P}) = \{x_1, x_2, \ldots, x_t\}$, and for $i \in [t]$ we have $B_i = \beta(x_i)$. Furthermore, let $\beta(\mathbb{P}) = \cup_{i=1}^{t} \beta(x_i)$. Let B_i be a bag such that $|B_i| > \eta$. Towards bounding the size of B_i we mark some of vertices in B_i, and delete all the unmarked vertices in B_i from G. Next, we move to the description of the marking scheme.

Marking Scheme. Towards defining the marking scheme we introduce the following function. For $i \in [t]$, we define two functions namely, $\mathsf{id}^i_\ell, \mathsf{id}^i_r : B_i \to [t]$. Intuitively, these functions denote how far or close a vertex appears in bags that are in left and right direction from the bag B_i. For a vertex $v \in B_i$, $\mathsf{id}^i_\ell(v)$ is the smallest integer $x \in [t]$ such that $v \in B_x$, and $\mathsf{id}^i_r(v)$ is the largest integer $y \in [t]$ such that $v \in B_y$. We now move to the description of creating the set $H_i \subseteq B_i$, of marked vertices. A frame $\mathbb{F} = (X, Y)$ in G is a tuple such that $X \subseteq M$ of size at most 10 and $Y \subseteq X$. A vertex $v \in V(G)$ is said

to *fit* a frame $\mathbb{F} = (X, Y)$ if $N(v) \cap X = Y$. For each frame $\mathbb{F}$ in G we create four sets $L_f^{\mathbb{F},i}, L_c^{\mathbb{F},i}, R_f^{\mathbb{F},i}, R_c^{\mathbb{F},i} \subseteq B_i$ of marked vertices each of size as large as $k+5$ (and add these vertices to H_i) as follows.

- We create the set $L_f^{\mathbb{F},i}$ by marking as large as $k+5$ vertices in B_i with (at most) $k+5$ lowest values of id_ℓ^i among the unmarked vertices which fit the frame $\mathbb{F}$.
- We create the set $L_c^{\mathbb{F},i}$ by marking as large as $k+5$ vertices in B_i with (at most) $k+5$ highest values of id_ℓ^i among the unmarked vertices which fit the frame $\mathbb{F}$.
- We create the set $R_f^{\mathbb{F},i}$ by marking as large as $k+5$ vertices in B_i with (at most) $k+5$ highest values of id_r^i among the unmarked vertices which fit the frame $\mathbb{F}$.
- We create the set $R_c^{\mathbb{F},i}$ by marking as large as $k+5$ vertices in B_i with (at most) $k+5$ lowest values of id_r^i among the unmarked vertices which fit the frame $\mathbb{F}$.

We denote the set of marked vertices by H_i. Notice that $|H_i| \leq 2^{10} \cdot 4(k+5)\binom{|M|}{10}$. Next, we argue that an unmarked vertex is irrelevant, and hence can be deleted from G.

Reduction Rule 8.5. *Let v be a vertex in $B_i \setminus H_i$. Delete v from G i.e., the resulting instance is $(G - \{v\}, k)$.*

Lemma 8.17. *Reduction Rule 8.5 is safe.*

Before proving Lemma 8.17, we prove the bound that we can obtain on the size of B_i, where $i \in [t]$ using it.

Theorem 8.2. *If Reduction Rule 8.5 is not applicable then for each $i \in [t]$ we have $|B_i| \leq \eta$.*

Proof. Follows from safeness of Reduction Rule 8.5 (Lemma 8.17) and $|H_i| \leq 2^{10} \cdot 4(k+5)\binom{|M|}{10} = \eta$, where $i \in [t]$. □

In the remainder of this section we focus on the proof of Lemma 8.17. Let v be a vertex in $B_i \setminus H_i$ and $G' = G - \{v\}$. We will show that (G, k) is a *yes* instance of IVD if and only if (G', k) is a *yes* instance of IVD. In the forward direction, let S be a solution to IVD in (G, k). Since $G \setminus S$ is an interval graph implies that all its induced subgraphs are also interval graphs. Hence, it follows that S is a solution to IVD in (G', k).

In the reverse direction, let S be a solution to IVD in (G', k). We will show that $G \setminus S$ is an interval graph. Suppose not, then there must be an obstruction in $G \setminus S$ containing v. Here, all the obstructions in $G \setminus S$ are guaranteed to contain v because otherwise the obstruction is also present in $G' \setminus S$, which contradicts that S is a solution to IVD in (G', k). This implies that $S \cup \{v\}$ is a $(k+1)$-Solution for G. Recall that $\mathcal{W}$ is $(k+1)$-necessary, therefore $S \cup \{v\}$ hits $\mathcal{W}$. Since $v \notin M$ and $\mathcal{W} \subseteq 2^M$, we derive that S hits $\mathcal{W}$. But then any construction in $G \setminus S$ is not covered by $\mathcal{W}$ since $v \notin M$. This together with the fact that M is a 9-redundant modulator implies that for an obstruction $\mathbb{O}'$ in $G \setminus S$ we have $|V(\mathbb{O}') \cap M| \geq 10$. Consider an obstruction $\mathbb{O}$ containing v in $G \setminus S$ which is of minimum possible size, and over all such minimum sized obstructions, $\mathbb{O}$ maximizes the number of vertices from B_i. Note that $|V(\mathbb{O}') \cap M| \geq 10$, therefore $\mathbb{O}$ must be a cycle, a †-AW, or a ‡-AW on at least 10 vertices. Next, we consider cases depending on which

type of obstruction $\mathbb{O}$ is, and the role that v plays in $\mathbb{O}$. In each of these cases our goal will be either to construct a smaller sized obstruction, an obstruction not containing v, or an obstruction which has same number of vertices as $\mathbb{O}$ but has more vertices from B_i than $\mathbb{O}$ has from B_i. In case each case this will contradict the choice of $\mathbb{O}$.

Before proceeding further, let us observe the following. Consider a frame $\mathbb{F} = (X, Y)$ and let $w \in N(v)$ such that either $w \in Y$ or $w \in V(G) \setminus M$. In the first case, it follows from the definitions that $L_f^{\mathbb{F},i} \cup R_f^{\mathbb{F},i} \subseteq N(w)$. In the second case, as both v and w lie in the clique-path $\mathbb{P}$ and v is an unmarked vertex, at least one of $L_f^{\mathbb{F},i} \subseteq N(w)$ or $R_f^{\mathbb{F},i} \subseteq N(w)$ holds. Similarly, if $w \notin N(v)$ and $w \in Y$, then $(L_c^{\mathbb{F},i} \cup R_c^{\mathbb{F},i}) \cap N(w) = \emptyset$. Else if $w \notin N(v)$ and $w \in V(G) - M$, then at least one of $L_c^{\mathbb{F},i} \cap N(w) = \emptyset$ or $R_c^{\mathbb{F},i} \cap N(w) = \emptyset$ holds.

$\mathbb{O}$ is a cycle

Let us first note that $|V(\mathbb{O}) \cap B_i| \leq 2$. Let x, y be the neighbors of v in $\mathbb{O}$, and note that they lie in $M \cup \beta(\mathbb{P})$. Furthermore, let $\hat{M} = M \cap V(\mathbb{O})$, $M' \subseteq \hat{M}$ of size 3 such that $\hat{M} \cap \{x, y\} \subseteq M'$, and $\mathbb{F} = (M', M' \cap \{x, y\})$. Next, consider the sets $L_f = L_f^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$ and $R_f = R_f^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$. Since $|S| \leq k$, $v \notin H_i$, and B_i is a clique therefore, $L_f, R_f \neq \emptyset$. Let $z \in M' \setminus \{x, y\}$, which exists since $|M'| = 3$. Now suppose that there is $v^* \in L_f \cup R_f$ such that $(v^*, x), (v^*, y) \in E(G)$ then we claim that we can obtain a cycle on at least four vertices not containing v in $G \setminus S$. Since v^* fits $\mathbb{F}$, therefore $(v^*, z) \notin E(G)$. Consider the paths P_{xz} and P_{yz} from x to z and y to z in $\mathbb{O} - \{v\}$, respectively. Furthermore, let x^* and y^* be the last vertices in paths P_{xz} and P_{yz} which are adjacent to v^*. Note that x^* and y^* exists since $(x, v^*), (y, v^*) \in E(G)$. But then the path from x^* to y^* in $\mathbb{O}$ along with v^* forms an induced cycle on at least 4 vertices in $G \setminus S$ which does not contain v.

Hence, we assume that any vertex in $L_f \cup R_f$ is adjacent at most one of x, y. Moreover, by the construction of L_f and R_r we have that, either $L_f \subseteq N(x)$ and $R_f \subseteq N(y)$, or $R_f \subseteq N(x)$ and $L_f \subseteq N(y)$, must hold. Suppose that $L_f \subseteq N(x)$ and $R_f \subseteq N(y)$ (the other case is symmetric). Consider vertices $u^* \in L_f$ and $v^* \in R_f$. Note that $(u^*, x), (v^*, y), (u^*, v^*) \in E(G)$ and $(u^*, y), (v^*, x), (u^*, z), (v^*, z) \notin E(G)$. Consider the paths P_{xz} and P_{yz} from x to z and y to z in $\mathbb{O} - \{v\}$, respectively. Let x^* be the last vertex in the path P_{xz} such that $N(x^*) \cap \{u^*, v^*\} \neq \emptyset$. Similarly, let y^* be the last vertex in the path P_{yz} such that $N(y^*) \cap \{u^*, v^*\} \neq \emptyset$. Let P_{x^*z} and P_{zy^*} be the paths from x^* to z and z to y^* in $\mathbb{O} - \{v\}$, respectively. But then $G[V(P_{x^*z}) \cup V(P_{zy^*}) \cup \{u^*, v^*\}]$ contains a cycle on at least 4 vertices.

From now onwards we may assume that there are no chordless cycles in $G \setminus S$. Now we consider the case when $\mathcal{O}$ is an AW.

$\mathbb{O}$ is a †-AW

Let $\mathbb{O}$ comprise of the base path $\mathsf{base}(\mathbb{O}) = (b_1, b_2, \ldots, b_z)$, base terminals t_ℓ, t_r, shallow terminal t, and centre c. Furthermore, let $P(\mathbb{O}) = (t_\ell, b_1, b_2, \ldots, b_z, t_r)$, and let $b_0 = t_\ell$, and $b_{z+1} = t_r$. Let $\hat{M} = M \cap V(\mathbb{O})$, M' be a subset of $\hat{M}$ of size 8 such that $\hat{M} \cap \{c, t, t_\ell, t_r, b_1, b_2, b_{z-1}, b_z\} \subseteq M'$, and $\mathbb{F} = (M', M' \cap N(v))$. Next, we define the following sets, such that the vertices contained in them which will be used to either

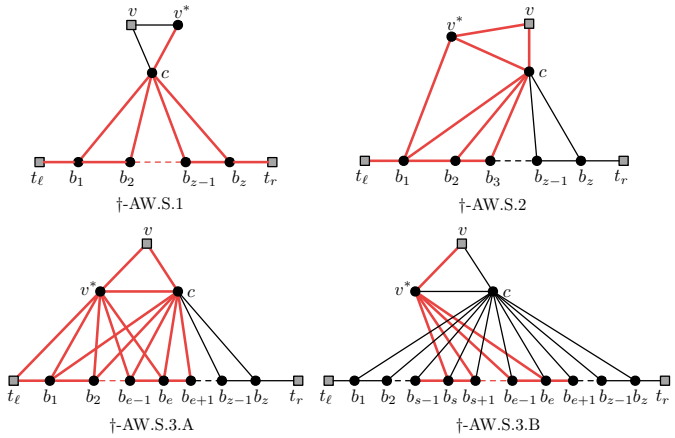

Figure 8.2: Construction of an obstruction when $\mathbb{O}$ is †-AW and $v = t$.

construct an obstruction not containing v, or an obstruction containing v but with (strictly) smaller size, or an obstruction with same number of vertices as $\mathbb{O}$ but containing strictly more vertices from B_i than $\mathbb{O}$. Let $L_f = L_f^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$, $L_c = L_c^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$, $R_f = R_f^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$, and $R_c = R_c^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$. Notice that $|V(\mathbb{O}) \cap B_i| \leq 4$, since no obstruction contains a clique of size 5 and $G[B_i]$ is a clique. This together with the fact that $v \notin H_i$ and $|S| \leq k$ implies that $L_f, L_c, R_f, R_c \neq \emptyset$. Next, we consider cases depending on the role that v plays in the obstruction $\mathbb{O}$.

Suppose v is the shallow terminal. In this case, $(v, c) \in E(G)$ therefore, $L_f \subseteq N(c)$ or $R_f \subseteq N(c)$ must hold. Consider the case when $L_f \subseteq N(c)$ (the other case is symmetric), and let v^* be a vertex in L_f. Next, we consider the following cases based on the neighborhood of v^* in $\mathbb{O}$. (see Figure 8.2).

Case †-AW.S.1. $|N(v^*) \cap V(P(\mathbb{O}))| = 0$. In this case, $G[(V(\mathbb{O}) \setminus \{v\}) \cup \{v^*\}]$ is a †-AW in $G' \setminus S$.

Case †-AW.S.2 If $|N(v^*) \cap V(P(\mathbb{O}))| = 1$. If $(v^*, t_\ell) \in E(G)$ then $G[\{v^*, c, t_\ell, b_1\}]$ is a C_4, which contradicts that $\mathbb{O}$ is the smallest obstruction in $G \setminus S$ containing v. Analogous argument can be given when $(v^*, t_r) \in E(G)$. Therefore, we assume that $N(v^*) \cap V(P(\mathbb{O})) = \{b_i\}$, where $i \in [z]$. If $i \in [z] \setminus \{1, z\}$ then $G[\{v^*, c, b_i, b_{i-1}, b_{i-2}, b_{i+1}, b_{i+2}\}]$ is a long claw in $G \setminus S$. If none of the above cases are applicable then $N(v^*) \cap V(P(\mathbb{O})) \in \{\{b_1\}, \{b_z\}\}$. Suppose that $N(v^*) \cap V(P(\mathbb{O})) = \{b_1\}$ (the other case is symmetric) then $G[\{c, v, v^*, b_1, b_2, b_3, t_\ell\}]$ is a whipping top in $G \setminus S$.

Case †-AW.S.3 $|N(v^*) \cap V(P(\mathbb{O}))| \geq 2$. If neighbors of v^* are not consecutive in the path $P(\mathbb{O})$ then we can obtain an induced cycle on at least 4 vertices in $G[\{v^*\} \cup V(P(\mathbb{O}))]$, therefore we assume that the neighbors of v^* in $P(\mathbb{O})$ are consecutive. By the construction of $\mathbb{F}$ and v^* we know that there are at least 9 vertices in $P(\mathbb{O})$ which

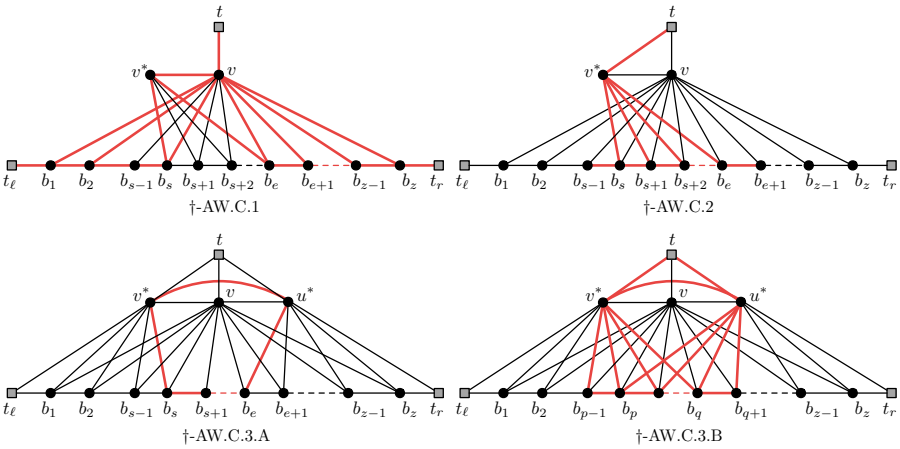

Figure 8.3: Construction of an obstruction when $\mathbb{O}$ is †-AW and $v = c$.

are non-adjacent to v^*. This also implies that $|\{t_\ell, t_r\} \cap N(v^*)| \leq 1$. Without loss of generality we assume that $(v^*, t_r) \notin E(G)$. Next, we consider the following cases based on whether or not $(v^*, t_\ell) \in E(G)$.

A) $(v^*, t_\ell) \in E(G)$. In this case, there exists $e \in [z-2]$ such that $b_e \in N(v^*)$ and $b_{e+1} \notin N(v^*)$. Let $V' = \{v, v^*, c, t_\ell\} \cup \{b_1, b_2, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a ‡-AW with $|V'| < |V(\mathbb{O})|$, a contradiction to the choice of $\mathbb{O}$.

B) $(v^*, t_\ell) \notin E(G)$. Let b_s and b_e be the first and last vertices in $P(\mathbb{O})$ which are adjacent to v^*, respectively. Notice that $s \neq e$ (since $|N(v^*) \cap V(P(\mathbb{O}))| \geq 2$), and $\{b_s, b_{s+1}, \ldots b_e, b_{e+1}\} \subset \{b_1, b_2, \ldots b_z\}$ (strict subset). Let $V' = \{v, v^*\} \cup \{b_{s-1}, b_s, b_{s+1}, \ldots, b_e, b_{e+1}\}$. Observe that $|V'| < |V(\mathbb{O})|$, and $G[V']$ is a †-AW.

Suppose v is the centre. In this case, $(t_\ell, v), (t_r, v) \notin E(G)$. Since $v \notin H_i$ and each vertex in $L_c \cup R_c$ fits the frame $\mathbb{F}$, one of the following holds. (1) $N(t_\ell) \cap L_c = \emptyset$ and $N(t_r) \cap R_c = \emptyset$; (2) $N(t_r) \cap L_c = \emptyset$ and $N(t_\ell) \cap R_c = \emptyset$; (3) $N(t_\ell) \cap L_c = \emptyset$ and $N(t_r) \cap L_c = \emptyset$; (4) $N(t_\ell) \cap R_c = \emptyset$ and $N(t_r) \cap R_c = \emptyset$. Consider a vertex $v^* \in L_c \cup R_c$, and let b_s and b_e be the first and the last vertices in the path $P(\mathbb{O})$ which are adjacent to v^*, respectively. The existence and distinctness of b_s, b_e follows from the fact that $|N(v^*) \cap V(\mathsf{base}(\mathbb{O}))| \geq 7$, which is implied from M being a 9-redundant modulator and v^* fitting the frame $\mathbb{F}$. The neighbors of v^* in $P(\mathbb{O})$ must be consecutive, otherwise we can obtain an induced cycle of length at least 4, which does not contain v. We further consider sub-cases based on whether or not the following two criterions are satisfied (see Figure 8.3).

1. $t \in N(v^*)$;

2. $N(v^*) \cap \{t_\ell, t_r\} = \emptyset$.

Case †-AW.C.1. $t \notin N(v^*)$. If $\{t_\ell, t_r\} \subseteq N(v^*)$ then $G[\{v^*, t_\ell, b_1, v, b_z, t_r, t\}]$ is a whipping top. Therefore, we can assume that $|\{t_\ell, t_r\} \subseteq N(v^*)| \leq 1$. Let $V' = (V(\mathbb{O}) \setminus \{b_{s+1}, b_{s+2}, \ldots, b_{e-1}\}) \cup \{v^*\}$. Notice that $|V'| < |V(\mathbb{O})|$ since $|N(v^*) \cap V(\mathsf{base}(\mathbb{O}))| \geq 7$ and neighbors of v^* are consecutive. Moreover, $G[V']$ is an (induced) †-AW or a net, which is of strictly smaller size than $\mathbb{O}$, contradicting the choice of $\mathbb{O}$. Here, we crucially rely on the fact that $|N(v^*) \cap \{t_\ell, t_r\}| \leq 1$.

Case †-AW.C.2. $t \in N(v^*)$ and $N(v^*) \cap \{t_\ell, t_r\} = \emptyset$. In this case, $G[\{v^*, t, b_{s-1}, b_s, b_{s+1}, \ldots, b_e, b_{e-1}\}]$ forms an (induced) †-AW in $G \setminus S$ which does not contain v.

If Cases †-AW.C.1 and †-AW.C.2 are not applicable then for each $u \in L_c \cup R_c$ we have $t \in N(u)$ and $N(u) \cap \{t_\ell, t_r\} \neq \emptyset$. Furthermore, $v \notin H_i$, $(t_\ell, v), (t_r, v) \notin E(G)$, each vertex in $L_c \cup R_c$ fits the frame $\mathbb{F}$. Therefore, one of the following must holds. 1) $N(t_\ell) \cap L_c = \emptyset$ and $N(t_r) \cap R_c = \emptyset$; 2) $N(t_r) \cap L_c = \emptyset$ and $N(t_\ell) \cap R_c = \emptyset$. Thus for each $u \in L_c \cup R_c$ we have $|N(u) \cap \{t_\ell, t_r\}| = 1$. We assume that $N(t_\ell) \cap L_c = \emptyset$ and $N(t_r) \cap R_c = \emptyset$ (the other case is symmetric). Next, we consider a vertex $u^* \in L_c$ and a vertex $v^* \in R_c$. Notice that (by the above discussion) $t \in N(u^*) \cap N(v^*)$, $t_\ell \notin N(u^*)$, $t_r \in N(u^*)$, $t_r \notin N(v^*)$, and $t_\ell \in N(v^*)$. Also, since $u^*, v^* \in B_i$ we have $(u^*, v^*) \in E(G)$. We now consider the remaining case.

Case †-AW.C.3. $t \in N(v_\ell) \cap N(v_r)$, $N(v_\ell) \cap \{t_\ell, t_r\} \neq \emptyset$, and $N(v_r) \cap \{t_\ell, t_r\} \neq \emptyset$. We consider the following sub-cases.

A) If u^* and v^* have no common neighbor in $P(\mathbb{O})$ then $G[\{u^*, v^*\} \cup V(P(\mathbb{O}))]$ contains an (induced) cycle on at least 4 vertices.

B) Otherwise, u^* and v^* have at least one common neighbor in $P(\mathbb{O})$. Let b_p and b_q be the first and the last common neighbors of u^* and v^* in $P(\mathbb{O})$, respectively. Notice that $b_{p-1} \in N(v^*)$ and $b_{p-1} \notin N(u^*)$. This follows from the fact that $t_\ell, b_q \in N(v^*)$, neighbors of v^* are consecutive vertices in $P(\mathbb{O})$, $t_\ell \notin N(u^*)$, and p is the first common neighbor of u^* and v^* in $P(\mathbb{O})$. Similarly, we can argue that $b_{q+1} \in N(u^*)$ and $b_{q+1} \notin N(v^*)$. Consider the set $V' = \{t, v^*, u^*\} \cup \{b_{p-1}, b_p, \ldots, b_q, b_{q+1}\}$. Notice that $G[V']$ is a ‡-AW or a tent which does not contain v.

Suppose v is one of the base terminals. We consider the case when $v = t_\ell$. By a symmetric argument we can handle the case when $v = t_r$. If $c \notin \beta(\mathbb{P})$ then for each $u \in L_c \cup R_c$ we have $(u, c) \notin E(G)$ as it fits the frame $\mathbb{F}$ and $N(u) \setminus (M \cup \beta(\mathbb{P})) = N(v) \setminus (M \cup \beta(\mathbb{P}))$. Otherwise $c \in \beta(\mathbb{P})$, and then at least one of $L_c \cap N(c) = \emptyset$ or $R_c \cap N(c) = \emptyset$ holds. Let $X_c \in \{L_c, R_c\}$ be a set such that $X_c \cap N(c) = \emptyset$. Similarly, if $b_1 \notin \beta(\mathbb{P})$ then for each $u \in L_f \cup R_f$ we have $(u, b_1) \in E(G)$ as it fits the frame $\mathbb{F}$ and $N(u) \setminus (M \cup \beta(\mathbb{P})) = N(v) \setminus (M \cup \beta(\mathbb{P}))$. Otherwise, $b_1 \in \beta(\mathbb{P})$, and then at least one of $L_f \subseteq N(b_1)$ or $R_f \subseteq N(b_1)$ holds. Let $Y_f \in \{L_f, R_f\}$ be a set such that $Y_f \subseteq N(b_1)$. Next, we consider cases based on whether or not $b_1 \in B_i$ (see Figure 8.4).

Case †-AW.T.1. $b_1 \in B_i$. Consider a vertex $u^* \in X_c$. Note that $(u^*, b_1) \in E(G)$ since $b_1 \in B_i$, and $(u^*, c) \notin E(G)$ by the choice of u^*. Also, $(u^*, t) \notin E(G)$ otherwise, $G[\{t, c, b_1, u^*\}]$ is cycle on 4 vertices in $G \setminus S$. Recall that u^* fits the frame $\mathbb{F}$ (and

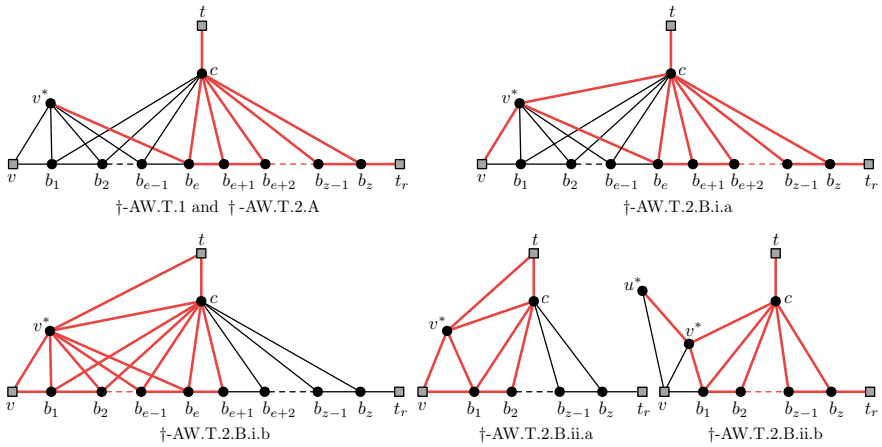

Figure 8.4: Construction of an obstruction when $\mathbb{O}$ is †-AW and $v = t_\ell$.

$(b_1, u^*) \in E(G)$), therefore there exists b_e such that $b_e \in N(u^*)$ and $b_{e+1} \notin N(u^*)$, where $e \in [z]$ (possibly $e = 1$). This together with the fact that neighbors of u^* in $P(\mathbb{O})$ are consecutive (otherwise, we obtain an induced cycle on at least 4 vertices not containing v) implies that $(u^*, t_r) \notin E(G)$. But then $G[\{t, c, u^*\} \cup \{b_e, b_{e+1}, \ldots, b_z, t_r\}]$ is a †-AW (or a net) which does not contain v.

Case †-AW.T.2. $b_1 \notin B_i$. Consider a vertex $v^* \in Y_f \cup \{u \in X_c \mid (u, b_1) \in E(G)\}$, and the following cases based on its neighborhood in $\mathbb{O}$.

A) $(v^*, c) \notin E(G)$. In this case, $(v^*, t) \notin E(G)$, otherwise $G[\{v^*, t, c, b_1\}]$ is a cycle on 4 vertices. Recall that v^* fits the frame $\mathbb{F}$, therefore there are at least 7 vertices in $P(\mathbb{O})$ which are non-adjacent to v^*. This together with the fact that $(b_1, v^*) \in E(G)$ implies that there exists $e \in [z-1]$ such that $b_e \in N(v^*)$ and $b_{e+1} \notin N(v^*)$. But then $G[V']$ is a †-AW not containing v in $G \setminus S$, where $V' = \{t, c, v^*, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$.

B) $(v^*, c) \in E(G)$. We further consider the following cases.

i) There exists $e \in [z] \setminus \{1\}$ such that $(v^*, b_e) \in N(v^*)$ and $(v^*, b_{e+1}) \notin N(v^*)$. Observe that since M is a 9-redundant modulator and v^* fits the frame $\mathbb{F}$, therefore $e < z$. Consider the following cases based on whether or not $(t, v^*) \in E(G)$.

a) $(t, v^*) \notin E(G)$. Let $V' = \{t, c, v^*, v, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a †-AW in $G \setminus S$. Furthermore, either $|V'| < |V(\mathbb{O})|$ or $|V'| = |V(\mathbb{O})|$ and $|V' \cap B_i| > |V(\mathbb{O}) \cap B_i|$. Here, we rely on the fact that $b_1 \notin B_i$. In either case we obtain a contradiction to the choice of $\mathbb{O}$.

b) $(t, v^*) \in E(G)$. Let $V' = \{t, c, v^*, v\} \cup \{b_1, b_2, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$ and $|V'| < |V(\mathbb{O})|$, which contradicts the choice of $\mathbb{O}$.

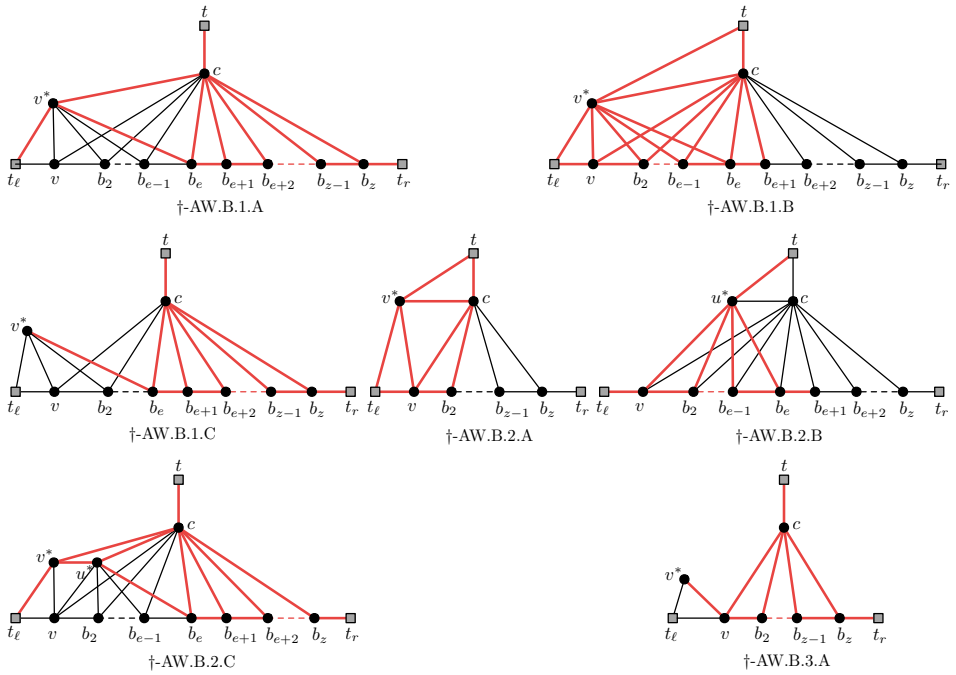

Figure 8.5: Construction of an obstruction when $\mathbb{O}$ is †-AW and $v = b_1$.

ii) Otherwise, if i) does not hold then the only neighbors of v^* in $P(\mathbb{O})$ are b_1 and v. Consider the following cases based on whether or not $(t, v^*) \in E(G)$.

a) $(t, v^*) \in E(G)$. In this case, $G[\{v, v^*, t, c, b_1, b_2\}]$ is a tent.

b) $(t, v^*) \notin E(G)$. We consider a vertex in $u^* \in X_c$ to obtain the desired obstruction. Notice that $(b_1, u^*) \notin E(G)$ as Case †-AW.T.2.A is not applicable. Furthermore, $(b_j, u^*) \notin E(G)$, for each $j \in [z] \setminus \{1\}$, otherwise $G[\{v, u^*\} \cup \{b_1, b_2, \ldots b_j\}]$ will contain an induced cycle on at least 4 vertices, which is an obstruction containing v with strictly less number of vertices than $\mathbb{O}$. Let $V' = (V(\mathbb{O}) \setminus \{v\}) \cup \{v^*, u^*\}$. Observe that $G[V']$ is a †-AW which does not contain v.

Suppose v is either b_1 or b_z. Suppose $v = b_1$ (the other case is symmetric). If $t_\ell \notin \beta(\mathbb{P})$ then for each $u \in L_f \cup R_f$ we have $(u, t_\ell) \in E(G)$ as it fits the frame $\mathbb{F}$ and $N(u) \setminus (M \cup \beta(\mathbb{P})) = N(v) \setminus (M \cup \beta(\mathbb{P}))$. Otherwise, $t_\ell \in \beta(\mathbb{P})$, and then at least one of $L_f \subseteq N(t_\ell)$ or $R_f \subseteq N(t_\ell)$ holds. Let $X_f \in \{L_f, R_f\}$ be a set such that $X_f \subseteq N(t_\ell)$. Similarly, if $b_2 \notin \beta(\mathbb{P})$ then for each $u \in L_f \cup R_f$ we have $(u, b_2) \in E(G)$ as it fits the frame $\mathbb{F}$ and $N(u) \setminus (M \cup \beta(\mathbb{P})) = N(v) \setminus (M \cup \beta(\mathbb{P}))$. Otherwise, $b_2 \in \beta(\mathbb{P})$, and then at least one of $L_f \subseteq N(b_2)$ or $R_f \subseteq N(b_2)$ holds. Let $Y_f \in \{L_f, R_f\}$ be a set such that $Y_f \subseteq N(b_2)$. Next, we consider cases depending on the neighborhood of vertices in $X_f \cup Y_f$ in $\mathbb{O}$ (see Figure 8.5).

Case †-AW.B.1. There is a vertex $v^* \in X_f \cup Y_f$ such that $\{t_\ell, b_2\} \subseteq N(v^*)$. There exists $e \leq z-2$ such that $b_e \in N(v^*)$ and $b_{e+1} \notin N(v^*)$. This follows from the fact that $(v^*, b_2) \in E(G)$ and v^* fits the frame $\mathbb{F}$. Next, we consider the sub-cases based on whether or not $(v^*, c), (v^*, t) \in E(G)$.

A) $(v^*, c) \in E(G), (v^*, t) \notin E(G)$. Let $V' = \{t, c, v^*, t_\ell, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a †-AW which does not contain v.

B) $(v^*, c) \in E(G), (v^*, t) \in E(G)$. Let $V' = \{t, c, v^*, v, t_\ell\} \cup \{b_2, b_3, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a ‡-AW which has strictly fewer vertices than $\mathbb{O}$.

C) $(v^*, c) \notin E(G)$. Notice that in this case $(v^*, t) \notin E(G)$, otherwise $G[\{v^*, t, c, b_2\}]$ is an induced cycle on 4 vertices. Let $V' = \{t, c, v^*, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is an induced †-AW which does not contain v.

Case †-AW.B.2. Suppose that for every $u \in X_f \cup Y_f$ we have $(u, c) \in E(G)$. Since Case †-AW.B.1 is not applicable, we can assume that for each $u \in X_f \cup Y_f$ we have $\{t_\ell, b_2\} \not\subseteq N(u)$. By the construction of X_f and Y_f we know that for each $u \in X_f \cup Y_f$ we have $\{t_\ell, b_2\} \cap N(u) \neq \emptyset$, and $X_f, Y_f \neq \emptyset$. Consider a vertex $v^* \in X_f$ and a vertex $u^* \in Y_f$. We have that $(v^*, c), (u^*, c), (v^*, t_\ell), (u^*, b_2) \in E(G)$ and $(v^*, b_2), (u^*, t_\ell) \notin E(G)$. Next, we consider cases based on whether or not t adjacent to v^* and u^*.

A) $(t, v^*) \in E(G)$. Recall that $b_2 \notin N(v^*)$ and $t_\ell, t, c \in N(v^*)$. But then $G[\{c, v, v^*, b_2, t_\ell, t\}]$ is a tent in $G \setminus S$.

B) $(t, u^*) \in E(G)$. There exists $e \in [z-2]$ such that $b_e \in N(u^*)$ and $b_{e+1} \notin N(u^*)$. This follows from the fact that $(u^*, b_2) \in E(G)$ and u^* fits the frame $\mathbb{F}$. Let $V' = \{b_2, b_3, \ldots, b_e, b_{e+1}\} \cup \{t, u^*, t_\ell, v\}$. Then $G[V']$ is a †-AW in $G \setminus S$ which has strictly fewer vertices than $\mathbb{O}$.

C) $(t, v^*), (t, u^*) \notin E(G)$. We start by arguing that v^* cannot be adjacent to b_j, where $j \in [z] \setminus \{1\}$. For $j = 2$ it follows from the choice of v^*. Next consider the smallest $j > 2$ such that $(v^*, b_j) \in E(G)$. Then $G[\{v, v^*\} \cup \{b_2, b_3, \ldots, b_j\}]$ is an induced cycle on at least 4 vertices, which has strictly less number of vertices than $\mathbb{O}$. Therefore, we assume that the only neighbor of v^* in $P(\mathbb{O})$ are v and t_ℓ. Next, we argue about neighbors of u^* in $P(\mathbb{O})$. There exists $e \in [z-2]$ such that $b_e \in N(u^*)$ and $b_{e+1} \notin N(u^*)$. This follows from the fact that $(u^*, b_2) \in E(G)$ and u^* fits the frame $\mathbb{F}$. Let $V' = \{t, c, t_\ell, t_r, v^*, u^*\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a †-AW in $G \setminus S$ which does not contain v.

Case †-AW.B.3. Suppose that there is $u \in X_f \cup Y_f$ such that $(u, c) \notin E(G)$, and for all $u \in X_f \cup Y_f$ we have $\{t_\ell, b_2\} \not\subseteq N(u)$. Consider vertices $v^* \in X_f$ and $u^* \in Y_f$, and the following sub-cases.

A) Consider the case when $(v^*, c) \notin E(G)$. This implies that $(v^*, t) \notin E(G)$, otherwise $G[v^*, c, t, v]$ is a cycle on 4 vertices. Notice that $t_\ell \notin B_i$, which follows from the fact that $Y_f \neq \emptyset$ as for each $u \in Y_f$ we have $(u, b_2) \in E(G)$ and therefore $(u, t_\ell) \notin E(G)$. Also, v^* is not adjacent to any b_j, where $j \geq 2$, since the neighbors

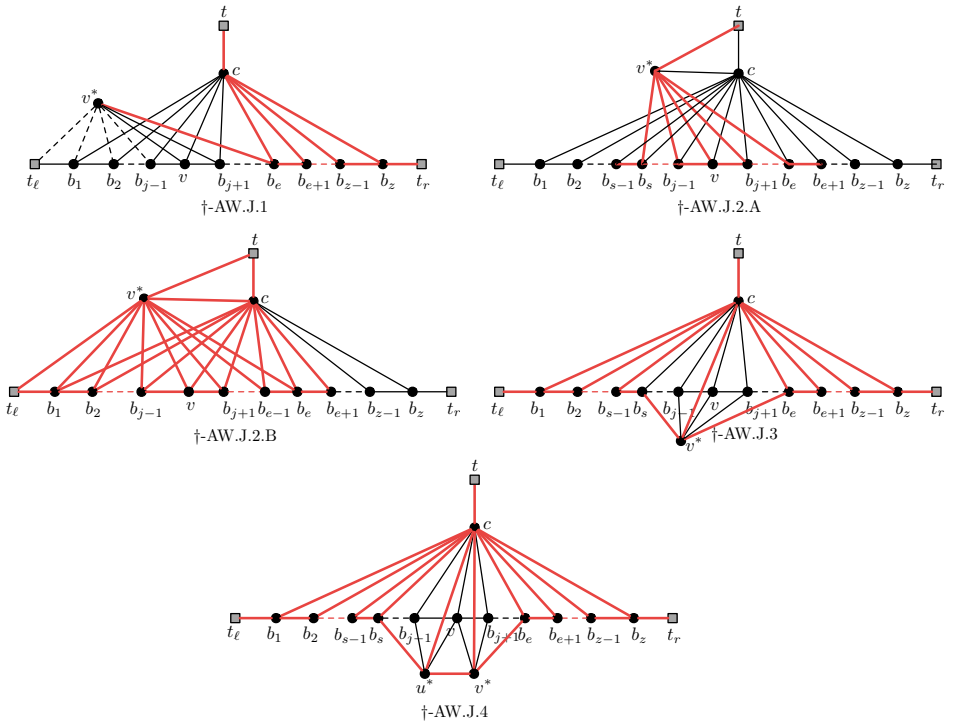

Figure 8.6: Construction of an obstruction when $\mathbb{O}$ is †-AW and $v = b_j$, where $j \in [z-1] \setminus \{1\}$.

of v^* in $P(\mathbb{O})$ must be consecutive, $(v^*, t_\ell) \in E(G)$, and $(v^*, b_2) \notin E(G)$. But then $G[(V(\mathbb{O}) \setminus \{t_\ell\}) \cup \{v^*\}]$ is a †-AW with same number of vertices as $\mathbb{O}$ but with more vertices from B_i.

B) Consider the case when $(u^*, c) \notin E(G)$. Since Case †-AW.B.3.A is not applicable we can assume that $(v^*, c) \in E(G)$. Observe that $G[\{c, v^*, u^*, b_2\}]$ is a cycle on 4 vertices. Here, we rely on the fact that $(v^*, b_2) \notin E(G)$.

Suppose that v is a base vertex b_j, where $j \in [z] \setminus \{1, z\}$. Let $X_f \in \{L_f, R_f\}$ be a set such that $X_f \subseteq N(b_{j-1})$ and $Y_f \in \{L_f, R_f\}$ be a set such that $Y_f \subseteq N(b_{j+1})$. Next, we consider cases based on neighborhood in $\mathbb{O}$ of vertices in sets X_f and Y_f (see Figure 8.6).

Case †-AW.J.1. If there is $v^* \in X_f \cup Y_f$ such that $(v^*, c) \notin E(G)$. Note that, as $(v^*, c) \notin E(G)$, we have $(v^*, t) \notin E(G)$, otherwise $G[\{v, v^*, c, t\}]$ is a cycle on 4 vertices. Notice that all the neighbors of v^* on $P(\mathbb{O})$ must be consecutive, and one of (a) $\{t_\ell, b_1\} \cap N(v^*) = \emptyset$ or (b) $\{t_r, b_z\} \cap N(v^*) = \emptyset$ must hold. Suppose that $\{t_r, b_z\} \cap N(v^*) = \emptyset$ (the other case is symmetric). Let $e \in [z-1]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $t_r, b_z \notin N(v^*)$. We

note that e could possibly be equal to j. Let $V' = \{t, c, v^*, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $|V'| < |V(\mathbb{O})|$ since $j \in [z] \setminus \{1, z\}$. Moreover, $G[V']$ is a †-AW in $G \setminus S$, which contradicts the choice of $\mathbb{O}$.

Note that if Case †-AW.J.1 is not applicable then for each $u \in X_f \cup Y_f$ we have $(u, c) \in E(G)$. Next, we consider cases based on whether or not the following conditions are satisfied for a vertex $u \in X_f \cup Y_f$.

1. $(u, t) \in E(G)$;
2. $\{b_{j-1}, b_{j+1}\} \subseteq N(u)$.

Case †-AW.J.2 If there is $v^* \in X_f \cup Y_f$ such that $(v^*, t) \in E(G)$. Since M is a 9-redundant modulator, one of $\{b_1, b_2, t_\ell\} \cap N(v^*) = \emptyset$ or $\{b_{z-1}, b_z, t_r\} \cap N(v^*) = \emptyset$ must hold. Suppose $\{b_{z-1}, b_z, t_r\} \cap N(v^*) = \emptyset$ holds (the other case is symmetric). We further consider the following sub-cases based on whether or not $t_\ell \in N(v^*)$.

A) $t_\ell \notin N(v^*)$. Let $s \in [j]$ such that b_s is the first vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $(t_\ell, v^*) \notin E(G)$ and $(v^*, v) \in E(G)$. Also, let $e \in [z-2]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $(t_r, v^*), (b_z, v^*), (b_{z-1}, v^*) \notin E(G)$ and $(v^*, v) \in E(G)$. Notice that $s \neq e$ since by the construction of the sets X_f and Y_f we have that v^* is incident to v and at least one of the vertices in $\{b_{j-1}, b_{j+1}\}$. Let $V' = \{t, v^*\} \cup \{b_{s-1}, b_s, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a †-AW in $G \setminus S$. Moreover, $|V'| < |V(\mathbb{O})|$ since $t_r, c, b_z \notin V'$ and $V' \subseteq V(\mathbb{O}) \cup \{v^*\}$.

B) $t_\ell \in N(v^*)$. Let $e \in [z-2]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $(t_r, v^*), (b_z, v^*), (b_{z-1}, v^*) \notin E(G)$ and $(v^*, v) \in E(G)$. Let $V' = \{t, v^*, c, t_\ell\} \cup \{b_1, b_2, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$. Moreover, $|V'| < |V(\mathbb{O})|$ since $t_r, b_z \notin V'$ and $V' \subseteq V(\mathbb{O}) \cup \{v^*\}$.

Case †-AW.J.3. If there is $v^* \in X_f \cup Y_f$ such that $(v^*, t) \notin E(G)$ and $\{b_{j-1}, b_{j+1}\} \subseteq N(v^*)$. Notice that all the neighbors of v^* on $P(\mathbb{O})$ must be consecutive, and there exists at least three vertices on $P(\mathbb{O})$ that are non-adjacent to v^*, which follows from M being a 9-redundant modulator. Therefore, one of $\{t_\ell, b_1\} \cap N(v^*) = \emptyset$ or $\{t_r, b_z\} \cap N(v^*) = \emptyset$ must holds. Suppose that $\{t_r, b_z\} \cap N(v^*) = \emptyset$ (other case is symmetric). Let $e \in [z-1]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $t_r, b_z \notin N(v^*)$. Also, let $s \in [z-1] \cup \{0\}$ be the lowest integer such that $(v^*, b_s) \in E(G)$ (b_s could possibly be same as b_{j-1} or $b_0 = t_\ell$). Let $V' = \{t, c, v^*, t_\ell, t_r\} \cup \{b_1, b_2, \ldots, b_s\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $|V'| < |V(\mathbb{O})|$ since $j \in [z-1] \setminus \{1\}$. Moreover, $G[V']$ is an induced †-AW in $G \setminus S$, which does not contain v.

Case †-AW.J.4 For all $u \in X_f \cup Y_f$ we have $(v^*, t) \notin E(G)$, and $\{b_{j-1}, b_{j+1}\} \not\subseteq N(v^*)$. Since Cases †-AW.J.1, †-AW.J.2, and †-AW.J.3 are not applicable then (together with the construction of X_f and Y_f) for each $u \in X_f \cup Y_f$ we have $(u, c) \in E(G)$, $(u, t) \notin E(G)$, and $|N(u) \cap \{b_{j-1}, b_{j+1}\}| = 1$. Next, consider a vertex $u^* \in X_f$ and $v^* \in Y_f$. Let $s \in [j-1] \cup \{0\}$ such that b_s is the first vertex in $P(\mathbb{O})$ which is adjacent to u^*, which

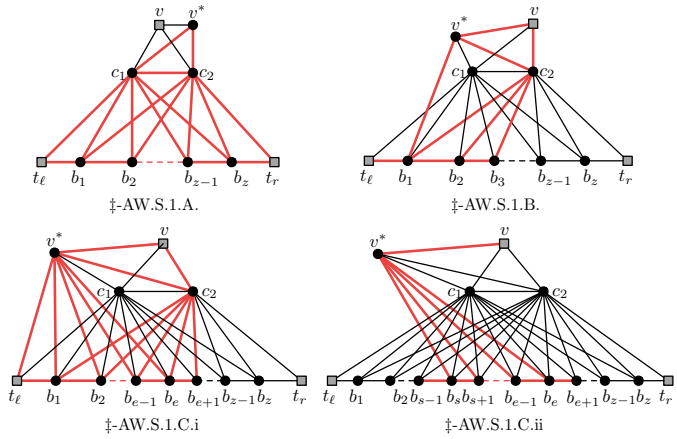

Figure 8.7: Construction of an obstruction when $\mathbb{O}$ is ‡-AW and $v = t$.

exists since $(u^*, b_{j-1}) \in E(G)$. Also, let $e \in [z-1]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $(t_r, v^*), (b_z, v^*) \notin E(G)$ and $(v^*, b_{j+1}) \in E(G)$. Notice that $s \neq e$. Let $V' = \{t, c, v^*, u^*\} \cup \{t_\ell, b_1, b_2, b_{s-1}, b_s\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a †-AW in $G \setminus S$ which does not contain v.

$\mathbb{O}$ is a ‡-AW

Let $\mathbb{O}$ comprise of the base path $\mathsf{base}(\mathbb{O}) = (b_1, b_2, \ldots, b_z)$ with terminals t_ℓ, t_r, shallow terminal t, and centres c_1, c_2. Furthermore, let $P(\mathbb{O}) = (t_\ell, b_1, b_2, \ldots, b_z, t_r)$, $b_0 = t_\ell$, and $b_{z+1} = t_r$. Let $\hat{M} = M \cap V(\mathbb{O})$, M' be a subset of $\hat{M}$ of size 9 such that $\hat{M} \cap \{c_1, c_2, t, t_\ell, t_r, b_1, b_2, b_{z-1}, b_z\} \subseteq M'$, and $\mathbb{F} = (M', M' \cap N(v))$. Next, we define sets, the vertices from which will be used to either construct an obstruction not containing v, an obstruction containing v but with (strictly) smaller size, or and an obstruction with same number of vertices as $\mathbb{O}$ but containing more vertices from B_i. Let $L_f = L_f^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$, $L_c = L_c^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$, $R_f = R_f^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$, and $R_c = R_c^{\mathbb{F},i} \setminus (S \cup V(\mathbb{O}))$. Notice that $|V(\mathbb{O}) \cap B_i| \leq 4$, since no obstruction contains a clique of size 5 and $G[B_i]$ is a clique. This together with the fact that $v \notin H_i$ and $|S| \leq k$ implies that $L_f, L_c, R_f, R_c \neq \emptyset$. Next, we consider cases depending on the role that v plays in $\mathbb{O}$.

Suppose that v is the shallow terminal. For a vertex $u \in L_f \cup R_f$ we have $\{c_1, c_2\} \cap N(u) \neq \emptyset$. This follows from the fact that $v \notin H_i$ and $(v, c_1), (v, c_2) \in E(G)$. Next, consider the following cases depending on the neighborhood in $\mathbb{O}$ of vertices in $L_f \cup R_f$ (see Figure 8.7).

Case ‡-AW.S.1. There is $v^* \in L_f \cup R_f$ such that $c_1, c_2 \in N(v^*)$. We further consider sub-cases based on other neighbors (if any) of v^* in $\mathbb{O}$.

A) $|N(v^*) \cap V(P(\mathbb{O}))| = 0$. In this case, $G[(V(\mathbb{O}) \setminus \{v\}) \cup \{v^*\}]$ is a ‡-AW in $G \setminus S$.

B) If $|N(v^*) \cap V(P(\mathbb{O}))| = 1$. If $(v^*, t_\ell) \in E(G)$ then $G[\{v^*, c_2, t_\ell, b_1\}]$ is a cycle on 4 vertices. Analogous argument can be given when $(v^*, t_r) \in E(G)$. Therefore, we assume that $N(v^*) \cap V(P(\mathbb{O})) = \{b_i\}$, where $i \in [z]$. If $i \in [z] \setminus \{1, z\}$ then $G[\{v^*, v, b_i, b_{i-1}, b_{i-2}, b_{i+1}, b_{i+2}\}]$ is a long claw in $G \setminus S$. If none of the above cases are applicable then $N(v^*) \cap V(P(\mathbb{O}))$ is $\{b_1\}$ or $\{b_z\}$. Suppose that $N(v^*) \cap V(P(\mathbb{O})) = \{b_1\}$ (the other case is symmetric) then $G[\{c_2, v, v^*, b_1, b_2, b_3, t_\ell\}]$ is a whipping top in $G \setminus S$.

C) $|N(v^*) \cap V(P(\mathbb{O}))| \geq 2$. If neighbors of v^* are not consecutive in the path $P(\mathbb{O})$ then we can obtain an induced cycle on at least 4 vertices in $G[\{v^*\} \cup V(P(\mathbb{O}))]$, therefore we assume that the neighbors of v^* in $P(\mathbb{O})$ are consecutive. By the construction of $\mathbb{F}$ and v^* we know that there are at least 3 vertices in $P(\mathbb{O})$ which are non-adjacent to v^*. This also implies that $|\{t_\ell, t_r\} \cap N(v^*)| \leq 1$. Without loss of generality we assume that $(v^*, t_r) \notin E(G)$. Next, we consider the following cases based on whether or not $(v^*, t_\ell) \in E(G)$.

i) $(v^*, t_\ell) \in E(G)$. In this case, there exists $e \in [z-2]$ such that $b_e \in N(v^*)$ and $b_{e+1} \notin N(v^*)$. Let $V' = \{v, v^*, c_2, t_\ell\} \cup \{b_1, b_2, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a ‡-AW with $|V'| < |V(\mathbb{O})|$.

ii) $(v^*, t_\ell) \notin E(G)$. Let b_s and b_e be the first and the last vertex in $P(\mathbb{O})$ which are adjacent to v^*, respectively. Notice that $s \neq e$ (since $|N(v^*) \cap V(P(\mathbb{O}))| \geq 2$), and $\{b_s, b_{s+1}, \cdots b_e, b_{e+1}\} \subset \{b_1, b_2, \ldots b_z\}$. Let $V' = \{v, v^*\} \cup \{b_{s-1}, b_s, b_{s+1}, \ldots, b_e, b_{e+1}\}$. Observe that $|V'| < |V(\mathbb{O})|$, and $G[V']$ is a †-AW.

Case ‡-AW.S.2. For all $u \in L_f \cup R_f$ we have $|\{c_1, c_2\} \cap N(v^*)| = 1$. Observe that either $L_f \subseteq N(c_1)$ and $R_f \subseteq N(c_2)$ or $R_f \subseteq N(c_1)$ and $L_f \subseteq N(c_2)$ holds. Suppose $L_f \subseteq N(c_1)$ and $R_f \subseteq N(c_2)$ (the other case is symmetric). Next, consider a vertex $u^* \in L_f$ and a vertex $v^* \in R_f$. Since Case ‡-AW.S.1 is not applicable we have $(u^*, c_1), (v^*, c_2) \in E(G)$ and $(u^*, c_2), (v^*, c_1) \notin E(G)$. Moreover, $u^*, v^* \in B_i$ therefore, $(u^*, v^*) \in E(G)$. But then $G[\{u^*, v^*, c_1, c_2\}]$ is a cycle on 4 vertices.

Suppose v is one of the centres. Suppose $v = c_1$ (the other case is symmetric). Since $v \notin H_i$, $(t_r, v) \notin E(G)$, and each vertex in $L_c \cup R_c$ fits the frame $\mathbb{F}$, one of $N(t_r) \cap L_c = \emptyset$ or $N(t_r) \cap R_c = \emptyset$ must hold. Consider the case when $N(t_r) \cap L_c = \emptyset$ (the other case is symmetric), and a vertex $v^* \in L_c$. Let b_s and b_e be the first and the last vertex in the path $P(\mathbb{O})$ which are adjacent to v^*, respectively. The existence and distinctness of b_s, b_e follows from the fact that $|N(v^*) \cap V(\mathsf{base}(\mathbb{O}))| \geq 4$, which is implied by M being a 9-redundant modulator and v^* fitting the frame $\mathbb{F}$. Also, $e \leq z$ since $(v^*, t_r) \notin E(G)$. The neighbors of v^* in $P(\mathbb{O})$ must be consecutive, otherwise we can obtain an induced cycle of length at least 4 which does not contain v. We further consider sub-cases based on whether or not $t, c_2 \in N(v^*)$ (see Figure 8.8).

Case ‡-AW.C.1. $t, c_2 \notin N(v^*)$. Let $V' = \{v^*, v, c_2, t, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Notice that $|V'| < |V(\mathbb{O})|$ since $|N(v^*) \cap V(\mathsf{base}(\mathbb{O}))| \geq 4|$ and neighbors of v^* are consecutive. Moreover, $G[V']$ is a ‡-AW or a tent, which is of strictly smaller size than $\mathbb{O}$, contradicting the choice of $\mathbb{O}$. Here, we crucially rely on the fact that $t_r \notin N(v^*)$.

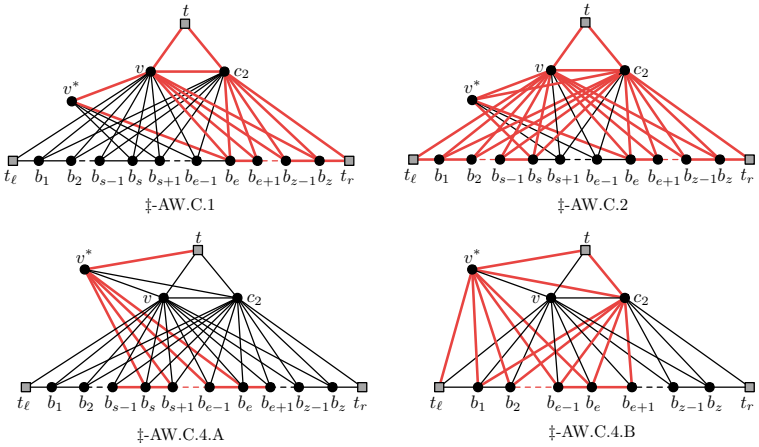

Figure 8.8: Construction of an obstruction when $\mathbb{O}$ is ‡-AW and $v = c_1$.

Case ‡-AW.C.2. $t \notin N(v^*)$ and $c_2 \in N(v^*)$. Let $V' = (V(\mathbb{O}) \setminus \{b_{s+1}, b_{s+2}, \ldots, b_{e-2}, b_{e-1}\}) \cup \{v^*\}$. Notice that $|V'| < |V(\mathbb{O})|$, and $G[V']$ is a ‡-AW.

Case ‡-AW.C.3. $t \in N(v^*)$ and $c_2 \notin N(v^*)$. Recall that $N(v^*) \cap \{b_1, b_2, \ldots, b_z\} \neq \emptyset$. Consider a vertex $b_j \in N(v^*) \cap \{b_1, b_2, \ldots, b_z\}$. The graph $G[\{v^*, t, c_2, b_j\}]$ is a cycle on 4 vertices.

Case ‡-AW.C.4. $t \in N(v^*)$ and $c_2 \in N(v^*)$. In this case we consider the following sub-cases based on whether or not $(t_\ell, v^*) \in E(G)$.

A) $(t_\ell, v^*) \notin E(G)$. Let $V' = \{t, v^*, t_\ell\} \cup \{b_{s-1}, b_s, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a †-AW in $G \setminus S$ which does not contain v.

B) $(t_\ell, v^*) \in E(G)$. Let $V' = \{t, v^*, c_2, t_\ell\} \cup \{b_1, b_2, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$ which does not contain v.

Suppose v is one of the base terminals. We consider the case when $v = t_\ell$. By a symmetric argument we can handle the case when $v = t_r$. If $c_2 \notin \beta(\mathbb{P})$ then for each $u \in L_c \cup R_c$ we have $(u, c_2) \notin E(G)$ as it fits the frame $\mathbb{F}$ and $N(u) \setminus (M \cup \beta(\mathbb{P})) = N(v) \setminus (M \cup \beta(\mathbb{P}))$. Otherwise, $c_2 \in \beta(\mathbb{P})$, and then at least one of $L_c \cap N(c_2) = \emptyset$ or $R_c \cap N(c_2) = \emptyset$ holds. Let $X_c \in \{L_c, R_c\}$ be a set such that $X_c \cap N(c_2) = \emptyset$. Similarly, if $b_1 \notin \beta(\mathbb{P})$ then for each $u \in L_f \cup R_f$ we have $(u, b_1) \in E(G)$ as it fits the frame $\mathbb{F}$ and $N(u) \setminus (M \cup \beta(\mathbb{P})) = N(v) \setminus (M \cup \beta(\mathbb{P}))$. Otherwise, $b_1 \in \beta(\mathbb{P})$, and then at least one of $L_f \subseteq N(b_1)$ or $R_f \subseteq N(b_1)$ holds. Let $Y_f \in \{L_f, R_f\}$ be a set such that $Y_f \subseteq N(b_1)$. Next, we consider cases based on whether or not $b_1 \in B_i$ (see Figure 8.9).

Case ‡-AW.T.1. $b_1 \in B_i$. Consider a vertex $v^* \in X_c$. Note that $(b_1, v^*) \in E(G)$ since $b_1 \in B_i$, and $(v^*, c_2) \notin E(G)$, by the choice of v^*. Also, $(v^*, t) \notin E(G)$ otherwise, $G[\{t, c_2, b_1, v^*\}]$ is a cycle on 4 vertices in $G \setminus S$. Recall that v^* fits the frame $\mathbb{F}$ (and

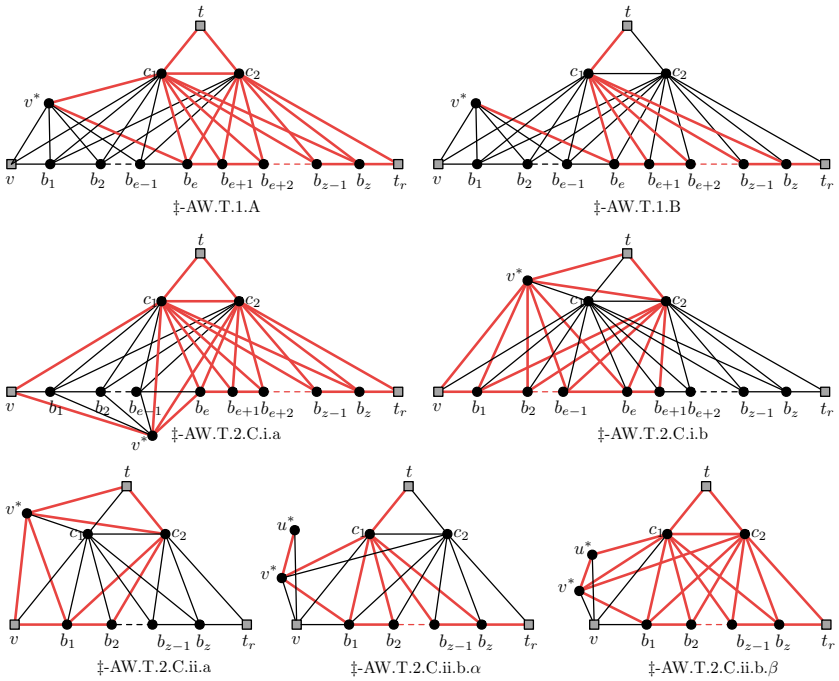

Figure 8.9: Construction of an obstruction when $\mathbb{O}$ is ‡-AW and $v = t_\ell$.

$(b_1, v^*) \in E(G)$), therefore there exists $e \in [z-2]$ such that $b_e \in N(v^*)$ and $b_{e+1} \notin N(v^*)$. This together with the fact that neighbors of v^* in $P(\mathbb{O})$ are consecutive (otherwise, we obtain an induced cycle on at least 4 vertices not containing v) implies that $(v^*, t_r) \notin E(G)$. Next, we consider cases based on whether or not $(v^*, c_1) \in E(G)$.

A) $(v^*, c_1) \in E(G)$. Let $V' = \{t, c_1, c_2, v^*, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$ not containing v.

B) $(v^*, c_1) \notin E(G)$. Let $V' = \{t, c_1, v^*, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a †-AW in $G \setminus S$ not containing v.

Case ‡-AW.T.2. $b_1 \notin B_i$. Consider a vertex $v^* \in Y_f \cup \{u \in X_c \mid (u, b_1) \in E(G)\}$, and the following cases based on its neighborhood in $\mathbb{O}$.

A) $(v^*, c_2) \notin E(G)$. Notice that this case is the same as Case ‡-AW.T.1, therefore we can obtain a desired obstruction in the same manner.

B) $(v^*, c_1) \notin E(G)$. Observe that $(v^*, t) \notin E(G)$, otherwise $G[\{v^*, b_1, c_1, t\}]$ is a cycle on 4 vertices in $G \setminus S$. Now, we can obtain an obstruction as in the Case ‡-AW.T.1.B.

C) $(v^*, c_1), (v^*, c_2) \in E(G)$. We further consider the following cases based on neighborhood of v^* in $P(\mathbb{O})$.

i) There exists $e \in [z] \setminus \{1\}$ such that $(v^*, b_e) \in N(v^*)$ and $(v^*, b_{e+1}) \notin N(v^*)$. Observe that since M is a 9-redundant modulator and v^* fits the frame $\mathbb{F}$, therefore $e < z - 1$. Consider the following cases based on whether or not $(t, v^*) \in E(G)$.

a) $(t, v^*) \notin E(G)$. Let $V' = \{t, c_1, c_2, v^*, v, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$. Furthermore, either $|V'| < |V(\mathbb{O})|$ or $|V'| = |V(\mathbb{O})|$ and $|V' \cap B_i| > |V(\mathbb{O}) \cap B_i|$. Here, we rely on the fact that $b_1 \notin B_i$. In either case we obtain a contradiction to the choice of $\mathbb{O}$.

b) $(t, v^*) \in E(G)$. Let $V' = \{t, v^*, c_2, v\} \cup \{b_1, b_2, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$ and $|V'| < |V(\mathbb{O})|$.

ii) If i) does not hold then the only neighbors of v^* in $P(\mathbb{O})$ are b_1 and v. Consider the following cases based on whether or not $(t, v^*) \in E(G)$.

a) $(t, v^*) \in E(G)$. In this case, $G[\{v, v^*, t, c_2, b_1, b_2\}]$ is a tent.

b) $(t, v^*) \notin E(G)$. We consider a vertex $u^* \in X_c$ to obtain the desired obstruction. Recall that from the construction of X_c we have $(u^*, c_2) \notin E(G)$. This further implies that $(u^*, t) \notin E(G)$, otherwise $G[\{u^*, v^*, c_2, t\}]$ is a cycle on 4 vertices. Moreover, we assume that $(u^*, b_1) \notin E(G)$, otherwise u^* satisfies the premise of Case ‡-AW.T.2.A. Also, $(u^*, b_j) \notin E(G)$, for each $j \in [z] \setminus \{1\}$, otherwise $G[\{v, u^*\} \cup \{b_1, b_2, \cdots b_j\}]$ will contain an induced cycle on at least 4 vertices, which is an obstruction containing v with strictly less number of vertices than $\mathbb{O}$. Next, we consider the following cases depending on whether or not $(u^*, c_1) \in E(G)$.

α) $(u^*, c_1) \notin E(G)$. Let $V' = \{t, c_1, u^*, v^*, t_r\} \cup \{b_1, b_2, \ldots, b_z\}$. Observe that $G[V']$ is a †-AW in $G \setminus S$, which does not contain v.

β) $(u^*, c_1) \in E(G)$. Let $V' = \{t, c_1, c_2, u^*, v^*, t_r\} \cup \{b_1, b_2, \ldots, b_z\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$, which does not contain v.

Suppose v is b_1 or b_z. Suppose $v = b_1$ (the other case is symmetric). If $t_\ell \notin \beta(\mathbb{P})$ then for each $u \in L_f \cup R_f$ we have $(u, t_\ell) \in E(G)$ as it fits the frame $\mathbb{F}$ and $N(u) \setminus (M \cup \beta(\mathbb{P})) = N(v) \setminus (M \cup \beta(\mathbb{P}))$. Otherwise, $t_\ell \in \beta(\mathbb{P})$, and then at least one of $L_f \subseteq N(t_\ell)$ or $R_f \subseteq N(t_\ell)$ holds. Let $X_f \in \{L_f, R_f\}$ be a set such that $X_f \subseteq N(t_\ell)$. Similarly, if $b_2 \notin \beta(\mathbb{P})$ then for each $u \in L_f \cup R_f$ we have $(u, b_2) \in E(G)$ as it fits the frame $\mathbb{F}$ and $N(u) \setminus (M \cup \beta(\mathbb{P})) = N(v) \setminus (M \cup \beta(\mathbb{P}))$. Otherwise, $b_2 \in \beta(\mathbb{P})$, and then at least one of $L_f \subseteq N(b_2)$ or $R_f \subseteq N(b_2)$ holds. Let $Y_f \in \{L_f, R_f\}$ be a set such that $Y_f \subseteq N(b_2)$. Next, we consider cases depending on the neighborhood of vertices in $X_f \cup Y_f$ in $\mathbb{O}$ (see Figure 8.10).

Case ‡-AW.B.1. There is $v^* \in X_f \cup Y_f$ such that $\{t_\ell, b_2\} \subseteq N(v^*)$. There exists $e \in [z-2]$ such that $b_e \in N(v^*)$ and $b_{e+1} \notin N(v^*)$. This follows from the fact that $(v^*, b_2) \in E(G)$ and v^* fits the frame $\mathbb{F}$. Next, we consider the sub-cases based on whether or not $(v^*, c_1), (v^*, c_2), (v^*, t) \in E(G)$.

A) $(v^*, c_2) \in E(G), (v^*, t) \in E(G)$. Let $V' = \{t, c_2, v^*, t_\ell\} \cup \{b_1, b_2, \cdots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a ‡-AW such that $|V'| < |V(\mathbb{O})|$.

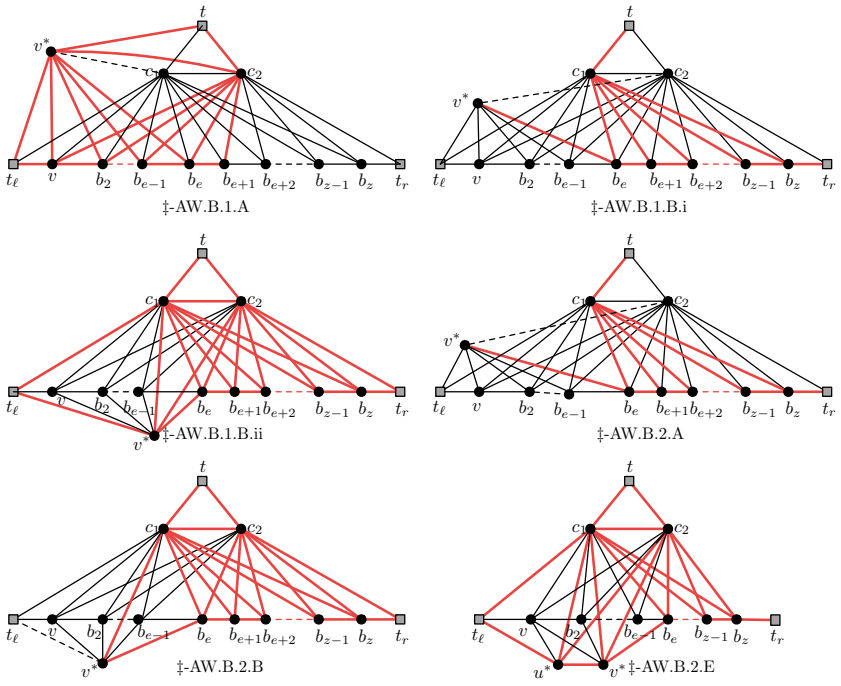

Figure 8.10: Construction of an obstruction when $\mathbb{O}$ is ‡-AW and $v = b_1$.

If Case ‡-AW.B.1.A is not-applicable then $(v^*, c_2) \notin E(G)$ or $(v^*, t) \notin E(G)$ must hold.

B) $(v^*, t) \notin E(G)$. We consider the following cases.

i) $(v^*, c_1) \notin E(G)$. Let $V' = \{t, c, v^*, t_r\} \cup \{b_e, b_{e+1}, \dots b_z\}$. Observe that $G[V']$ is a †-AW not containing v.

ii) $(v^*, c_1) \in E(G)$. Let $V' = \{t, c_1, c_2, v^*, t_r, t_\ell\} \cup \{b_e, b_{e+1} \dots b_z\}$. Observe that $G[V']$ contains a ‡-AW not containing v, that is present in $G \setminus S$.

C) $(v^*, c_2) \notin E(G)$. Since Case ‡-AW.B.1.B is not applicable we can assume that $(v^*, t) \in E(G)$. But then $G[\{v^*, b_2, c_2, t\}]$ is a cycle on 4 vertices.

Case †-AW.B.2. For all $u \in X_f \cup Y_f$ we have $\{t_\ell, b_2\} \not\subseteq N(u)$. Further, by the construction of X_f and Y_f we know that for each $u \in X_f \cup Y_f$ we have $\{t_\ell, b_2\} \cap N(u) \neq \emptyset$, and $X_f, Y_f \neq \emptyset$. Hence, for any pair of vertices $u^* \in X_f$ and $v^* \in Y_f$, we have that $(u^*, t_\ell), (v^*, b_2) \in E(G)$ and $(u^*, b_2), (v^*, t_\ell) \notin E(G)$ (since Case ‡-AW.B.1 is not applicable). Next, we consider cases based on whether or not t, c_2 are adjacent to vertices in $X_f \cup Y_f$.

A) Consider the case when there is $v^* \in X_f \cup Y_f$ such that $(v^*, c_1) \notin E(G)$. In this case, $(v^*, t) \notin E(G)$, otherwise we obtain an induced cycle $G[\{v^*, v, c_1, t\}]$ on 4 vertices. Let $e \in [z-2]$ such that b_e is the last vertex in $\mathsf{base}(\mathbb{O})$ that is adjacent

to v^*. Let $V' = \{t, c_1, v^*, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Notice that $G[V']$ is a †-AW that has fewer vertices than $\mathbb{O}$ as we exclude c_2 and t_ℓ and include v^*.

Hereafter, we assume that for each $u \in X_f \cup Y_f$ we have $(u, c_1) \in E(G)$.

B) Consider the case when there is $v^* \in X_f \cup Y_f$ such that $(v^*, c_2) \notin E(G)$. In this case, $(v^*, t) \notin E(G)$, otherwise $G[v^*, t, c_2, v]$ is a cycle on 4 vertices. Let $e \in [z-2]$ such that b_e is the last vertex in $\mathsf{base}(\mathbb{O})$ that is adjacent to v^*. Let $V' = \{t, c_1, c_2, v^*, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Notice that $G[V']$ is a ‡-AW that has either fewer vertices than $\mathbb{O}$ or has same number of vertices as $\mathbb{O}$ but has more vertices from B_i. Here, we rely on the fact that $t_\ell \notin B_i$, which is ensured by the fact that $Y_f \neq \emptyset$ and $Y_f \cap N(t_\ell) = \emptyset$.

Hereafter, we will assume that for each $u \in X_f \cup Y_f$ we have $c_1, c_2 \in N(u)$.

C) If there is $u^* \in X_f$ such that $(u^*, t) \in E(G)$. Recall that, $(u^*, t_\ell) \in E(G)$ and $(u^*, b_2) \notin E(G)$. In this case, $G[\{t, u^*, c_2, t_\ell, v, b_2\}]$ is a tent.

D) If there is $v^* \in Y_f$ such that $(v^*, t) \in E(G)$. Recall that, $(v^*, b_2) \in E(G)$ and $(v^*, t_\ell) \notin E(G)$. Let $e \in [z-2]$ such that b_e is the last vertex in $\mathsf{base}(\mathbb{O})$ that is adjacent to v^*. Note that $e \geq 2$ as $v^* \in Y_f$. Let $V' = \{t, v^*, t_\ell, b_{e+1}\} \cup \{v, b_3, \ldots, b_e\}$. Observe that $G[V']$ is a †-AW in $G \setminus S$ with strictly fewer vertices than $\mathbb{O}$, as we exclude c_1, c_2 and include v^*.

E) Consider a vertex $u^* \in X_f$ and a vertex $v^* \in Y_f$. Since all the previous cases are not applicable, therefore $(u^*, c_1), (u^*, c_2), (v^*, c_1), (v^*, c_2) \in E(G)$, and $(u^*, t), (v^*, t) \notin E(G)$. Recall that neighbors of u^*, v^* in $P(\mathbb{O})$ are consecutive. Furthermore, $(v^*, t_\ell) \notin E(G)$ and there is no b_j adjacent to u^*, where $j \geq 2$. Let $e \in [z-2]$ such that b_e is the last neighbor of v^* in $P(\mathbb{O})$. Now, let $V' = \{t_\ell, u^*, v^*, c_1, c_2, t\} \cup \{b_e, b_{e+1}, \cdots, b_z\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$ which does not contain v.

Suppose $v = b_j$, where $j \in [z] \setminus \{1, z\}$. Let $X_f \in \{L_f, R_f\}$ be a set such that $X_f \subseteq N(b_{j-1})$ and $Y_f \in \{L_f, R_f\}$ be a set such that $Y_f \subseteq N(b_{j+1})$. Since M is a 9-redundant modulator, $N(\{t_r, b_z, b_{z-1}\}) \cap (X_f \cup Y_f) = \emptyset$ or $N(\{t_\ell, b_1, b_2\}) \cap (X_f \cup Y_f) = \emptyset$. We assume that $N(\{t_r, b_z, b_{z-1}\}) \cap (X_f \cup Y_f) = \emptyset$ (the other case is symmetric). Next, we consider cases based on neighborhood in $\mathbb{O}$ of vertices in sets X_f, and Y_f (see Figure 8.11).

Case ‡-AW.J.1. If there is $v^* \in X_f \cup Y_f$ such that $(v^*, c_1) \notin E(G)$. Note that if $(v^*, c_1) \notin E(G)$ then $(v^*, t) \notin E(G)$, otherwise $G[\{v, v^*, c_1, t\}]$ is a cycle on 4 vertices. Also, the neighbors of v^* in $P(\mathbb{O})$ must be consecutive. Let $e \in [z-2]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $t_r, b_z, b_{z-1} \notin N(v^*)$. Let $V' = \{t, c_1, v^*, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a †-AW with strictly fewer vertices than $\mathbb{O}$.

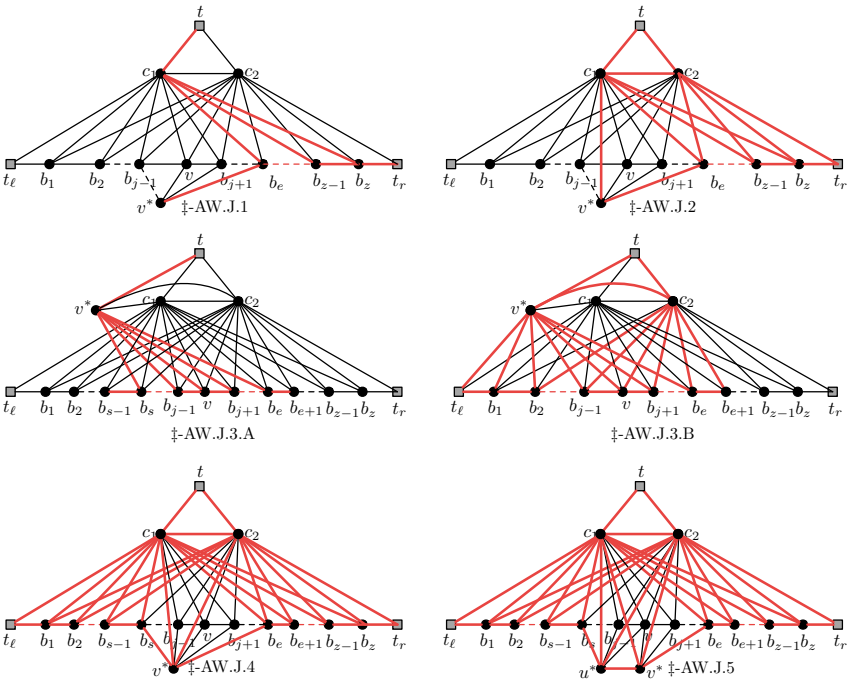

Figure 8.11: Construction of an obstruction when $\mathbb{O}$ is ‡-AW and $v = b_j$, where $j \in [z-1] \setminus \{1\}$.

Case ‡-AW.J.2. If there is $v^* \in X_f \cup Y_f$ such that $(v^*, c_2) \notin E(G)$. Since Case ‡-AW.J.1 is not applicable we can assume that $(v^*, c_1) \in E(G)$. Note that if $(v^*, c_2) \notin E(G)$ then $(v^*, t) \notin E(G)$, otherwise $G[\{v, v^*, c_2, t\}]$ is a cycle on 4 vertices. Also, the neighbors of v^* in $P(\mathbb{O})$ must be consecutive. Let $e \in [z-2]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $t_r, b_z, b_{z-1} \notin N(v^*)$. Let $V' = \{t, c_1, c_2, v^*, t_r\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a ‡-AW with strictly fewer vertices than $\mathbb{O}$.

Note that if Cases ‡-AW.J.1 and ‡-AW.J.2 are not applicable then for each $u \in X_f \cup Y_f$ we have $(u, c_1), (u, c_2) \in E(G)$. The cases we consider next are based on whether or not the following conditions are satisfied for a vertex $u \in X_f \cup Y_f$.

1. $(u, t) \in E(G)$;
2. $\{b_{j-1}, b_{j+1}\} \subseteq N(u)$.

Case ‡-AW.J.3. If there is $v^* \in X_f \cup Y_f$ such that $(v^*, t) \in E(G)$. We further consider the following sub-cases based on whether or not $t_\ell \in N(v^*)$.

A) $t_\ell \notin N(v^*)$. Let $s \in [j]$ such that b_s is the first vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $(t_\ell, v^*) \notin E(G)$ and $(v^*, v) \in E(G)$. Also, let $e \in [z-2]$

such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $(t_r, v^*), (b_z, v^*), (b_{z-1}, v^*) \notin E(G)$ and $(v^*, v) \in E(G)$. Notice that $s \neq e$ since by the construction of the sets X_f and Y_f we have that v^* is incident to v and at least one of the vertices in $\{b_{j-1}, b_{j+1}\}$. Let $V' = \{t, v^*\} \cup \{b_{s-1}, b_s, \ldots, b_e, b_{e+1}\}$. Observe that $G[V']$ is a †-AW in $G \setminus S$ with $|V'| < |V(\mathbb{O})|$.

B) $t_\ell \in N(v^*)$. Let $e \in [z-2]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $(t_r, v^*), (b_z, v^*), (b_{z-1}, v^*) \notin E(G)$ and $(v^*, v) \in E(G)$. Let $V' = \{t, v^*, c_2, t_\ell\} \cup \{b_1, b_2, \ldots, b_e, b_{e+1}\}$ is a ‡-AW in $G \setminus S$. Moreover, $|V'| < |V(\mathbb{O})|$ since $t_r, b_z \notin V'$ and $V' \subseteq V(\mathbb{O}) \cup \{v^*\}$.

Case †-AW.J.4. If there is $v^* \in X_f \cup Y_f$ such that $(v^*, t) \notin E(G)$ and $\{b_{j-1}, b_{j+1}\} \subseteq N(v^*)$. Notice that all the neighbors of v^* on $P(\mathbb{O})$ must be consecutive, and there exists at least three vertices on $P(\mathbb{O})$ that are non-adjacent to v^*, which follows from M being a 9-redundant modulator. Also, we have $\{t_r, b_z, b_{z-1}\} \cap N(v^*) = \emptyset$. Let $e \in [z-2]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $t_r, b_z, b_{z-1} \notin N(v^*)$. Also, let $s \in [z-1] \cup \{0\}$ be the lowest integer such that $(v^*, b_s) \in E(G)$ (b_s could possibly be same as b_{j-1} or $b_0 = t_\ell$). Let $V' = \{t, c_1, c_2, v^*, t_r\} \cup \{b_1, b_2, \ldots, b_s\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$ which does not contain v.

Case †-AW.J.5. For all $u \in X_f \cup Y_f$ we have $c_1, c_2 \in N(u)$, $(u, t) \notin E(G)$, and $\{b_{j-1}, b_{j+1}\} \not\subseteq N(u)$. Notice that by the construction of X_f and Y_f we have for each $u \in X_f \cup Y_f$, $|N(u) \cap \{b_{j-1}, b_{j+1}\}| = 1$. Next, consider a vertex $u^* \in X_f$ and $v^* \in Y_f$. Let $s \in [j-1] \cup \{0\}$ such that b_s is the first vertex in $P(\mathbb{O})$ which is adjacent to u^*, which exists since $(u^*, b_{j-1}) \in E(G)$. Also, let $e \in [z-1]$ such that b_e is the last vertex in $P(\mathbb{O})$ which is adjacent to v^*, which exists since $(t_r, v^*), (b_z, v^*) \notin E(G)$ and $(v^*, b_{j+1}) \in E(G)$. Notice that $s \neq e$. Let $V' = \{t, c_1, c_2, v^*, u^*\} \cup \{t_\ell, b_1, b_2, b_{s-1}, b_s\} \cup \{b_e, b_{e+1}, \ldots, b_z\}$. Observe that $G[V']$ is a ‡-AW in $G \setminus S$ which does not contain v.

8.4 Bounding the Length of a Clique Path

Let us now turn to the problem of bounding the sizes of non-module components. The approach mimics the one presented in Chapter 7 (also, [ALM+17a, JP17], but requires additional structural observations corresponding to interval graphs and it's obstructions [BLS99, Gol04]. Each non-module component is a clique path in $G - M$, where M is a 9-redundant modulator. Let K be a clique path of a non-module component C. Note that there is a vertex in M that has a neighbor as well as a non-neighbor in C. In this section we consider the problem of reducing the number of bags in a clique path K. We assume that any bag B_i in the clique path K has at most $\eta = 2^{10} \cdot 4(k+5)\binom{|M|}{10}$ vertices. We will devise a collection of "marking schemes" that mark some polynomial in k number of bags in K, such that the obstructions are "well behaved" in the region between any two consecutive marked bags are. In particular, our marking schemes ensure that if any obstruction intersects an unmarked region of the clique-path, then the intersection is an induced path. Then, we design reduction rules that "preserve" a minimum separator of the unmarked region. More precisely, we identify an irrelevant vertex or an irrelevant

edge, and then delete it or contract it in the graph. The correctness of these reduction rules follow from the structural properties ensured by the marking schemes.

Let us recall the graph G^+ that is obtained from G by adding a new edge for each set of cardinality 2 in $\mathcal{W}$. Note that the new edges in G^+ have both endpoints in M. This graph will be helpful in obtaining structural properties of the graph G. In particular, the structural properties of any obstructions that is not covered $\mathcal{W}$, hold in both G^+ and G. Let us also note the following. Recall that M is a 9-redundant solution, and we have a $(k+2)$-necessary family $\mathcal{W} \subseteq 2^{V(G)}$, such that any solution of size at most $k+2$ must hit every set in $\mathcal{W}$. In the following, whenever we consider a set Z of $k+2$ or fewer vertices and the obstructions in $G - Z$, we will also assume that Z is a hitting set for $\mathcal{W}$. We call such a set Z a *good set*. We will justify this assumption when we present our reduction rules and prove their correctness. In particular, we will show that any "candidate solution" for G, that is obtained from a solution to the "reduced instance" will always be a good set. Let us recall that all obstructions in G on up to 10 vertices are present in $\mathcal{W}$ and therefore they are hit by the good set Z. Hence, we may assume that any obstruction, $\mathbb{O}$, in $G - Z$ is either a chord-less cycle, or an †-AW or a ‡-AW such that $|\mathbb{O} \cap M| \geq 10$.

Let us now define a few notations that will be required in this section. Note that these notations apply to K as well as any sub-clique-path of K. We fix an ordering of the bags in the clique path K from left to right. or two bags B_ℓ, B_r in K by $K[B_\ell, B_r]$ we denote the sub-clique path in K between B_ℓ and B_r (including B_ℓ and B_r). We say that a vertex $v \in K$ is a *marked vertex* if there is a marked bag that contains it, otherwise it is an *unmarked vertex*. We say that two marked bags B, B' are *consecutive* if $K[B, B']$ contains no marked bags other than B and B'. We say that two bags of a clique path are *adjacent* if there is no other bag that lies between them. Notice that for a bag B in a clique path, B^{-1} and B^{+1} denote the adjacent bags of B, on the left and no the right, respectively.

8.4.1 Partition into manageable-clique paths

In this section, we partition a clique path K into a collection of so called "manageable-clique paths", that are well structured with respect to the modulator M. We shall construct a collection of bags, denoted by $K(M)$, based on the edges between K and M and mark the bags in $K(M)$. Let us initialize $K(M)$ as the collection containing the two end bags of K.

Let us begin with the following property of interval graphs.

Observation 32. *Let H be an interval graph and let H' be the graph obtained by one of the following operations.*

(a) For $v \in V(H)$, $H' = H - \{v\}$ or

(b) For $(u, v) \in E(H)$, $H' = H/(u, v)$.

Then H' is an interval graph. Furthermore the size of any clique in H' is upper-bounded by the size of a maximum clique in H.

The above observation follows from the definition of interval graphs and their interval representation [Gol04]. In particular, statement (b) follows from the observation that an

interval representation of $H/(u,v)$ can be obtained by taking an interval representation of H and "merging" the intervals of u and v.

Now consider a vertex $m \in M$ and the let H_m be the bipartite graph with vertex bipartition $N(m) \cap K$ and $\overline{N(m)} \cap K$, where $u \in N(m) \cap K$ and $v \in \overline{N(m)} \cap K$ are adjacent in H_m if and only if $(u,v) \in E(G)$. Here, $\overline{N(m)} = V(G) \setminus N(m)$. Next, we prove the following lemma about the graph H_m.

Lemma 8.18. *For $m \in M$, let Y_m be a maximum matching in H_m. Then $|Y_m| \leq 2\eta$.*

Proof. Consider the case when $|Y_m| > 2\eta$, otherwise the claim trivially holds. Let $\tilde{K}$ be the graph obtained from K by contracting each edge (u,v) in Y_m, and let uv^* be the vertex resulting after its contraction. Next, let K' be the graph $\tilde{K}[\{uv^* \mid (u,v) \in Y_m\}]$. We note that the definition of K' relies on the fact that Y_m is a matching in H_m. From the construction of K' and Observation 32 it follows that K' is an interval graph on $|Y_m|$ vertices. Furthermore, the size of a maximal clique in K' is bounded by η. Interval graphs are perfect graphs, and on a perfect graph G we know that $\omega(G)\alpha(G) \geq |V(G)|$, where $\omega(G)$ and $\alpha(G)$ are the cardinalities of maximum clique and maximum independent set in G, respectively [Lov72] (or Theorem 3.3 [Gol04]). This implies that there is an independent set in K' of size greater than $|Y_m|/\eta > 2$. Consider an independent set of size 3 in K', and the corresponding edges of the matching Y_m in K. It follows that these three edges and the vertex m form a long-claw in G^+. However, this contradicts the fact that M is a 9-redundant solution and each set in $\mathcal{W}$ has at least 2 vertices. □

For each each $m \in M$, we compute a maximum matching Y_m in the graph H_m. Then for each edge in Y_m we pick a bag in K that contains this edge and add it to $K(M)$. Let us observe that we add at most $2\eta|M|$ bags to $K(M)$. Before proceeding further, we add some more bags to $K(M)$ that gives us some additional structural properties. For notational convenience, we introduce the graph G^+, which is obtained from G by adding the edge (m_1, m_2), for each $m_1, m_2 \in M$ such that $(m_1, m_2) \notin E(G)$ and $\{m_1, m_2\} \in \mathcal{W}$. Next, we state the following observation, which will be useful in designing one of our marking schemes for bags in the clique path.

Observation 33. *Let m_1 and m_2 be a pair of (distinct) vertices in M such that $(m_1, m_2) \notin E(G^+)$. Then $(N(m_1) \cap N(m_2)) \setminus M$ is a clique in G^+.*

The above observation follows from the fact that if there were a non-adjacent pair of vertices v_1 and v_2 in the common neighborhood of m_1 and m_2, then we would have a cycle of length 4, that intersects M in only two vertices. Therefore, either $\{m_1, m_2\}$ is in $\mathcal{W}$ and hence, they must be adjacent in G^+, or we contradict the fact that M is a 9-redundant modulator. It follows that there is a bag in K that contains the common neighborhood of m_1 and m_2 whenever they are not adjacent in G^+. We add a collection of up to $|M|^2$ bags to $K(M)$, each containing the common neighborhood of a pair of non-adjacent vertices in M in G^+. This gives us the following marking scheme.

Marking Scheme I. We mark all bags in $K(M)$.

Observe that we have marked at most $\boxed{2\eta|M| + |M|^2 + 2 < 4\eta|M|}$ bags of K in the above marking scheme. Here, we use the fact that $\eta \geq |M|$. Next, we state an observation that will be useful later.

Observation 34. *If $v \in K$ such that there is no bag in $K(M)$ that contains v, then $N_{G^+}(v) \cap M$ $(= N_G(v) \cap M)$ is a clique in G^+.*

Let $B_\ell, B_r \in K(M)$ be two consecutive marked bags in K. Now, consider the graph $K[B_\ell, B_r] - (B_\ell \cup B_r)$, and observe that it may have several connected components. For the sake of brevity, we call these connected components the *components of* $K[B_\ell, B_r]$. We extend this definition to say that an induced subgraph of K is a *component of* K if it is a component of $K[B_\ell, B_r]$ for some pair of consecutive marked bags B_ℓ, B_r in K. Note that each component of K is also a clique path. These components have the following property.

Lemma 8.19. *Let X be a component of K. Then any $m \in M$ is adjacent to, either all vertices in X (in both G and G^+), or none of them.*

Proof. Note that for any $m \in M$, $N_G(m) \cap X = N_{G^+}(m) \cap X$, therefore, it is enough to argue in the graph G. Suppose the claim is not true, then choose an $m \in M$ for which the claim is false. Then m has both a neighbor and a non-neighbor in X in G. Hence, there is an edge $e \in X$ such that one endpoint of e lies in $N_G(m) \cap K$ and the other-endpoint lies in $\overline{N_G(m)} \cap K$, i.e. $e \in H_m$. Furthermore, by construction, both these endpoints are disjoint from the vertices of the matching Y_m in H_m. Therefore $Y_m \cup \{e\}$ is also a matching in H_m. However, this is a contradiction as Y_m is a maximum matching in H_m. This concludes the proof. □

Let us fix a pair of consecutive marked bags B_ℓ, B_r and consider the components of $K[B_\ell, B_r]$. Note that the above lemma may be stated as follows. Any component of $K[B_\ell, B_r]$ is a "module with respect to M". The following lemma shows that all but at most 4η of these components are actually modules in the graph G^+ (and G as well).

Lemma 8.20. *All but at most 4η of the components of $K[B_\ell, B_r]$ are modules in the graph G^+ (and G).*

Proof. Let us recall that $G^+ - M = G - M$. Let X be a component of $K[B_\ell, B_r]$. Observe that for any vertex $v \in B_\ell \cup B_r$ there are at most two components such that v has a neighbor and a non-neighbor in the component. Indeed, if this were not the case then we obtain a long-claw in $K \subseteq G^+ - M$, which is a contradiction. Now recall that there are at most 2η vertices in $B_\ell \cup B_r$. Hence it follows that all but at most 4η components have the following property. Each vertex $v \in B_\ell \cup B_r$ is adjacent to, either every vertex of the component, or none of them. Finally, observe that the neighborhood of any component X is a subset of $M \cup B_\ell \cup B_r$. Hence, it follows from the above arguments and Lemma 8.19 that all but at most 4η connected components of $K[B_\ell, B_r] - (B_\ell \cup B_r)$ are modules in G^+ (and G as well). □

Let us note another useful property of these components.

Observation 35. *Let X be a component of $K[B_\ell, B_r]$. Then there is a sub-clique path K_X of $K[B_\ell, B_r]$ such that $X \subseteq V(K_X) \subseteq X \cup B_\ell \cup B_r$. Furthermore, if X and Y are two distinct components of K, then K_X and K_Y have no common bags.*

Proof. Since X is a connected graph and K is a tree-decomposition, it follows from the definitions that the collection of bags of K, such that each bag contains a vertex of

X, forms a sub-clique path K_X of K. Furthermore, as X is a connected component of $K[B_\ell, B_r] - (B_\ell \cup B_r)$, it follows that K_X is a sub-clique path of $K[B_\ell, B_r]$ and $X \subseteq V(K_X) \subseteq X \cup B_\ell \cup B_r$. Also note that K_X doesn't contain the bags B_ℓ and B_r, this will be useful for proving the second part of the claim.

Next, consider another component Y of K, and the following cases depending on whether or not Y is a component of $K[B_\ell, B_r]$.

- Y is a component of $K[B_p, B_q]$ where $K[B_p, B_q]$ is distinct from $K[B_\ell, B_r]$. Recall that B_p and B_q are consecutive marked bags. Similarly, B_ℓ and B_r are marked bags. Let K_X and K_Y be the sub-clique paths returned by first part of the claim. Then $X \subseteq K_X \subseteq X \cup B_\ell \cup B_r$ and K_X does not contain the bags B_ℓ and B_r. Similarly, $Y \subseteq K_Y \subseteq Y \cup B_p \cup B_q$ and K_Y does not contain the bags B_p and B_q. This together with the distinctness of $K[B_p, B_q]$ where $K[B_p, B_q]$ implies that K_X and K_Y have no bags in common.

- Next suppose that Y were another component of $K[B_\ell, B_r]$, and let K_Y be the corresponding sub-clique path of $K[B_\ell, B_r]$. If there is a bag B that lies in both K_X and K_Y, then as B is a clique, there is a pair of vertices $u \in X \cap B$ and $v \in Y \cap B$ that are adjacent to each other. But this is a contradiction.

This concludes the proof. □

Recall that the components of each $K[B_\ell, B_r]$ can be divided into two groups, those that are modules in G and the rest. We will first consider the problem of reducing these module components.

Module components of K

Let us now consider the problem of reducing the total number of vertices in a module component of K. Consider a pair of consecutive marked bags B_ℓ, B_r in K and the number of vertices in the module components of $K[B_\ell, B_r]$. Let $\widehat{\mathcal{C}}$ be the collection of connected components in $K[B_\ell, B_r] - (B_\ell \cup B_r)$ that are modules in G. Note that $|M \cup B_\ell \cup B_r| \leq |M| + 2\eta$. Now we may apply Lemma 8.7 for $\widehat{M} = B_\ell \cup B_r$ and obtain a subset B of $V(\widehat{\mathcal{C}})$ of size $4(k+2)^2(|M|+2\eta)^6$ such that the following holds. If $S \subseteq V(G)$ of size at most k and $\mathbb{O}$ is an obstruction in $G - S$ that is not covered by $\mathcal{W}$, then there is another obstruction $\mathbb{O}'$ in $G - S$ such that $\mathbb{O}' \cap (V(\widehat{\mathcal{C}}) \setminus B) = \emptyset$. This gives the following reduction rule.

Reduction Rule 8.6. *Suppose $v \in V(\widehat{\mathcal{C}}) \setminus B$, then delete v from the graph G.*

Lemma 8.21. *Reduction rule 8.6 is safe.*

Proof. Let $v \in V(\widehat{\mathcal{C}}) \setminus B$, and $G' = G - \{v\}$. We will show that (G, k) is a *yes*-instance of IVD if and only if (G', k) is a *yes*-instance of IVD. Moreover, any solution S to IVD in (G, k) (or (G', k)) is a hitting set for $\mathcal{W}$.

In the forward direction, let S be a solution to IVD in (G, k). By the definition of $\mathcal{W}$, $S \cap M$ ($\subseteq V(G')$) hits each set in $\mathcal{W}$. Moreover, as $G' - S$ is an induced subgraph of $G - S$, it follows from Observation 32 that S is a solution to IVD in (G', k).

In the reverse direction, let S be a solution to IVD in (G', k) and $S = S' \cup \{v\}$. Since $G' - S'$ is same as the graph $G - S$, therefore it follows that $G - S$ is an interval graph.

Moreover, as $|S| \leq k+1$, it hits each set in $\mathcal{W}$, which is $(k+2)$-necessary. But since $v \notin M$, it follows that S' hits each set in $\mathcal{W}$. Next, we want to show that S' is a solution to IVD in (G,k). Suppose not, then consider an obstruction $\mathbb{O}$ in $G-S'$. Notice that $\mathbb{O}$ is not covered by $\mathcal{W}$. But then, from Lemma 8.7, there is an obstruction $\mathbb{O}'$ in $G-S'$ that is disjoint from $V(\widehat{\mathcal{C}}) \setminus B$ which is not covered by $\mathcal{W}$, and hence it doesn't contain the vertex v. Therefore, $\mathbb{O}'$ is also an obstruction in $(G-\{v\})-S' = G'-S'$, which is a contradiction. This concludes the proof. □

By the above reduction rule, we may assume the total number of vertices in $V(\widehat{\mathcal{C}})$ is bounded by $4(k+2)^2(|M|+2\eta)^6$. Note that any bag in K that contains a vertex of $V(\widehat{\mathcal{C}})$ is a subset of $V(\widehat{\mathcal{C}}) \cup B_\ell \cup B_r$. Therefore, there are at most $|V(\widehat{\mathcal{C}})|$ bags in K that contain a vertex of $V(\widehat{\mathcal{C}})$. Finally, observe that there are at most $4\eta|M|$ pairs of consecutive marked bags in K. Applying the above reduction rule for every such pair, we obtain the following. There are at most $4(k+2)^2(|M|+2\eta)^6 \cdot 4\eta|M|$ vertices in K that lie in the union of all module components. Let $\mathcal{C}(K)$ denote the above collection of vertices.

Marking Scheme II: We mark all bags in K that contain a vertex from $\mathcal{C}(K)$.

Note that using Marking Scheme II, at most $\boxed{4(k+2)^2(|M|+2\eta)^6 \cdot 4\eta|M|}$ bags in K have been marked.

Obtaining Manageable-Clique-Paths

Now we consider the non-module components of K. We now mark some more bags in K that will partition them into sub-clique path with additional structural properties. We will call these sub-clique paths so obtained, *manageable-clique paths*. We initialize an empty-collection $\widehat{K(M)}$. For each non-module component X of $K[B_\ell, B_r]$ that contains an unmarked vertex, where B_ℓ, B_r are consecutive marked bags in $K(M)$; for each pair of adjacent bags B, B' in K_X such that $B \cap B_\ell \supsetneq B' \cap B_\ell$ or $B \cap B_r \subsetneq B' \cap B_r$, we add B, B' to $\widehat{K(M)}$. Note that we add up to 4η bags to $\widehat{K(M)}$ for each component X of K, and therefore we have added at most $4\eta \cdot 4\eta \cdot 4\eta|M| = 64\eta^3|M|$ bags to $\widehat{K(M)}$. This bound follows from Marking Scheme I, Lemma 8.20, and the above discussion.

Marking Scheme III: We mark all bags in $\widehat{K(M)}$.

Note that we marked at most $\boxed{64\eta^3|M|}$ bags using the above marking scheme. We now further partition K using the bags marked in the above scheme. For a sub-clique path K_X (returned by Observation 35) of a component X of $K[B_\ell, B_r]$, and two consecutive marked bags $B_p, B_q \in \widehat{K(M)}$ in K_X, the sub-clique path $K_X[B_p, B_q]$ is a (B_ℓ, B_r)-manageable clique path (or simple a *manageable clique path*). Next, we derive the following property.

Observation 36. *Let X be a component of $K[B_\ell, B_r]$, where B_ℓ and B_r are marked consecutive bags in $K(M)$. Furthermore, let K_X be the sub-clique path of K returned by Observation 35 for X. Let $B_p, B_q \in K(M)$ be a pair of consecutive marked bags in K_X*

such that $K_X[B_p, B_q]$ contains an unmarked vertex. Then for any bag B in $K_X[B_p, B_q]$ we have $B_p \cap B_\ell = B \cap B_\ell = B_q \cap B_\ell$ and $B_p \cap B_r = B \cap B_r = B_q \cap B_r$.

Let us derive a few properties of the manageable-clique paths. Consider a (B_ℓ, B_r)-manageable clique path Q, and let $C_Q = B_\ell \cap B_r$. We have the following observation, that follows from the construction of Q (together with the Marking Scheme I) and Observation 34.

Observation 37. *Any vertex $m \in M$ is adjacent to either all vertices in $V(Q) \setminus C_Q$ in the graph G^+ (and G as well), or none of them. Furthermore, $N(V(Q) \setminus C_Q) \cap M$ is a clique in G^+.*

Let us define $M_A = M \cap N(Q \setminus C_Q)$, and $M_P = M \setminus M_A$. Let us observe that, by construction, $N(M_P) \cap Q \subseteq C_Q$. Let us note that there may be a vertex $v \in C_Q$ and a vertex $m \in M_A$ such that $(v, m) \notin E(G)$.

Observation 38. *Let $v \in C_Q$ and $m \in M_A$ be a pair of non-adjacent vertices. Then Q is a clique in G^+ (and G).*

Proof. Let us consider these vertices in the graph G. Observe that every vertex in $C_Q \cup M_A$ is adjacent to every vertex in $Q \setminus C_Q$ in the graph G. Therefore, if there is a pair of non-adjacent vertices $u, w \in Q \setminus C_Q$, then we obtain a cycle on 4 vertices, namely, $C = (u, v, w, m)$ in G which intersects the 9-redundant solution M in only one vertex, which is m. This implies C is covered by $\mathcal{W}$. Since $V(C) \cap M = \{m\}$, this implies that $\{m\} \in \mathcal{W}$. This contradicts that each set in $\mathcal{W}$ is of size at least 2. Therefore, we conclude that Q is a clique in G. □

Hence, we may assume that Q is not a clique, and then we have the following observation.

Observation 39. *Suppose Q is not a clique. Then for any $v \in V(Q)$ and $m \in M_A$, we have $(v, m) \in E(G)$ (and $E(G^+)$). Furthermore, each vertex in $M_A \cup C_Q$ is adjacent to all vertices in $V(Q) \cup M_A$ in G^+.*

Proof. The first part follows from Observation 37, definition of M_A, and Observation 38. Also, by construction C_Q is a clique that is contained in every bag of Q in G, and $G - M = G^+ - M$. Therefore, we next argue that M_A is a clique in G^+. Suppose not, then there are $m_1, m_2 \in M_A$ such that $(m_1, m_2) \notin E(G^+)$. By our assumption there exists $u, v \in V(Q)$ such that $(u, v) \notin E(G)$ (and hence $(u, v) \notin E(G^+)$). But then, $C = (m_1, u, m_2, v)$ is an induced cycle on 4 vertices in G which intersects the 9-redundant solution M in 2 vertices. This implies that C is covered by $\mathcal{W}$. Since each set in $\mathcal{W}$ is of size at least 2, therefore we have that $\{m_1, m_2\} \in \mathcal{W}$. This contradicts that $(m_1, m_2) \notin E(G^+)$. This concludes the proof. □

8.4.2 Manageable clique paths

We start by recalling the number of manageable clique paths, which is bounded by $64\eta^3|M|$. Thus, if a manageable clique path is a clique in G, then we add it to the set of marked bags. By doing this we add at most $64\eta^3|M|$ bags to $\widehat{K(M)}$. Hereafter, we deal with only those manageable clique paths that are not cliques in G. Next, consider

a manageable clique path $Q = K[B_\ell, B_r]$, where B_ℓ and B_r are its first and last bags, respectively. Also, let $C_Q = B_\ell \cap B_r$ and $Q_I = V(Q) \setminus (B_\ell \cup B_r)$. Observe that all vertices in Q_I are unmarked. Let $M_A = M \cap N(V(Q) \setminus C_Q)$, and $M_P = M \setminus M_A$. Note that from Observation 39 it follows that M_A is adjacent to every vertex in Q, while $N(M_P) \cap V(Q) \subseteq C_Q$. Finally, from the definition of M_A and Observation 37 we have that M_A is a clique in G^+.

We will devise a sequence of marking schemes, that mark a bounded number of bags in Q, such that the obstructions are well-behaved with respect to the marked bags. We have the following definition for AW.

Definition 8.3. An AW $\mathbb{O}$ is called *Q-manageable* if all terminals in $\mathbb{O} \cap Q$ appear in marked bags.

More precisely, our goal is to mark a bounded number of bags in Q so that, for any set S of $k+2$ or fewer vertices, if there is an AW $\mathbb{O}$ in $G - S$, then there must be another AW $\mathbb{O}$ that is Q-manageable. Let $\mathcal{B}_Q$ be the set of marked bags in Q, and initially it contains only B_ℓ and B_r, which are its first and last bags, respectively.

We have the following lemma (Lemma 8.22) that characterizes the intersection between a manageable clique path Q and an induced path P in G. Let us note the this lemma holds for any sub-clique path Q' of Q when we define the associated vertex sets B'_ℓ, B'_r, Q'_I and C'_Q accordingly. Note that the sets M_A and M_P remain unchanged for Q'. We present the proof of this lemma for the clique path Q only, but the same arguments will hold for Q'.

Lemma 8.22. *Let $P = (v_1, v_2, \ldots, v_t)$ be an induced path in G^+ such that $(V(P) \setminus \{v_1, v_t\}) \cap Q_I \neq \emptyset$, $V(P) \cap M_A = \emptyset$ and $V(P) \cap (V(G) \setminus Q_I) \neq \emptyset$. Then $V(P) \cap C_Q = \emptyset$. Furthermore, if $v_1, v_t \notin Q_I$, then $P_Q = P[V(P) \cap V(Q)]$ is an induced path in G^+ from a vertex in $B_\ell \setminus C_Q$ to a vertex in $B_r \setminus C_Q$ such that $P_Q - (B_\ell \cup B_r)$ is an induced path contained in Q_I.*

Proof. Consider a vertex $v \in (V(P) \cap Q_I) \setminus \{v_1, v_t\}$, and let v_{-1} and v_{+1} be its two neighbors in P. Observe that, as $N_{G^+}(v) \subseteq V(Q) \cup M_A$ and $V(P) \cap M_A = \emptyset$, the vertices v_{-1} and v_{+1} must belong to $V(Q)$. Furthermore, C_Q is a clique and for any $w \in C_Q$ we have $N_{G^+}(v) \subseteq N_{G^+}(w)$ as Q is not a clique in G^+ (Observation 39). Therefore, $V(P) \cap C_Q = \emptyset$, as otherwise we obtain a chord in the induced path P between a vertex $w \in V(P) \cap C_Q$ and one of v_{-1} or v_{+1}. This concludes the proof of first part of the lemma.

Now we prove the second part of the lemma. Consider the set $P_Q = V(P) \cap V(Q)$, and let $v_s \in P_Q$ be a vertex with lowest possible index in P that belongs to the set Q_I. Note that $s \in [t] \setminus \{1, t\}$ by our assumption that $v_1, v_t \notin Q_I$ and $(V(P) \cap Q_I) \setminus \{v_1, v_t\} \neq \emptyset$. Let v_e (possibly same as v_s) be the vertex with highest index in P which belongs to Q_I such that for each $i \in \{s, s+1, \ldots, e\}$, $v_i \in Q_I$. By our assumption we have that $e \in [t] \setminus \{1, t\}$. Next, we consider the vertices v_{s-1} and v_{e+1}. From the construction of v_{s-1} and v_{e+1}, the assumption that $V(P) \cap M_A = \emptyset$, and first part of the lemma it follows that $v_{s-1}, v_{e+1} \notin Q_I \cup M_A \cup C_Q$. Moreover, $(v_{s-1}, v_s), (v_e, v_{e+1}) \in E(G^+)$, and for $v^* \in \{v_s, v_e\}$, we have $N_{G^+}(v^*) \subseteq V(Q) \cup M_A$. Therefore, $v_{s-1}, v_{e+1} \in (B_\ell \cup B_r) \setminus C_Q$. Without loss of generality, we assume that $v_{s-1} \in B_\ell \setminus C_Q$. But then $v_{e+1} \notin B_\ell \setminus C_Q$, otherwise we have a chord (v_{s-1}, v_{e+1}) in P. This implies that $v_{e+1} \in B_r \setminus C_Q$. Therefore,

$P' = P[\{v_{s-1}, v_s, \ldots, v_{e+1}\}]$ is an induced path from a vertex in $B_\ell \setminus C_Q$ to a vertex in $B_r \setminus C_Q$. Notice that v_{s-1-i}, where $i \geq 2$ cannot belong to B_ℓ, otherwise there will be a chord in P. We note that v_{s-2} could possibly belong to $B_\ell \setminus C_Q$ but not to C_Q. Similarly, we can show that v_{e+1+i}, where $i \geq 2$ cannot belong to B_r, while v_{e+2} could possible belong to $B_r \setminus C_Q$ but not to C_Q. Let $s^* \in \{s-1, s-2\}$ be the lowest index such that $v_{s^*} \in V(P) \cap (B_\ell \setminus C_Q)$ and $e^* \in \{e+1, e+2\}$ be the highest index such that $v_e{}^* \in V(P) \cap (B_r \setminus C_Q)$. Observe that if $v_{s-2}, v_{e+2} \notin Q_I$, then $P^* = P[\{v_{s^*}, v_{s^*+1}, \ldots, v_{e^*}\}]$ is an induced path from a vertex in $B_\ell \setminus C_Q$ to a vertex in $B_r \setminus C_Q$. Thus to prove the lemma it is enough to show that $v_i \notin Q_I$, where $i \in [s-2] \cup \{e+2, e+3, \ldots, t\}$. Suppose not, then there is an integer $i^* \in [s-2] \cup \{e+2, e+3, \ldots, t\}$ such that $v_{i^*} \in Q_I$. Since $v_{i^*} \in Q_I$, it must hold that v_{i^*} belong to a bag, say B^* in Q which is different from B_ℓ and B_r. Recall that P' is a sub-path of P from $v_{s-1} \in B_\ell \setminus C_Q$ to $v_{e+1} \in B_r \setminus C_Q$. Therefore, P' intersects every bag in the manageable clique path Q. In particular, it contains a vertex say, v^* from B^*, which is different from v_{i^*}. But then $(v^*, v_{i^*}) \in E(G^+)$ is a chord in the induced path P, which is a contradiction. This concludes the proof of the lemma. □

Observation 40. *Let $v \in V(Q) \setminus C_Q$. Then v not a center vertex of any AW in G^+ that is not covered by $\mathcal{W}$.*

Proof. Let $\mathbb{O}$ be an AW in G^+ that is not covered by $\mathcal{W}$, and suppose that v is a center vertex of $\mathbb{O}$. Then v must be adjacent to all vertices of $\mathsf{base}(\mathbb{O})$. As M is an 9-redundant modulator, there are at least 5 vertices of M in $\mathsf{base}(\mathbb{O})$, and therefore there are vertices $m_1, m_2 \in M$ such that $(m_1, m_2) \notin E(G^+)$, $(m_1, v), (m_2, v) \in E(G^+)$ and $(m_1, m_2) \in E(G^+)$. This follows from the fact that $v \in V(Q) \setminus C_Q$, $N_G(v) \cap M \subseteq M_A$, and M_A is a clique in G^+ (Observation 37). But this contradicts that $\mathbb{O}$ is an obstruction in G^+. □

Let $\mathbb{O}$ be an AW (not covered by $\mathcal{W}$) in G^+, and let $P(\mathbb{O})$ denote the induced path $(t_\ell, \mathsf{base}(\mathbb{O}), t_r)$. We have the following notion of distance of a vertex in Q from the end bags B_ℓ and B_r. We use this notion in marking bags that satisfy certain properties and are closest to the endpoints of Q. Recall that we have an ordering of the bags from left to right, where B_ℓ is the leftmost bag and B_r is the rightmost bag in Q.

Definition 8.4. Let $v \in Q_I$. The *distance between v and B_ℓ* is defined as the number of bags between between B_ℓ and the right-most bag in Q that contains v. The *distance between v and B_r* is defined as the number of bags between between B_r and the left-most bag in Q that contains v.

Next, we consider the following cases based the intersection $\mathbb{O}$ and $Q_I \cup M_A$.

$\mathsf{base}(\mathbb{O}) \cap Q_I = \emptyset$ or $P(\mathbb{O}) \cap M_A \neq \emptyset$

Let us recall a few facts. $\mathbb{O}$ is an AW in G^+ that is not covered by $\mathcal{W}$. Q is a manageable clique path which is not a clique in G (and hence in G^+). The sets M_A, B_ℓ, B_r are cliques in G^+, and they separate Q_I from the rest of the graph in G^+ (and G as well). Furthermore, every vertex of M_A is adjacent to all vertices in $V(Q) \cup M_A$ in G^+ (Observation 39). The vertices of $V(Q) \setminus C_Q$, and in particular Q_I, cannot be the center vertices of any AW in G^+ that is not covered by $\mathcal{W}$ (Observation 40). Therefore,

the vertices of Q_I are either base vertices or terminals of such an AW. We now have the following easy observation.

Observation 41. *Let A be a clique in the graph and let $\mathbb{O}$ be an AW. Then $|A \cap \mathbb{O}| \leq 4$.*

From Observation 41 it follows that $|\mathbb{O} \cap (M_A \cup B_\ell \cup B_r)| \leq 12$. Let c_1, c_2 be the center vertices of $\mathbb{O}$ (in case of †-AW, $c = c_1 = c_2$). Now suppose that $\mathsf{base}(\mathbb{O}) \cap Q_I = \emptyset$. Then $\mathbb{O} \cap Q_I \subseteq \{t_\ell, t_r, t\}$. Next, suppose that there is a vertex $m \in \mathsf{base}(\mathbb{O}) \cap M_A$. Then observe that $|\mathbb{O} \cap Q_I| \leq 2$, otherwise $m \in \mathsf{base}(\mathbb{O})$ will be adjacent to 3 vertices of $\mathbb{O} \setminus \{c_1, c_2\}$. Hence, $|\mathbb{O} \cap Q_I| \leq 2$. In summary, $\mathbb{O} \cap (V(Q) \cup M_A)$ is a graph on at most 15 vertices, where up to 12 of these vertices are in $M_A \cup B_\ell \cup B_r$ and up to 3 of these vertices are in Q_I. This gives rise to the following marking scheme.

Marking Scheme IV: For each graph R on at most 15 vertices, a partition of $V(R)$ into R_B and R_I, where $|R_B| \leq 12$, and $X \subseteq M_A \cup B_\ell \cup B_r$ such that $G^+[X]$ is isomorphic to $R[R_B]$ consider the following. For each tuple $\mathcal{X} = (X, X_1, X_2, X_3)$ where $X_1 \cup X_2 \cup X_3 \subseteq X$, let $\mathcal{A}_{R,\mathcal{X}}$ be a family consisting of all subsets $Y \subseteq Q_I$, where $Y = \{y_i \mid i \in [|Y|]\}$ such that (i) $G^+[Y]$ is isomorphic to $R[R_I]$, (ii) $G^+[X \cup Y]$ is isomorphic to R, and (iii) $N_{G^+}(y_i) \cap X = X_i$ for $i \in [|Y|]$. Notice that sets in $\mathcal{A}_{R,\mathcal{X}}$ are of same size say, $y^* \leq 3$. Next, consider the matroid $\mathcal{M} = (U, \mathcal{I})$, where $U = V(G^+)$ and $\mathcal{I} = \{U' \subseteq U \mid |U'| \leq y^* + k + 2\}$. Notice that $\mathcal{M}$ is a uniform matroid, and therefore is representable over a field of size at least $y^* + k + 2$ [Oxl06]. Thus, using Theorem 2.1 we obtain a $k+2$-representable family $\widehat{\mathcal{A}_{R,X}} \subseteq^{k+2}_{rep} \mathcal{A}_{R,X}$. For every vertex contained in some set in $\widehat{\mathcal{A}_{R,X}}$, we mark a bag in Q that contains this vertex.

We observe that there are at most $\mathcal{O}(1)$ choices for the graph R and it's partition into R_B and R_I. Then there are at most $\binom{|M_A \cup B_\ell \cup B_r|}{\leq 12}$ choices for X, and $\mathcal{O}(1)$ choices for the tuple $\mathcal{X}$ for each X. The collection $\mathcal{A}_{R,\mathcal{X}}$ contains at most $\binom{|Q_S \cup Q_P|}{\leq 3}$ graphs. Now by Theorem 2.1 there are at most $\mathcal{O}((k+2)^3)$ sets in $\widehat{\mathcal{A}_{R,\mathcal{X}}}$ and each set contains $y^* \leq 3$ vertices. Hence, overall we mark at most $\boxed{\mathcal{O}((2\eta + |M|)^{12}(k+2)^3)}$ bags in Q.

Lemma 8.23. *Let S be a good set of size at most $k+2$, and $\mathbb{O}$ be an AW in $G^+ - S$ such that $\mathsf{base}(\mathbb{O}) \cap Q_I = \emptyset$ or $P(\mathbb{O}) \cap M_A \neq \emptyset$. Then there is an AW $\mathbb{O}'$ in $G^+ - S$ such that $\mathbb{O}'$ is Q-manageable. Furthermore, $\mathbb{O}' - Q_I = \mathbb{O} - Q_I$.*

Proof. As S is a good set of size at most $k+2$, it hits all sets in $\mathcal{W}$. Therefore, the obstruction $\mathbb{O}$ in $G^+ - S$ contains at least 10 vertices of M. Now consider the graph $R = \mathbb{O} \cap Q$. Let $X = V(R) \cap (M_A \cup B_\ell \cup B_r)$ and $Y = V(R) \cap Q_I$. From the previous discussions it follows that $V(R)$, X and Y contain at most $15, 12$ and 3 vertices, respectively. Let $Y = \{y_i \mid i \in [|Y|]\}$. Let X_i denote the set $X \cap N_{G^+}(y_i)$ for $i \in [|Y|]$, and $\emptyset$ otherwise. Let $\mathcal{X} = (X, X_1, X_2, X_3)$. Notice that $Y \in \mathcal{A}_{R,\mathcal{X}}$. Thus, from Theorem 2.1 there is a set $Y' \in \widehat{\mathcal{A}_{R,\mathcal{X}}}$, where $Y' = \{y'_i \mid i \in [|Y|]\}$, such that (i) $G^+[Y']$ is isomorphic to $G^+[Y]$, (ii) $G^+[X \cup Y']$ is isomorphic to R, and (iii) $N_{G^+}(y'_i) \cap X = X_i$ for $i \in [|Y|]$. Now observe that $\mathbb{O}' = (\mathbb{O} - Y) \cup Y'$ is isomorphic to $\mathbb{O}$. Hence $\mathbb{O}'$ is an AW in $G^+ - S$ such that all vertices of $\mathbb{O}$ from Q appear in marked bags of Q. Hence, $\mathbb{O}'$ is Q-manageable. Finally observe that, by construction, $\mathbb{O}' - Q_I = \mathbb{O} - Q_I$. This concludes the proof of the lemma. □

In the following subsection, we consider the problem of obtaining Q-manageable obstructions when $\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$ and $P(\mathbb{O}) \cap M_A = \emptyset$. Let us also note, in the rest of this subsection, we treat the bags marked by Marking Scheme IV as unmarked and only consider the bags marked by Marking Schemes I,II and III as marked.

$\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$ and $P(\mathbb{O}) \cap M_A = \emptyset$

We start with the following observations, which will be useful later. We let c_1 and c_2 to be the centers of $\mathbb{O}$. For the case when $\mathbb{O}$ is a †-AW, we let $c = c_1 = c_2$.

Observation 42. *If $\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$, then $c_1, c_2 \in C_Q \cup M_A$.*

Proof. Let $v \in \mathsf{base}(\mathbb{O}) \cap Q_I$. Observe that $N_{G^+}(v) \subseteq V(Q) \cup M_A$, and no vertex of $V(Q) \setminus C_Q$ can be a center vertex of an AW in G^+ (Observation 40). As c_1, c_2 must be in $N_{G^+}(v)$, therefore we have that $c_1, c_2 \in M_A \cup C_Q$. □

Observation 43. *If $\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$, then $t_\ell, t_r \notin V(Q) \cup M_A$. Furthermore, $\mathsf{base}(\mathbb{O}) \cap (C_Q \cup M_A) = \emptyset$ and $t \notin C_Q \cup M_A$.*

Proof. From Observation 42, $\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$ implies that $c_1, c_2 \in C_Q \cup M_A$. Observation 39 implies that any vertex of $C_Q \cup M_A$ is adjacent to every vertex in $V(Q)$ in G^+. Also, M_A is a clique in G^+. If $t_\ell \in V(Q) \cup M_A$, then we have the edge (t_ℓ, c_2) in $\mathbb{O}$, which is a contradiction. Hence we have that $t_\ell \notin V(Q)$. An analogous argument can be given to show that $t_r \notin V(Q)$.

Next, suppose that $w \in \mathsf{base}(\mathbb{O}) \cap (C_Q \cup M_A)$, and by assumption that $P(\mathbb{O}) \cap M_A = \emptyset$ we have $w \notin M_A$. Hence $w \in C_Q$, and by Observation 39 w is adjacent (in G^+) to every vertex in $V(Q) \cup M_A$. Let $v \in \mathsf{base}(\mathbb{O}) \cap Q_I$ and u be the neighbor of v in $P(\mathbb{O})$, which is different than w. Recall that $N_{G^+}(v) \subseteq V(Q) \cup M_A$, therefore, $u \in V(Q) \cup M_A$. But then, $P(\mathbb{O})[\{v, u, w\}]$ is a cycle on 3 vertices, contradicting that $P(\mathbb{O})$ is an induced path. Finally, if $t \in C_Q \cup M_A$, then $(t, v) \in E(G^+)$, which is a contradiction. □

It follows from Observation 43 that, if $\mathbb{O}$ is an obstruction such that $\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$ and $P(\mathbb{O}) \cap M_A = \emptyset$, then we have the following. Both endpoints of $P(\mathbb{O})$, i.e. the base terminals of $\mathbb{O}$, lie outside Q. Therefore, by Lemma 8.22 we obtain the following.

Lemma 8.24. *If $\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$ and $P(\mathbb{O}) \cap M_A = \emptyset$, then $P_Q = \mathbb{O} \cap (Q - C_Q)$ is an induced path between a vertex in $B_\ell \setminus C_Q$ and $B_r \setminus C_Q$, and $P_Q \subseteq \mathsf{base}(\mathbb{O})$.*

This leads us to the following observation.

Observation 44. *If $\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$ and $P(\mathbb{O}) \cap M_A = \emptyset$. Then $t \notin V(Q) \cup M_A$.*

The next lemma follows directly from the above results and the definition of Q-manageable obstructions.

Lemma 8.25. *Let S be a good set of at most $k+2$ vertices. Let $\mathbb{O}$ be an AW in $G^+ - S$ such that $\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$ and $P(\mathbb{O}) \cap M_A = \emptyset$. Then $\mathbb{O}$ is a Q-manageable obstruction.*

8.4.3 Nice-clique paths and nice obstructions

We now consider a pair of consecutive marked bags in K that were marked by the Marking Schemes I to IV. In particular, for each manageable clique path Q' we mark a collection of bags in Q' via Marking Scheme IV, that partition Q into sub-clique paths which are called nice-clique paths. Formally, a *nice-clique path* is a sub-clique path $K[B_\ell, B_r]$ such that B_ℓ and B_r are consecutive marked bags. We note that a nice-clique path is not a clique, since it contains at least two distinct bags of K. Further, any nice-clique path is contained in a manageable-clique path, and therefore it also has the properties of a manageable-clique path. Now, for any nice-clique path Q, that is contained in a manageable-clique path Q', we define the sets B_ℓ, B_r, Q_I, C_Q, M_A and M_P in the same way as before. Let us note that the sets M_A and M_P remain the same for both Q and Q', by Observation 39 for the manageable-clique path Q'. Furthermore, $C'_Q \subseteq C_Q$, since Q is a sub-clique-path of Q'.

Definition 8.5. An obstruction $\mathbb{O}$ is called *nice*, if for every nice-clique path Q, and $X = V(\mathbb{O}) \cap (V(Q) \setminus C_Q)$ either $X \subseteq B_\ell \cup B_r$, or $\mathbb{O}[X]$ is an induced path between a vertex in $B_\ell \setminus C_Q$ and a vertex in $B_r \setminus C_Q$, such that $(V(\mathbb{O}) \cap V(Q)) \setminus (B_\ell \cup B_r) \subseteq Q_I$.

The following lemma shows that a chordless cycle is always a nice obstruction.

Lemma 8.26. *Let S be a good set of size at most $k+2$ and let $\mathbb{O}$ be a chordless cycle in $G^+ - S$. Then $\mathbb{O}$ is a nice obstruction.*

Proof. By definition, S hits all set in $\mathcal{W}$, and therefore $|M \cap \mathbb{O}| \geq 10$. Let us consider a nice-clique path Q and suppose that $X = V(\mathbb{O}) \cap (V(Q) \setminus C_Q) \not\subseteq B_\ell \cup B_r$, i.e. there is a vertex $v \in X \cap Q_I$. Therefore, there is a pair of (distinct) vertices $m_1, m_2 \in M$ such that the path segment P between m_1 and m_2 in $\mathbb{O}$ contains the vertex v and $V(\mathbb{O}) \setminus V(P) \neq \emptyset$. Let P^* be the sub-path of P that contains v and exactly two vertices from M which are its end vertices. Note that P^* exists, and could possibly be same as P. Furthermore, P^* is an induced path in G^+. Let P^* be the path from $m_1^* \in M$ to $m_2^* \in M$. Next, we argue that $m_1^*, m_2^* \notin M_A$. As M_A is a clique (Q is not a clique and Observation 39) and P^* is an induced path in $G^+ - S$, at least one of $m_1^* \notin M_A$ or $m_2^* \notin M_A$ holds. Next, suppose that $m_1^* \in M_A$, $m_2^* \in M_P$ (the other case is symmetric), and therefore $(v, m_2^*) \notin E(G^+)$. Observe that v has no neighbor outside $V(Q) \cup M_A$ and $m_1 \in M_A$ is adjacent to all vertices in $V(Q) \cup M_A$ (Observation 39). Now let u be the neighbor of v in the sub-path of P^* from v to m_2. Observe that $u \in V(Q)$, and therefore we obtain a chord (m_1, u) in P^*, which is a contradiction. Therefore, $m_1^*, m_2^* \notin M_A$, and thus we have that $V(P^*) \cap M_A = \emptyset$. Observe that P^* satisfies the premise of Lemma 8.22, as the endpoints of P^* lie outside $V(Q)$, it contains an internal vertex from Q_I, and $V(P) \cap M_A = \emptyset$. Therefore, $P^* \cap Q$ is an induced path from a vertex in $B_\ell \setminus C_Q$ to a vertex in $B_r \setminus C_Q$ such that $P^* - (B_\ell \cup B_r)$ is an induced path contained in Q_I. Finally, as this argument holds for every nice-clique path Q, the lemma follows. □

Lemma 8.27. *Let S be a good set of size at most $k+2$ such that $G^+ - S$ has an obstruction $\mathbb{O}$ (that is not covered by $\mathcal{W}$). Then there is a nice obstruction $\mathbb{O}'$ in $G^+ - S$ (that is not covered by $\mathcal{W}$).*

Proof. As S is a good set of size at most $k+2$, it hits all sets in $\mathcal{W}$. Therefore, $\mathbb{O}$ contains at least 10 vertices from M. If $\mathbb{O}$ is a chordless cycle, then by Lemma 8.26, it is a nice

obstruction. Otherwise $\mathbb{O}$ is an AW, and suppose that it is not a nice obstruction. Let $\mathbb{O}'$ be an obstruction in $G^+ - S$ that is Q'-manageable for every manageable-path Q'. It is obtained by iteratively applying Lemma 8.23 or Lemma 8.25 for every manageable-clique path Q', depending on the sets $\mathsf{base}(O) \cap Q'_I$ and $P(\mathbb{O}) \cap M_A$. Note that, each application of these lemmas modifies the obstruction only within the corresponding manageable-clique path.

We claim that $\mathbb{O}'$ is a nice obstruction in $G^+ - S$. Consider a nice-clique path $Q = Q'[B_i, B_j]$ (i.e. it is nice-clique path that is contained in the manageable-clique path Q', and it the sub-clique path between a pair of consecutive marked bags B_i, B_j in Q'). We must show that $V(\mathbb{O}') \cap (V(Q) \setminus C_Q)$ is either a subset of $B_i \cup B_j$, or an induced path between a vertex in $B_i \setminus C_Q$ and a vertex in $B_j \setminus C_Q$ such that $(V(\mathbb{O}) \cap V(Q)) \setminus (B_\ell \cup B_r) \subseteq Q_I$. Here, $C_Q = B_i \cap B_j$. We note that the partition of M into M_A and M_P that be obtain with respect to Q' is same as the partition that we obtain with respect to Q. Thus we deal with the partition of M into M_A and M_P which is defined by Q'. First suppose that either $\mathsf{base}(\mathbb{O}) \cap Q'_I \neq \emptyset$ or $P(\mathbb{O}) \cap M_A = \emptyset$. Then, as $\mathbb{O}$ is Q'-manageable by Lemma 8.23, it follows that $V(\mathbb{O}) \cap (V(Q') \setminus C'_Q)$ is always contained in marked bags of Q'. Therefore $V(\mathbb{O}) \cap (V(Q) \setminus C_Q) \subseteq B_i \cup B_j$.

Otherwise, $\mathsf{base}(\mathbb{O}) \cap Q'_I \neq \emptyset$ and $P(\mathbb{O}) \cap M_A = \emptyset$. Then by Lemma 8.25, $\mathbb{O}'$ is Q-manageable for the manageable-clique path Q'. Therefore, from Lemma 8.24 we know that $P = \mathbb{O}' \cap (Q' - C_{Q'})$ is an induced path from a vertex in $B'_\ell \setminus C_{Q'}$ to a vertex in $B'_r \setminus C_{Q'}$, where B'_ℓ and B'_r are the first and the last bags of Q' respectively, and $C'_Q = B'_\ell \cap B'_r$. Observe that P visits every bag in $Q = Q'[B_i, B_j]$. Moreover, $P \subseteq \mathsf{base}(\mathbb{O}')$ and by Observation 42 $\{c_1, c_2\} \subseteq C'_Q \cup M_A$. And Observation 43 implies that $t_\ell, t_r \notin V(Q')$ (the endpoints of $P(\mathbb{O})$). Also note that $C'_Q \subseteq C_Q$ by definition. Now, if $V(P) \cap (V(Q) \setminus C_Q) \subseteq B_i \cup B_j$, then $\mathbb{O}'$ is Q-nice. Otherwise, $V(P) \cap Q_I \neq \emptyset$. Now observe that the path $P(\mathbb{O})$ and Q satisfy the following conditions: (i) $P(\mathbb{O})$ contains has an internal vertex from Q_I, (ii) $V(P(\mathbb{O})) \cap M_A = \emptyset$, (iii) $V(P(\mathbb{O})) \cap (V(G^+) \setminus Q_I) \neq \emptyset$ and (iv) the endpoints of $P(\mathbb{O})$ lie outside Q_I. Thus, $P(\mathbb{O})$ satisfies the premise of Lemma 8.22. Therefore, $P(\mathbb{O}) \cap (Q \setminus C_Q)$ is an induced path from a vertex in $B_i \setminus C_Q$ to a vertex in $B_j \setminus C_Q$ such that $(V(P(\mathbb{O}')) \cap V(Q)) \setminus (B_i \cup B_j) \subseteq Q_I$. Hence $V(\mathbb{O}') \cap (V(Q) \setminus C_Q)$ is an induced path between a vertex in $B_i \setminus C_Q$ and a vertex in $B_j \setminus C_Q$ such that $(V(\mathbb{O}) \cap V(Q)) \setminus (B_\ell \cup B_r) \subseteq Q_I$. Thus, $\mathbb{O}$ is Q-nice. This concludes the proof. □

Let us remark that the proof of Lemma 8.27, Definition 8.5 and earlier results show the following corollary.

Corollary 8.3. *If $\mathbb{O}$ is a nice obstruction in G^+ that is not covered by $\mathcal{W}$, then for any nice-clique path Q, $\mathbb{O} \cap Q$ is either a subset of $B_\ell \cup B_r$, or $\mathbb{O} \cap (Q - C_Q)$ is an induced path between a vertex in $B_\ell \setminus C_Q$ and a vertex in $B_r \setminus C_Q$ that contains a vertex of Q_I. Furthermore, in the second case and when $\mathbb{O}$ is an AW, $V(\mathbb{O}) \cap (C_Q \cup M_A) = \{c_1, c_2\}$, and $(V(\mathbb{O}) \cap V(Q)) \setminus (B_\ell \cup B_r)$ is an induced path in Q_I which is a sub-path of $\mathsf{base}(\mathbb{O})$. Here, B_ℓ and B_r are the first and last bags of Q, respectively.*

Let us note that the induced path $V(\mathbb{O}) \cap (V(Q) \setminus C_Q)$, in the above corollary, is disjoint from C_Q. Furthermore, this path lies in $P(\mathbb{O})$ whenever $\mathbb{O}$ is an AW. This follows from the proof of Lemma 8.27. We will require a strengthening of the above corollary that allows us to "replace" the path $P = \mathbb{O} \cap (Q - C_Q)$ in $\mathbb{O}$ with another path P' between the endpoint bags of Q and obtain a new obstruction. To obtain this property,

we need to further partition a nice-clique path by marking the following collection of bags.

Marking Scheme V. For every nice-clique path Q with endpoint bags B_ℓ, B_r, mark every pair of adjacent bags B, B' in Q such that $B \cap B_\ell \supsetneq B' \cap B_\ell$, or $B \cap B_r \subsetneq B' \cap B_r$.

Clearly, we mark at most 4η bags for each nice-clique path Q. Recall that we have at most $64\eta^3|M|$ manageable-clique paths and for each manageable-clique path we marked at most $\mathcal{O}((2\eta + |M|)^{12}(k+2)^3)$ bags using Marking Scheme IV. Hence, in Marking Scheme V we mark at most $\boxed{\mathcal{O}(\eta^{16}|M|k^3)}$ bags. Here, we rely on the fact that $\eta > |M|$.

Observation 45. *Let B_i, B_j be a pair of consecutive marked bags in a nice-clique path Q. Then for every $B \in Q[B_i, B_j]$, we have $B_i \cap B_\ell = B \cap B_\ell = B_j \cap B_\ell$, and $B_i \cap B_r = B \cap B_r = B_j \cap B_r$. Here, B_ℓ and B_r are endpoint bags of Q.*

Let us review the structural results we have obtained till now. Let $\mathbb{O}$ be a nice AW in G which is not covered by $\mathcal{W}$. Then $\mathbb{O}$ is also a nice AW in G^+. Observe that the converse holds as well, i.e. if $\mathbb{O}$ is a nice AW in G^+ then it is a nice AW in G. Next, the terminal vertices and center vertices of $\mathbb{O}$ either lie in marked bags in K, or lie in the modulator M, or lie outside K (from Definition 8.5, Lemma 8.26 and Lemma 8.27). Let Q be a nice-clique path such that $\mathbb{O} \cap (Q - C_Q)$ is an induced path P between a vertex in $B_\ell \setminus C_Q$ and a vertex in $B_r \setminus C_Q$ that contains a vertex in $Q - (B_\ell \cup B_r)$. Here, B_ℓ and B_r are endpoint bags of Q. Note that the vertices in $Q - (B_\ell \cup B_r)$ are unmarked vertices up till Marking Scheme IV, and therefore it lies in $\mathsf{base}(\mathbb{O})$. From our arguments in Lemma 8.27, we have the following properties. As P contains an unmarked vertex (up till Marking Scheme IV), $\mathsf{base}(\mathbb{O}) \cap Q_I \neq \emptyset$ and $P(\mathbb{O}) \cap M_A = \emptyset$. The vertices of P lie in $P(\mathbb{O})$. Furthermore, the internal vertices of P lie in $\mathsf{base}(\mathbb{O})$ and $P - (B_\ell \cup B_r)$ is an induced path contained in Q_I. The centers c_1, c_2 of $\mathbb{O}$ lie in $C_Q \cup M_A$. The shallow terminal t of $\mathbb{O}$ lies outside $V(Q)$ as P visits every bag in Q. Consider any pair of consecutive marked bags B_i, B_j in Q under Marking Scheme V. Let Q_{ij} denote the subclique path $Q[B_i, B_j]$, and let C_{ij} denote the set $B_i \cap B_j$. Consider the path $P_{ij} = P \cap Q_{ij}$ and suppose that P_{ij} contains a vertex in $Q_{ij} - C_{ij}$. Then, as P_{ij} is an induced path, it is disjoint from C_{ij}, which is part of every bag of Q_{ij}. Therefore, P_{ij} contains no vertex in $B_\ell \cup B_r$ (since $Q_{ij} \cap (B_\ell \cup B_r) \subseteq C_{ij}$). In other words $P_{ij} \subseteq Q_I$. Furthermore, as every vertex of P_{ij} is an internal vertex of $P \subseteq P(\mathbb{O})$, we have $P_{ij} \subseteq \mathsf{base}(\mathbb{O})$. Let $u \in B_i$ and $v \in B_j$ be the endpoints of P_{ij}. Now, let P'_{ij} be another induced path between u and v in $Q[B_i, B_j] - C_{ij}$, and observe that $P'_{ij} \subseteq Q_I$. Let us note that when $\mathbb{O}$ is a nice chordless cycle, we define the paths P, P_{ij} and P'_{ij} in a similar manner. We then have the following lemma.

Lemma 8.28. *There is a nice obstruction $\mathbb{O}'$ which is not covered by $\mathcal{W}$ such that $\mathbb{O}' \subseteq (\mathbb{O} - P_{ij}) \cup P'_{ij}$.*

Proof. Since every vertex of P'_{ij} lies in Q_I, the neighborhood of these vertices is contained in $V(Q) \cup M_A$. Recall that $P = \mathbb{O} \cap (Q - C_Q)$ is an induced path from a vertex say, $x \in B_\ell$ to a vertex say, $y \in B_r$ that contains a vertex in $Q - (B_\ell \cup B_r)$ and $P \subseteq \mathsf{base}(\mathbb{O})$. By x_o and y_o we denote the neighbors of x and y in P, respectively. Let P' be an induced path between x and y in $G[(V(P) \setminus V(P_{ij})) \cup V(P'_{ij})]$. Also, let $\mathbb{O}' = G[(V(\mathbb{O}) \setminus V(P)) \cup V(P')]$,

and x_c and y_c be the neighbors of x and y in P', respectively. Notice that all the internal vertices of P' except possibly x_c and y_c are contained in Q_I. Moreover, if $x_c \notin Q_I$ then $x_c \in B_\ell$ and hence $x_c = x_o$, which follows from Observation 45 and the fact that P_{ij}, P'_{ij} do not contain any vertex from $B_\ell \cup B_r$. Similarly, if $y_c \notin Q_I$ then $y_c \in B_r$ and $y_c = y_o$. Let $T = (N_{P'}[x] \cap B_\ell) \cup (N_{P'}[y] \cap B_r)$. The above discussion implies that $P_f = G[(V(P(\mathbb{O})) \setminus V(P)) \cup V(P')]$ is an induced path from t_ℓ to t_r. Here, t_ℓ and t_r are the base terminals of $\mathbb{O}$ if $\mathbb{O}$ is an AW, and otherwise $\mathbb{O}$ is a chordless cycle in which case t_ℓ and t_r are neighbors of x and y, respectively in $\mathbb{O}$ which are different from their neighbors in P. Notice that for each $w \in V(P') \setminus T$ we have $N(w) \subseteq V(Q) \cup M_A$. Therefore, from the above discussions, if $\mathbb{O}$ is a chordless cycle then $\mathbb{O}'$ is a chordless cycle on at least 4 vertices. Here, we rely on the fact that there are at least 10 vertices from M in $\mathbb{O}$ and they all lie in M_P. Next, consider the case when $\mathbb{O}$ is an AW. Notice that as P contains a vertex in $Q_{ij} - C_{ij}$, therefore the centers c_1, c_2 of $\mathbb{O}$ belong to $C_Q \cup M_A$ (see Observation 40). This implies that each vertex in P' is adjacent to c_1 and c_2. Finally, recall that there are at least 4 vertices in $\mathbb{O} - Q$, as M is a 9-redundant solution and $\mathbb{O}$ is not covered by $\mathcal{W}$. Now it follows that $\mathbb{O}' = \mathbb{O} - P \cup P'$ is also an obstruction. In each of the cases, by construction we have that $\mathbb{O}'$ is Q-nice and is not covered by $\mathcal{W}$. Moreover, $\mathbb{O} - V(Q) = \mathbb{O}' - V(Q)$. Therefore, it follows that $\mathbb{O}$ is a nice obstruction which is not covered by $\mathcal{W}$. □

In the following, by a *separator*, we mean a $B_i \setminus C_{ij}$ and $B_j \setminus C_{ij}$ separator in $Q_{ij} - C_{ij}$. Let us note that any such separator in G is also a separator in G^+, and vice-versa. We apply the above lemma (Lemma 8.28) to derive the fact that any minimal solution either does not intersect $V(Q_{ij}) \setminus C_{ij}$ or contains a separator. Furthermore, we can replace this separator with any other separator and the resulting set is also a solution. For a set $S \subseteq V(G)$, by S_{ij} we denote the set $S \cap (V(Q_{ij}) \setminus C_{ij})$.

Lemma 8.29. *Let S be a solution of size at most $k+2$ in G (or G^+) such that it contains a vertex in $V(Q_{ij}) \setminus C_{ij}$. Then either S_{ij} is a separator, or else $S \setminus S_{ij}$ is also a solution. Furthermore, if S_{ij} is a separator then for any separator S^*_{ij} such that $S^* = (S \setminus S_{ij}) \cup S^*_{ij}$ has size at most $k + 2$, the set S^* is also a solution.*

Proof. First suppose that S_{ij} is not a separator. As S is a solution of size at most $k + 2$ in G, it hits all the sets in $\mathcal{W}$. Hence $G - S = G^+ - S$, i.e. S is a solution in G^+. By our assumptions, S does not separate $B_i \setminus C_{ij}$ and $B_j \setminus C_{ij}$ in $Q_{ij} - C_{ij}$. Let $S' = S \setminus S_{ij}$. If S' is not a solution, there is an obstruction $\mathbb{O}'$ in the graph $G - S'$, and note that $\mathbb{O}'$ contains a vertex of $S \setminus S' \subseteq V(Q_{ij}) \setminus C_{ij}$. Since $M \cap S = M \cap S'$, it follows that S' also hits all the sets in $\mathcal{W}$. Hence, $|\mathbb{O}' \cap M| \geq 10$ and $\mathbb{O}'$ is an obstruction in $G^+ - S'$. Therefore by Lemma 8.27, we obtain a nice obstruction $\mathbb{O}$ in $G^+ - S'$, which is also present in $G - S'$. Here, we rely on the fact that $G^+ - S' = G - S'$ which is implied by the fact that S' is a hitting set for $\mathcal{W}$. Now we consider the obstruction $\mathbb{O}'$ in the graph G^+. Clearly, $\mathbb{O}$ also contains a vertex in $S \setminus S'$, and by Corollary 8.3, it follows that $\mathbb{O} \cap (Q - C_Q)$ is a path P between a vertex of $B_\ell \setminus C_Q$ and a vertex of $B_r \setminus C_Q$ that is disjoint from C_Q. Let $P_{ij} = P \cap Q_{ij}$ and note that P_{ij} contains a vertex of $S \setminus S' \subseteq V(Q_{ij}) \setminus C_{ij}$. Let $u \in B_i$ and $v \in B_j$ be the endpoints P_{ij}. Since S_{ij} is not a separator, there is an induced path P'_{ij} between u and v in $Q_{ij} - C_{ij}$ that is disjoint from S. Here, we rely on the fact that B_ℓ and B_r are cliques. Now, by Lemma 8.28, we have an obstruction $\mathbb{O}'' \subseteq G[(V(\mathbb{O}) \setminus V(P_{ij})) \cup V(P'_{ij})$ that is not covered by $\mathcal{W}$ in G^+ (and

G), and by construction it is disjoint from S. But this is a contradiction. Therefore S' must also be a solution.

Now suppose that S_{ij} is a separator. We now argue that if S^*_{ij} is another separator, such that $S^* = (S \setminus S_{ij}) \cup S^*_{ij}$ has size at most $k+2$, then S^* is also a solution. Suppose not, then we can argue that there is a nice obstruction $\mathbb{O}$ in $G - S^*$, such that it contains a vertex of $S_{ij} \subseteq V(Q_{ij}) \setminus C_{ij}$. Then, as before, we obtain a path P_{ij} in $G[V(\mathbb{O}) \cap (V(Q_{ij}) \setminus C_{ij})]$ between a vertex in $B_i \setminus C_{ij}$ and a vertex in $B_j \setminus C_{ij}$ in $Q_{ij} - C_{ij}$. But this contradicts the fact that S^*_{ij} is a separator. Therefore S^* is also a solution. □

Corollary 8.4. *(i) If S is a minimal solution in G (or G^+) and $|S| \leq k+2$, then S_{ij} is either a minimal separator or $\emptyset$.*

(ii) If S is an optimum solution in G (or G^+) and $|S| \leq k+2$, then S_{ij} is either a minimum separator or $\emptyset$.

For each pair of consecutive bags B_i, B_j, let us select a minimum sized separator S^*_{ij}. Then we have the following lemma which follows from the proof of Lemma 8.29.

Lemma 8.30. *Let S be any minimal solution to the instance G (or G^+) of size at most $k+2$. Then there is a solution S' of cardinality at most $|S|$ such that $S' \cap (V(Q_{ij}) \setminus C_{ij})$ is either the empty set or S^*_{ij}.*

We call the solutions of the above form *good solutions*. Let us recall the following fact about interval graphs and their clique-path decomposition. In a clique path, any separator is intersection of two adjacent bags. Observe that, by definition $S^*_{ij} \cup C_{ij}$ is a separator in Q (and more generally in K). We now have the following marking rule.

Marking Scheme VI. For each pair of consecutive marked bags B_i, B_j in Q, mark a pair of bags $B, B' \in Q_{ij}$ such that $B \cap B' = S^*_{ij} \cup C_{ij}$.

Following this marking scheme, we consider the problem of reducing the set of unmarked vertices in Q_{ij}, where Q is a nice-clique path and B_i, B_j are two consecutive marked bags.

Lemma 8.31. *Let v be an unmarked vertex in Q_{ij} such that v is contained in only one bag. Then (G, k) is a yes instance of* IVD *if and only if $(G - \{v\}, k)$ is a yes instance of* IVD*.*

Proof. In the forward direction, let S be a solution in G of size at most k. Clearly, S is a solution in $G - \{v\}$ as well. Now we consider the reverse direction. Let S be a solution of size at most k in $G - \{v\}$ and suppose that it is not a solution in G. Observe that $S \cup \{v\}$ is a solution in G of cardinality at most $k+1$, and therefore it hits each set in $\mathcal{W}$. Hence $G - (S \cup \{v\}) = G^+ - (S \cup \{v\})$ and $S \cup \{v\}$ is also a solution in G^+. Further observe that, as $v \notin M$, we have that S hits every set in $\mathcal{W}$. This implies that $G - S = G^+ - S$. Now consider an obstruction $\mathbb{O}$ in $G - S$, and clearly it includes v. It follows that the obstruction $\mathbb{O}$ is also present in $G^+ - S$, and furthermore $V(\mathbb{O}) \cap M$ contains at least 10 vertices as $\mathbb{O}$ is not covered by $\mathcal{W}$. Let us consider $\mathbb{O}$ in the graph G^+ along with the good set S. Observe that $N(v) \subseteq B \cup M_A$, where B is the (unique) bag containing v, and $(B \cup M_A) \setminus S$ is a clique in $G^+ - S$ (using Observation 39). Therefore,

$\mathbb{O}$ is not a chordless cycle, and hence $\mathbb{O}$ is an AW. Now, by Lemma 8.27, there is a nice obstruction $\mathbb{O}'$ in $G^+ - S$, and note that all terminals of $\mathbb{O}'$ lie in marked bags. If $v \in \mathbb{O}'$, then as v is an unmarked vertex, by Observation 40 and Corollary 8.3, v lies in $\mathsf{base}(\mathbb{O})$ and therefore $N(v)$ must contain a pair of non-adjacent vertices. But this is a contradiction. Hence, v is not part of the obstruction $\mathbb{O}'$. This implies that that $\mathbb{O}'$ is an obstruction in $G^+ - (S \cup \{v\})$ (and in $G - (S \cup \{v\})$ as well). But then $\mathbb{O}'$ is also present in $(G - \{v\}) - S$, which is also a contradiction. Hence, S must also be a solution in G. This concludes the proof of this lemma. □

The above lemma gives the following reduction rule.

Reduction Rule 8.7. *Let Q be a nice-clique path, and let B_i, B_j a pair of consecutive marked bags. Then pick a unmarked vertex in $Q[B_i, B_j]$ that is contained in only one bag, and delete it from the graph G. The resulting instance is $(G - \{v\}, k)$.*

If the above reduction rule is not applicable, then there are no unmarked vertices in any nice-clique path Q that are contained in only one bag. Then observe that for any bag unmarked B in Q we have $B = (B \cap B^{-1}) \cup (B \cap B^{+1})$. Let us now consider the remaining of the unmarked vertices in Q_{ij}.

Lemma 8.32. *Let Q_{ij} contain an unmarked vertex. Then there is an edge (u, v) such that at least one of its endpoint is an unmarked vertex, and there is only one bag in Q_{ij} that contains this edge.*

Proof. Let us traverse in $Q[B_i, B_j]$ from B_i, and let B be the first bag in Q_{ij} that contains an unmarked vertex. Let us partition the bag B into three parts as follows, $B_2 = B^{-1} \cap B^{+1} \subseteq B$, $B_1 = (B \cap B^{-1}) \setminus B_2$ and $B_3 = B \cap B^{+1} \setminus B_2$. Note that, $B \cap B^{-1} = B_1 \cup B_2$, and $B \cap B^{+1} = B_2 \cap B_3$. Furthermore, if $B_1 = \emptyset$ then $B = B_2 \cup B_3 \subseteq B^{+1}$, which is a contradiction as B is a maximal clique in the clique path Q, and hence $B \not\subseteq B^{+1}$. Therefore $B_1 \neq \emptyset$, and similarly $B_3 \neq \emptyset$. Now consider an unmarked vertex $u \in B$ and observe that $u \in B_3$. Next we choose a vertex $v \in B_1$ and clearly it is distinct from u. Furthermore, as $v \notin B^{+1}$ and $u \notin B^{-1}$, we have that the edge (u, v) is present only in B. □

In the following, we select an edge $e = (u, v)$ given by the above lemma (Lemma 8.32), that lies in Q_{ij} for some pair of consecutive marked bags B_i, B_j in the nice-clique path Q. We call such an edge an *irrelevant edge*. Note that, by construction, $u, v \notin C_{ij}$ and therefore they belong to Q_I.

Lemma 8.33. *Let (u, v) be an irrelevant edge in Q_{ij}. Then there is no minimal separator of $B_i \setminus C_{ij}$ and $B_j \setminus C_{ij}$ in $Q_{ij} - C_{ij}$ that contains both u and v.*

Proof. Recall that $Q_{ij} - C_{ij}$ is a clique path (possibly disconnected) with endpoint bags $B_i - C_{ij}$ and $B_j - C_{ij}$. Therefore every minimal separator of these endpoint bags is the intersection of a pair of adjacent bags in $Q_{ij} - C_{ij}$. If both u and v were in a minimal-separator, then the edge (u, v) appears in at least two bags, which is a contradiction. Therefore, there is no minimal separator that contains both u and v. □

Observation 46. *A minimal solution of size at most $k + 2$ in G (or G^+) contains at most one of u and v, where (u, v) is an irrelevant edge.*

Proof. Let S be a minimal solution in G that contains both of u and v. As before, we conclude that S is also a solution in G^+. Then, as S contains a vertex of $V(Q_{ij}) \setminus C_{ij}$, by Corollary 8.4 we have $S_{ij} = S \cap (V(Q_{ij}) \setminus C_{ij})$ is a minimal separator. Now, by our assumptions, S_{ij} contains both u and v, whereas by Lemma 8.33, no minimal separator can contain both these vertices. This is a contradiction. □

Let us also recall that G^+ is a super-graph obtained by adding additional edges in M for sets of size 2 in $\mathcal{W}$. Therefore, any induced subgraph of G^+ that is not covered $\mathcal{W}$ is present in G and vice-versa. In particular, any edge in G^+ whose end-points don't form a set of $\mathcal{W}$, is also present in G.

Lemma 8.34. *Let $e = (u, v)$ be an irrelevant edge in $Q_{ij} - C_{ij}$, where u is an unmarked vertex. Then (G, k) is a yes instance of* IVD *if and only if $(G/e, k)$ is a yes instance of* IVD*.*

Proof. Let z^* denote the vertex obtained by contracting the irrelevant edge $e = (u, v)$. Let S be a solution of size k in G. Observe that, we can assume S is a minimal solution, and therefore it does not contain both u and v. Let $S' = (S \setminus \{u, v\}) \cup \{z^*\}$ whenever S contains at least one of u, v and $S' = S$ otherwise. In the first case, observe that $G/e - S'$ is isomorphic to $G - (S \cup \{u, v\})$. And in the second case $G/e - S'$ is isomorphic to $(G - S)/e$. As interval graphs are closed under edge-contractions and vertex deletions (Observation 32), we have that S' is a solution in G/e of size at most k.

Now suppose that S' is a solution of size at most k in G/e. We have two cases depending on whether or not $z^* \in S'$. First consider the case when $z^* \in S'$. Then $S = (S' \setminus \{z^*\}) \cup \{u, v\}$ is a solution of size $k + 1$ in G, as $G - S$ is isomorphic to $G/e - S'$. Now, as S is a solution of size at most $k + 1$, it must hit each set in $\mathcal{W}$. Therefore, $G^+ - S = G - S$, i.e. S is a solution in G^+. Furthermore $S \setminus \{u, v\}$ hits each set in $\mathcal{W}$, as $u, v \notin M$. Let us now consider the graph G^+ and the good set (also a solution) S of size at most $k + 1$ in it. Observe that S contains an unmarked vertex in Q_{ij} (since $u, v \in Q_{ij}$). Therefore by Lemma 8.29, it follows that either there is a strict subset S'' of S that is also a solution, or $S_{ij} = S \cap (V(Q_{ij}) \setminus C_{ij})$ is a separator of $B_i \setminus C_{ij}$ and $B_j \setminus C_{ij}$ in $Q_{ij} - C_{ij}$. In the first case, we obtain a solution $S^+ = S''$ in G^+ of size at most k. In the second case, observe that S_{ij} cannot be a minimal separator, as that will contradict Lemma 8.33. Therefore, as $u, v \in S_{ij}$, there is a strict subset S'_{ij} that includes at most one of u an v, which is also separator. Then by Lemma 8.29, $S^* = (S \setminus S_{ij}) \cup S'_{ij}$ is also a solution in G^+, and note that it has size at most k. Hence we have obtained a solution S^* in G^+ of size at most k. Further observe that S^* hits every set in $\mathcal{W}$. Therefore $G^+ - S^* = G - S^*$, i.e. S^* is a solution of size at most k in G.

Now consider the case when $z^* \notin S'$. In this case, let $S = S' \cup \{u, v\}$, and observe that it has size $k + 2$. As $G - S$ is isomorphic to $G/e - (S' \cup \{z^*\})$, we have that S is a solution in G. As $\mathcal{W}$ is a (k+2)-necessary set, S hits each set in $\mathcal{W}$, which then implies that S' hits each set in $\mathcal{W}$. Therefore $G - S' = G^+ - S'$. Also note that S' is a good set in G^+ of cardinality at most k. Now we claim that S' is a solution of cardinality k in G. Suppose not and let there be an obstruction $\mathbb{O}'$ in $G - S'$. As S' hits $\mathcal{W}$, we have that $\mathbb{O}'$ is not covered by $\mathcal{W}$ and $\mathbb{O}' \cap M$ contains at least 10 vertices. Further note that $\mathbb{O}'$ is also present in the graph $G^+ - S'$. Now consider the graph G^+, the obstruction $\mathbb{O}'$ and the good set S'. By Lemma 8.27, there is a nice obstruction $\mathbb{O}$ in $G^+ - S'$ and note

that is not covered by $\mathcal{W}$. Also note that, since $G - S' = G^+ - S'$ and $V(\mathbb{O}) \cap S' = \emptyset$, the properties of $\mathbb{O}$ derived in G^+ also hold in G.

First suppose that $V(\mathbb{O}) \cap \{u, v\} = \emptyset$. Then clearly $\mathbb{O}$ is present in G/e, and furthermore it is disjoint from S'. This is a contradiction. Next, suppose that $V(\mathbb{O}) \cap \{u, v\}$ is one of u or v. We claim that $G/e[(V(\mathbb{O}) \setminus \{u, v\}) \cup \{z^*\}]$ contains an obstruction. Now consider the obstruction $\mathbb{O}$ in the graph G^+ and the good set S'. And note that $\mathbb{O}$ is not covered by $\mathcal{W}$. As $u, v \in V(Q_{ij}) \setminus C_{ij}$, they lie in Q_I. Therefore $N(u) \cup N(v) \subseteq V(Q) \cup M_A$, and hence u, v have no neighbors in $V(\mathbb{O}) \setminus (V(Q) \cup M_A)$. Now, as $P = \mathbb{O} \cap (Q - C_Q)$ contains a vertex from Q_I, by Corollary 8.3, P must be an induced path between a vertex in $B_\ell \setminus C_Q$ and a vertex in $B_r \setminus C_Q$ such that $P - (B_\ell \cup B_r)$ is an induced path contained in Q_I. Let us also note that P must contain at least 3 vertices. Now we have two following cases.

- Consider the case when $\mathbb{O}$ is a chordless cycle in G^+ (and G as well). As $\mathbb{O}$ is a nice obstruction we have $|V(\mathbb{O}) \cap M| \geq 10$. And as P contains at least 3 vertices, $V(\mathbb{O}) \cap M_A = \emptyset$ (using Observation 39). Hence $(N(u) \cup N(v)) \cap (V(\mathbb{O}) \cap M) = \emptyset$ in G^+ (and G as well). Now we can conclude that $G/e[(V(\mathbb{O}) \setminus \{u, v\}) \cup \{z^*\}]$ contains a chordless cycle, by considering a vertex $m \in V(\mathbb{O}) \cap M$ and the two induced paths between m and z^* in $G/e[(V(\mathbb{O}) \setminus \{u, v\}) \cup \{z^*\}]$.

- Next, we consider the case when $\mathbb{O}$ is an AW in G^+ (and G as well). As $\mathbb{O}$ is a nice obstruction that contains a vertex from Q_I, it follows that $P \subseteq P(\mathbb{O})$ and $P \cap Q_I \subseteq \mathsf{base}(\mathbb{O})$ (see the proof of Lemma 8.27, Lemma 8.28 and Lemma 8.24). Note that $\{u, v\} \cap P \subset \mathsf{base}(\mathbb{O})$. Furthermore, $P(\mathbb{O}) \cap M_A = \emptyset$, as P contains at least 3 vertices and any vertex in M_A is adjacent to every vertex in Q (using Observation 39). Also note that the shallow terminal t lies outside Q, as P visits every bag in Q (using Corollary 8.3). Hence, $V(\mathbb{O}) \cap (V(Q) \cup M_A) = V(P) \cup \{c_1, c_2\}$. Therefore $(N(u) \cup N(v)) \cap V(\mathbb{O}) \subseteq V(P) \cup \{c_1, c_2\}$ in G^+ (and G as well). Furthermore $\{c_1, c_2\} \subseteq C_Q \cup M_A$ (see the proof of Lemma 8.27). And since $|\mathsf{base}(\mathbb{O}) \cap M| \geq 5$, we have that P is a strict subset of $P(\mathbb{O})$. Therefore, $(N(u) \cup N(v)) \cap (V(\mathbb{O}) \setminus \{c_1, c_2\})$ is a strict subset of $V(P(\mathbb{O}))$ and $u, v \in N(c_1) \cap N(c_2)$. Hence $G/e[(V(P(\mathbb{O})) \setminus \{u, v\}) \cup \{z^*\}]$ contains an induced path from t_ℓ to t_r with at least 6 internal vertices, where t_ℓ and t_r are base terminals of $\mathbb{O}$. Now it follows that $G/e[(V(\mathbb{O}) \setminus \{u, v\}) \cup \{z^*\}]$ contains an AW of the same type as $\mathbb{O}$. Further observe that this obstruction lies in $G/e - S'$, which is a contradiction.

Now we consider the case that both u, v are present in $\mathbb{O}$. We claim that $\mathbb{O}/e$ is an obstruction in G/e. Indeed, if $\mathbb{O}$ is a chordless cycle, then as it contains at least 10 vertices in M, it follows that $\mathbb{O}/e$ is also a chordless cycle on at least 9 vertices. Otherwise, $\mathbb{O}$ is a nice AW. Now, recall that u is an unmarked vertex in $Q_{ij} \subseteq Q$ and observe that the vertex u lies in Q_I. Let $P = \mathbb{O} \cap (Q - C_Q)$ and observe that $P \cap Q_I \neq \emptyset$. Therefore, by Corollary 8.3, we have that P is an induced path between a vertex in $B_\ell \setminus C_Q$ and a vertex in $B_r \setminus C_Q$, such that $P - (B_\ell \cup B_r)$ is an induced path contained in Q_I. Let $P_{ij} = P \cap Q_{ij}$ and observe that it contains the edge (u, v). Furthermore, $P_{ij} \cap (B_\ell \cup B_r) = \emptyset$ by the construction of Q_{ij} (from the marking schemes). Therefore all vertices in P_{ij}, and in particular the vertices u and v, must be internal vertices of $P(\mathbb{O})$, i.e. they are in $\mathsf{base}(\mathbb{O})$ (see the proof of Lemma 8.27). Finally, recall that $\mathsf{base}(\mathbb{O})$ contains at least 5 vertices of M. Therefore, $P(\mathbb{O})/e$ is an induced path on

at least 6 vertices between t_ℓ and t_r. Hence, it follows that $\mathbb{O}/e$ is an AW of the same type as $\mathbb{O}$, and further it is present in G/e. Finally observe that $\mathbb{O}/e$ is an obstruction in G/e that is disjoint from S'. This is a contradiction.

Having obtained a contradiction in all cases, we must conclude that S' is a solution in G, and recall that it has size at most k. This concludes the proof of this lemma. □

The above lemma (Lemma 8.34) gives us the following reduction rule.

Reduction Rule 8.8. *Let (u, v) be an irrelevant edge in $Q_{ij} - C_{ij}$ where u is an unmarked vertex. Then contract the edge (u, v) in the graph G. The resulting instance is $(G/e, k)$.*

By Reduction Rule 8.8, we may assume that there are no unmarked vertices in Q_{ij}. Then applying this reduction rule over all pairs of consecutive marked bags in every nice-clique path, we conclude that all vertices in the clique path K are marked. Finally, we apply the above marking schemes and reduction rules for every clique path in $G - M$, and conclude that all the vertices in $G - M$ are marked. We now proceed to bounding the number of vertices in the graph.

8.5 The Number of Vertices in the Kernel

Suppose that none of the reduction rules apply to the given instance. Then we obtain the following bound on the number of vertices in G. First recall that $|M| = \mathcal{O}(k^{10})$, and the total number of vertices in the module components of $G - M$ is bounded by $\mathcal{O}(k^3|M|^6) = \mathcal{O}(k^{63})$. Now let us bound the number of vertices in the non-module components of $G - M$. There are at most $\mathcal{O}(|M|)$ such components, and each of them is a clique-path. Consider one such component K. Recall that, the size of any clique in $G - M$ is upper bounded by $\eta = \mathcal{O}(k|M|^{10}) = \mathcal{O}(k^{101})$. Now, in Marking Scheme I, we mark at most $\mathcal{O}(\eta|M|)$ bags in K. Then for each pair of consecutive marked bags in K, Marking Scheme II and III, which consider two disjoint cases, together mark $\mathcal{O}(\eta^7|M|k^2) + \mathcal{O}(\eta^3|M|)$ bags in K. Note that the bags marked by Marking Scheme II lie in module components of K. Next, for each pair of consecutive marked bags in Marking Scheme I and III, gives us $\mathcal{O}(\eta^3|M|)$ manageable clique paths in K. For each manageable clique path we mark $\mathcal{O}(k^3\eta^{12})$ bags to partition it into nice clique paths. Hence, Marking Scheme IV marks a total of $\mathcal{O}(\eta^{15}|M|k^3)$ bags in K. Then again, for each pair of consecutive marked bags, Marking Scheme V marks $\mathcal{O}(\eta)$ bags. Hence the total number of marked bags in $G - M$ is $\mathcal{O}(|M|\eta^{16}k^3) = \mathcal{O}(k^{1629})$. Finally, Marking Scheme VI marks only 2 bags for each pair of consecutive marked bags in Marking Scheme V. Therefore, the total number of marked bags in K is $\mathcal{O}(\eta^7|M|k^2) + \mathcal{O}(\eta^{16}|M|k^3)$ which is $\mathcal{O}(\eta^{16}|M|k^3) = \mathcal{O}(k^{1629})$. Now, we can bound the total number of vertices in the clique path K as $\mathcal{O}(k^{1629}) \cdot \mathcal{O}(\eta) = \mathcal{O}(k^{1730})$. Finally, we can bound the total number of vertices contained in non-module components of $G - M$ as $\mathcal{O}(k^{1730}) \cdot \mathcal{O}(|M|) = \mathcal{O}(k^{1740})$. Clearly, this is also an upper bound on the total number of vertices in G.

Chapter 9

Split Contraction

In this chapter, we look at the problem of contracting a given graph to a split graph. Recall that a split graph is a graph whose vertex set can be partitioned into two sets, A and B, such that A is a clique while B is an independent set in the graph. The problem we study is called Split Contraction, which is formally defined below.

Split Contraction	**Parameter:** k
Input: A graph G and an integer k.	
Question: Does there exist $X \subseteq E(G)$ such that G/X is a split graph and $\lvert X\rvert \leq k$?	

It seemed plausible that Split Contraction, like $\mathcal{F}$-Edge Contraction where $\mathcal{F}$ is the family of cliques, is solvable in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$. From the proposed algorithm for the problem [GC15] it seems like the bottleneck of the problem is captured by graphs having small vertex covers. Interestingly, the first result in this chapter, given in Section 9.1, proves that it is unlikely to overcome the difficulty imposed by such graphs. In particular, we show that unless the ETH fails, Split Contraction parameterized by ℓ, the size of a minimum vertex cover of the input graph, does not have an algorithm running in time $2^{o(\ell^2)} \cdot n^{\mathcal{O}(1)}$. Here, n denotes the number of vertices in the input graph. To the best of our knowledge, under the Exponential Time Hypothesis (ETH) [IPZ01, CFK+15], this is the *first* tight lower bound of this form for problems parameterized by the vertex cover number of the input graph. Lately, there has been increasing scientific interest in the examination of lower bounds of forms other than $2^{o(s)} \cdot n^{\mathcal{O}(1)}$ for some parameters s. For example, lower bounds that are "slightly super-exponential", i.e. of the form $2^{o(s \log s)} \cdot n^{\mathcal{O}(1)}$ for various parameters s, have been studied in [LMS11]. Cygan et al. [CPP16] obtained a lower bound of the form $2^{2^{o(k)}} \cdot n^{\mathcal{O}(1)}$, where k is the solution size, for the Edge Clique Cover problem. Very recently, Marx and Mitsoue [MM16] have further obtained lower bounds of the forms $2^{2^{o(w)}} \cdot n^{\mathcal{O}(1)}$ and $2^{2^{2^{o(w)}}} \cdot n^{\mathcal{O}(1)}$, where w is the treewidth of the input graph, for choosability problems. In order to derive the lower bound result when parameterized by size of vertex cover, we make use of a partitioning of the vertex set $V(G)$ into sets $C_1, \ldots, C_t$, which is obtained by using harmonious coloring. Recall that in Chapter 4 (Section 4.2.1) we illustrated usage of such a partitioning in deriving lower bound results.

In Section 9.2 we prove the Split Contraction is W[1]-hard when parmeterized by the size of a solution. We find this result surprising: one might a priori expect that "contraction to split graphs" should be easy as split graphs have structures that seem relatively simple. Indeed, many NP-hard problems admit simple polynomial-time

algorithms if restricted to split graphs. Consequently, this result can also be viewed as a strong evidence of the inherent complexity of the edit operation which contracts edges. Furthermore, some of the ideas underlying the constructions of this reduction, such as the exploitation of properties of a special case of the Perfect Code problem to analyze budget constraints involving edge contractions, might be used to establish other W[1]-hard results for problems of similar flavors.

We design a (standalone) FPT algorithm for Split Contraction that runs in time $2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$, where ℓ is the size of the minimum vertex cover in the input graph (Section 9.3). This matches the lower bound result that we prove.

9.1 Lower Bound for Split-Contraction Parameterized by Vertex Cover

In this section, we show that unless the ETH fails, Split Contraction does not admit an algorithm running in time $2^{o(\ell^2)}n^{\mathcal{O}(1)}$, where ℓ is the size of a minimum vertex cover of the input graph G on n vertices.

To obtain our lower bound, we give an appropriate reduction from Vertex Cover on sub-cubic graphs. For this we utilize the fact that Vertex Cover on sub-cubic graphs does not have an algorithm running in time $2^{o(n)}n^{\mathcal{O}(1)}$ unless the ETH fails [IPZ01, Kom15]. For the ease of presentation we split the reduction into two steps. The first step comprises of reducing a special case of Vertex Cover on sub-cubic graphs, which we call Sub-Cubic Partitioned Vertex Cover (Sub-Cubic PVC) to Split Contraction. In the second step we show that there does not exist an algorithm running in time $2^{o(n)}n^{\mathcal{O}(1)}$ for Sub-Cubic PVC. We remark that the reduction from Vertex Cover on sub-cubic graphs (Sub-Cubic VC) to Sub-Cubic PVC is a Turing reduction.

9.1.1 Reduction from Sub-Cubic Partitioned Vertex Cover to Split Contraction

In this section, we give a reduction from Sub-Cubic Partitioned Vertex Cover to Split Contraction. Next, we formally define Sub-Cubic Partitioned Vertex Cover.

Sub-Cubic Partitioned Vertex Cover (Sub-Cubic PVC)
Input: A sub-cubic graph G, an integer t, for each $i \in [t]$, an integer $k_i \geq 0$, a partition $\mathcal{P} = \{C_1, \ldots, C_t\}$ of $V(G)$ such that $t \in \mathcal{O}(\sqrt{|V(G)|})$ and for all $i \in [t]$, C_i is an independent set and $|C_i| \in \mathcal{O}(\sqrt{|V(G)|})$. Furthermore, for $i, j \in [t], i \neq j$, $|E(G[C_i \cup C_j]) \cap E(G)| = 1$.
Question: Does G have a vertex cover X such that for all $i \in [t]$, $|X \cap C_i| \leq k_i$?

We first explain (informally) the ideas behind our reduction. Let X be a *hypothetical* vertex cover we are looking for. Recall that we assume the ETH holds and thus we are allowed to use $2^{o(n)}n^{\mathcal{O}(1)}$ time to obtain our reduction. We will use this freedom to design our reduction and to construct an instance (G', k') of Split Contraction. For $i \in [t]$, in $V(G')$, we have a *vertex* corresponding to each possible intersection of X with C_i on at most k_i vertices. Furthermore, we have a vertex $c_i \in V(G')$ corresponding to each

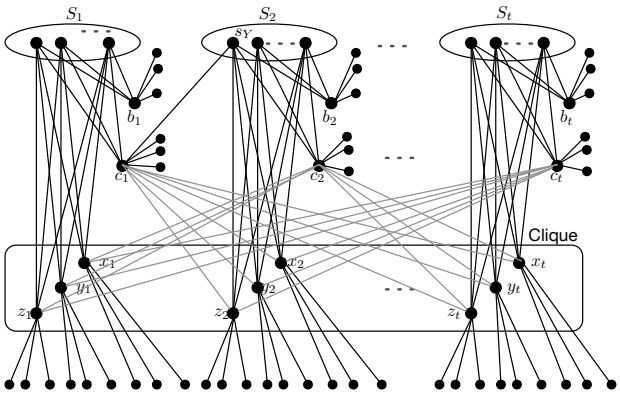

Figure 9.1: Reduction from Sub-Cubic PVC to Split Contraction.

C_i, for $i \in [t]$. We want to make sure that for each $(u, v) \in E(G)$, we choose an edge in $E(G')$ (in the solution to Split Contraction) that is incident to a vertex which corresponds to a subset containing one of u or v and one of c_i or c_j. Furthermore, we want to force these selected vertices to be contracted to the clique side in the resulting split graph. We crucially exploit the fact that there is exactly one edge between every C_i, C_j pair, where $i, j \in [t], i \neq j$. Finally, we will add a clique, say Γ, of size $3t$ and make each of its vertices adjacent to many pendant vertices, which ensures that after contracting the solution edges, the vertices of Γ remain in the clique side. We will assign appropriate adjacencies between the vertices of Γ and c_i, for $i \in [t]$. This will guide us in selecting edges for the solution of the contraction problem. We now move to the formal description of the construction used in the reduction.

Construction. Let $(G, \mathcal{P} = \{C_1, C_2, \ldots, C_t\}, k_1, \ldots, k_t)$ be an instance of Sub-Cubic PVC and $n = |V(G)|$. We create an instance of Split Contraction (G', k') as follows. For $i \in [t]$, let $S_i = \{v_Y \mid Y \subseteq C_i \text{ and } |Y| \leq k_i\}$. That is, S_i comprises of vertices corresponding to subsets of C_i of size at most k_i. For each $i \in [t]$, we add five vertices b_i, c_i, x_i, y_i, z_i to $V(G')$. The vertices $\{x_i, y_i, z_i \mid i \in [t]\}$ induce a clique (on $3t$ vertices) in G'. We add the edges $(b_i, s_Y), (c_i, s_Y), (x_i, s_Y), (y_i, s_Y), (z_i, s_Y)$ for all $s_Y \in S_i$ to $E(G')$. For $i, j \in [t]$, $i \neq j$, we add the edges $(c_i, x_j), (c_i, y_j), (c_i, z_j)$ to $E(G')$. For $i, j \in [t]$, $i \neq j$ and $s_Y \in S_j$, we add the edge (c_i, s_Y) in $E(G')$ if and only if Y covers the unique edge between C_i and C_j. For all $i \in [t]$, we add $4t + 2$ pendant vertices, b'^i_j, $j \in [4t + 2]$, to b_i. Similarly, for all $i \in [t]$, we add $4t + 2$ pendant vertices c'^i_j, x'^i_j, y'^i_j, and z'^i_j, $j \in [4t + 2]$, to c_i, x_i, y_i and z_i, respectively. The pendant vertices are added in order to make sure that the vertices resulting after the contraction of their witness sets belong to the clique side. This completes the construction of the graph G'. Observe that $\{b_i, c_i, x_i, y_i, z_i \mid i \in [t]\}$ forms a minimum vertex cover of G' of size $5t$. Finally, we set $k' = 2t$. The resulting instance of Split Contraction is (G', k'). We refer the reader to Figure 9.1 for an illustration of the construction.

In the next few lemmata (Lemmata 9.1 to 9.6) we prove certain properties of the instance (G', k') of Split Contraction. This will be helpful later for establishing the equivalence between the original instance $(G, \mathcal{P} = \{C_1, C_2, \ldots, C_t\}, k_1, \ldots, k_t)$ of Sub-Cubic PVC and the instance (G', k') of Split Contraction. In Lemmas 9.1 to 9.6 we

will use the following notations. We use T to denote a solution to Split Contraction in (G', k') and $H = G'/T$ with $\hat{C}$, $\hat{I}$ being a partition of $V(H)$ inducing a clique and an independent set, respectively, in H. We let $\varphi : V(G') \to V(H)$ be the surjective function defining the contractibility of G' to H, and $\mathcal{W}$ be the *H-witness structure* of G'.

Lemma 9.1. *Let (G', k') be a yes instance of* Split Contraction. *Then, for all $v \in \{b_i, c_i, x_i, y_i, z_i \mid i \in [t]\}$, we have $\varphi(v) \in \hat{C}$.*

Proof. Consider $v \in \{b_i, c_i, x_i, y_i, z_i \mid i \in [t]\}$. Recall that there are $4t + 2 = 2k' + 2$ pendant vertices v'^i_j, for $j \in [2k' + 2]$ adjacent to v. At most k' edges in $\{(v'^i_j, v) \mid j \in [2k' + 2]\}$ can belong to T. Therefore, there exist $j_1, j_2 \in [2k' + 2]$, $j_1 \neq j_2$ such that no edge incident to $v'^i_{j_1}$ or $v'^i_{j_2}$ is in T. In other words, for $h_1 = \varphi(v'^i_{j_1})$ and $h_2 = \varphi(v'^i_{j_2})$, $W(h_1)$ and $W(h_2)$ are singleton sets. Since $\mathcal{W}$ is a H-witness structure of G', $(h_1, h_2) \notin E(H)$. Therefore, at least one of h_1, h_2 belongs to $\hat{I}$, say $h_1 \in \hat{I}$. This implies that $\varphi(v) \in \hat{C}$. □

Lemma 9.2. *Let (G', k') be a yes instance of* Split Contraction. *Then, for all $i \in [t]$, there exists $s_{Y_i} \in S_i$ such that $(b_i, s_{Y_i}) \in T$.*

Proof. Towards a contradiction assume that there is $i \in [t]$ such that for all $s_Y \in S_i$, $(b_i, s_Y) \notin T$. Recall that $N_{G'}(b_i) = S_i \cup \{b'^i_j \mid j \in [4t + 2]\}$. Let $h = \varphi(b_i)$ and $A = \{b_j, c_j, x_j, y_j, z_j \mid j \in [t], j \neq i\}$. There exists $v \in A$ such that $|W(h')| = 1$, where $h' = \varphi(v)$. This follows from the fact that at most $2k' = 4t$ vertices in A can be incident to an edge in T, although $|A| = 5(t - 1) > 4t$, as t can be assumed to be larger than 6, else the graph has constantly many edges and we can solve the problem in polynomial time. From Lemma 9.1 it follows that $(h, h') \in E(H)$, but $W(h), W(h')$ are not adjacent in G', contradicting that $\mathcal{W}$ is an *H-witness structure* of G'. Hence the claim follows. □

For each $i \in [t]$, we arbitrarily choose a vertex $s^\star_{Y_i} \in S_i$ such that $(b_i, s^\star_{Y_i}) \in T$. The existence of such a vertex is guaranteed by Lemma 9.2.

Lemma 9.3. *Let (G', k') be a yes instance of* Split Contraction *and $(b_i, s^\star_{Y_i}) \in T$ for $i \in [t]$. Then, for $h_i = \varphi(s^\star_{Y_i})$, we have $|W(h_i)| \geq 3$. Furthermore, there is an edge in T incident to b_i or $s^\star_{Y_i}$ other than $(b_i, s^\star_{Y_i})$.*

Proof. Suppose there exists $i \in [t]$, $h_i = \varphi(s^\star_{Y_i})$ such that $|W(h_i)| < 3$. Recall that $|W(h_i)| \geq 2$, since $b_i \in W(h_i)$. Let $A = \{x_j, y_j, z_j \mid j \in [t], j \neq i\}$. From Lemma 9.2, it follows that for each $j \in [t]$, there is an edge $(b_j, s^\star_{Y_j}) \in T$, therefore the number of edges in T incident to a vertex in A is bounded by $k' - t = t$. But $|A| = 3t - 3 > 2t$, therefore, there exists $a \in A$ such that for $h_a = \varphi(a)$, $|W(h_a)| = 1$. From Lemma 9.1, $(h_i, h_a) \in E(H)$, therefore $W(h_i)$ and $W(h_a)$ must be adjacent in G'. But $a \notin N(\{b_i, s^\star_{Y_i}\})$, hence $W(h_i)$ and $W(h_a)$ are not adjacent in G', contradicting that $\mathcal{W}$ is an *H-witness structure* of G'.

Since $|W(h_i)| \geq 3$ and $G[W(h_i)]$ is connected, at least one of $s^\star_{Y_i}, b_i$ must be adjacent to an edge in T which is not $(s^\star_{Y_i}, b_i)$. □

Lemma 9.4. *Let (G', k') be a yes instance of* Split Contraction. *Then, for all $i \in [t]$, we have $|W(h_i)| \geq 2$ where $h_i = \varphi(c_i)$.*

Proof. Towards a contradiction assume that there exists $i \in [t]$, $h_i = \varphi(c_i)$, such that $|W(h_i)| < 2$. Let $A = \{c_j \mid j \in [t], j \neq i\} \cup \{x_i, y_i, z_i\}$. From Lemma 9.2 it follows that the edge $(b_j, s^\star_{Y_j}) \in T$, for each $j \in [t]$. By Lemma 9.3 it follows that there is an edge in

T that is adjacent to exactly one of $\{b_j, s^\star_{Y_j}\}$ in T, for all $j \in [t]$. Therefore, at most t vertices in A can be incident to an edge in T, while $|A| = t+2$. This implies that there exists $a \in A$, $h_a = \varphi(a)$ such that $|W(h_a)| = 1$. Observe that none of the vertices in A are adjacent to c_i in G'. Therefore, it follows that $W(h_i), W(h_a)$ are not adjacent in G'. But Lemma 9.1 implies that $(h_i, h_a) \in E(H)$, a contradiction to $\mathcal{W}$ being an *H-witness structure* of G'. □

Lemma 9.5. *Let (G', k') be a yes instance of* Split Contraction *and $(b_i, s^\star_{Y_i}) \in T$ for $i \in [t]$. Then, for each $i \in [t]$, we have $|W(h_i)| = 3$ where $h_i = \varphi(s^\star_{Y_i})$.*

Proof. For $i \in [t]$, let $h_i = \varphi(s^\star_{Y_i})$. From Lemma 9.3 we know that $|W(h_i)| \geq 3$. Let $C = \{c_i \mid i \in [t]\}$ and $\mathcal{S} = \{\{b_i, s^\star_{Y_i}\} \mid i \in [t]\}$. From Lemmata 9.3 and 9.4 it follows that each $c \in C$ must be incident to an edge in T and each $S \in \mathcal{S}$ must have a vertex which is incident to an edge in T with the other endpoint not in S. Since $|C| = |\mathcal{S}| = t$ and $(b_i, s^\star_{Y_i}) \in T$, for all $i \in [t]$, there are at most t edges in T that are incident to a vertex in C and a vertex in $S \in \mathcal{S}$. Therefore, each $c \in C$ is incident to exactly one edge in T. Similarly, each $S \in \mathcal{S}$ is incident to exactly one edge with one endpoint in S and the other not in S. This implies that exactly one vertex $c \in C$ belongs to $W(h_i)$ for $i \in [t]$, and c does not belong to $W(h_j)$, where $i \neq j$, $i, j \in [t]$. Also note that none of the vertices in $\{x_i, y_i, z_i \mid i \in [t]\}$ can be incident to an edge in T. Similarly, none of the vertices in $\{b'^i_j, c'^i_j, x'^i_j, y'^i_j, z'^i_j \mid i \in [t], j \in [4t+2]\}$ can be incident to an edge in T. Hence, we get that $|W(h_i)| = 3$, concluding the proof. □

Lemma 9.6. *Let (G', k') be a yes instance of* Split Contraction *and $(b_i, s^\star_{Y_i}) \in T$ for $i \in [t]$. Then, for all $i \in [t]$, we have $c_i \in W(h_i)$ where $h_i = \varphi(s^\star_{Y_i})$.*

Proof. Suppose for some $i \in [t]$, $c_i \notin W(h_i)$ where $h_i = \varphi(s^\star_{Y_i})$. From Lemmata 9.3, 9.4 and $k' = 2t$, it follows that there exists some $j \in [t]$ such that $c_i \in W(h_j)$, where $h_j = \varphi(s^\star_{Y_j})$. By our assumption, $j \neq i$. From Lemma 9.5 we know that $|W(h_j)| = 3$, therefore $W(h_j) = \{b_j, s^\star_{Y_j}, c_i\}$. Moreover, by Lemmata 9.4 and 9.5 and since $k' = 2t$, $|W(x_i)| = 1$. However, we then get that $W(h_j), W(x_i)$ are not adjacent in G'. By Lemma 9.1, we obtain a contradiction to the assumption that $\mathcal{W}$ is an *H-witness structure* of G'. This completes the proof. □

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

Lemma 9.7. *$(G, \mathcal{P} = \{C_1, C_2, \ldots, C_t\}, k_1, \ldots, k_t)$ is a yes instance of* Sub-Cubic PVC *if and only if (G', k') is a yes instance of* Split Contraction.

Proof. In the forward direction, let Y be a vertex cover in G such that for each $i \in [t]$, $|Y \cap C_i| \leq k_i$. For $i \in [t]$, we let $Y_i = Y \cap C_i$. Let $T = \{(b_i, s_{Y_i}), (c_i, s_{Y_i}) \mid i \in [t]\}$. Let $H = G'/T$, $\varphi : V(G') \to V(H)$ be the underlying surjective map and $\mathcal{W}$ be the *H-witness structure* of G'. To show that T is a solution to Split Contraction in (G', k'), it is enough to show that H is a split graph. Let $I = \cup_{i \in [t]}(S_i \setminus \{s_{Y_i}\}) \cup \{b'^i_j, c'^i_j, x'^i_j, y'^i_j, z'^i_j \mid i \in [t], j \in [4t+2]\}$. Recall that for each $v \in I$, $|W(\varphi(v))| = 1$. Furthermore, for $v, v' \in I$, $(v, v') \notin E(G')$. Hence, it follows that $\hat{I} = \{\varphi(v) \mid v \in I\}$ induces an independent set in H. Let $\mathcal{C}_1 = \{x_i, y_i, z_i \mid i \in [t]\}$. Recall that $G'[\mathcal{C}_1]$ is a clique and from the construction of T, $|W(\varphi(c))| = 1$ for all $c \in \mathcal{C}_1$. Therefore, $\hat{\mathcal{C}}_1 = \{\varphi(c) \mid c \in \mathcal{C}_1\}$ induces a clique in H. Let $\mathcal{C}_2 = \{s_{Y_i} \mid i \in [t]\}$, $h_i = \varphi(s_{Y_i})$ for $i \in [t]$, and $\hat{\mathcal{C}}_2 = \{h_i \mid i \in [t]\}$. From the

construction of T, we have $W(h_i) = \{b_i, c_i, s_{Y_i}\}$ for all $i \in [t]$. Observe that for $c_1 \in \hat{\mathcal{C}}_1$ and $c_2 \in \hat{\mathcal{C}}_2$, $W(c_1), W(c_2)$ are adjacent in G', therefore, $(c_1, c_2) \in E(H)$. Consider $h_i, h_j \in \hat{\mathcal{C}}_2$, where $i, j \in [t], i \neq j$. Recall $W(h_i) = \{b_i, s_{Y_i}, c_i\}$ and $W(h_j) = \{b_j, s_{Y_j}, c_j\}$. Since Y is a vertex cover, at least one of Y_i or Y_j covers the unique edge between C_i and C_j in G, say Y_i covers the edge between C_i and C_j. But then $(s_{Y_i}, c_j) \in E(G')$, therefore $(h_i, h_j) \in E(H)$. The above argument implies that $\hat{\mathcal{C}} = \hat{\mathcal{C}}_1 \cup \hat{\mathcal{C}}_2$ induces a clique in H. Furthermore, $V(H) = \hat{I} \cup \hat{\mathcal{C}}$. This implies that H is a split graph.

In the reverse direction, let T be a solution to Split Contraction in (G', k'). Let $H = G'/T$, $\varphi : V(G') \to V(H)$ be the underlying surjective map and $\mathcal{W}$ be the *H-witness structure* of G'. From Lemma 9.2, it follows that for all $i \in [t]$, there exists $s_{Y_i} \in S_i$ such that $(b_i, s_{Y_i}) \in T$. For $i \in [t]$, let Y_i be the set such that $(b_i, s_{Y_i}) \in T$. We let $Y = \cup_{i \in [t]} Y_i$. For $i \in [t]$, from the definition of the vertices in S_i, it follows that $|Y \cap C_i| \leq k_i$. We will show that Y is a vertex cover in G. Towards a contradiction assume that there exists $i, j \in [t], i \neq j$, such that Y does not cover the unique edge between C_i and C_j. From Lemmas 9.2 and 9.6 it follows that $W(h_i) = \{b_i, s_{Y_i}, c_i\}$ and $W(h_j) = \{b_j, s_{Y_j}, c_j\}$, where $h_i = \varphi(s_{Y_i})$ and $h_j = \varphi(s_{Y_j})$. From Lemma 9.1 it follows that $(h_i, h_j) \in E(H)$. Therefore, $W(h_i)$ and $W(h_j)$ are adjacent in G'. Recall that $N_{G'}(b_i) \cap W(h_j) = \emptyset$, $N_{G'}(b_j) \cap W(h_i) = \emptyset$, $(c_i, c_j), (s_{Y_i}, s_{Y_j}) \notin E(G')$. Therefore, at least one of $(c_i, s_{Y_j}), (c_j, s_{Y_i})$ must belong to $E(G')$, say $(c_i, s_{Y_j}) \in E(G')$. But then by construction it follows that $Y_j \subseteq Y$ covers the unique edge between C_i and C_j in G, a contradiction. This completes the proof. □

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

Theorem 9.1. *Unless the* ETH *fails,* Split Contraction *parameterized by ℓ, the size of a minimum vertex cover of the input graph, does not have an algorithm running in time $2^{o(\ell^2)} \cdot n^{\mathcal{O}(1)}$. Here, n denotes the number of vertices in the input graph.*

Proof. Towards a contradiction assume that there is an algorithm $\mathcal{A}$ for Split Contraction, parameterized by ℓ, the size of a minimum vertex cover, running in time $2^{o(\ell^2)} n^{\mathcal{O}(1)}$. Let $(G, \mathcal{P} = \{C_1, C_2, \ldots, C_t\}, k_1, \ldots, k_t)$ be an instance of Sub-Cubic PVC. We create an instance (G', k') of Split Contraction as described in the **Construction**, running in time $2^{o(n)} \cdot n^{\mathcal{O}(1)}$, where $n = |V(G)|$. Recall that in the instance created, the size of a minimum vertex cover is $\ell = 5t = \mathcal{O}(\sqrt{n})$. Then we use algorithm $\mathcal{A}$ for deciding if (G', k') is a *yes* instance of Split Contraction and return the same answer for Sub-Cubic PVC on $(G, \mathcal{P}, k_1, \ldots, k_t)$. The correctness of the answer returned follows from Lemma 9.7. But then we can decide whether $(G, \mathcal{P}, k_1, \ldots, k_t)$ is a *yes* instance of Sub-Cubic PVC in time $2^{o(n)} \cdot n^{\mathcal{O}(1)}$, which contradicts ETH assuming Theorem 9.2. This concludes the proof. □

9.1.2 Reduction from Sub-Cubic VC to Sub-Cubic PVC

Finally, to complete our proof we show that Sub-Cubic PVC on graphs with n vertices can not be solved in time $2^{o(n)} n^{\mathcal{O}(1)}$ unless the ETH fails. In this section, we give a Turing reduction from Sub-Cubic VC to Sub-Cubic PVC that will imply our desired assertion.

Let (G, k) be an instance of Sub-Cubic VC and $n = |V(G)|$. We first create a new instance (G', k') of Sub-Cubic VC satisfying certain properties. We start by computing

(in polynomial time) a harmonious coloring of G using $t \in \mathcal{O}(\sqrt{n})$ color classes such that each color class contains at most $\mathcal{O}(\sqrt{n})$ vertices using Proposition 2.1. Let $C_1, \ldots, C_t$ be the color classes. Recall that between each pair of the color classes, C_i, C_j for $i, j \in [t]$, $i \neq j$, we have at most one edge. If for some $i, j \in [t], i \neq j$, there is no edge between a vertex in C_i and a vertex in C_j, then we add a new vertex x_{ij} in C_i and a new vertex x_{ji} in C_j and add the edge (x_{ij}, x_{ji}). Observe that we add a matching corresponding to a missing edge between a pair of color classes. In this process we can add at most $t-1$ new vertices to a color class C_i, for $i \in [t]$. Therefore, the number of vertices in C_i for $i \in [t]$ after addition of new vertices is also bounded by $\mathcal{O}(\sqrt{n})$. We denote the resulting graph by G' with partition of vertices $C_1, \ldots, C_t$ (including the newly added vertices, if any). Observe that the number of vertices n' in G' is at most $\mathcal{O}(n)$. Let m be the number of matching edges added in G to obtain G' and let $k' = k + m$. It is easy to see that (G, k) is a *yes* instance of Sub-Cubic VC if and only if (G', k') is a yes instance of Sub-Cubic VC.

We will now be working with the instance (G', k') of Sub-Cubic VC with the partition of vertices $C_1, \ldots, C_t$ obtained by extending the color classes of the harmonious coloring of G we started with. We guess the size of the intersection of the vertex cover in G' with each C_i, for $i \in [t]$. That is, for $i \in [t]$, we guess an integer $0 \leq k'_i \leq \min(|C_i|, k')$, such that $\sum_{i \in [t]} k'_i = k'$. Finally, we let $(G', \mathcal{P} = \{C_1, \ldots, C_t\}, k'_1, \ldots, k'_t)$ be an instance of Sub-Cubic PVC. Notice that G' and $\mathcal{P}$ satisfies all the requirements for it to be an instance of Sub-Cubic PVC. It is easy to see that (G', k') is a *yes* instance of Sub-Cubic VC if and only if for some guess of k_i, for $i \in [t]$, $(G', \mathcal{P} = \{C_1, \ldots, C_t\}, k'_1, \ldots, k'_t)$ is a *yes* instance of Sub-Cubic PVC. This finishes the reduction from Sub-Cubic VC to Sub-Cubic PVC.

Theorem 9.2. *Unless the* ETH *fails,* Sub-Cubic PVC *does not admit an algorithm running in time* $2^{o(n)} \cdot n^{\mathcal{O}(1)}$. *Here,* n *is the number of vertices in the input graph.*

Proof. Towards a contradiction assume that there is an algorithm $\mathcal{A}$ for Sub-Cubic PVC running in time $2^{o(n)} \cdot n^{\mathcal{O}(1)}$. Let (G, k) be an instance of Sub-Cubic VC. We apply the above mentioned reduction to create an instance (G', k') of Sub-Cubic VC with vertex partitions $C_1, \ldots, C_t$ such that $t \in \mathcal{O}(\sqrt{n})$ and $|C_i| \in \mathcal{O}(\sqrt{n})$, for all $i \in [t]$. Furthermore, there is exactly one edge between C_i, C_j, for $i, j \in [t]$, $i \neq j$, and C_i induces an independent set in G'. For each guess $0 \leq k'_i \leq \min(|C_i|, k')$ of the size of intersection of vertex cover with C_i, for $i \in [t]$, we solve the instance $(G', \mathcal{P}, k'_1, \ldots, k'_t)$. By the exhaustiveness of the guesses of the size of intersection for each partition, (G', k') is a *yes* instance of Sub-Cubic VC if and only if for some guess $k'_1, \ldots, k'_t$, $(G', \mathcal{P}, k'_1, \ldots, k'_t)$ is a *yes* instance of Sub-Cubic PVC. We emphasize the fact that the number of guesses we make is bounded by $\sqrt{n}^{\mathcal{O}(\sqrt{n})} = 2^{o(n)}$, since $|C_i| \in \mathcal{O}(\sqrt{n})$ and $t \in \mathcal{O}(\sqrt{n})$. But then we have an algorithm for Sub-Cubic VC running in time $2^{o(n)} \cdot n^{\mathcal{O}(1)}$, contradicting the ETH. This concludes the proof. □

9.2 W[1]-hardness of Split Contraction

In this section, we show that Split Contraction parameterized by the solution size is W[1]-hard. Towards this we first define an intermediate problem from which we give the desired reduction.

Special Red-Blue Perfect Code (SRBPC) **Parameter:** k
Input: A bipartite graph G with vertex set $V(G)$ partitioned into $\mathcal{R}$ (red set) and $\mathcal{B}$ (blue set). Furthermore, $\mathcal{R}$ is partitioned (disjoint) into $R_1 \uplus R_2 \uplus \ldots \uplus R_k$ and for all $r, r' \in \mathcal{R}$, $d_G(r) = d_G(r')$. That is, every vertex in $\mathcal{R}$ has same degree, say d.
Question: Does there exist $X \subseteq \mathcal{R}$, such that for all $b \in \mathcal{B}$, $\lvert N(b) \cap X \rvert = 1$ and for all $i \in [k]$, $\lvert R_i \cap X \rvert = 1$?

SRBPC is a variant of Perfect Code which is known to be W[1]-hard [DF95]. We postpone the W[1]-hardness proof of SRBPC to Section 9.2.2 and first give a parameterized reduction from SRBPC to Split Contraction, showing that Split Contraction is W[1]-hard.

9.2.1 Reduction from SRBPC to Split Contraction

Let $(G, \mathcal{R} = R_1 \uplus, R_2 \uplus \ldots \uplus R_k, \mathcal{B})$ be an instance of SRBPC. We will assume that $|\mathcal{B}| = dk$, otherwise, the instance is a trivial *no* instance of SRBPC. For technical reasons we assume that $|\mathcal{B}| = \ell > 4k$ (and hence $d > 4$). Such an assumption is valid because otherwise, the problem is FPT. Indeed, if $|\mathcal{B}| = \ell \leq 4k$ then for every partition $P_1, \ldots, P_k$ of $\mathcal{B}$ into k parts such that each part is non-empty, we first guess a permutation π on k elements and then for every $i \in [k]$, we check whether there exists a vertex $r_{\pi(i)} \in R_{\pi(i)}$ that dominates exactly all the vertices in P_i (and none in other parts P_j, $j \neq i$). Clearly, all this can be done in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$. Furthermore, we also assume that $k \geq 2$, else the problem is solvable in polynomial time. Now we give the desired reduction. We construct an instance (G', k') of Split Contraction as follows. Initially, $V(G') = \mathcal{R} \cup \mathcal{B}$ and $E(G') = E(G)$. For all $b, b' \in \mathcal{B}$, $b \neq b'$, we add the edge (b, b') to $E(G')$. That is, we transform $\mathcal{B}$ into a clique. Let $t = 2k + 2$. For each $b_i \in \mathcal{B}$, we add a set of t vertices $y_1^i, \ldots, y_t^i$ each adjacent to b_i in G'. We add a vertex s adjacent to every vertex $r \in \mathcal{R}$ in G'. Also, we add a set of t vertices $q_1, \ldots, q_t$ each adjacent to s in G'. For each $i \in [k]$, we add a vertex x_i adjacent to each vertex $r \in R_i$. Finally, for all $i \in [k]$, we add a set of t vertices $w_1^i, \ldots, w_t^i$ adjacent to x_i in G'. We set the new parameter k' to be $2k$. This completes the description of the reduction. We refer the reader to Figure 9.2 for an illustration of the reduction.

In the next four lemmata (Lemmata 9.8 to 9.11) we prove certain structural properties of the instance (G', k') of Split Contraction. These will later be used in showing that $(G, \mathcal{R} = R_1 \uplus, R_2 \uplus \ldots \uplus R_k, \mathcal{B})$ is a *yes* instance of SRBPC if and only if (G', k') is a *yes* instance of Split Contraction. For the next four lemmata, we let S be a solution to Split Contraction in (G', k') and $H = G'/S$ with $\hat{C}$, $\hat{I}$ being a partition of $V(H)$ inducing a clique and an independent set, respectively, in H. Let $\varphi : V(G) \to V(H)$ denote the function defining the contractibility of G to H, and $\mathcal{W}$ be the *H-witness structure* of G.

Lemma 9.8. *Let (G', k') be a yes instance of* Split Contraction. *Then, for all $v \in (\{s\} \cup \mathcal{B} \cup \{x_i \mid i \in [k]\})$, we have $\varphi(v) \in \hat{C}$.*

Proof. We only give an argument for the vertex s. The argument for vertices in $\mathcal{B} \cup \{x_i \mid i \in [k]\}$ is analogous and thus omitted. Recall that there are t pendant vertices $q_1, \ldots, q_t$ adjacent to s, where $t = 2k + 2$. At most $2k < t$ edges in $\{(q_i, s) \mid i \in [t]\}$ can belong to S. Therefore, there exist $j_1, j_2 \in [t]$, $j_1 \neq j_2$ such that no edge incident to q_{j_1} or q_{j_2} is

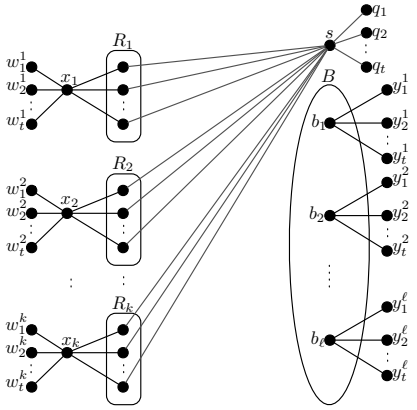

Figure 9.2: W[1]-hardness of Split Contraction.

in S. In other words, for $h_1 = \varphi(q_{j_1})$ and $h_2 = \varphi(q_{j_2})$, $W(h_1)$ and $W(h_2)$ are singleton sets. Since $\mathcal{W}$ is a H-witness structure of G', $(h_1, h_2) \notin E(H)$. Therefore, at least one of h_1, h_2 belongs to $\hat{I}$, say $h_1 \in \hat{I}$. This implies that $\varphi(s) \in \hat{C}$. □

Lemma 9.9. *Let (G', k') be a yes instance of* Split Contraction. *Then, for all $i \in [k]$, there exists $r_i \in R_i$ such that $(x_i, r_i) \in S$.*

Proof. Towards a contradiction assume that for some $i \in [k]$ and $h_i = \varphi(r_i^\star)$, $|W(h_i)| < 3$. From our assumption that $(x_i, r_i^\star) \in S$ we have that $x_i \in W(h_i)$. Also, note that there is a set $\mathcal{B}' \subseteq \mathcal{B}$ of at least $\ell - 2k$ vertices such that for $h_b = \varphi(b)$, $|W(h_b)| = 1$. This follows from the fact that at most $2k$ vertices in $\mathcal{B}$ can be incident to an edge in S. Let $\hat{\mathcal{B}} = \mathcal{B}' \setminus N(r_i^\star)$. We claim that $|\hat{\mathcal{B}}| \geq \ell - 2k - d > 0$. Towards the claim observe that if $(G, \mathcal{R}, \mathcal{B})$ is a *yes* instance of SRBPC then $\ell = dk$. The last assertion follows from the fact that every vertex in $\mathcal{R}$ has degree exactly d and we are seeking a solution $X \subseteq \mathcal{R}$, such that for all $b \in \mathcal{B}$, $|N(b) \cap X| = 1$ and for all $i \in [k]$, $|R_i \cap X| = 1$. That is, the set X is of size k and it partitions $\mathcal{B}$. This implies that $d > 4$, since $\ell = dk > 4k$. Thus, combining this with the fact that $k \geq 2$ we have that $|\hat{\mathcal{B}}| \geq \ell - 2k - d = (d-2)k - d > 0$. This completes the claim. Since the size of $|W(h_i)| < 3$ and it contains x_i and $r_i^\star$ we have that $W(h_i) = \{x_i, r_i^\star\}$. Now, consider $\hat{b} \in \hat{\mathcal{B}}$ with $\hat{h} = \varphi(\hat{b})$. Observe that $W(h_i)$ and $W(\hat{h})$ are not adjacent in G, however since $x_i \in W(h_i)$ Lemma 9.8 implies that $h_i \in \hat{C}$. But then $(\hat{h}, h_i) \in E(H)$, a contradiction. This implies that for all $i \in [k]$ and $h_i = \varphi(r_i^\star)$ we have $|W(h_i)| \geq 3$. However, since $h_i, \hat{h} \in \hat{C}$ there must be a vertex in $W(h_i)$ that is adjacent to a vertex in $W(\hat{h})$. But since $W(\hat{h}) = \{\hat{b}\}$, $W(h_i)$ must contain a vertex that is adjacent to $\hat{b}$. But, none of the vertices in $\{w_1^i, \cdots, w_t^i\}$ are adjacent to $\hat{b}$. Thus, $W(h_i)$ must contain a vertex that is adjacent to either x_i or $r_i^\star$ but not to any of the vertices in $\{w_1^i, \cdots, w_t^i\}$. Let such a vertex be z_i and let it be adjacent to $r_i^\star$ (or x_i). Since a solution to (G', k') can be formed by taking spanning trees of each of the witness sets, we can assume that S contains a spanning tree of $W(h_i)$ that contains the edge $e_i = (z_i, r_i^\star)$ (or $e_i = (z_i, x_i)$) and $e_i^\star$. This completes the proof of the lemma. □

For each $i \in [k]$ we arbitrarily choose a vertex $r_i^\star \in R_i$ such that $e_i^\star = (x_i, r_i^\star) \in S$. The existence of such a vertex is guaranteed by Lemma 9.9.

Lemma 9.10. *Let (G', k') be a yes instance of* Split Contraction. *Then, for all $i \in [k]$ and $h_i = \varphi(r_i^\star)$, we have $|W(h_i)| \geq 3$. Furthermore, there is an edge $e_i \neq e_i^\star$ in S incident to exactly one of $x_i, r_i^\star$ and not incident to the vertices in $\{w_1^i, \ldots, w_t^i\}$.*

Proof. Towards a contradiction assume that for some $i \in [k]$ and $h_i = \varphi(r_i^\star)$, $|W(h_i)| < 3$. From our assumption that $(x_i, r_i^\star) \in S$ we have that $x_i \in W(h_i)$. Also, note that there is a set $\mathcal{B}' \subseteq \mathcal{B}$ of at least $\ell - 2k$ vertices such that for $h_b = \varphi(b)$, $|W(h_b)| = 1$. This follows from the fact that at most $2k$ vertices in $\mathcal{B}$ can be incident to an edge in S. Let $\hat{\mathcal{B}} = \mathcal{B}' \setminus N(r_i^\star)$. We claim that $|\hat{\mathcal{B}}| \geq \ell - 2k - d > 0$. Towards the claim observe that if (G, k) is a *yes* instance of SRBPC then $\ell = dk$. The last assertion follows from the fact that every vertex in $\mathcal{R}$ has degree exactly d and we are seeking a solution $X \subseteq \mathcal{R}$, such that for all $b \in \mathcal{B}$, $|N(b) \cap X| = 1$ and for all $i \in [k]$, $|R_i \cap X| = 1$. That is, the set X is of size k and it partitions $\mathcal{B}$. This implies that $d > 4$, since $\ell = dk > 4k$. Thus, combining this with the fact that $k \geq 2$ we have that $|\hat{\mathcal{B}}| \geq \ell - 2k - d = (d-2)k - d > 0$. This completes the claim. Since the size of $|W(h_i)| < 3$ and it contains x_i and $r_i^\star$ we have that $W(h_i) = \{x_i, r_i^\star\}$. Now, consider $\hat{b} \in \hat{\mathcal{B}}$ with $\hat{h} = \varphi(b)$. Observe that $W(h_i)$ and $W(\hat{h})$ are not adjacent in G, however since $x_i \in W(h_i)$ Lemma 9.8 implies that $h_i \in \hat{C}$. But then $(\hat{h}, h_i) \in E(H)$, a contradiction. This implies that for all $i \in [k]$ and $h_i = \varphi(r_i^\star)$ we have $|W(h_i)| \geq 3$. However, since $h_i, \hat{h} \in \hat{C}$ there must be a vertex in $W(h_i)$ that is adjacent to a vertex in $W(\hat{h})$. But since $W(\hat{h}) = \{\hat{b}\}$, $W(h_i)$ must contain a vertex that is adjacent to $\hat{b}$. But, none of the vertices in $\{w_1^i, \cdots, w_t^i\}$ are adjacent to $\hat{b}$. Thus, $W(h_i)$ must contain a vertex that is adjacent to either x_i or $r_i^\star$ but not to any of the vertices in $\{w_1^i, \cdots, w_t^i\}$. Let such a vertex be z_i and let it be adjacent to $r_i^\star$ (or x_i). Since an optimal solution to (G', k') can be formed by taking spanning trees of each of the witness sets, we can assume that S contains a spanning tree of $W(h_i)$ that contains the edge $e_i = (z_i, r_i^\star)$ (or $e_i = (z_i, x_i)$) and $e_i^\star$. This completes the proof of the lemma. □

From Lemma 9.9 we know that for each $i \in [k]$, we have $r_i^\star \in R_i$ such that $(x_i, r_i^\star) \in S$. Similarly, from Lemma 9.10 we know that, for each $i \in [k]$, there is an edge incident to one of x_i, r_i other than $e_i^\star = (x_i, r_i^\star)$ in every solution. Recall that for $i, j \in [k]$, $i \neq j$ none of x_i, r_i is adjacent to x_j, r_j. Hence, it follows that we have already used up our budget of $k' = 2k$ by forcing certain types of edges to be in S. Finally, we prove Lemma 9.11 that forces even more structure on the witness sets.

Lemma 9.11. *Let (G', k') be a yes instance of* Split Contraction. *Then, for all $i \in [k]$, $r_i^\star \in W(\varphi(s))$.*

Proof. Let $h_s = \varphi(s)$ and $\hat{R} = \{r_i^\star \mid i \in [k], r_i^\star \in W(h_s)\}$. For a contradiction assume that $|\hat{R}| < k$, otherwise the claim trivially holds. By Lemma 9.9, for each $i \in [k]$, $e_i^\star = (x_i, r_i^\star) \in S$. This implies that for all $r_i^\star \in \hat{R}$, $x_i \in W(h_s)$ and hence $|W(h_s)| \geq 2|\hat{\mathcal{R}}| + 1$. From Lemma 9.10 we know that there exists an edge $e_i \neq e_i^\star \in S$ incident to either x_i or $r_i^\star$ and not incident to any vertex in $\{w_1^i, \ldots, w_t^i\}$. Thus, every edge in S is incident to either x_i or $r_i^\star$. This implies that for every vertex $z \in \{q_1, \ldots, q_t\} \cup \{y_1^j, \ldots, y_t^j \mid j \in [\ell]\}$, $|W(\varphi(z))| = 1$. Now we show that there exists a vertex in $\mathcal{B}$ that is not adjacent to any vertex in $W(h_s)$. We start proving the claim that S does not contain an edge of the form $(r_i^\star, b_j)$, where $i \in [k]$ and $b_j \in \mathcal{B}$. Suppose not, then consider the sets $\hat{R}_b = \{r_i^\star \in \hat{R} \mid (r_i^\star, b) \in S, b \in \mathcal{B}\}$ and $\hat{\mathcal{B}} = \{b \in \mathcal{B} \mid (r_i^\star, b) \in S, i \in [k]\}$. By our

assumption we have $|\hat{R}_b| = q > 0$. Moreover, for each $b \in \hat{\mathcal{B}}$, we have $\varphi(s)$ and $\varphi(b)$ are adjacent in H and $|\hat{\mathcal{B}}| \leq q$. Observe that $|W(\varphi(s)) \cap \mathcal{R}| \leq k - q$, and $W(\varphi(s)) \cap \hat{R}_b = \emptyset$. From Lemma 9.8, $\varphi(s)$ must be adjacent in H to each $\varphi(b)$, where $b \in \mathcal{B}$. Since degree of each vertex in $\mathcal{R}$ is d therefore, $\varphi(s)$ can be adjacent in H to at most $d(k-q)$ vertices $\varphi(b)$, where $b \in \mathcal{B} \setminus \hat{\mathcal{B}}$. As $d > 4$, there is a vertex $b \in \mathcal{B} \setminus \hat{\mathcal{B}}$ such that $\varphi(s)$ and $\varphi(b)$ are non-adjacent in H, which is not possible. This concludes the proof of the claim. The claim allows us to assume that the only vertices in $W(h_s)$ that can be adjacent to a vertex in $\mathcal{B}$ are in $\hat{R}$. However, every vertex in $\hat{R}$ has exactly d neighbours in $\mathcal{B}$. This together with the fact that $|\mathcal{B}| = \ell = dk > d|\hat{R}|$ implies that there exists a subset $\mathcal{B}'$ of size $d(k - |\hat{R}|)$ such that none of these vertices are adjacent to any vertex in $\hat{R}$. However, at most $(k - |\hat{R}|)$ vertices in $\mathcal{B}'$ can be incident to an edge in S. This implies that there exists a vertex $b \in \mathcal{B}'$ with $h = \varphi(b)$ such that it is not incident to any edge in S and thus $|W(h)| = 1$. But then we can conclude that $W(h)$ and $W(h_s)$ are not adjacent in G'. However, by Lemma 9.8 we know that $h_s, h \in \hat{C}$ and thus there is an edge $(h = \varphi(b), h_s) \in E(H')$, a contradiction. This contradicts our assumption that $|\hat{R}| < k$ and gives us the desired result. □

We are now ready to prove the equivalence between the instance $(G, \mathcal{R}, \mathcal{B})$ of SRBPC and the instance (G', k') of Split Contraction.

Lemma 9.12. *$(G, \mathcal{R} = R_1 \uplus \ldots \uplus R_k, \mathcal{B})$ is a yes instance of* SRBPC *if and only if (G', k') is a yes instance of* Split Contraction.

Proof. In the forward direction, let $Z = \{r_i \mid r_i \in R_i, i \in [k]\} \subseteq \mathcal{R}$ be a solution to SRBPCin $(G, \mathcal{R}, \mathcal{B})$. Let $Z' = \{(r_i, x_i), (r_i, s) \mid i \in [k]\}$. Observe that $|Z'| = 2k$. Let $T = \{r_i, x_i \mid i \in [k]\}$. We define the following surjective function $\varphi : V(G') \to V(G') \setminus T$. If $v \in T \cup \{s\}$ then $\varphi(v) = s$, else $\varphi(v) = v$. Observe that $G'[W(s)]$ is connected and for all $v \in V(G') \setminus (T \cup \{s\})$, $W(v)$ is a singleton set. Consider the graph H with $V(H) = V(G') \setminus T$ and $(v, u) \in E(H)$ if and only if $\varphi^{-1}(v), \varphi^{-1}(u)$ are adjacent in G'. Note that the graphs G'/Z' and H are isomorphic, therefore we prove that H is a split graph. Let $\hat{C} = \{\varphi(v) \mid \mathcal{B} \cup \{s\}\}$ and $\hat{I} = V(H) \setminus \hat{C}$. For $v, u \in \hat{I}$, $\varphi^{-1}(v) = \{v\}$ and $\varphi^{-1}(u) = \{u\}$ and $\{v\}, \{u\}$ are non-adjacent in G'. Therefore, $(v, u) \notin E(H)$. This proves that $\hat{I}$ is an independent set in H. For $b, b' \in \mathcal{B} \subset \hat{C}$, $(b, b') \in E(G')$, therefore $(\varphi(v), \varphi(u)) \in E(H)$. Since Z is a solution to SRBPCin $(G, \mathcal{R}, \mathcal{B})$, for $b \in \mathcal{B}$, there exists $r_i \in Z$ such that $(b, r_i) \in E(G')$, therefore, $W(s)$ and b are adjacent in G'. Hence, $(\varphi(s), \varphi(b)) \in E(H')$. This finishes the proof that $\hat{C}$ induces a clique in H and that H is a split graph.

In the reverse direction, let S be a solution to (G', k') of Split Contraction, and denote $H = G'/S$. Let $\mathcal{W}$ be the *H-witness structure* of G, φ be the associated surjective function and $h_s = \varphi(s)$. From Lemmas 9.9 and 9.11 it follows that for all $i \in [k]$, there exists $r_i^\star \in R_i$ such that $(x_i, r_i^\star) \in S$ and $r_i^\star, x_i \in W(h_s)$. Let $Z = \{r_i^\star \mid i \in [k]\}$. We will show that Z is a solution to SRBPC in $(G, \mathcal{R}, \mathcal{B})$. Since $|W(h_s)| \geq k' + 1 = 2k + 1$, it holds that for all $v \in V(H) \setminus \{h_s\}$, $|W(v)| = 1$. This implies that for all $b \in \mathcal{B}$, $b \notin W(h_s)$. Also observe that since $x_i \in W(h_s)$ for all $i \in [k]$ and $|W(h_s)| = k' + 1 = 2k + 1$, we have that $|W(h_s) \cap R_i| = 1$. This implies that $|Z| = k$ and $|Z \cap R_i| = 1$, for all $i \in [k]$. To show that Z is indeed a solution, it is enough to show that for all $b_j \in \mathcal{B}$, $|Z \cap N(b_j)| = 1$. Towards a contradiction, assume there exists $b_j \in \mathcal{B}$ such that $|Z \cap N(b_j)| \neq 1$. Let $h_{b_j} = \varphi(b_j)$. We consider the following two cases.

- If $|Z \cap N_{G'}(b_j)| < 1$. Recall that $W(h_{b_j}) = \{b_j\}$. Further, $N_{G'}(b_j) \subseteq \mathcal{R} \cup \{y_1^j, \ldots, y_t^j\}$, $Z = W(h_s) \cap \mathcal{R}$ and by our assumption $Z \cap N_{G'}(b_j) = \emptyset$. But then $W(h_s)$ and $W(h_{b_j})$ are not adjacent in G'. However, Lemma 9.8 implies that $(h_s, h_{b_j}) \in E(H)$, contradicting our assumption that $|Z \cap N(b_j)| < 1$.

- If $|Z \cap N_{G'}(b_j)| > 1$, then there exists $j, j' \in [k]$, $j \neq j'$ such that $r_j^\star, r_{j'}^\star \in N_{G'}(b)$. Then it follows that $|\cup_{i \in [k]} N(r_i^\star)| < \ell = dk$. But then there exists $b' \in \mathcal{B}$ such that $W(\varphi(b'))$ and $W(h_s)$ are non-adjacent in G', contradicting that $(\varphi(b'), h_s) \in E(H)$ from Lemma 9.8.

This completes the proof. □

Next, we prove the main theorem of this section.

Theorem 9.3. Split Contraction *is* W[1]-hard *when parmeterized by the size of a solution.*

Proof. Proof follows from construction, Lemma 9.12 and the W[1]-hardness of SRBPC (Theorem 9.4). □

9.2.2 W[1]-hardness of Special Red-Blue Perfect Code

In this section, we show that SRBPC is W[1]-hard, when parameterized by the solution size. We give a reduction from Multi-Colored Clique (MCC) to SRBPC. Recall that MCC takes as an input a graph G with a partition $V_1, V_2, \ldots, V_k$ of $V(G)$, and the objective is to check if there a set $X \subseteq V(G)$ such that for each $i \in [k]$, $|X \cap V_i| = 1$ and $G[X]$ is a clique. It is known that MCC is W[1]-hard when parameterized by k [FHRV09].

The intuitive description of the reduction we are going to construct below is as follows. Let $(G, V_1, \ldots, V_k)$ of be an instance of MCC. We will often refer to the sets V_i as color classes. For each color class we create a vertex selection gadget. Then we have edge selection gadgets which ensure that between every pair of color classes an edge is selected. The vertex selection gadget ensures that the vertex chosen is same as the one incident to the edge chosen by the edge selection gadget. Finally, we have a coherence gadget which ensures that all the edges that are incident to a color class are incident to the same vertex in this color class.

For technical reasons we will assume that the number of vertices in G is 2^t, for some $t \in \mathbb{N}$. Note that this can be easily achieved by adding dummy vertices to an arbitrary color class with no edge incident to them. This results in at most doubling of the number of vertices in the graph. For our purposes, we also assign a unique t-bit-string to each vertex $v \in V(G)$. Next, we move to the description of the instance $(G', \mathcal{R}, \mathcal{B})$ of SRBPC that we create.

Edge Selection Gadget. For $i, j \in [k]$, $i \neq j$, we create an edge selection gadget E_{ij} as follows. For each edge $(u, v) \in E(G)$, such that $u \in V_i$ and $v \in V_j$, we add a vertex e_{uv} to E_{ij}. We emphasize the fact that E_{ij} and E_{ji} denote the same set. Similarly, for an edge $(u, v) \in E(G)$, the vertices e_{vu} and e_{uv} are the same vertex. The symmetry in the indices/subscripts holds only for the *edge selection* gadgets.

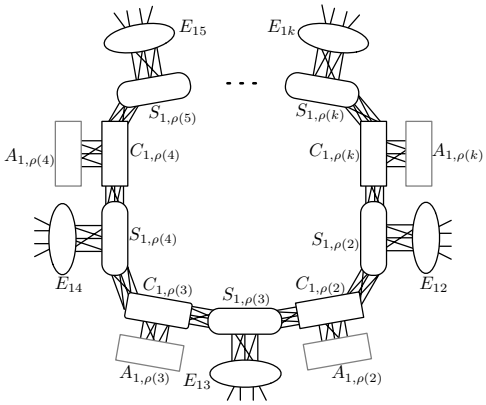

Figure 9.3: Illustration of edges between Vertex Selection Gadget, Coherence Gadget for $i = 1$, and Edge Selection Gadget.

For the description of the *vertex selection* and *coherence* gadgets we will need the following notation. For $i \in [k]$, the set $T_i = \{j \in [k] \mid j \neq i\}$ has a natural total ordering ρ_i, specifically the order given by the relation $<$ defined on $\mathbb{N}$. Therefore, by $\rho_i(j)$ we denote the position of j in the total ordering of T_i (1st position is denoted by 1). We will slightly abuse the notation and drop the subscript i from ρ_i whenever it is clear from the context.

Vertex Selection Gadget. For each color class $i \in [k]$ we have a vertex selection gadget $\mathcal{S}_i$. For $i \in [k]$, $\mathcal{S}_i$ consists of $k-1$ sets of vertices $S_{i,\rho(j)}$, where $j \in [k] \setminus \{i\}$. Here, $S_{i,\rho(j)}$ is a set of $2t$ vertices denoted by $x_0^{i,\rho(j)}, x_1^{i,\rho(j)}, \ldots, x_{t-1}^{i,\rho(j)}, y_0^{i,\rho(j)}, y_1^{i,\rho(j)}, \ldots, y_{t-1}^{i,\rho(j)}$. The intuition behind the construction of the set $S_{i,\rho(j)}$ is to encode the bit representation of the vertices in V_i. The size of $S_{i,\rho(j)}$ is twice the size of the bit-representation for achieving the degree constraints of the vertices in the instance of SRBPC to be created.

Coherence Gadget. Consider $i \in [k]$ and $j \in [k] \setminus \{i\}$. We have a set $C_{i,\rho(j)}$ containing copies of vertices in V_i, *i.e.* $|C_{i,\rho(j)}| = |V_i|$. For a vertex $v \in V_i$, its copy in $C_{i,\rho(j)}$ is denoted by $c_v^{i,\rho(j)}$. Also, we have a set $A_{i,\rho(j)}$ containing a vertex $a_\ell^{i,\rho(j)}$, for each $\ell \in [t]$. The set $A_{i,\rho(j)}$ is added only to ensure some degree constraints in the construction. For each $u \in A_{i,\rho(j)}$ and $v \in C_{i,\rho(j)}$, we add the edge (u, v) to $E(G')$, i.e., $G'[A_{i,\rho(j)} \cup C_{i,\rho(j)}]$ is a complete bipartite graph. By $\mathcal{A}_i$ we denote the set $\cup_{j \in [k] \setminus \{i\}} A_{i,\rho(j)}$.

We now move to the description of the edges between vertex selection, edge selection and coherence gadgets. We refer the reader to Figure 9.3 for an illustration of the reduction.

Edges between gadgets. Let $i, j \in [k], i \neq j$, and $u \in V_i, v \in V_j$ such that $(u, v) \in E(G)$. Recall that corresponding to the edge (u, v), we have a vertex e_{uv} in E_{ij} (which is same as E_{ji}). Let $b_0 b_1 \ldots b_{t-1}$ be the unique bit-string assigned to u. We add an edge between $x_\ell^{i,\rho(j)} \in S_{i,\rho(j)}$ and e_{uv} in G' if and only if $b_\ell = 1$, here $\ell \in \{0, \ldots, t-1\}$. Similarly, we add an edge between $y_\ell^{i,\rho(j)} \in S_{i,\rho(j)}$ and e_{uv} in G' if and only if $b_\ell = 0$; here, $\ell \in \{0, \ldots, t-1\}$. Refer to Figure 9.4 for a pictorial illustration.

We now describe the edges between $C_{i,\rho(j)}$ and $S_{i,\rho(j)}$. We will assume modulo k-

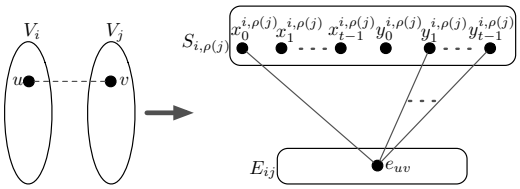

Figure 9.4: Edges between E_{ij} and S_{ij}, assuming the bit-string associated with v has $b_0 = 1$ and $b_\ell = 0$ for all $\ell \in [t-1]$.

arithmetics for the computation of indices. We note that the notation ρ is used only for ease in specification and modulo index computation to work properly. For $i, j \in [k]$, $i \neq j$ and $v \in V_i$, there is a vertex $c_v^{i,\rho(j)} \in C_{i,\rho(j)}$. Let $b_0 b_1 \dots b_{t-1}$ be the unique bit-string assigned to v. We add an edge between $x_\ell^{i,\rho(j)} \in S_{i,\rho(j)}$ and $c_v^{i,\rho(j)}$ in G' if and only if $b_\ell = 0$, here $\ell \in \{0, \dots, t-1\}$. Similarly, we add an edge between $y_\ell^{i,\rho(j)+1} \in S_{i,\rho(j)+1}$ and $c_v^{i,\rho(j)}$ in G' if and only if $b_\ell = 1$, here $\ell \in \{0, \dots, t-1\}$. This finishes the description of the graph G'.

Now we move on to partitioning the vertices in $V(G')$ into two sets $\mathcal{R}$ and $\mathcal{B}$. Then we further partition $\mathcal{R}$. For $i, j \in [k]$, $i \neq j$ we add all the vertices in $C_{i,\rho(j)}$ and E_{ij} to $\mathcal{R}$. All the remaining vertices are added to the set $\mathcal{B}$. The set $\mathcal{R}$ is partitioned into E_{ij} and $C_{i,\rho(j)}$, where $i \neq j$. Observe that since $E_{ij} = E_{ji}$ for all $i \neq j$ we have $k(k-1) + \binom{k}{2}$ parts of $\mathcal{R}$ and the degree of each vertex in $\mathcal{R}$ is $2t$. This completes the description of the instance $(G', \mathcal{R}, \mathcal{B})$ of SRBPC.

Next, we prove some lemmata that will help us in establishing the equivalence between the two instances.

Lemma 9.13. *Let $(G', \mathcal{R}, \mathcal{B})$ be a yes instance of* SRBPC *and R be one of its solution. If for some $i, j \in [k], i \neq j$, $u \in V_i$, $v \in V_j$ we have $e_{uv} \in R$ then the following holds.*

- $c_u^{i,\rho(j)}, c_u^{i,\rho(j)-1} \in R$.
- $c_v^{j,\rho(i)}, c_v^{j,\rho(i)-1} \in R$.

Proof. We give proof only for the first part of the lemma. The second one follows from an analogous argument. Consider $i, j \in [k], i \neq j$, $u \in V_i$, $v \in V_j$, such that $e_{uv} \in R$. Let $\bar{b}_u = b_0 b_1 \dots b_{t-1}$ be the unique bit-string assigned to u. Observe that all the vertices $x_\ell^{i,\rho(j)}$ with $b_\ell = 1$, for $\ell \in \{0, \dots, t-1\}$ are adjacent to e_{uv}. Since R is a solution, it must contain a vertex from $C_{i,\rho(j)}$. Let the unique vertex in $R \cap C_{i,\rho(j)}$ be $c_w^{i,\rho(j)}$. Suppose $w \neq u$. Consider the difference in the bit-string representation $\bar{b}_w$, of w and $\bar{b}_u$. Since $w \neq u$, $\bar{b}_w$ and $\bar{b}_u$ differs in at least one position, let the first such position be q. If $b_q = 1$ (q^{th} bit in $\bar{b}_u$) then q^{th} bit in $\bar{b}_w$ is 0. But then, $x_q^{i,\rho(j)}$ is adjacent to two vertices, namely e_{uv} and $c_w^{i,\rho(j)}$, contradicting that R is a solution. If $b_q = 0$, then $x_q^{i,\rho(j)}$ is not adjacent to e_{uv} and $c_u^{i,\rho(j)}$. Recall that $N(x_q^{i,\rho(j)}) \subseteq E_{ij} \cup C_{i,\rho(j)}$. Hence, $x_q^{i,\rho(j)}$ is non-adjacent to any vertex in R, a contradiction. Therefore, $u = w$ and $c_u^{i,\rho(j)} \in R$. A similar argument can be given for proving $c_u^{i,\rho(j)-1} \in R$. This completes the proof. □

Lemma 9.14. *Let $(G', \mathcal{R}, \mathcal{B})$ be a yes instance of* SRBPC *and R be a solution. If for some $i, j \in [k], i \neq j$ and $u \in V_i$ we have $c_u^{i,\rho(j)} \in R$ then there exists some $v \in V_j$ such that $e_{uv} \in R$.*

Proof. Towards a contradiction assume that for some $i, j \in [k], i \neq j$ and $u \in V_i$ we have $c_u^{i,\rho(j)} \in R$ and for all $v \in V_j$, $e_{uv} \notin R$. Let $\bar{b}_u = b_0 b_1 \dots b_{t-1}$ be the unique bit-string assigned to u. For all $\ell \in \{0, \dots, t-1\}$ such that $b_\ell = 0$, $x_\ell^{i,\rho(j)}$ is adjacent to $c_u^{i,\rho(j)}$. Since R is a solution it must contain a vertex $e_{wz} \in E_{ij}$, where $w \in V_i$ and $z \in V_j$. By assumption $w \neq u$. But by Lemma 9.13, $c_w^{i,\rho(j)} \in R$, contradicting that $|R \cap C_{i,\rho(j)}| = 1$. This implies that $w = u$. □

Lemma 9.15. *Let $(G', \mathcal{R}, \mathcal{B})$ be a yes instance of* SRBPC *and R be a solution. If for some $i, j \in [k], i \neq j$ and $u \in V_i$ we have $c_u^{i,\rho(j)} \in R$ then for all $\ell \in [k] \setminus \{i\}$ we have $c_u^{i,\rho(\ell)} \in R$.*

Proof. Follows from Lemmas 9.13 and 9.14. □

Lemma 9.16. *(G, k) is a yes instance of* MCC *if and only if $(G', \mathcal{R}, \mathcal{B})$ is a yes instance of* SRBPC*.*

Proof. In the forward direction, let $V = \{v_i \mid i \in [k]\}$ be a solution to MCC for $(G, V_1, \dots, V_k)$. Let $\bar{b}_i$ be the unique bit-string assigned to v_i, for $i \in [k]$. Also, we let $R = \{c_{v_i}^{i,\rho(j)} \mid i, j \in [k], i \neq j\} \cup \{e_{v_i v_j} \mid i, j \in [k], i \neq j\}$. Observe that $|R \cap C_{ij}| = 1$, for all $i, j \in [k], i \neq j$. Similarly, $|R \cap E_{ij}| = 1$, for all $i, j \in [k], i \neq j$. Recall that $\mathcal{B} = V(G') \setminus \mathcal{R} = (\cup_{i \in [k]} \mathcal{S}_i) \cup (\cup_{i \in [k]} \mathcal{A}_i)$. Here, for $i \in [k]$, we have $\mathcal{S}_i = \cup_{j \in [k] \setminus \{i\}} S_{i,\rho(j)}$ and $\mathcal{A}_i = \cup_{j \in [k] \setminus \{i\}} A_{i,\rho(j)}$. Observe that for each $i \in [k]$, each vertex in $\mathcal{A}_i$ is adjacent to exactly one vertex in R. Next, we show that for $i, j \in [k]$, $i \neq j$, each vertex in $S_{i,\rho(j)}$ is adjacent to exactly one vertex in R. Recall that $S_{i,\rho(j)}$ is adjacent only to vertices in $C_{i,\rho(j)}, C_{i,\rho(j)-1}$ and E_{ij}. Consider a vertex $x_\ell^{i,\rho(j)} \in S_{i,\rho(j)}$, for $\ell \in \{0, \dots, t-1\}$. Assume that ℓ^{th} bit of $\bar{b}_i$ is 1. This implies that $x_\ell^{i,\rho(j)}$ is adjacent to $e_{v_i v_j}$ and not adjacent to $c_{v_i}^{i,\rho(j)}$. Also, $x_\ell^{i,\rho(j)}$ is non-adjacent to any other vertex in R. Hence it follows that $|R \cap N(x_{v_i}^{i,\rho(j)})| = 1$. An analogous argument can be given for the case when ℓ^{th} bit of $\bar{b}_i$ is 0. Furthermore, we can give a symmetric argument for a vertex $y_\ell^{i,\rho(j)} \in S_{i,\rho(j)}$, where $\ell \in \{0, \dots, t-1\}$. This finishes the proof of the forward direction.

In the reverse direction, let R be a solution to SRBPC for $(G', \mathcal{R}, \mathcal{B})$. Note that for $i, j \in [k]$, $i \neq j$, $|R \cap E_{ij}| = 1$ and $|R \cap C_{i,\rho(j)}| = 1$. Let $X = \{v \in V(G) \mid c_v^{i,\rho(j)} \in R\}$. It follows from Lemma 9.15 that for all $i \in [k]$, $|X \cap V_i| = 1$. Consider $u, v \in X$, where $u \in V_i$, $v \in V_j$ and $i \neq j$. From Lemma 9.15 for all $\ell \in [k]$, $i \neq \ell$ we have $c_u^{i,\rho(\ell)} \in R$ and for all $\ell' \in [k]$, $j \neq \ell'$ we have $c_u^{j,\rho(\ell')} \in R$. This together with Lemma 9.14 imply that $e_{uv} \in R$. Hence $(u, v) \in E(G)$. Since choice of u, v was arbitrary, it implies that $G[X]$ is a clique. □

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

Theorem 9.4. SRBPC *when parameterized by the number of parts in $\mathcal{R}$ is* W[1]-hard*.*

Proof. Follows from construction of the instance $(G', \mathcal{R}, \mathcal{B})$ of SRBPC for the given instance (G, k) of MCC, Lemma 9.16, and W[1]-hardness of MCC. □

9.3 FPT Algorithm for Split Contraction Parameterized by Vertex Cover

In this section, we give an FPT algorithm for Split Contraction when parameterized by the size of a minimum vertex cover. In Section 9.3.1, we give an algorithm running in time $2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$ for Split Contraction parameterized by ℓ, the size of minimum vertex cover, when the input graph is connected. In this section we use the algorithm for solving Split Contraction parameterized by the size of a minimum vertex cover on connected graphs to solve Split Contraction on general graphs.

Let (G, k) be an instance of Split Contraction and $C_1, \ldots, C_t$ be the set of connected components of G. Observe that except for one connected component in G, every other component must be contracted to a single vertex, since all the vertices in these components must be part of the independent set. Also, note that for contracting a component to a single vertex we need to contract a spanning tree in it. Therefore, for each $i \in [t]$ let $k_i = k - \sum_{j \in [t] \setminus \{i\}} |V(C_j) - 1|$ and solve the instance (C_i, k_i). If for any $i \in [t]$ the algorithm returns a *yes* instance then we return that (G, k) is a *yes* instance, otherwise return that (G, k) is a *no* instance. The correctness of the above algorithm relies on the correctness of the algorithm for connected graphs and thus results in the following theorem.

Theorem 9.5. Split Contraction *admits an algorithm running in time* $2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$, *where* ℓ *is the size of the minimum vertex cover of the input graph.*

9.3.1 Algorithm for Split Contraction on connected graphs

In this section, we give an FPT algorithm for Split Contraction parameterized by the size of a minimum vertex cover when the input graph is a connected. Let (G, k) be an instance of Split Contraction, where G is a connected graph. We start by computing a minimum sized vertex cover S in G. Computing a minimum vertex cover in a graph can be done in time $1.2738^{\ell} \cdot n^{\mathcal{O}(1)}$, where ℓ is the size of a minimum vertex cover in the graph [CKX10]. We first prove the following Lemma which will be useful for the algorithm.

Lemma 9.17. *Let* G *be a connected graph,* S *be a minimum vertex cover in* G *and* $K \subseteq E(G)$ *be a set of minimum size such that* G/K *is a split graph, then* $|K| < 2|S|$.

Proof. Let T be a *dfs*-tree of G and L_T denote the set of leaves in T. It is well known that $V(T) \setminus L_T$ is a connected vertex cover in G and $|V(T) \setminus L_T| \leq 2|S|$ [Sav82]. Let E_T be the edges in T that are non-adjacent to vertices in L_T. Observe that G/E_T is a split graph. Thus, $|K| \leq |E_T| < |V(T) \setminus L_T| \leq 2|S|$. □

Let $I = V(G) \setminus S$. Since S is a vertex cover, I is an independent set in G. We define an equivalence relation $\mathcal{R}$ among the vertices in I based on their neighborhood in S. Basically, $u, v \in I$ belong to the same equivalence class if and only if $N(u) = N(v)$. Let $I_1, \ldots, I_t$ be the equivalence classes of $\mathcal{R}$. Note that $t \leq 2^{|S|}$. We apply the following reduction rules exhaustively.

Reduction Rule 9.1. *If* $k \geq 2|S|$, *then return that* (G, k) *is a yes instance.*

Lemma 9.18. *Reduction Rule 9.1 is safe.*

Proof. The proof follows from Lemma 9.17. □

Reduction Rule 9.2. *For $j \in [t]$, if there is an equivalence class I_j, such that $|I_j| > 2k+2$, then delete an arbitrary vertex $v \in I_j$ from G. That is, the resulting instance is $(G - \{v\}, k)$.*

Lemma 9.19. *Reduction Rule 9.2 is safe.*

Proof. Let (G, k) be an instance of Split Contraction. Furthermore, for some $j \in [t]$ we have $|I_j| > 2k+2$ and let $v \in I_j$ and let $(G' = G - \{v\}, k)$. In the forward direction, let X be a solution to (G, k), $\mathcal{W}$ be the $H = G/X$-*witness structure* of G with φ being the underlying surjective function. If no edge in X is incident to v, then X is also a solution in (G', k) as G'/X is an induced subgraph of G/X. Let $X_v \subseteq X$ be those edges which are incident to v. There is a vertex $v' \in I_j$ that is not adjacent to any edge in X since $|I_j| > 2k+2$. Let $X_{v'} = \{(u, v') \mid (u, v) \in X_v\}$, *i.e.*, $X_{v'}$ is the set of edges obtained by replacing v by v' in X_v. Note that such a replacement is possible because $N(v) = N(v')$. Let $X' = (X \setminus X_v) \cup X_{v'}$. Clearly, the size of $|X'| \leq |X| \leq k$. We define the surjective function $\varphi' : V(G') \to V(H) \setminus \{\varphi(v')\}$ as follows. For $u \in V(G')$, $u \neq v'$, $\varphi'(u) = \varphi(u)$ and $\varphi'(v') = \varphi(v)$ (recall, $\varphi(v) \neq \varphi(v')$). For $h \in V(H) \setminus \{\varphi(v')\}$ we let $W'(h) = \varphi^{-1}(h)$. Let H' to be the graph with $V(H') = V(H) \setminus \{\varphi(v')\}$ and $(h_1, h_2) \in E(H')$ if and only if $W'(h_1)$ and $W'(h_2)$ are adjacent in G'. Since, $|W(\varphi(v'))| = 1$ we have that for any vertex $h \in V(H') \setminus \{\varphi' v'\}$, $W'(h) = W(h)$ and $W'(\varphi'(v')) = (W(\varphi(v)) \setminus \{v\}) \cup \{v'\}$. Observe that since $N_G(v) = N_G(v')$, we have that for all $h \in V(H')$, $G'[W(h)]$ is connected, and hence it follows that G' is contractible to H'. Furthermore, to show that G'/X' is a split graph, it is enough to show that H' is a split graph. Since $N_G(v) = N_G(v')$, the graphs H, H' differs only in the vertex $\varphi(v') \in V(H)$ ($\varphi(v') \notin V(H')$). But any induced subgraph of a split graph, is a split graph, hence it follows that H' is a split graph.

In the reverse direction, let X be a solution to Split Contraction in (G', k), $H = G'/X$ and $\varphi, \mathcal{W}$ be the underlying surjective function and H-*witness structure* of G', respectively. Observe that X can be incident to at most $2k$ vertices in I_j, therefore there are vertices $u, u' \in V(G') \cap I_j$, $u \neq u'$ which are not incident to any edge in X *i.e.* $|W(\varphi(u))| = |W(\varphi(u')| = 1$. Let $\mathcal{C}'$ and $\mathcal{I}'$ be the clique and independent set respectively in H. Note that at least one of $\varphi(u), \varphi(u')$ belongs to $\mathcal{I}'$, say $\varphi(u) \in \mathcal{I}'$. We define the surjective function $\varphi_v : V(G) \to V(H) \cup \{v\}$ as follows. For $x \in V(G) \setminus \{v\}$, $\varphi_v(x) = \varphi(x)$ and $\varphi(v) = v$. Let H_v be the graph with vertex set $V(H) \cup \{v\}$ and $(h, h') \in E(H_v)$ if and only if $W_v(h)$ and $W_v(h')$ are adjacent in G. Notice that φ_v satisfies all the properties for it to define the contractibility of G to H_v. Recall that $N(v) = N(u)$. But then $\mathcal{I}' \cup \{v\}$ is an independent set and $\mathcal{C}'$ is a clique, partitioning the vertices of H_v, therefore H_v is a split graph. But notice that indeed $H_v = G/X$, hence the claim follows. □

Given an instance (G, k) to Split Contraction, we apply Reduction Rules 9.1 and 9.2 until no longer applicable. For simplicity we denote the resulting instance where none of the reduction rules are applicable by (G, k) itself. Observe that the number of vertices in G is upper bounded by $(2k+2) \cdot 2^{\ell} + \ell \leq (4\ell+2) \cdot 2^{\ell} + \ell = 2^{\mathcal{O}(\ell)}$, where $\ell = |S|$. This follows from the fact that the reduction rules are not applicable and Lemma 9.17.

Observe that the number of vertices in G that are incident to an edge in the solution is bounded by $2k$. We guess $X \subseteq V(G)$ of size at most $2k$, which is incident to at least one edge in the solution. Note that the number of such guesses is upper bounded by $\binom{2^{\mathcal{O}(\ell)}}{2\ell} = 2^{\mathcal{O}(\ell^2)}$. The number of edges in $G[X]$ is bounded by $\mathcal{O}(\ell^2)$. For each $E' \subseteq E(G[X])$ of size at most k, we check if G/E' is a split graph. If for all $X \subseteq V(G)$ of size at most $2k$ and $E' \subseteq E(G[X])$, G/E' is not a split graph then we return that (G, k) is a *no* instance, otherwise we return that (G, k) is a *yes* instance of Split Contraction.

Correctness and running time analysis. Given an instance (G, k), where G is a connected graph on n vertices, the algorithm starts by computing a minimum sized vertex cover S in G and an equivalence relation based on the neighborhood in G. The time required for this step of the algorithm is bounded by $\mathcal{O}(1.2738^{\ell} \cdot n^{\mathcal{O}(1)})$, where $\ell = |S|$ [CKX10]. The algorithm then applies one of the reduction rule, if applicable. The reduction rules can be applied in polynomial time and their safeness follows from Lemma 9.18 and 9.19. When none of the reduction rules are applicable then the algorithm solves the instance in a brute force way and here its correctness is immediate. In the brute force step the algorithm guess a subset $X \subseteq V(G)$ of size at most $2k$ which are incident to an edge in the solution. The number of such subsets is bounded by $2^{\mathcal{O}(\ell \cdot k)}$, which in turn is bounded by $2^{\mathcal{O}(\ell^2)}$. For the guessed subset X, the algorithm tries for all possible sets of edges E' of size at most k in $E(G[X])$. The number of such edge subsets is upper bounded by $2^{\mathcal{O}(k \log k)}$ which is bounded by $2^{\mathcal{O}(\ell^2)}$. Checking if G/E' is a split graph takes linear time [Gol04]. Hence, the total running time is bounded by $1.2738^{\ell} \cdot n^{\mathcal{O}(1)} + 2^{\mathcal{O}(\ell^2)} \cdot 2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$.

Theorem 9.6. Split Contraction *on connected graphs admits an algorithm running in time* $2^{\mathcal{O}(\ell^2)} \cdot n^{\mathcal{O}(1)}$*, where* ℓ *is the size of a minimum vertex cover of the input graph.*

Part III

New Results on Simultaneous $\mathcal{F}$-Deletion Problems

Chapter 10

Simultaneous Feedback Vertex Set

For an α-edge-colored graph G, an *α-simultaneous feedback vertex set* (or *α-simfvs* for short) is a set $S \subseteq V(G)$ such that $G_i - S$ is a forest for each $i \in [\alpha]$. In this chapter, we look at the problem Simultaneous Feedback Vertex Set, which is formally defined below.

Simultaneous Feedback Vertex Set (Sim-FVS) **Parameter:** k and α **Input:** Given an α-edge-colored graph $G = (V, E_1, \ldots, E_\alpha)$ and an integer k. **Question:** Is there a set $S \subseteq V(G)$ of size at most k such that for each $i \in [\alpha]$, $G_i - S$ is a forest?

We show that, like its classical counterpart ($\alpha = 1$), Sim-FVS parameterized by k is in FPT and admits a polynomial kernel, for any fixed constant α. In particular, we obtain the following results.

- An FPT algorithm running in $\mathcal{O}^\star(23^{\alpha k})$ time. For the special case of $\alpha = 2$, we give a faster algorithm running in $\mathcal{O}^\star(81^k)$ time. Using a completely different approach, Ye [Ye15] independently (and simultaneously) gave an algorithm running in $\mathcal{O}^\star(k^{\alpha k})$ time for the general case and an algorithm running in $\mathcal{O}^\star(c^k)$ time, $c > 81$, for the case $\alpha = 2$.
- For every constant α, we obtain a kernel with $\mathcal{O}(\alpha k^{3(\alpha+1)})$ vertices.
- The running time of our algorithm implies that Sim-FVS is FPT even when $\alpha \in o(\log n)$. We complement this positive result by showing that for $\alpha \in \mathcal{O}(\log n)$, where n is the number of vertices in the input graph, Sim-FVS is W[1]-hard.

Our algorithms and kernel build on the tools and methods developed for Feedback Vertex Set [CFK$^+$15]. However, we need to develop both new branching rules as well as new reduction rules. The main reason why our results do not follow directly from earlier work on FVS is the following. Many (if not all) parameterized algorithms, as well as kernelization algorithms, developed for the FVS problem [CFK$^+$15] exploit the fact that vertices of degree two or less in the input graph are, in some sense, irrelevant. In other words, vertices of degree one or zero cannot participate in any cycle and every cycle containing any degree-two vertex must contain both of its neighbors. Hence, if this degree-two vertex is part of a feedback vertex set then it can be replaced by either one of its neighbors. Unfortunately (or fortunately for us), this property does not hold for the Sim-FVS problem, even when $\alpha = 2$. For instance, if a vertex is incident to

edges of color 1 and two edges of color 2, it might in fact be participating in two distinct cycles. Hence, it is not possible to neglect (or shortcut) this vertex in any graph induced on edges of one color. As we shall see, most of the new algorithmic techniques that we present deal with vertices of exactly this type. Although very tightly related to one another, we show that there are subtle and interesting differences separating the FVS problem from the Sim-FVS problem, even for $\alpha = 2$.

In Section 10.1, we present an algorithm solving the Sim-FVS problem, parameterized by solution size k, in $\mathcal{O}^\star(23^{\alpha k})$ time. Our algorithm follows the iterative compression paradigm introduced by Reed et al. [RSV04] combined with new reduction and branching rules. Our main new branching rule can be described as follows: Given a maximal degree-two path in some G_i, $i \in [\alpha]$, we branch depending on whether there is a vertex from this path participating in an α-simultaneous feedback vertex set or not. In the branch where we guess that a solution contains a vertex from this path, we construct a color i cycle which is isolated from the rest of the graph. In the other branch, we are able to follow known strategies by "simulating" the classical FVS problem. Observe that we can never have more than k isolated cycles of the same color. Hence, by incorporating this fact into our measure we are guaranteed to make "progress" in both branches. For the base case, each G_i is a disjoint union of cycles (though not G) and to find an α-simultaneous feedback vertex set for G we cast the remaining problem as an instance of Hitting Set parameterized by the size of the family. For $\alpha = 2$, we can instead use an algorithm for finding maximum matchings in an auxiliary graph. Using this fact we give a faster, $\mathcal{O}^\star(81^k)$ time, algorithm for the case $\alpha = 2$.

In Section 10.2, we tackle the question of kernelization and present a polynomial kernel for the problem, for constant α. Our kernel has $\mathcal{O}(\alpha k^{3(\alpha+1)})$ vertices and requires new insights into the possible structures induced by those special vertices discussed above. In particular, we enumerate all maximal degree-two paths in each G_i after deleting a feedback vertex set in G_i and study how such paths interact with each other. Using marking techniques, we are able to "unwind" long degree-two paths by making a private copy of each unmarked vertices for each color class. This unwinding leads to "normal" degree-two paths on which classical reduction rules can be applied, and hence we obtain the desired kernel.

Finally, we consider the dependence between α and both the size of our kernel and the running time of our algorithm in Section 10.3. We show that even for $\alpha \in \mathcal{O}(\log n)$, where n is the number of vertices in the input graph, Sim-FVS becomes W[1]-hard. We show hardness via a new problem of independent interest which we denote by Partitioned Hitting Set. Partitioned Hitting Set is a special variant of the well-known Hitting Set problem. Recall that in the Hitting Set problem, we are given a universe $\mathcal{U}$, a family $\mathcal{F} = \{f_1, f_2, \ldots\}$ of subsets of $\mathcal{U}$, and an integer k, and the goal is to "hit" every set in $\mathcal{F}$ using at most k elements from $\mathcal{U}$. Formally, the goal is to determine whether there exists $U \subseteq \mathcal{U}$ such that $|U| \leq k$ and $U \cap f \neq \emptyset$, for all $f \in \mathcal{F}$. The input to the Partitioned Hitting Set problem consists of a tuple $(\mathcal{U}, \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha, k)$, where each $\mathcal{F}_i$, $i \in [\alpha]$, is a collection of subsets of the finite universe $\mathcal{U}$, k is a positive integer, and all the sets within a family $\mathcal{F}_i$, $i \in [\alpha]$, are pairwise disjoint. As in the Hitting Set problem, the goal is to determine whether there exists a subset U of $\mathcal{U}$ of cardinality at most k such that for every $f \in \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha$, $f \cap U$ is nonempty. We show that Partitioned Hitting Set is W[1]-hard for $\alpha \in \mathcal{O}(\log |\mathcal{U}||\mathcal{F}|)$ via a reduction from Partitioned Subgraph Isomorphism, and we show that Sim-FVS is W[1]-hard for

$\alpha \in \mathcal{O}(\log n)$ via a reduction from Partitioned Hitting Set with $\alpha \in \mathcal{O}(\log |\mathcal{U}||\mathcal{F}|)$. Along the way, we also show, using a somewhat simpler reduction from Hitting Set, that Sim-FVS is W[2]-hard for $\alpha \in \mathcal{O}(n)$.

10.1 FPT Algorithm for Simultaneous Feedback Vertex Set

We give an algorithm for the Sim-FVS problem using the method of iterative compression [RSV04, CFK+15]. We briefly describe the general scheme of iterative compression and then focus on an algorithm for solving the disjoint version of Sim-FVS. We refer the reader to [RSV04, CFK+15] for more details. In the Disjoint Sim-FVS problem, we are given an α-colored graph $G = (V, E_1, \ldots, E_\alpha)$, an integer k, and an α-simfvs W in G of size $k+1$. The objective is to find an α-simfvs $S \subseteq V(G) \setminus W$ of size at most k, or correctly conclude the non-existence of such an α-simfvs.

Let $(G = (V, E_1, \ldots, E_\alpha), k)$ be an instance of Sim-FVS. We fix an arbitrary ordering $(v_1, v_2, \ldots, v_n)$ on the vertices of G. For $j \in [n]$, we let $G^{(j)}$ denote the subgraph of G induced on the first j vertices. Note that when $j = k$, we can take the vertex set of $G^{(k)}$ as an α-simfvs in $G^{(k)}$ of size k. Now suppose that for some $j > k$, we have constructed an α-simfvs $S^{(j)}$ of $G^{(j)}$ of size at most k. Then, in the graph $G^{(j+1)}$, the set $Z^{(j+1)} = S^{(j)} \cup \{v_{j+1}\}$ is an α-simfvs of size at most $k+1$. If in fact $|Z^{(j+1)}| \leq k$ then we are done, i.e. we let $S^{(j+1)} = Z^{(j+1)}$ and proceed to the next iteration. Otherwise, $|Z^{(j+1)}| = k+1$ and we need to "compress" into a smaller solution (if it exists). To that end, we first guess the intersection X of $S^{(j+1)}$ with $Z^{(j+1)}$. In other words, for every $q \in \{0\} \cup [k]$ and every subset X of $Z^{(j+1)}$ of size q, we construct an instance (G', W', k') of Disjoint Sim-FVS as follows. We let $G' = G^{(j+1)} - X$, $W' = Z^{(j+1)} \setminus X$, and $k' = k - q$. Note that $|W'| = k - q + 1$, so $|W'|$ is one larger than k'. If G' does not admit an α-simfvs of size at most k, then of course neither does G, and we may terminate (and declare a no-instance). If on the other hand $S^{(j+1)}$ has been successfully found (which is the union of X and the solution to instance (G', W', k')), then we proceed to the next graph $G^{(j+2)}$, and so on. Finally, observe that $G^{(n)} = G$, so we eventually either find an α-simfvs of size at most k in G or conclude that no solution exists. A simple calculation shows that the existence of an algorithm running in $c^k \cdot n^{\mathcal{O}(1)}$ time for the disjoint variant implies that Sim-FVS can be solved in time $(1+c)^k \cdot n^{\mathcal{O}(1)}$ [CFK+15]. In the next section we describe such an algorithm for Disjoint Sim-FVS.

10.1.1 Algorithm for Disjoint Sim-FVS

Let $(G = (V, E_1, \ldots, E_\alpha), W, k)$ be an instance of Disjoint Sim-FVS, and $F = G - W$. We start with some simple reduction rules that clean up the graph. Whenever some reduction rule applies, we apply the lowest-numbered applicable rule.

Reduction Rule 10.1. *Delete isolated vertices.*

Reduction Rule 10.2. *If there is a vertex v which has only one neighbor u in G_i, for some $i \in [\alpha]$, then delete the edge (v, u) from E_i.*

Algorithm 2: DISJOINT SIM-FVS

Input: $G = (V, E_1, E_2, \ldots, E_\alpha)$, W, k, and $\mathfrak{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_\alpha\}$
Output: *yes* or *no*.

Apply SIM-FVS R.1 to SIM-FVS R.5 exhaustively;
if $k < 0$ *or for any* $i \in [\alpha]$, $|\mathcal{C}_i| > k$ **then**
  return *no*
while *for some* $i \in [\alpha]$, $G_i[V(F_i) \cup V(W_i)]$ *is not a forest* **do**
  find a cordate vertex v_c of highest index in some tree of F_i;
  Let u_c, w_c be the vertices in tree $T^i_{v_c}$ with a neighbor u, w respectively in W_i;
  Let $P = (u_c, x_1, \ldots, x_t, v_c)$ and $P' = (v_c, y_1, \ldots, y_{t'}, w_c)$ be the paths in F_i from u_c to v_c and v_c to w_c, respectively;
  $\mathcal{G}_1 = (G - \{v_c\}, W, k-1, \mathfrak{C})$, Add $\mathcal{G}_1$ to $\mathfrak{G}$;
  if $V' = V(P) \setminus \{v_c\} \neq \emptyset$ **then**
    $\mathcal{C}_i = \mathcal{C}_i \cup \{(u_c, x_1, \ldots, x_t)\}$;
    $\mathcal{G}_2 = (G - V', W, k-1, \mathfrak{C})$, Add $\mathcal{G}_2$ to $\mathfrak{G}$;
  if $V' = V(P') \setminus \{v_c\} \neq \emptyset$ **then**
    $\mathcal{C}_i = \mathcal{C}_i \cup \{(y_1, \ldots, y_{t'}, w_c)\}$;
    $\mathcal{G}_3 = (G - V', W, k-1, \mathfrak{C})$, Add $\mathcal{G}_3$ to $\mathfrak{G}$;
  if u, w *are in the same component of* W_i **then**
    return $\bigvee_{\mathcal{G} \in \mathfrak{G}}$DISJOINT SIM-FVS($\mathcal{G}$)
  else
    return ($\bigvee_{\mathcal{G} \in \mathfrak{G}}$DISJOINT SIM-FVS($\mathcal{G}$)) $\vee$ DISJOINT SIM-FVS($G - (V(P) \cup V(P')), W \cup V(P) \cup V(P'), k, \mathfrak{C}$)
  end
end
`// Solve the remaining instance using the hitting set problem`
For $i \in [\alpha]$ let $V(\mathcal{C}_i) = \cup_{C \in \mathcal{C}_i} V(C)$, $\mathcal{U} = \cup_{i \in [\alpha]} V(\mathcal{C}_i)$;
$\mathcal{F} = \cup_{i \in [\alpha]} \mathcal{C}_i$;
Find a hitting set $U =$ HITTING SET($\mathcal{F}$,$\mathcal{U}$);
if $|U| \leq k$ **then**
  return *yes*
return *no*

Reduction Rule 10.3. *If there is a vertex $v \in V(G)$ with exactly two neighbors u, w (the total degree of v is two), delete edges (v, u) and (v, w) from E_i and add an edge (u, w) to E_i, where i is the color of edges (v, u) and (v, w). Note that after Reduction Rule 10.2 has been applied, both edges (v, u) and (v, w) must be of the same color.*

Reduction Rule 10.4. *If for some $i \in [\alpha]$ there is an edge of multiplicity larger than two in E_i, reduce its multiplicity to two.*

Reduction Rule 10.5. *If there is a vertex v with a self-loop, then add v to the solution set, delete v (and all edges incident on v) from the graph and decrease k by 1.*

The safeness of reduction rule 10.1 is immediate (as isolated vertices do not participate in any cycle). The safeness of Reduction Rule 10.4 follows from the fact that edges of multiplicity greater than two do not influence the set of feasible solutions. Safeness of Reduction Rule 10.5 follows from the fact that any vertex with a self-loop must be

present in every solution. Note that all of the above reduction rules can be applied in polynomial time. Moreover, after exhaustively applying all rules, the resulting graph G satisfies the following properties:

(P1) G contains no self-loops;
(P2) Every edge in G_i, for $i \in [\alpha]$, is of multiplicity at most two;
(P3) Every vertex in G has either degree zero or degree at least two in each G_i, for $i \in [\alpha]$;
(P4) The total degree of every vertex in G is at least three.

Lemma 10.1. *Reduction rule 10.2 is safe.*

Proof. Let G be an α-colored graph and v be a vertex whose only neighbor in G_i is u, for some $i \in [\alpha]$. Consider the α-colored graph G' with vertex set $V(G)$ and edge sets $E_i(G') = E_i(G) \setminus \{(v,u)\}$ and $E_j(G') = E_j(G)$, for $j \in [\alpha] \setminus \{i\}$. We show that G has an α-simfvs of size at most k if and only if G' has an α-simfvs of size at most k.

In the forward direction, consider an α-simfvs S in G of size at most k. Since $G'_j = G_j$, S intersects all the cycles in G'_j, $j \in [\alpha] \setminus \{i\}$. Note that in G_i, there is no cycle containing the edge (u,v) as v is a degree-one vertex in G_i. Hence, all the cycles in G_i are also cycles in G'_i. S intersects all cycles in G_i and, in particular, S intersects all cycles in G'_i. Therefore, S is an α-simfvs in G' of size at most k.

For the reverse direction, consider an α-simfvs S in G' of size at most k. If S is not an α-simfvs of G then there is a cycle C in some G_t, for $t \in [\alpha]$. Note that C cannot be a cycle in G_j as $G_j = G'_j$, for $j \in [\alpha] \setminus \{i\}$. Therefore C must be a cycle in G_i. The cycle C must contain the edge (v,u), as this is the only edge in G_i which is not an edge in G'_i. But v is a degree-one vertex in G_i, so it cannot be part of any cycle in G_i, contradicting the existence of cycle C. Thus S is an α-simfvs of G of size at most k. □

Lemma 10.2. *Reduction rule 10.3 is safe.*

Proof. Consider an α-colored graph G. Let v be a vertex in $V(G)$ such that v has total degree 2 and let u, w be the neighbors of v in G_i, where $u \neq w$ and $i \in [\alpha]$. Consider the α-colored graph G' with vertex set $V(G)$ and edge sets $E_i(G') = (E_i(G) \setminus \{(v,u),(v,w)\}) \cup \{(u,w)\}$ and $E_j(G') = E_j(G)$, for $j \in [\alpha] \setminus \{i\}$. We show that G has an α-simfvs of size at most k if and only if G' has an α-simfvs of size at most k.

In the forward direction, let S be an α-simfvs in G of size at most k. Suppose S is not an α-simfvs of G'. Then, there is a cycle C in G'_t, for some $t \in [\alpha]$. Note that C cannot be a cycle in G'_j as $G'_j = G_j$, for $j \in [\alpha] \setminus \{i\}$. Therefore C must be a cycle in G'_i. All the cycles C' not containing the edge (u,w) are also cycles in G_i and therefore S must contain some vertex from C'. It follows that C must contain the edge (u,w). Note that the edges $(E(C) \setminus \{(u,w)\}) \cup \{(v,u),(w,v)\}$ form a cycle in G_i. Therefore S must contain a vertex from $V(C) \cup \{v\}$. We consider the following cases:

- Case 1: $v \notin S$. In this case, S must contains a vertex from $V(C)$. Hence, S is an α-simfvs in G'.

- Case 2: $v \in S$. Let $S' = (S \setminus \{v\}) \cup \{u\}$. Any cycle C' containing v in G_i must contain u and w (since $d_{G_i}(v) = 2$). But S' intersects all such cycles C', as $u \in S'$. Therefore S' is an α-simfvs of G' of size at most k.

In the reverse direction, consider an α-simfvs S of G'. S intersects all cycles in G_j, since $G_j = G'_j$, for $j \in [\alpha] \setminus \{i\}$. All cycles in G_i not containing v are also cycles in G'_i and therefore S intersects all such cycles. A cycle C in G_i containing v must contain u and w (v is a degree-two vertex in G_i). Note that $(E(C) \setminus \{(v,u),(v,w)\}) \cup \{(u,w)\}$ is a cycle in G'_i and S, being an α-simfvs in G', must contain a vertex from $V(C) \setminus \{v\}$. Therefore $S \cap V(C) \neq \emptyset$, so S intersects cycle C in G'_i. Hence S an α-simfvs in G'. □

Algorithm. We give an algorithm for the decision version of the Disjoint Sim-FVS problem, which only verifies whether a solution exists or not. Such an algorithm can be easily modified to find an actual solution. Let (G, W, k) be an instance of the problem, where G is an α-colored graph. If $G[W]$ is not an α-forest then we can safely return that (G, W, k) is a no-instance. This follows from the fact that we are looking for an α-simfvs in G which is disjoint from W. Recall that, in the "compression" step of the iterative compression routine, we always "guess" the intersection of $S^{(j+1)}$ with $Z^{(j+1)}$. Hence, we assume that $G[W]$ is an α-forest in what follows. Whenever any of our Reduction Rules 10.1 to 10.5 apply, the algorithm exhaustively does so (in order). If at any point in our algorithm the parameter k drops below zero, then the resulting instance is again a *no*-instance.

Recall that initially $F = G - W$ is an α-forest, as W is an α-simfvs. We will consider each forest F_i, for $i \in [\alpha]$, separately (where F_i is the color i graph of the α-forest F). For $i \in [\alpha]$, we let $W_i = (W, E_i(G[W]))$ and η_i be the number of components in W_i. Some of the branching rules that we apply create special vertex-disjoint cycles. We will maintain this set of special cycles in $\mathcal{C}_i$, for each i, and we let $\mathfrak{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_\alpha\}$. Initially, $\mathcal{C}_i = \emptyset$. Each cycle that we add to $\mathcal{C}_i$ will be vertex disjoint from cycles previously added to $\mathcal{C}_i$. Hence, if at any point $|\mathcal{C}_i| > k$, for any $i \in [\alpha]$, then we can stop exploring the corresponding branch. Moreover, whenever we "guess" that some vertex v must belong to a solution, we also traverse the family $\mathfrak{C}$ and remove any cycles containing v. For the running time analysis of our algorithm we will consider the following measure:

$$\mu = \mu(G, W, k, \mathfrak{C}) = \alpha k + (\sum_{i \in [\alpha]} \eta_i) - (\sum_{i \in [\alpha]} |\mathcal{C}_i|)$$

The input to our algorithm consists of a tuple $(G, W, k, \mathfrak{C})$. For clarity, we will denote a reduced input by $(G, W, k, \mathfrak{C})$ (the one where reduction rules do not apply).

We root each tree in F_i at some arbitrary vertex. Assign an index t to each vertex v in the forest F_i, which is the distance of v from the root of the tree it belongs to (the root is assigned index zero). A vertex v in F_i is called *cordate* if one of the following holds:

- v is a leaf (or degree-zero vertex) in F_i with at least two color i neighbors in W_i.
- The subtree T^i_v rooted at v contains two vertices u and w which have at least one color i neighbor in W_i (v can be equal to u or w).

Lemma 10.3. *For $i \in [\alpha]$, let v_c be a cordate vertex of highest index in some tree of the forest F_i and let $T^i_{v_c}$ denote the subtree rooted at v_c. Furthermore, let u_c be one of the vertices in $T^i_{v_c}$ such that u_c has a neighbor in W_i. Then, in the path $P = (u_c, x_1, \ldots, x_t, v_c)$ (t could be equal to zero) between u_c and v_c the vertices $x_1, \ldots, x_t$ are degree-two vertices in G_i.*

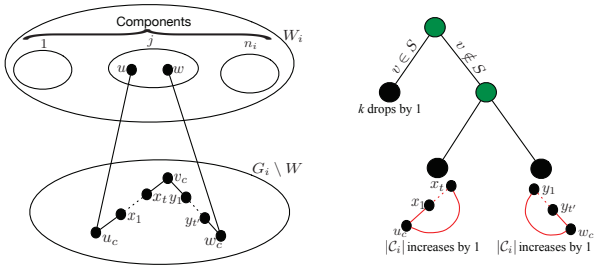

Figure 10.1: Branching in Case 1.a

Proof. If $u_c = v_c$ or $t = 0$ then there is nothing to prove. Otherwise, let $P = (u_c, x_1, \ldots, x_t, v_c)$ be the path from v_c to u_c, where $t \geq 1$. In P, if there is a vertex x (other than u_c and v_c) which has an edge of color i to a vertex in W_i, then x is a cordate vertex of higher index, contradicting the choice of v_c. Also, if there is a vertex x in P other than v_c and u_c of degree at least three in F_i, the subtree rooted at x has at least two leaves, and all the leaves have a color-i neighbor in W_i. Therefore, x is a cordate vertex and has a higher index than v_c, contradicting the choice of v_c. It follows that $x_1, \ldots, x_t$ (if they exist) are degree-two vertices in G_i. □

We consider the following cases depending on whether there is a cordate vertex in F_i or not.

- Case 1: There is a cordate vertex in F_i. Let v_c be a cordate vertex with the highest index in some tree in F_i and let the two vertices with neighbors in W_i be u_c and w_c (v_c can be equal to u_c or w_c). Let $P = (u_c, x_1, x_2, \ldots, x_t, v_c)$ and $P' = (v_c, y_1, y_2, \ldots, y_{t'}, w_c)$ be the unique paths in F_i from u_c to v_c and from v_c to w_c, respectively. Let $P_v = (u_c, x_1, \ldots, x_t, v_c, y_1, \ldots, y_{t'}, w_c)$ be the unique path in F_i from u_c to w_c. Consider the following sub-cases:

  - Case 1.a: u_c and w_c have neighbors in the same component of W_i. In this case one of the vertices from path P_v must be in the solution (Figure 10.1). We branch as follows:

    * v_c belongs to the solution. We delete v_c from G and decrease k by 1. In this branch, μ decreases by α. When v_c does not belong to the solution, then at least one vertex from $u_c, x_1, x_2, \ldots, x_t$ or $y_1, y_2, \ldots, y_{t'}, w_c$ must be in the solution. But note that these are vertices of degree at most two in G_i by Lemma 10.3. So with respect to color i, it does not matter which vertex is chosen in the solution. The only issue comes from some color j cycle, where $j \neq i$, in which choosing a particular vertex from $u_c, x_1, \ldots, x_t$ or $y_1, y_2, \ldots, y_{t'}, w_c$ would be more beneficial. We consider the following two cases.
    * One of the vertices from $u_c, x_1, x_2, \ldots, x_t$ is in the solution. In this case, we add an edge (u_c, x_t) (or (u_c, u_c) when u_c and v_c are adjacent) to G_i and delete the edge (x_t, v_c) from G_i. This creates a cycle C in $G_i - W$, which is itself a component in $G_i - W$. We remove the edges in C from G_i and add the cycle C to $\mathcal{C}_i$. We will be handling these sets of cycles

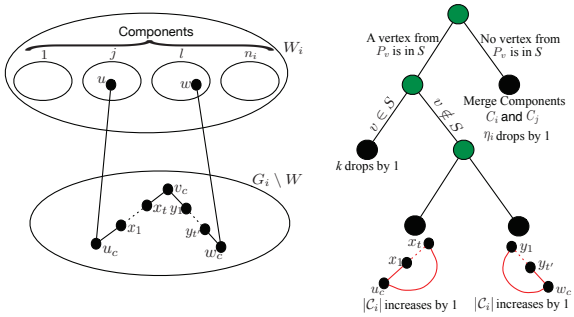

Figure 10.2: Branching: Case 1.b

independently. In this case $|\mathcal{C}_i|$ increases by 1, so the measure μ decreases by 1.

* One of the vertices from $y_1, y_2, \ldots, y_t, w_c$ is in the solution. In this case we add an edge (y_1, w_c) to G_i and delete the edge (v_c, y_1) from G_i. This creates a cycle C in $G_i - W$ as a component. We add C to $\mathcal{C}_i$ and delete edges in C from $G_i - W$. In this branch $|\mathcal{C}_i|$ increases by 1, so the measure μ decreases by 1. The resulting branching vector is $(\alpha, 1, 1)$.

– Case 1.b: u_c and w_c do not have neighbors in the same component. We branch as follows (Figure 10.2):

* v_c belongs to the solution. We delete v_c from G and decrease k by 1. In this branch μ decreases by α.
* One of the vertices from $u_c, x_1, x_2, \ldots, x_t$ is in the solution. In this case, we add an edge (u_c, x_t) to G_i and delete the edge (x_t, v_c) from G_i. This creates a cycle C in $G_i - W$ as a component. As in Case 1, we add C to $\mathcal{C}_i$ and delete edges in C from $G_i - W$. $|\mathcal{C}_i|$ increases by 1, so the measure μ decreases by 1.
* One of the vertices from $y_1, y_2, \ldots, y_t, w_c$ is in the solution. In this case, we add an edge (y_1, w_c) to G_i and delete the edge (v_c, y_1) from G_i. This creates a cycle C in $G_i - W$ as a component. We add C to $\mathcal{C}_i$ and delete edges in C from $G_i - W$. In this branch $|\mathcal{C}_i|$ increases by 1, so the measure μ decreases by 1.
* No vertex from path P_v is in the solution. In this case, we add the vertices in P_v to W, the resulting instance is $(G - P_v, W \cup P_v, k)$. The number of components in W_i decreases and we get a drop of 1 in η_i, so μ decreases by 1. Note that if $G[W \cup P_v]$ is not acyclic we can safely ignore this branch. The resulting branching vector is $(\alpha, 1, 1, 1)$.

- Case 2: There is no cordate vertex in F_i. Let $\mathcal{F}$ be a family of sets containing a set $f_C = V(C)$ for each $C \in \cup_{i=1}^{\alpha} \mathcal{C}_i$ and let $\mathcal{U} = \cup_{i=1}^{\alpha} (\cup_{C \in \mathcal{C}_i} V(C))$. Note that $|\mathcal{F}| \leq \alpha k$. We find a subset $U \subseteq \mathcal{U}$ (if it exists) which hits all the sets in $\mathcal{F}$, such that $|U| \leq k$.

Note that in Case 1, if the cordate vertex v_c is a leaf, then $u_c = w_c = v_c$. Therefore, from Case 1.a we are left with one branching rule. Similarly, we are left with the first

and the last branching rules for Case 1.b. If v_c is not a leaf but v_c is equal to u_c or w_c, say $v_c = w_c$, then for both Case 1.a and Case 1.b we do not have to consider the third branch. Finally, when none of the reduction or branching rules apply, we solve the problem by invoking an algorithm for the Hitting Set problem as a subroutine.

Lemma 10.4. *The presented algorithm for* Disjoint Sim-FVS *is correct.*

Proof. Consider an input $(G, W, k, \mathfrak{C})$ to the algorithm for Disjoint Sim-FVS, where G is an α-colored graph, W is an α-simfvs of size $k+1$, k is an integer, and $\mathfrak{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_\alpha\}$. Let $\mu = \mu(G, W, k, \mathfrak{C})$ be the measure as defined earlier. We prove the correctness of the algorithm by induction on the measure μ. The base case occurs when one of the following holds:

- $k < 0$,
- for some $i \in [\alpha]$, $|\mathcal{C}_i| > k$, or
- $\mu \leq 0$.

If $k < 0$, then we can safely conclude that G is a no-instance. If for some $i \in [\alpha]$ we have $|\mathcal{C}_i| > k$, then we need to pick at least one vertex from each of the vertex-disjoint cycles in $\mathcal{C}_i$ and there are at least $k+1$ of them. Our algorithm correctly concludes that the graph is also a *no*-instance in such cases. If $\mu = \alpha k + (\sum_{i=1}^{\alpha} \eta_i) - (\sum_{i=1}^{\alpha} |\mathcal{C}_i|) \leq 0$ then $\alpha k \leq \sum_{i=1}^{\alpha} |\mathcal{C}_i|$. But for each $i \in [\alpha]$, we have $|\mathcal{C}_i| \leq k$. Therefore $\alpha k \leq \sum_{i=1}^{\alpha} |\mathcal{C}_i| \leq \alpha k$, $\sum_{i=1}^{\alpha} |\mathcal{C}_i| = \alpha k$, and $|\mathcal{C}_i| = k$, for all $i \in [\alpha]$. This implies that for each $i \in [\alpha]$, $G_i[V(F_i) \cup V(W_i)]$ must be acyclic. Assume otherwise. Then, for some $i \in [\alpha]$, $G_i[V(F_i) \cup V(W_i)]$ contains a cycle which is vertex disjoint from the k cycles in $\mathcal{C}_i$. Therefore, at least $k+1$ vertices are needed to intersect these cycles and we again have a *no*-instance. Recall that when a new vertex v is added to the solution set we delete all those cycles in $\cup_{i=1}^{\alpha} \mathcal{C}_i$ which contain v.

We are now left with cycles in $\cup_{i=1}^{\alpha} \mathcal{C}_i$. Intersecting a cycle $C \in \cup_{i=1}^{\alpha} \mathcal{C}_i$ is equivalent to hitting the set $V(C)$. Hence, we construct a family $\mathcal{F}$ consisting of a set $f_C = V(C)$ for each $C \in \cup_{i=1}^{\alpha} \mathcal{C}_i$ and we let $\mathcal{U} = \cup_{i=1}^{\alpha} (\cup_{C \in \mathcal{C}_i} V(C))$. Note that $|\mathcal{F}| \leq \alpha k$. If we can find a subset $U \subseteq \mathcal{U}$ which hits all the sets in $\mathcal{F}$, such that $|U| \leq k$, then U is the required solution. Otherwise, we have a *no*-instance. It is known that the Hitting Set problem parameterized by the size of the family $\mathcal{F}$ is fixed-parameter tractable and can be solved in $\mathcal{O}^\star(2^{|\mathcal{F}|})$ time [CFK+15]. In particular, we can find an optimum hitting set $U \subseteq \mathcal{U}$, hitting all the sets in $\mathcal{F}$. Therefore, we have a subset of vertices that intersects all the cycles in $\mathcal{C}_i$, for $i \in [\alpha]$.

Putting it all together, at a base case, our algorithm correctly decides whether or not $(G, W, k, \mathfrak{C})$ is a *yes*-instance. For the induction hypothesis, assume that the algorithm correctly decides an instance for $\mu \leq t$. Now consider the case $\mu = t+1$. If some reduction rule applies then we create an equivalent instance (since all reduction rules are safe). Therefore, either we get an equivalent instance with the same measure or we get an equivalent instance with $\mu \leq t$ (the case when Reduction Rule 10.5 is applied). In the latter case, by the induction hypothesis, our algorithm correctly decides the instance where $\mu \leq t$. In the former case, we apply one of the branching rules. Each branching rule is exhaustive and covers all possible cases. In addition, the measure decreases at each branch by at least one. Therefore, by the induction hypothesis, the algorithm correctly decides whether or not the input is a *yes*-instance. □

Lemma 10.5. DISJOINT SIM-FVS *is solvable in time* $\mathcal{O}^\star(22^{\alpha k})$.

Proof. Reduction Rules 10.1 to 10.5 can be applied in time polynomial in the input size. Also, at each branch we spend a polynomial amount of time. For each of the recursive calls at a branch, the measure μ decreases at least by 1. When $\mu \leq 0$, then we are able to solve the remaining instance in time $\mathcal{O}(2^{\alpha k})$ or correctly conclude that the corresponding branch cannot lead to a solution. At the start of the algorithm $\mu \leq 2\alpha k$. Therefore, the height of the search tree is bounded by $2\alpha k$. The worst-case branching vector for the algorithm is $(\alpha, 1, 1, 1)$. The recurrence relation for the worst case branching vector is: $T(\mu) \leq T(\mu - \alpha) + 3T(\mu - 1) \leq T(\mu - 2) + 3T(\mu - 1)$, since $\alpha \geq 2$. The running time corresponding to the above recurrence relation is $3.303^{2\alpha k}$. At each branch we spend a polynomial amount of time but we might require $\mathcal{O}(2^{\alpha k})$ time for solving the base case. Therefore, the running time of the algorithm is $\mathcal{O}^\star(2^{\alpha k} \cdot 3.303^{2\alpha k}) = \mathcal{O}^\star(22^{\alpha k})$. □

Theorem 10.1. SIMULTANEOUS FEEDBACK VERTEX SET *is solvable in time* $\mathcal{O}^\star(23^{\alpha k})$.

10.1.2 Faster algorithm for SIM-FVS with $\alpha = 2$

We improve the running time of the FPT algorithm for SIM-FVS when $\alpha = 2$. Given two sets of disjoint cycles $\mathcal{C}_1$ and $\mathcal{C}_2$ and a set $V = \cup_{C \in \mathcal{C}_1 \cup \mathcal{C}_2} V(C)$, we want to find a subset $H \subseteq V$ such that H contains at least one vertex from $V(C)$, for each $C \in \mathcal{C}_1 \cup \mathcal{C}_2$. We construct a bipartite graph G_M as follows. We set $V(G_M) = \{c_x^1 \mid C_x \in \mathcal{C}_1\} \cup \{c_y^2 \mid C_y \in \mathcal{C}_2\}$. In other words, we create one vertex for each cycle in $\mathcal{C}_1 \cup \mathcal{C}_2$. We add an edge between c_x^1 and c_y^2 if and only if $V(C_x) \cap V(C_y) \neq \emptyset$. Note that for $i \in [2]$ and $C, C' \in \mathcal{C}_i$, $V(C) \cap V(C') = \emptyset$. In Lemma 10.6, we show that finding a matching M in G_M, such that $|M| + |V(G_M) \setminus V(M)| \leq k$, corresponds to finding a set H of size at most k, such that H contains at least one vertex from each cycle $C \in \mathcal{C}_1 \cup \mathcal{C}_2$.

Lemma 10.6. *For $i \in [2]$, let $\mathcal{C}_i$ be a set of vertex-disjoint cycles, i.e. for each $C, C' \in \mathcal{C}_i$, $C \neq C'$ implies $V(C) \cap V(C') = \emptyset$. Let $\mathcal{F} = \{V(C) \mid C \in \mathcal{C}_1 \cup \mathcal{C}_2\}$ and $\mathcal{U} = \cup_{C \in \mathcal{C}_1 \cup \mathcal{C}_2} V(C)$. There exists a vertex subset $H \subseteq \cup_{C \in \mathcal{C}_1 \cup \mathcal{C}_2} V(C)$ of size k such that $H \cap V(C) \neq \emptyset$, for each $C \in \mathcal{C}_1 \cup \mathcal{C}_2$, if and only if G_M has a matching M, such that $|M| + |V(G_M) \setminus V(M)| \leq k$.*

Proof. For the forward direction, consider a minimal vertex subset $H \subseteq V(C_1) \cup V(C_2)$ of size at most k such that for each $C \in \mathcal{C}_1 \cup \mathcal{C}_2$, $H \cap V(C) \neq \emptyset$. Note that a vertex $h \in H$ can be present in at most one cycle from $\mathcal{C}_i$, for $i \in [2]$, since $\mathcal{C}_i$ is a set of vertex-disjoint cycles. Therefore, h can be present in at most 2 cycles from $\mathcal{C}_1 \cup \mathcal{C}_2$. If h is present in 2 cycles, say $C_x \in \mathcal{C}_1$ and $C_y \in \mathcal{C}_2$, then in G_M we must have an edge between c_x^1 and c_y^2 (since h belongs to both C_x and C_y). We include the edge (c_x^1, c_y^2) in the matching M. If h belongs to only one cycle, say $C_z^i \in \mathcal{C}_1 \cup \mathcal{C}_2$, then we include vertex c_z^i in a set I. Note that $(V(G_M) \setminus V(M)) \subseteq I$. For each $h \in H$, we either add a matching edge or add a vertex to I. Therefore $|M| + |V(G_M) \setminus V(M)| \leq |M| + |I| \leq k$.

In the reverse direction, consider a matching M such that $|M| + |V(G_M) \setminus V(M)| \leq k$. We construct a set H of size at most k containing a vertex from each cycle in $\mathcal{C}_1 \cup \mathcal{C}_2$. For each edge (c_x^1, c_y^2) in the matching, where $C_x \in \mathcal{C}_1$ and $C_y \in \mathcal{C}_2$, there is a vertex h that belongs to both $V(C_x)$ and $V(C_y)$. Include h in H. For each $c_z^i \in V(G_M) \setminus V(M)$, add an arbitrary vertex $v \in V(C_z)$ to H. Note that $|H| \leq k$, since for each matching

edge and each unmatched vertex we added one vertex to H. Moreover, for each cycle $C \in \mathcal{C}_1 \cup \mathcal{C}_2$, its corresponding vertex in G_M is either part of the matching or is an unmatched vertex; in both cases there is a vertex in H that belongs to C. Therefore, H is a subset of size at most k which contains at least one vertex from each cycle in $\mathcal{C}_1 \cup \mathcal{C}_2$. □

Note that a matching M in G_M minimizing $|M|+|V(G_M)\setminus V(M)|$ is one of maximum size. Therefore, at the base case, we compute a maximum matching of the corresponding graph G_M, which is a polynomial-time solvable problem, and return an optimal solution for intersecting all cycles in $\mathcal{C}_1 \cup \mathcal{C}_2$. Moreover, if we set $\mu = 2k+(\eta_1/2+\eta_2/2)-(|\mathcal{C}_1|+|\mathcal{C}_2|)$, then the worst case branching vector is $(2, 1, 1, 1/2)$. Corresponding to this worst case branching vector, the running time of the algorithm is $\mathcal{O}^\star(81^k)$.

Theorem 10.2. Simultaneous Feedback Vertex Set *is solvable in time* $\mathcal{O}^\star(81^k)$ *when* $\alpha = 2$.

10.2 Kernel for Simultaneous Feedback Vertex Set

In this section, we give a kernel with $\mathcal{O}(\alpha k^{3(\alpha+1)})$ vertices for Sim-FVS. Let (G, k) be an instance of Sim-FVS, where G is an α-colored graph and k is a positive integer. We assume that Reduction Rules 10.1 to 10.5 have been exhaustively applied. The kernelization algorithm then proceeds in two stages. In stage one, we bound the maximum degree of G. In the second stage, we present new reduction rules to deal with degree-two vertices and, conclude a bound on the total number of vertices.

To bound the total degree of each vertex $v \in V(G)$, we bound the degree of v in G_i, for $i \in [\alpha]$. To do so, we need the Expansion Lemma [CFK+15] as well as the 2-approximation algorithm for the classical Feedback Vertex Set problem [BBF99].

10.2.1 Bounding the degree of vertices in G_i

We now describe reduction rules that allow us to bound the maximum degree of a vertex $v \in V(G)$.

Lemma 10.7 (Lemma 6.8 [MRRS12])**.** *Let G be an undirected (multi) graph and x be a vertex of G without a loop. Then in polynomial time we can either decide that (G, k) is a no-instance of* Feedback Vertex Set *or check whether there is an x-flower of order $k+1$, or find a set of vertices $Z \subseteq V(G) \setminus \{x\}$ of size at most $3k$ intersecting every cycle in G, i.e. Z is a feedback vertex set of G.*

The next proposition easily follows from Lemma 10.7.

Proposition 10.1. *Let G be an undirected α-colored multigraph and x be a vertex without a loop in G_i, for some $i \in [\alpha]$. Then in polynomial time we can either decide that (G, k) is a no-instance of* Sim-FVS *or check whether there is an x-flower of order $k+1$ in G_i, or find a set of vertices $Z \subseteq V(G) \setminus \{x\}$ of size at most $3k$ intersecting every cycle in G_i.*

After applying Reduction Rules 10.1 to 10.5 exhaustively, we know that the degree of a vertex in each G_i is either 0 or at least 2 and no vertex has a self-loop. Now consider

a vertex v whose degree in G_i is more than $3k(k+4)$. By Proposition 10.1, we know that one of three cases must apply:

(1) (G, k) is a *no*-instance of Sim-FVS,

(2) we can find (in polynomial time) a v-flower of order $k+1$ in G_i, or

(3) we can find (in polynomial time) a set $H_v \subseteq V(G_i)$ of size at most $3k$ such that $v \notin H_v$ and $G_i - H_v$ is a forest.

The following reduction rule allows us to deal with case (2). The safeness of the rule follows from the fact that if v in not included in the solution then we need to have at least $k+1$ vertices in the solution.

Reduction Rule 10.6. *For $i \in [\alpha]$, if G_i has a vertex v such that there is a v-flower of order at least $k+1$ in G_i, then delete v from G and decrease k by 1. The resulting instance is $(G - \{v\}, k-1)$.*

When in case (3), we bound the degree of v as follows. Consider the graph $G_i' = G_i - (H_v \cup \{v\} \cup V_0^i)$, where V_0^i is the set of degree-zero vertices in G_i. Let $\mathcal{D}$ be the set of components in the graph G_i' which have a vertex adjacent to v. Note that each $D \in \mathcal{D}$ is a tree and v cannot have two neighbors in D, since H_v is a feedback vertex set in G_i. We will now argue that each component $D \in \mathcal{D}$ has a vertex u such that u is adjacent to a vertex in H_v. Suppose for a contradiction that there is a component $D \in \mathcal{D}$ such that D has no vertex which is adjacent to a vertex in H_v. $D \cup \{v\}$ is a tree with at least 2 vertices, so D has a vertex w, such that w is a degree-one vertex in G_i, contradicting the fact that each vertex in G_i is either of degree zero or of degree at least two.

After exhaustive application of Reduction Rule 10.4, every pair of vertices in G_i can have at most two edges between them. In particular, there can be at most two edges between $h \in H_v$ and v. If the degree of v in G_i is more than $3k(k+4)$, then the number of components $|\mathcal{D}|$ in G_i' is more than $3k(k+2)$, since $|H_v| \leq 3k$.

Consider the bipartite graph $\mathcal{B}$, with bipartition (H_v, Q), where Q has a vertex q_D corresponding to each component $D \in \mathcal{D}$. We add an edge between $h \in H_v$ and $q_D \in Q$ to $E(\mathcal{B})$ if and only if D has a vertex d which is adjacent to h in G_i.

Reduction Rule 10.7. *Let v be a vertex of degree at least $3k(k+4)$ in G_i, for some $i \in [\alpha]$, and let H_v be a feedback vertex set in G_i not containing v and of size at most $3k$.*

- *Let $Q' \subseteq Q$ and $H \subseteq H_v$ be the sets of vertices obtained after applying Lemma 2.1 with $q = k+2$, $A = H_v$, and $B = Q$, such that H has a $(k+2)$-expansion into Q' in $\mathcal{B}$;*
- *Delete all the edges (d, v) in G_i, where $d \in V(D)$ and $q_D \in Q'$;*
- *Add double edges between v and h in G_i, for all $h \in H$ (unless such edges already exist).*

By Lemma 2.1 and Proposition 10.1, Reduction Rule 10.7 can be applied in time polynomial in the input size.

Lemma 10.8. *Reduction Rule 10.7 is safe.*

Proof. Let G be an α-colored graph where Reduction Rules 10.1 to 10.6 do not apply. Let v be a vertex of degree more than $3k(k+4)$ in G_i, for $i \in [\alpha]$. Let $H \subseteq H_v$, $Q' \subseteq Q$ be the sets defined above and let G' be the instance obtained after an application of Reduction Rule 10.7. We show that G has an α-simfvs of size at most k if and only if G' has an α-simfvs of size at most k. We need the following claim.

Claim 10.1. *Any k-sized α-simfvs S of G or G' either contains v or contains all the vertices in H.*

Proof. Since there exists a cycle (double edge) between v and every vertex $h \in H$ in G'_i, it easily follows that either v or all vertices in H must be in any solution for G'.

Consider the case of G. We assume $v \notin S$ and there is a vertex $h \in H$ such that $h \notin S$. Note that H has a $(k+2)$-expansion into Q' in $\mathcal{B}$, therefore h is the center of a $(k+2)$-star in $\mathcal{B}[H \cup Q']$. Let Q_h be the set of neighbors of h in $\mathcal{B}[H \cup Q']$ ($|Q_h| \geq k+2$). For each $q_D, q_{D'} \in Q_h$, their corresponding components $D, D' \in \mathcal{D}$ form a cycle with v and h. If both h and v are not in S, then we need to pick at least $k+1$ vertices to intersect the cycles formed by D, D', h, and v, for each $q_D, q_{D'} \in Q'$. Therefore, $H \subseteq S$, as needed. □

In the forward direction, consider an α-simfvs S of size at most k in G. For $j \in [\alpha] \setminus \{i\}$, $G'_j = G_j$ and therefore S intersects all the cycles in G'_j. By the previous claim, we can assume that either $v \in S$ or $H \subseteq S$. In both cases, S intersects all the new cycles created in G'_i by adding double edges between v and $h \in H$. Moreover, apart from the double edges between v and $h \in H$, all the cycles in G'_i are also cycles in G_i, therefore S intersects all those cycles in G'_i. It follows that S is an α-simfvs in G'.

In the reverse direction, consider an α-simfvs S in G' of size at most k. Note that for $j \in [\alpha] \setminus \{i\}$, $G'_j = G_j$. Therefore S intersects all the cycles in G_j. By the previous claim, at least one of the following must hold: (1) $v \in S$ or (2) $H \subseteq S$. Suppose that (1) $v \in S$ holds. Since $G'_i - \{v\} = G_i - \{v\}$, $S \setminus \{v\}$ intersects all the cycles in $G'_i - \{v\}$ and $G_i - \{v\}$. Therefore S intersects all the cycles in G_i and S is an α-simfvs in G. In case (2), i.e. when $v \notin S$ but $H \subseteq S$, any cycle in G which does not intersect with S is also a cycle in G' (since such a cycle does not intersect with H and the only deleted edges from G' belong to cycles passing through H). In other words, $S \setminus H$ intersects all cycles in both $G'_i - H$ and $G_i - H$ and, consequently, S is an α-simfvs in G. □

After exhaustively applying Reduction Rules 10.1 to 10.7, the degree of a vertex $v \in V(G_i)$ is at most $3k(k+4)-1$ in G_i, for each $i \in [\alpha]$.

10.2.2 Bounding the number of vertices in G

Having bounded the maximum total degree of a vertex in G, we now focus on bounding the number of vertices in the entire graph. To do so, we first compute an approximate solution for the Sim-FVS instance using the polynomial-time 2-approximation algorithm of Bafna et al. [BBF99] for the Feedback Vertex Set problem in undirected graphs. In particular, we compute a 2-approximate solution S_i in G_i, for $i \in [\alpha]$. We let $S = \cup_{i \in [\alpha]} S_i$. Note that S is an α-simfvs in G and has size at most $2\alpha|S_{OPT}|$, where $|S_{OPT}|$ is an optimal α-simfvs in G. For $i \in [\alpha]$, let $F_i = G_i - S_i$, and let $T^i_{\leq 1}$, T^i_2, and $T^i_{\geq 3}$, be

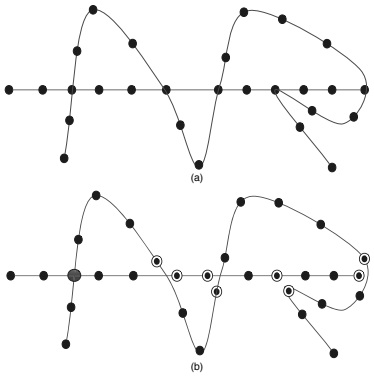

Figure 10.3: Unravelling two paths with five common vertices (a) to obtain two paths with one common vertex (b).

the sets of vertices in F_i having degree at most one in F_i, degree exactly two in F_i, and degree greater than two in F_i, respectively.

Later, we shall prove that bounding the maximum degree in G is sufficient for bounding the sizes of $T^i_{\leq 1}$ and $T^i_{\geq 3}$, for all $i \in [\alpha]$. We now focus on bounding the size of T^i_2 which, for each $i \in [\alpha]$, corresponds to a set of degree-two paths. In other words, for a fixed i, the graph induced in F_i by the vertices in T^i_2, i.e. $F_i[T^i_2]$, is a set of vertex-disjoint paths. We say a set of distinct vertices $\{v_1, \ldots, v_\ell\}$ in T^i_2 forms a *maximal degree-two path* if (v_j, v_{j+1}) is an edge in G_i, for all $j \in [\ell - 1]$, and all vertices $\{v_1, \ldots, v_\ell\}$ have degree exactly two in G_i.

We enumerate all the maximal degree-two paths in $G_i - S_i$, for $i \in [\alpha]$. Let this set of paths in $G_i - S_i$ be $\mathcal{P}_i = \{P^i_1, P^i_2, \ldots, P^i_{n_i}\}$, where n_i is the number of maximal degree-two paths in $G_i - S_i$. We introduce a special symbol ϕ and add ϕ to each set $\mathcal{P}_i$, for $i \in [\alpha]$. The special symbol will be used later to indicate that no path is chosen from the set $\mathcal{P}_i$.

Let $\mathfrak{S} = \mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_\alpha$ be the set of all tuples of maximal degree-two paths of different colors. For $\tau \in \mathfrak{S}$, $j \in [\alpha]$, $j(\tau)$ denotes the element from the set $\mathcal{P}_j$ in the tuple τ, i.e. for $\tau = (Q_1, \phi, \ldots, Q_j, \ldots, Q_\alpha)$, $j(\tau) = Q_j$ (for example $2(\tau) = \phi$).

For a maximal degree-two path $P^i_j \in \mathcal{P}_i$ and $\tau \in \mathfrak{S}$, we define $\mathsf{Intercept}(P^i_j, \tau) = \emptyset$ if $P^i_j \neq i(\tau)$. Otherwise, we define $\mathsf{Intercept}(P^i_j, \tau) = \{v \in V(P^i_j) \mid \text{for all } t \in [\alpha], \text{ if } t(\tau) \neq \phi \text{ then } v \in V(t(\tau))\}$. Stated differently, $\mathsf{Intercept}(P^i_j, \tau)$ is either the empty set or is the set of vertices which are present in all the paths in the tuple τ (of course a ϕ entry does not contribute to this set).

We define the notion of *unravelling* a path $P^i_j \in \mathcal{P}_i$ from all other paths of different colors in $\tau \in \mathfrak{S}$ at a vertex $u \in \mathsf{Intercept}(P^i_j, \tau)$ by creating a separate copy of u for each path. Formally, for a path $P^i_j \in \mathcal{P}_i$, $\tau \in \mathfrak{S}$, and a vertex $u \in \mathsf{Intercept}(P^i_j, \tau)$, the $\mathsf{Unravel}(P^i_j, \tau, u)$ operation does the following. For each $t \in [\alpha]$ with $t(\tau) \neq \phi$, let x_t and y_t be the unique neighbors of u on path $t(\tau)$ (and also in G_i). Create a vertex $u_{t(\tau)}$ for the path $t(\tau)$, delete the edges (x_t, u) and (u, y_t) from G_t and add the edges $(x_t, u_{t(\tau)})$ and $(u_{t(\tau)}, y_t)$ in G_t. Figure 10.3 illustrates the unravel operation for two paths of different colors.

Reduction Rule 10.8. *For a path $P_j^i \in \mathcal{P}_i$, $\tau \in \mathfrak{S}$, if $|\mathsf{Intercept}(P_j^i, \tau)| > 1$, then for a vertex $u \in \mathsf{Intercept}(P_j^i, \tau)$, $\mathsf{Unravel}(P_j^i, \tau, u)$.*

Lemma 10.9. *Reduction rule 10.8 is safe.*

Proof. Let G be an α-colored graph and S_i be a 2-approximate feedback vertex set in G_i, for $i \in [\alpha]$. Let $\mathcal{P}_i$ be the set of maximal degree-two paths in $G_i - S_i$ and $\mathfrak{S} = \mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_\alpha$. For a path $P_j^i \in \mathcal{P}_i$, $\tau \in \mathfrak{S}$, $|\mathsf{Intercept}(P_j^i, \tau)| > 1$, and $u \in \mathsf{Intercept}(P_j^i, \tau)$, let G' be the α-colored graph obtained after applying $\mathsf{Unravel}(P_j^i, \tau, u)$ in G. We show that G has an α-simfvs of size at most k, if and only if G' has an α-simfvs of size at most k.

In the forward direction, consider an α-simfvs S in G of size at most k. Let x be a vertex in $\mathsf{Intercept}(P_j^i, \tau) \setminus \{u\}$. We define $S' = S$ if $u \notin S$ and $S' = (S \setminus \{u\}) \cup \{x\}$ otherwise. A cycle C in the graph G'_t not containing $u_{t(\tau)}$, where $u_{t(\tau)}$ is the copy of u created for path $t(\tau)$, $\tau \in \mathfrak{S}$, and $t \in [\alpha]$, is also a cycle in G_t. Therefore S' intersects C. Let P_t be the path in $\mathcal{P}_t$ containing u, for $t \in [\alpha]$. Note that in $\mathcal{P}_i$, there is exactly one maximal degree-two path containing u and all the cycles in G_t containing u must contain P_t. All the cycles in G'_t containing $u_{t(\tau)}$ must contain x, since $u_{t(\tau)}$ is the private copy of u for the degree-two path $t(\tau)$ containing x. We consider the following cases depending on whether u belongs to S or not.

- $u \in S$: A cycle C in G'_t, $t \in [\alpha]$, containing $u_{t(\tau)}$ also contains x. Therefore S' intersects C.
- $u \notin S$: Corresponding to a cycle C in G'_t, $t \in [\alpha]$, containing $u_{t(\tau)}$, there is a cycle C' on vertices $(V(C) \cup \{u\}) \setminus \{u_{t(\tau)}\}$ in G_t. But S is an α-simfvs in G and therefore both S and S' must contain a vertex $y \in V(C') \setminus \{u\}$.

In the reverse direction, let S be an α-simfvs in G'. We define $S' = S$ if $\{u_{l(\tau)} | u_{l(\tau)} \in S, 1 \leq l \leq \alpha\} \cap S \neq \emptyset$ and $S' = (S \setminus \{u_{l(\tau)} | u_{l(\tau)} \in S, 1 \leq l \leq \alpha\}) \cup \{u\}$ otherwise. All the cycles in G_t not containing u are the cycles in G'_t not containing $u_{t(\tau)}$. Therefore S' intersects all those cycles. We consider the following cases depending on whether there is some $t' \in [\alpha]$ for which $u_{t'(\tau)}$ belongs to S or not.

- For all $t' \in [\alpha]$, $u_{t'(\tau)} \notin S$. Let C be a cycle in G_t containing u, for $t \in [\alpha]$. Note that G'_t has a cycle C' corresponding to C, with $V(C') = (V(C) \setminus \{u\}) \cup \{u_{t(\tau)}\}$. S intersects C', therefore both S and S' have a vertex $y \in V(C') \setminus \{u_{t(\tau)}\}$. Since $y \in V(C)$, S' intersects the cycle C in G_t.
- For some $t' \in [\alpha]$, $u_{t'(\tau)} \in S$. Note that S' intersects all the cycles in G_t containing u, for $t \in [\alpha]$. Moreover, the only purpose of $u_{t'(\tau)}$ being in S is to intersect a cycle C' in G'_t containing $u_{t'(\tau)}$. However, the corresponding cycle in G_t can be intersected by a single vertex, namely u. Therefore, S' is an α-simfvs in G.

This completes the proof. □

Theorem 10.3. Sim-FVS *admits a kernel on $\mathcal{O}(\alpha k^{3(\alpha+1)})$ vertices.*

Proof. Consider an α-colored graph G on which Reduction Rules 10.1 to 10.8 have been exhaustively applied. Note that all of our reduction rules are safe. Reduction Rules 10.1

to 10.5 can clearly be applied in time polynomial in $|V(G)|$ and k (for constant α). The fact that we can apply Reduction Rules 10.6 and 10.7 in polynomial time (for constant α) follows from Lemma 2.1 and Proposition 10.1. Moreover, observe that Reduction Rules 10.1 to 10.7 strictly decrease either the number of vertices or the number of edges in the graph, and therefore can only be applied a polynomial number of times. As for Reduction Rule 10.8, we shall show in what follows that the number of maximal degree-two paths is bounded by $k^{\mathcal{O}(\alpha)}$ (assuming Reduction Rules 10.1 to 10.7 have been exhaustively applied), and we can enumerate all of them in polynomial time (for constant α). Even though Reduction Rule 10.8 increases the number of vertices in the graph, such vertices will always have degree exactly two in the graph, and will subsequently be removed. Every application of Reduction Rule 10.8 decreases the number of vertices sharing the same set of maximal degree-two paths (and no reduction rule increases the number of such vertices). Hence, It remains to bound the number of vertices.

For $i \in [\alpha]$, the degree of a vertex $v \in V(G_i)$ is either 0 or at least 2 in G_i. In what follows, we do not count the vertices of degree 0 in G_i while counting the vertices in G_i; since the total degree of a vertex $v \in V(G)$ is at least three, there is some $j \in [\alpha]$ such that the degree of $v \in V(G_j)$ is at least 2.

Let S_i be a 2-approximate feedback vertex set in G_i, for $i \in [\alpha]$. Note that $S = \cup_{i=1}^{\alpha} S_i$ is a 2α-approximate α-simfvs in G. Let $F_i = G_i - S_i$, and let $T^i_{\leq 1}$, T^i_2, and $T^i_{\geq 3}$, be the sets of vertices in F_i having degree at most one in F_i, degree exactly two in F_i, and degree greater than two in F_i, respectively. The degree of each vertex $v \in V(G_i)$ is bounded by $\mathcal{O}(k^2)$ in G_i, for $i \in [\alpha]$. In particular, the degree of each $s \in S$ is bounded by $\mathcal{O}(k^2)$ in G_i. Moreover, each vertex $v \in T^i_{\leq 1}$ has degree at least 2 in G_i, and therefore must be adjacent to some vertex in S. It follows that $|T^i_{\leq 1}| \in \mathcal{O}(k^3)$.

In a tree, the number t of vertices of degree at least three is bounded by $\ell - 2$, where ℓ is the number of leaves. Hence, $|T^i_{\geq 3}| \in \mathcal{O}(k^3)$. Also, in a tree, the number of maximal degree-two paths is bounded by $t + l$. Consequently, the number of degree-two paths in $G_i - S_i$ is in $\mathcal{O}(k^3)$. Moreover, no two maximal degree-two paths in a tree intersect.

Note that there are at most $\mathcal{O}(k^3)$ maximal degree-two paths in $\mathcal{P}_i$, for $i \in [\alpha]$, and therefore $|\mathfrak{S}| = \mathcal{O}(k^{3\alpha})$. After exhaustive application of Reduction Rule 10.8, for each path $P^i_j \in \mathcal{P}_i$, $i \in [\alpha]$, and $\tau \in \mathfrak{S}$, there is at most one vertex in $\mathsf{Intercept}(P^i_j, \tau)$. Also note that after exhaustive application of Reduction Rules 10.1 to 10.7, the total degree of a vertex in G is at least 3. Therefore, there can be at most $\mathcal{O}(k^{3\alpha})$ vertices in a degree-two path $P^i_j \in \mathcal{P}_i$. Furthermore, there are at most $\mathcal{O}(k^3)$ degree-two maximal paths in G_i, for $i \in [\alpha]$. It follows that $|T^i_2| \in \mathcal{O}(k^{3(\alpha+1)})$ and $|V(G_i)| \leq |T^i_{\leq 1}| + |T^i_2| + |T^i_{\geq 3}| + |S_i| = \mathcal{O}(k^3) + \mathcal{O}(k^{3(\alpha+1)}) + \mathcal{O}(k^3) + 2k \in \mathcal{O}(k^{3(\alpha+1)})$. Therefore, the number of vertices in G is in $\mathcal{O}(\alpha k^{3(\alpha+1)})$. □

10.3 Hardness Results

In this section, we show that Sim-FVS is W[1]-hard when $\alpha \in \mathcal{O}(\log n)$, where n is the number of vertices in the input graph. We give a reduction from a special version of the Hitting Set (HS) problem, which we denote by Partitioned Hitting Set (PHS). We believe this version of Hitting Set to be of independent interest with possible applications for showing hardness results of similar flavor. We prove W[1]-hardness of Partitioned Hitting Set by a reduction from a restricted version of the

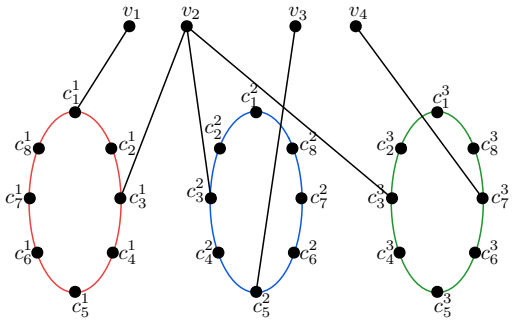

Figure 10.4: The graph G before contracting all edges colored zero for $\mathcal{U} = \{u_1, u_2, u_3, u_4\}$ and $\mathcal{F} = \{\{u_1, u_2\}, \{u_2, u_3\}, \{u_2, u_4\}\}$.

Partitioned Subgraph Isomorphism (PSI) problem.

Before we delve into the details, we start with a simpler reduction from Hitting Set showing that Sim-FVS is W[2]-hard when $\alpha \in \mathcal{O}(n)$. The reduction closely follows that of Lokshtanov [Lok08] for dealing with the Wheel-Free Deletion problem. Intuitively, starting with an instance $(\mathcal{U}, \mathcal{F}, k)$ of HS, we first construct a graph G on $2|\mathcal{U}||\mathcal{F}|$ vertices consisting of $|\mathcal{F}|$ vertex-disjoint cycles. Then, we use $|\mathcal{F}|$ colors to uniquely map each set to a separate cycle; carefully connecting these cycles together guarantees equivalence of both instances.

Theorem 10.4. Sim-FVS *parameterized by solution size is* W[2]-hard *when* $\alpha \in \mathcal{O}(n)$.

Proof. Given an instance $(\mathcal{U}, \mathcal{F}, k)$ of Hitting Set, we let $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$ and $\mathcal{F} = \{f_1, \ldots, f_{|\mathcal{F}|}\}$. We assume, without loss of generality, that each element in $\mathcal{U}$ belongs to at least one set in $\mathcal{F}$. For each $f_i \in \mathcal{F}$, $i \in [|\mathcal{F}|]$, we create a vertex-disjoint cycle C_i on $2|\mathcal{U}|$ vertices and assign all its edges color i. We let $V(C_i) = \{c_1^i, c_2^i, \ldots, c_{2|\mathcal{U}|}^i\}$ and we define $\beta(i, u_j) = c_{2j-1}^i$, for $i \in [|\mathcal{F}|]$ and $j \in [|\mathcal{U}|]$. In other words, every odd-numbered vertex of C_i is mapped to an element in $\mathcal{U}$. Now for every element $u_j \in \mathcal{U}$, $j \in [|\mathcal{U}|]$, we create a vertex v_j, we let $\gamma(u_j) = \{c_{2j-1}^i \mid i \in [|\mathcal{F}|] \wedge u_j \in f_i\}$, and we add an edge (of some special color, say zero) between v_j and every vertex in $\gamma(u_j)$ (see Figure 10.4).

To finalize the reduction, we contract all the edges colored zero to obtain an instance (G, k) of Sim-FVS. Note that $|V(G)| = |E(G)| \leq 2|\mathcal{U}||\mathcal{F}|$ and the total number of used colors is $|\mathcal{F}|$. Moreover, after contracting all 0-colored edges, $|\gamma(u_j)| = 1$ for all $u_j \in \mathcal{U}$.

Claim 10.2. *If $\mathcal{F}$ admits a hitting set of size at most k then G admits an $|\mathcal{F}|$-simfvs of size at most k.*

Proof. Let $X = \{u_{i_1}, \ldots, u_{i_k}\}$ be such a hitting set. We construct a vertex set $Y = \{\gamma(u_{i_1}), \ldots, \gamma(u_{i_k})\}$. If Y is not an $|\mathcal{F}|$-simfvs of G then $G[V(G) \setminus Y]$ must contain some cycle where all edges are assigned the same color. By construction, every set in $\mathcal{F}$ corresponds to a uniquely colored cycle in G. Hence, the contraction operations applied to obtain G cannot create new monochromatic cycles, i.e. every cycle in G which does not correspond to a set from $\mathcal{F}$ must include edges of at least two different colors. Therefore, if $G[V(G) \setminus Y]$ contains some monochromatic cycle then X cannot be a hitting set of $\mathcal{F}$. □

Claim 10.3. *If G admits an $|\mathcal{F}|$-simfvs of size at most k then $\mathcal{F}$ admits a hitting set of size at most k.*

Proof. Let $X = \{v_{i_1}, \ldots, v_{i_k}\}$ be such an $|\mathcal{F}|$-simfvs. First, note that if some vertex in X does not correspond to an element in $\mathcal{U}$, then we can safely replace that vertex with one that does (since any such vertex belongs to exactly one monochromatic cycle). We construct a set $Y = \{u_{i_1}, \ldots, u_{i_k}\}$. If there exists a set $f_i \in \mathcal{F}$ such that $Y \cap f_i = \emptyset$ then, by construction, there exists an i-colored cycle C_i in G such that $X \cap V(C_i) = \emptyset$, a contradiction. □

Combining the previous two claims with the fact that our reduction runs in time polynomial in $|\mathcal{U}|$, $|\mathcal{F}|$, and k, completes the proof of the lemma. □

Notice that the proof of Theorem 10.4 crucially relies on the fact that each cycle is "uniquely identified" by a separate color. In order to get around this limitation and prove W[1]-hardness for $\alpha \in \mathcal{O}(\log n)$ we need, in some sense, to group separate sets of a Hitting Set instance into $\mathcal{O}(\log(|\mathcal{U}||\mathcal{F}|))$ families such that sets inside each family are pairwise disjoint. By doing so, we can modify the proof of Theorem 10.4 to identify all sets inside a family using the same color, for a total of $\mathcal{O}(\log n)$ colors (instead of $\mathcal{O}(n)$). Next, we formally state the problems Partitioned Hitting Set (PHS) and Partitioned Subgraph Isomorphism (PSI).

Partitioned Hitting Set (PHS) **Parameter:** k
Input: A tuple $(\mathcal{U}, \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha, k)$, where $\mathcal{F}_i$, $i \in [\alpha]$, is a collection of subsets of the finite universe $\mathcal{U}$ and k is a positive integer. Moreover, all the sets within a family $\mathcal{F}_i$, for $i \in [\alpha]$, are pairwise disjoint.
Question: Is there a set $X \subseteq \mathcal{U}$ of size at most k such that for every $f \in \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha$, we have $f \cap X \neq \emptyset$?

Partitioned Subgraph Isomorphism (PSI) **Parameter:** $k = |E(G)|$
Input: A graph H, a graph G with $V(G) = \{g_1, \ldots, g_\ell\}$, and a coloring function $col : V(H) \to [\ell]$.
Question: Is there an injection $inj : V(G) \to V(H)$ such that for every $i \in [\ell]$, $col(inj(g_i)) = i$ and for every $(g_i, g_j) \in E(G)$, $(inj(g_i), inj(g_j)) \in E(H)$?

Theorem 10.5 ([GM09, Mar07])**.** Partitioned Subgraph Isomorphism *parameterized by $|E(G)|$ is* W[1]-hard*, even when the smaller graph G is connected and has maximum degree three. Moreover, the problem cannot be solved in time $f(k)n^{o(\frac{k}{\log k})}$, where f is an arbitrary function, $n = |V(H)|$, and $k = |E(G)|$, unless* ETH *fails.*

We make a few simplifying assumptions: For an instance of Partitioned Subgraph Isomorphism, we let H_i denote the subgraph of H induced on vertices colored i. We assume that $|V(H_i)| = 2^t$, for $i \in [\ell]$ and t some positive integer; adding isolated vertices to each set is enough to guarantee this size constraint. Moreover, we assume that whenever there is no edge $(g_i, g_j) \in E(G)$, then there are no edges between $V(H_i)$ and $V(H_j)$ in H (see Figure 10.5 for an example of an instance). Note that the PSI problem asks for a "colorful" subgraph of H isomorphic to G such that one vertex from H_i is mapped to the vertex g_i, $i \in [\ell]$. Therefore, it is also safe to assume that H_i, $i \in [\ell]$, is edgeless.

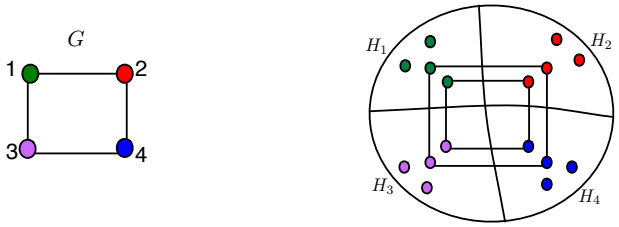

Figure 10.5: An instance of the PSI problem.

Theorem 10.6. Partitioned Hitting Set *parameterized by solution size is* W[1]-hard *when* $\alpha \in \mathcal{O}(\log(|\mathcal{U}||\mathcal{F}|))$. *Moreover, the problem cannot be solved in time* $f(k)n^{o(\frac{k}{\log k})}$, *where* f *is an arbitrary function,* $n = |\mathcal{U}|$, *and* k *is the required solution size, unless* ETH *fails.*

Proof. Given an instance $(H, G, col, \ell = |V(G)|, k = |E(G)|)$ of PSI, where G has maximum degree three, we reduce it into an instance $(\mathcal{U}, \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha, k' = k + \ell)$ of PHS, where $\alpha = 16 \log 2^t + 1 = 16t + 1$, $\mathcal{F}_i$, $i \in [\alpha]$, is a collection of subsets of the finite universe $\mathcal{U}$, and all the sets within a family $\mathcal{F}_i$ are pairwise disjoint.

We start by constructing the universe $\mathcal{U}$. For each vertex $h^i_j \in V(H_i)$, $i \in [\ell]$ and $0 \leq j \leq 2^t - 1$, we create an element v^i_j. For each edge $(h^{i_1}_{j_1}, h^{i_2}_{j_2}) \in E(H)$, we create an element $e^{i_1,i_2}_{j_1,j_2}$ where j_1 is the index of the vertex in H_{i_1}, j_2 is the index of the vertex in H_{i_2}, $i_1, i_2 \in [\ell]$, and $0 \leq j_1, j_2 \leq 2^t - 1$. Note that $|\mathcal{U}| = |V(H)| + |E(H)| = \ell 2^t + |E(H)| < 4^t 2\ell^2$.

We now create "selector gadgets" between elements corresponding to vertices and elements corresponding to edges. For every *ordered* pair (x, y), where $x, y \in [\ell]$, such that there exists an edge between H_x and H_y in H (or equivalently there exists an edge (g_x, g_y) in G), we create $2t$ sets. We denote half of those sets by $U_{x,y,p}$ and the order half by $D_{x,y,p}$, where $p \in [t]$. Let $\mathcal{U}_x$ denote the set of all elements corresponding to vertices in H_x and let $\mathcal{U}_{x,y}$ (x and y unordered in $\mathcal{U}_{x,y}$) denote the set of all elements corresponding to edges between vertices in H_x and vertices H_y. We let $bit(i)[p]$, $0 \leq i \leq 2^t - 1$ and $p \in [t]$, be the p^{th} bit in the bit representation of i (where position $p = 1$ holds the most significant bit). For each $v^x_i \in \mathcal{U}_x$ and for all p from 1 to t, if $bit(i)[p] = 0$ we add v^x_i to set $D_{x,y,p}$ and we add v^x_i to set $U_{x,y,p}$ otherwise. For each $e^{x,y}_{i,j} \in \mathcal{U}_{x,y}$ and for all p from 1 to t, if $bit(i)[p] = 0$ we add $e^{x,y}_{i,j}$ to set $U_{x,y,p}$ and we add $e^{x,y}_{i,j}$ to set $D_{x,y,p}$ otherwise. Recall that for $e^{x,y}_{i,j}$, i corresponds to the index of element $v^x_i \in \mathcal{U}_x$.

Finally, for each x, where $x \in [\ell]$, we add the set $Q_x = \mathcal{U}_x$, and for each (unordered) pair x,y such that $(g_x, g_y) \in E(G)$ we add the set $Q_{x,y} = \mathcal{U}_{x,y}$. Put differently, a set Q_x contains all elements corresponding to vertices in H_x and a set $Q_{x,y}$ contains all elements corresponding to edges between H_x and H_y. The role of these $\ell + k$ sets is simply to force a solution to pick at least one element from every $\mathcal{U}_x$ and one element from every $\mathcal{U}_{y,z}$, $x, y, z \in [\ell]$. Note that we have a total of $4t|E(G)| + |E(G)| + \ell < 4t\ell^2 + \ell^2 + \ell$ sets and therefore $16t + 1 \in \mathcal{O}(\log(|\mathcal{U}||\mathcal{F}|))$. We set $k' = |V(G)| + |E(G)| = \ell + k$. This completes the construction. An example of the construction for the instance given in Figure 10.5 is provided in Figure 10.6.

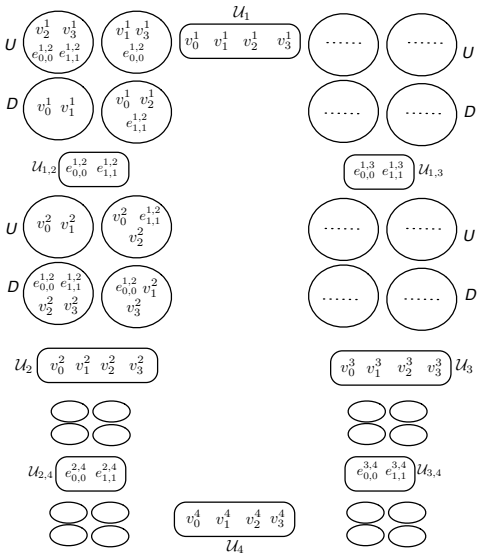

Figure 10.6: Parts of the reduction for the PSI instance from Figure 10.5. Rounded rectangles represents subsets of the universe and circles (and ellipses) represent sets in the family.

Claim 10.4. *In the resulting instance* $(\mathcal{U}, \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha, k' = k + \ell)$, $\alpha = 16 \log 2^t + 1 = 16t + 1$.

Proof. First, we note that all sets Q_x and $Q_{y,z}$, $x, y, z \in [\ell]$, are pairwise disjoint. Hence, we can group all these sets into a single partition. We now prove that $16t$ is enough to partition the remaining sets.

Since G has maximum degree three, we know by Vizing's theorem [Viz64] that G admits a proper 4-edge-coloring, i.e. no two edges incident on the same vertex receive the same color. Let us fix such a 4-edge-coloring and denote it by $\beta : E(G) \to \{1, 2, 3, 4\}$. Recall that for every ordered pair (x, y), where $x, y \in [\ell]$, we define two groups of sets $U_{x,y,p}$ and $D_{x,y,p}$, $p \in [t]$. Given any set $X_{x,y,p}$, $X \in \{U, D\}$, we define the partition to which $X_{x,y,p}$ belongs as $part(X, x, y, p) = (\beta(g_x, g_y), p, \{U, D\}, \{x < y, x > y\})$. In other words, we have a total of $16t$ partitions depending on the color of the edge (g_x, g_y) in G, the position p, whether $X = U$ or $X = D$, and whether $x < y$ or $x > y$ (recall that we assume $x \neq y$).

Since β is a proper 4-coloring of the edges of G, we know that if two sets belong to the same partition they must be of the form $X_{x_1,y_1,p}$ and $X_{x_2,y_2,p}$, where $X \in \{U, D\}$, $x_1 \neq x_2$, $y_1 \neq y_2$, $\beta(g_{x_1}, g_{y_1}) = \beta(g_{x_2}, g_{y_2})$, $x_1 < y_1$ ($x_1 > y_1$), and $x_2 < y_2$ ($x_2 > y_2$). It follows from our construction that $X_{x_1,y_1,p} \cap X_{x_2,y_2,p} = \emptyset$; $X_{x_1,y_1,p}$ only contains elements from $\mathcal{U}_{x_1} \cup \mathcal{U}_{x_1,y_1}$, $X_{x_2,y_2,p}$ only contains elements from $\mathcal{U}_{x_2} \cup \mathcal{U}_{x_2,y_2}$, and $(\mathcal{U}_{x_1} \cup \mathcal{U}_{x_1,y_1}) \cap (\mathcal{U}_{x_2} \cup \mathcal{U}_{x_2,y_2})$ is empty. □

Claim 10.5. *The resulting instance* $(\mathcal{U}, \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha, k' = k + \ell)$ *admits no hitting set of size* $k' - 1$.

Proof. If there exists a hitting set S of size $k'-1$, then either (1) there exists $\mathcal{U}_x$, where $x \in [\ell]$, such that $S \cap \mathcal{U}_x = \emptyset$ or (2) there exists $\mathcal{U}_{y,z}$, where $y, z \in [\ell]$, such that $S \cap \mathcal{U}_{y,z} = \emptyset$. In case (1), we are left with a set Q_x which is not hit by S. Similarly, for case (2), there exists a set $Q_{y,z}$ which is not hit by S. In both cases we get a contradiction as we assumed S to be a hitting set, as needed. □

Claim 10.6. *Any hitting set of size k' of the resulting instance $(\mathcal{U}, \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha, k' = k + \ell)$ must pick exactly one element from each set $\mathcal{U}_x$, where $x \in [\ell]$, and exactly one element from each set $\mathcal{U}_{y,z}$, where $y, z \in [\ell]$. Moreover, for every ordered pair (x, y), for $x, y \in [\ell]$, a hitting set of size k' must pick $v_i^x \in \mathcal{U}_x$ and $e_{i,j}^{x,y} \in \mathcal{U}_{x,y}$, $0 \leq i, j \leq 2^t - 1$. In other words, the vertex $h_i^x \in V(H)$ is incident to the edge $(h_i^x, h_j^y) \in E(H)$.*

Proof. The first part of the claim follows from the previous claim combined with the fact that $k' = k + \ell$. For the second part, assume that there exists a hitting set S of size k' such that for some ordered pair, (x, y), S includes $v_{i_1}^x \in \mathcal{U}_x$ and $e_{i_2,j}^{x,y} \in \mathcal{U}_{x,y}$, where $i_1 \neq i_2$. Since $i_1 \neq i_2$, then $bit(i_1)[p] \neq bit(i_2)[p]$ for at least one position p. For that position, we know that $v_{i_1}^x$ and $e_{i_2,j}^{x,y}$ must both belong to only one of $U_{x,y,p}$ or $D_{x,y,p}$. Hence, either $U_{x,y,p}$ or $D_{x,y,p}$ is not hit by $v_{i_1}^x$ and $e_{i_2,j}^{x,y}$ when $i_1 \neq i_2$. □

Claim 10.7. *If $(H, G, col, \ell = |V(G)|, k = |E(G)|)$, where G has maximum degree three, is a yes-instance of* PSI *then $(\mathcal{U}, \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha, k' = k + \ell)$ is a yes-instance of* PHS*.*

Proof. Let S, a subgraph of H, denote the solution graph and let $V(S) = \{h_{i_1}^1, \ldots, h_{i_\ell}^\ell\}$. We claim that $S' = \{v_{i_1}^1, \ldots, v_{i_\ell}^\ell\} \cup \{e_{j_1,j_2}^{x,y} \mid (g_x, g_y) \in E(G) \wedge j_1, j_2 \in \{i_1, \ldots, i_\ell\}\}$ is a hitting set of $\mathcal{F}$. That is, the hitting set picks ℓ elements corresponding to the ℓ vertices in S (or G) and k elements corresponding to the k edges in G.

Clearly, all sets Q_x and $Q_{y,z}$, for $x, y, z \in [\ell]$, are hit since we pick one element from each. We now show that all sets $U_{x,y,p}$ and $D_{x,y,p}$, $x, y \in [\ell]$ and $p \in [t]$, are also hit. Assume, without loss of generality, that for fixed x, y, and p, some set $U_{x,y,p}$ is not hit. Let $v_{i_1}^x \in \mathcal{U}_x$ be the element we picked from $\mathcal{U}_x$ and let $e_{i_2,j}^{x,y}$ be the element we picked from $\mathcal{U}_{x,y}$. If $U_{x,y,p}$ is not hit, it must be the case that $i_1 \neq i_2$ which, by the previous claim, is not possible. □

Claim 10.8. *If $(\mathcal{U}, \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha, k' = k + \ell)$ is a yes-instance of* PHS *then $(H, G, col, \ell = |V(G)|, k = |E(G)|)$ is a yes-instance of* PSI*.*

Proof. Let $S = \{v_{i_1}^1, \ldots, v_{i_\ell}^\ell\} \cup \{e_{j_1,j_2}^{x,y} | (g_x, g_y) \in E(G) \wedge j_1, j_2 \in \{i_1, \ldots, i_\ell\}\}$ be a hitting set of $\mathcal{F}$. Note that we can safely assume that the hitting set picks such elements since it has to hit all sets Q_x and $Q_{y,z}$, for $x, y, z \in [\ell]$. We claim that the subgraph S' of H with vertex set $V(S') = \{h_{i_1}^1, \ldots, h_{i_\ell}^\ell\}$ is a solution to the PSI instance.

By construction, there is an injection $inj : V(G) \to V(S')$ such that for every $i \in [\ell]$, $col(inj(g_i)) = i$. In fact, S' contains exactly one vertex for each color $i \in [\ell]$. Assume that there exists an edge $(g_i, g_j) \in E(G)$ such that $(inj(g_i), inj(g_j)) \notin E(S')$. This implies that there exists two vertices $h_i^x, h_j^y \in V(S')$ such that $(h_i^x, h_j^y) \notin E(S')$. But we know that there exists at least one edge, say $(h_{i'}^x, h_{j'}^y)$, between vertices in H_x and vertices in H_y (from our assumptions). Since $i' \neq i$, $j' \neq j$, $v_i^x, v_j^y \in S$, and $e_{i,j}^{x,y} \notin S$, it follows that S cannot be a hitting set of $\mathcal{F}$ as at least one set in $U_{x,y,p} \cup D_{x,y,p}$ and one set in $U_{y,x,p} \cup D_{y,x,p}$ is not hit by S, a contradiction. □

This completes the proof of the theorem. □

We are now ready to state the main result of this section. The proof of Theorem 10.7 follows the same steps as the proof of Theorem 10.4 with one exception, i.e we reduce from Partitioned Hitting Set with $\alpha \in \mathcal{O}(\log(|\mathcal{U}||\mathcal{F}|))$ and use $\mathcal{O}(\log(|\mathcal{U}||\mathcal{F}|))$ colors instead of $|\mathcal{F}|$.

Theorem 10.7. Sim-FVS *parameterized by solution size is* W[1]-hard *when* $\alpha \in \mathcal{O}(\log n)$.

Proof. Given an instance $(\mathcal{U}, \mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_\alpha, k)$ of PHS, we let $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$ and $\mathcal{F}_i = \{f_1^i, \ldots, f_{|\mathcal{F}_i|}^i\}$, where $i \in [\alpha]$. We assume, without loss of generality, that each element in $\mathcal{U}$ belongs to at least one set in $\mathcal{F}$.

For each $f_j^i \in \mathcal{F}_i$, where $i \in [\alpha]$ and $j \in [|\mathcal{F}_i|]$, we create a vertex-disjoint cycle C_j^i on $2|\mathcal{U}|$ vertices and assign all its edges color i. We let $V(C_j^i) = \{c_1^{i,j}, \ldots, c_{2|\mathcal{U}|}^{i,j}\}$ and we define $\beta(i, j, u_p) = c_{2p-1}^{i,j}$, where $i \in [\alpha]$, $j \in [|\mathcal{F}_i|]$, and $p \in [|\mathcal{U}|]$. In other words, every odd-numbered vertex of C_j^i is mapped to an element in $\mathcal{U}$. Now for every element $u_p \in \mathcal{U}$, where $p \in [|\mathcal{U}|]$, we create a vertex v_p, we let $\gamma(u_p) = \{c_{2p-1}^{i,j} \mid i \in [\alpha] \wedge j \in [|\mathcal{F}_i|] \wedge u_p \in f_j^i\}$, and we add an edge (of some special color, say 0) between v_p and every vertex in $\gamma(u_p)$. To finalize the reduction, we contract all the edges colored 0 to obtain an instance (G, k) of Sim-FVS. Note that $|V(G)| = |E(G)| = 2|\mathcal{U}||\mathcal{F}|$ and the total number of used colors is α. Moreover, after contracting all special edges, $|\gamma(u_p)| = 1$ for all $u_p \in \mathcal{U}$.

Claim 10.9. *If $\mathcal{F}$ admits a hitting set of size at most k then G admits an α-simfvs of size at most k.*

Proof. Let $X = \{u_{p_1}, \ldots, u_{p_k}\}$ be such a hitting set. We construct a vertex set $Y = \{\gamma(u_{p_1}), \ldots, \gamma(u_{p_k})\}$. If Y is not an α-simfvs of G then $G[V(G) \setminus Y]$ must contain some monochromatic cycle. By construction, only sets from the same family $\mathcal{F}_i$, for $i \in [\alpha]$, correspond to cycles assigned the same color in G. But since we started with an instance of PHS, no two such sets intersect. Hence, the contraction operations applied to obtain G cannot create new monochromatic cycles. Therefore, if $G[V(G) \setminus Y]$ contains some monochromatic cycle then X cannot be a hitting set of $\mathcal{F}$. □

Claim 10.10. *If G admits an α-simfvs of size at most k then $\mathcal{F}$ admits a hitting set of size at most k.*

Proof. Let $X = \{v_{p_1}, \ldots, v_{p_k}\}$ be such an α-simfvs. First, note that if some vertex in X does not correspond to an element in $\mathcal{U}$, then we can safely replace that vertex with one that does (since any such vertex belongs to exactly one monochromatic cycle). We construct a set $Y = \{u_{p_1}, \ldots, u_{p_k}\}$. If there exists a set $f_j^i \in \mathcal{F}_i$ such that $Y \cap f_j^i = \emptyset$ then, by construction, there exists an i-colored cycle C_i in G such that $X \cap V(C_i) = \emptyset$, a contradiction. □

Combining the previous two claims with the fact that our reduction runs in time polynomial in $|\mathcal{U}|$, $|\mathcal{F}|$, and k, completes the proof of the theorem. □

Chapter 11

Simultaneous Feedback Edge Set

In this chapter, we consider the edge variant, namely, Simultaneous Feedback Edge Set (Sim-FES for short) of the problem Simultaneous Feedback Vertex Set. It is more convenient (and logical) to view an α-edge-colored graph as a graph with a coloring function $\mathsf{col} : E(G) \to 2^{[\alpha]}$. Therefore, in this chapter, we use this notation for an α-edge-colored graphs. A *feedback edge set* in a graph G is $S \subseteq E(G)$ such that $G - S$ is a forest. For a graph G with a coloring function $\mathsf{col} : E(G) \to 2^{[\alpha]}$, *simultaneous feedback edge set* is a subset $S \subseteq E(G)$ such that $G_i - S$ is a forest for all $i \in [\alpha]$. Here, $G_i = (V(G), E_i)$, where $E_i = \{e \in E(G) \mid i \in \mathsf{col}(e)\}$. Next, we define the Sim-FES problem formally defined below.

Simultaneous Feedback Edge Set (Sim-FES) **Parameter:** k and α
Input: An α-edge-colored graph G with coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$ and an integer k.
Question: Is there a simultaneous feedback edge set of cardinality at most k in G?

Unlike the vertex variant of the problem, when $\alpha = 1$, the problem is equivalent to finding a maximal spanning forest, and hence is polynomial time solvable. In Section 11.1 we design an FPT algorithm for Sim-FES by reducing to α-Matroid Parity on direct sum of elongated co-graphic matroids of G_i, $i \in [\alpha]$ (see Chapter 2 for definitions related to matroids). This algorithm runs in time $\mathcal{O}(2^{\omega k\alpha+\alpha \log k} n^{\mathcal{O}(1)})$. We also show that (unlike the vertex counterpart) for $\alpha = 2$ (2-edge-colored graphs) Sim-FES is polynomial time solvable. This follows from the polynomial time algorithm for the Matroid parity problem. In Section 11.2 we show that for $\alpha = 3$, Sim-FES is NP-hard. Towards this, we give a reduction from the Vertex Cover in cubic graphs which is known to be NP-hard [Moh01]. Furthermore, the same reduction shows that the problem cannot be solved in $2^{o(k)} n^{\mathcal{O}(1)}$ time unless ETH fails. We complement our FPT algorithms by showing that Sim-FES is W[1]-hard when parameterized by the solution size k (Section 11.3). When $\alpha = \mathcal{O}(|V(G)|)$, we give a parameter preserving reduction from the Hitting Set problem, a well known W[2]-hard problem parameterized by the solution size [CFK+15]. However, Sim-FES remains W[1]-hard even when $\alpha = \mathcal{O}(\log(|V(G)|))$. We show this by giving a parameter preserving reduction from Partitioned Hitting Set problem, a variant of the Hitting set problem, which is W[1]-hard parameterized by the solution size (see Chapter 10, Section 10.3). In Section 11.4, we give a kernel with $\mathcal{O}((k\alpha)^{\mathcal{O}(\alpha)})$ vertices. Towards this we apply some of the standard preprocessing rules for obtaining kernel for Feedback Vertex Set and use the approach similar to the one used for

designing kernelization algorithm for Sim-FVS. In Section 11.5 we give an FPT algorithm for the problem, when parameterized by the dual parameter. Formally, this problem is defined as follows.

Max-Sim Acyclic-Subgraph (Max-Sim-Subgraph) **Parameter:** q
Input: An α-edge-colored graph G with coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$ and an integer q.
Question: Is there a subset $F \subseteq E(G)$ such that $\lvert F\rvert \geq q$ and for each $i \in [\alpha]$, the graph $G_i[F]$ is acyclic?

For solving Max-Sim-Subgraph we reduce it to an equivalent instance of the α-Matroid Parity problem. As an immediate corollary we get an exact algorithm for Sim-FES running in time $\mathcal{O}(2^{\omega n\alpha^2} n^{\mathcal{O}(1)})$.

11.1 FPT Algorithm for Simultaneous Feedback Edge Set

In this section, we design an algorithm for Sim-FES by giving a reduction to α-Matroid Parity on the direct sum of elongated co-graphic matroids associated with graphs restricted to different color classes.

We describe our algorithm, Algo-SimFES, for Sim-FES. Let $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ be an input instance to Sim-FES, where $\lvert V(G)\rvert = n$. Let η_i be the number of connected components in G_i. To make G_i acyclic we need to delete at least $\lvert E(G_i)\rvert - n + \eta_i$ edges from G_i. Therefore, if there is $i \in [\alpha]$ such that $\lvert E(G_i)\rvert - n + \eta_i > k$, then Algo-SimFES returns *no*. We let $k_i = \lvert E(G_i)\rvert - n + \eta_i$. Observe that for $i \in [\alpha]$, $0 \leq k_i \leq k$. We need to delete at least k_i edges from $E(G_i)$ to make G_i acyclic. Therefore, the algorithm Alg-SimFES for each $i \in [\alpha]$, guesses k'_i, where $k_i \leq k'_i \leq k$ and computes a solution S of Sim-FES such that $\lvert S \cap E(G_i)\rvert = k'_i$. Let $M_i = (E_i, \mathcal{I}_i)$ be the k'_i-elongation of the co-graphic matroid associated with G_i.

Proposition 11.1. *Let G be a graph with η connected components and M be an r-elongation of the co-graphic matroid associated with G, where $r \geq \lvert E(G)\rvert - \lvert V(G)\rvert + \eta$. Then $B \subseteq E(G)$ is a basis of M if and only if the subgraph $G - B$ is acyclic and $\lvert B\rvert = r$.*

Proof. In the forward direction let $B \subseteq E(G)$ be a basis of M. By Definition of M it follows that $\lvert B\rvert = r$ and B contains a basis B_c of the co-graphic matroid of G. Suppose $G - B$ has a cycle. This implies that $G - B_c$ has a cycle. But then, there is an edge $e \in E(G - B_c)$ whose removal from $G - B_c$ does not increase the number of connected components in $G - B_c$. This contradicts that B_c was a basis in the co-graphic matroid of G.

In the reverse direction let $B \subseteq E(G)$ such that $\lvert B\rvert = r$ and $G - B$ is acyclic. Consider a inclusion wise maximal subset $B' \subseteq B$ such that the number of connected components in $G - B'$ is η. Observe that $G - B'$ does not contain a cycle since $G - B$ is acyclic and B' is inclusion wise maximal. Therefore, it follows that B' is a basis in the co-graphic matroid of G. But then B contains a basis of the co-graphic matroid of G and $\lvert B\rvert = r$, therefore B is a basis in M. □

By Proposition 11.1, for any basis F_i in M_i, $G_i - F_i$ is acyclic. Therefore, our objective is to compute $F \subseteq E(G)$ such that $\lvert F\rvert = k$ and the elements of F restricted

Algorithm 3: Pseudocode of Algo-SimFES

Input: An α-edge-colored graph G with coloring $\mathsf{col} : E(G) \to 2^{[\alpha]}$ and an integer k.

Output: *yes* or *no*.

```
1  Let η_i be the number of connected components in G_i for each i ∈ [α]
2  k_i := |E(G_i)| − n + η_i for all i ∈ [α]
3  if there exists i ∈ [α] such that k_i > k then
4      return no
5  for (k'_1, ..., k'_α) ∈ ([k] ∪ {0})^α such that k_i ≤ k'_i for all i ∈ [α] do
6      Let M_i be the k'_i-elongation of the co-graphic matroid associated with G_i.
7      Let M_{α+1} = U_{τ,k'} over the ground set Fake(G), where, k' = Σ_{i∈[α]}(k − k'_i).
8      Let M := ⊕_{i∈[α+1]} M_i.
9      For each e ∈ E(G), let Copies(e) be the block of elements of M.
10     if there is an independent set of M composed of k blocks then
11         return yes
12 return no
```

to the elements of M_i form a basis for all $i \in [\alpha]$. For this we will construct an instance of α-Matroid Parity as follows. For each $e \in E(G)$ and $i \in \mathsf{col}(e)$, we use e^i to denote the corresponding element in M_i. For each $e \in E(G)$, by $\mathsf{Original}(e)$ we denote the set of elements $\{e^j \mid j \in \mathsf{col}(e)\}$. For each edge $e \in E(G)$, we define $\mathsf{Fake}(e) = \{e^j \mid j \in [\alpha] - \mathsf{col}(e)\}$. Finally, for each edge $e \in E(G)$, by $\mathsf{Copies}(e)$ we denote the set $\mathsf{Original}(e) \cup \mathsf{Fake}(e)$. Let $\mathsf{Fake}(G) = \bigcup_{e \in E(G)} \mathsf{Fake}(e)$. Furthermore, let $\tau = |\mathsf{Fake}(G)| = \sum_{e \in E(G)} |\mathsf{Fake}(e)|$ and $k' = \sum_{i \in [\alpha]}(k - k'_i)$. Let $M_{\alpha+1} = (E_{\alpha+1}, \mathcal{I}_{\alpha+1})$ be a uniform matroid over the ground set $\mathsf{Fake}(G)$. That is, $M_{\alpha+1} = U_{\tau,k'}$. By Propositions 2.4 to Proposition 2.6 we know that M_is are representable over $\mathbb{F}_p(X)$, where $p > \max(\tau, 2)$ is a prime number and their representation can be computed in polynomial time. Let A_i be the linear representation of M_i for all $i \in [\alpha + 1]$. Notice that $E_i \cap E_j = \emptyset$ for all $1 \leq i \neq j \leq \alpha + 1$. Let M denote the direct sum $M_1 \oplus \cdots \oplus M_{\alpha+1}$ with its representation matrix being A_M. Note that the ground set of M is $\bigcup_{e \in E(G)} \mathsf{Copies}(e)$. Now we define an instance of α-Matroid Parity, which is the linear representation A_M of M and the partition of ground set into $\mathsf{Copies}(e)$, $e \in E(G)$. Notice that for all $e \in E(G)$, $|\mathsf{Copies}(e)| = \alpha$. Also for each $i \in [\alpha]$, $\mathsf{rank}(M_i) = k'_i$ and $\mathsf{rank}(M_{\alpha+1}) = k' = \sum_{i \in [\alpha]}(k - k'_i)$. This implies that $\mathsf{rank}(M) = \alpha k$.

Now Algo-SimFES outputs *yes* if there is a basis (an independent set of cardinality αk) of M which is a union of k blocks in M and otherwise outputs *no*. Algo-SimFES uses the algorithm mentioned in Proposition 2.7 to check whether there is an independent set of M, composed of blocks. A pseudocode of Algo-SimFES can be found in Algorithm 3.

Lemma 11.1. Algo-SimFES *is correct.*

Proof. Let $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ be a *yes* instance of Sim-FES and let $F \subseteq E(G)$, where $|F| = k$ be a solution of $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$. Let $k_i = |E(G_i)| - n + \eta_i$, where η_i is the number of connected components in G_i, for all $i \in [\alpha]$. For all $i \in [\alpha]$, let $k'_i = |F \cap E(G_i)|$. Since F is a solution, $k_i \leq k'_i$ for all $i \in [\alpha]$. This implies that Algo-SimFES will not execute Step 4. Consider the **for** loop for the choice $(k'_1, \ldots, k'_\alpha)$.

We claim that the columns corresponding to $S = \bigcup_{e \in F} \mathsf{Copies}(e)$ form a basis in M and it is union of k blocks. Note that $|S| = \alpha k$ by construction. For all $i \in [\alpha]$, let $F^i = \{e^i \mid e \in F, i \in \mathsf{col}(e)\}$, which is subset of ground set of M_i. By Proposition 11.1, for all $i \in [\alpha]$, F^i is a basis for M_i. This takes care of all the edges in $\cup_{e \in F} \mathsf{Original}(e)$. Now let $S^* = S - \cup_{i \in [\alpha]} F^i = \cup_{e \in F} \mathsf{Fake}(e)$. Observe that $|S^*| = \sum_{i \in [\alpha]} (k - k'_i) = k'$. Also, S^* is a subset of ground set of $U_{\tau,k'}$ and thus is a basis since $|S^*| = k'$. Hence S is a basis of M. Note that S is the union of blocks corresponding to $e \in F$ and hence is union of k blocks. Therefore, Algo-SimFES will output *yes*.

In the reverse direction suppose Algo-SimFES outputs *yes*. This implies that there is a basis, say S, that is the union of k blocks. By construction S corresponds to union of the sets $\mathsf{Copies}(e)$ for some k edges in G. Let these edges be $F = \{e_1, \ldots, e_k\}$. We claim that F is a solution of $(G, k, \mathsf{col} : E(G) \to 2^{[\alpha]})$. Clearly $|F| = k$. Since S is a basis of M, for each $i \in [\alpha]$, $B(i) = S \cap \{e^i \mid e \in E(G_i)\}$ is a basis in M_i. Let $F(i) = \{e \mid e^i \in B(i)\} \subseteq F$. Since $B(i)$ is a basis of M_i, by Proposition 11.1, $G_i - F(i)$ is an acyclic graph. □

Lemma 11.2. Algo-SimFES *runs in deterministic time* $\mathcal{O}(2^{\omega k \alpha + \alpha \log k} |V(G)|^{\mathcal{O}(1)})$.

Proof. The for loop runs $(k+1)^\alpha$ times. The step 10 uses the algorithm mentioned in Proposition 2.7, which takes time $\mathcal{O}(2^{\omega k \alpha} ||A_M||^{\mathcal{O}(1)}) = \mathcal{O}(2^{\omega k \alpha} |V(G)|^{\mathcal{O}(1)})$. All other steps in the algorithm takes polynomial time. Thus, the total running time is $\mathcal{O}(2^{\omega k \alpha + \alpha \log k} |V(G)|^{\mathcal{O}(1)})$. □

Since α-Matroid Parity for $\alpha = 2$ can be solved in polynomial time [Lov80] algorithm Algo-SimFES runs in polynomial time for $\alpha = 2$. This gives us the following theorem.

Theorem 11.1. Sim-FES *is in* FPT *and when* $\alpha = 2$ Sim-FES *is in* P.

11.2 Hardness Results

In this section, we show that when $\alpha = 3$, Sim-FES is NP-Hard. Furthermore, from our reduction we conclude that it is unlikely that Sim-FES admits a subexponential-time algorithm. We give a reduction from Vertex Cover (VC) in cubic graphs to the special case of Sim-FES where $\alpha = 3$. Let (G, k) be an instance of VC in cubic graphs, which asks whether the graph G has a vertex cover of size at most k. We assume without loss of generality that $k \leq |V(G)|$. It is known that VC in cubic graphs is NP-hard [Moh01] and unless the ETH fails, it cannot be solved in time $\mathcal{O}^\star(2^{o(|V(G)|+|E(G)|)})$ [Kom15]. Thus, to prove that when $\alpha = 3$, it is unlikely that Sim-FES admits a parameterized subexponential time algorithm (an algorithm of running time $\mathcal{O}^\star(2^{o(k)})$), it is sufficient to construct (in polynomial time) an instance of the form $(G', \mathsf{col}' : E' \to 2^{[3]}, k' = \mathcal{O}(|V(G)| + |E(G)|))$ of Sim-FES that is equivalent to (G, k). Refer Figure 11.1 for an illustration of the construction.

To construct $(G', k', \mathsf{col}' : E(G') \to 2^{[3]})$, we first construct an instance $(\widehat{G}, \widehat{k})$ of VC in subcubic graphs which is equivalent to (G, k). We set

$$V(\widehat{G}) = V(G) \cup \Big(\bigcup_{(v,u) \in E(G)} \{x_{v,u}, x_{u,v}\} \Big), \text{ and}$$

$$E(\widehat{G}) = \{(x_{v,u}, x_{u,v}) \mid (v,u) \in E(G)\} \cup (\bigcup_{(v,u)\in E(G)} \{(v, x_{v,u}), (u, x_{u,v})\}).$$

That is, the graph $\widehat{G}$ is obtained from the graph G by subdividing each edge in $E(G)$ twice.

Lemma 11.3. *G has a vertex cover of size k if and only if $\widehat{G}$ has a vertex cover of size $\widehat{k} = k + |E(G)|$*

Proof. In the forward direction, let S be a vertex cover in G. We will construct a vertex cover $\widehat{S}$ in $\widehat{G}$ of size at most $k + |E(G)|$. Consider an edge $(v,u) \in E(G)$. If both u, v belongs to S, then we arbitrarily add one of the vertices from $\{x_{v,u}, x_{u,v}\}$ to $\widehat{S}$. If exactly one vertex in $\{v,u\}$ belongs to S, say $v \in S$ then, we add $x_{u,v}$ to $\widehat{S}$. If $u \in S$, then we add $x_{v,u}$ to $\widehat{S}$. Clearly, $\widehat{S}$ is a vertex cover in $\widehat{G}$ and is of size at most $k + |E(G)|$.

In the reverse direction, given a vertex cover in $\widehat{G}$. For each $(v,u) \in E(G)$ such that both $x_{v,u}$ and $x_{u,v}$ are in the vertex cover, we can replace $x_{u,v}$ by u, and then, by removing all of the remaining vertices of the form $x_{v,u}$ (whose number is exactly $|E(G)|$), we obtain a vertex cover of G. □

Observe that in $\widehat{G}$ every path between two degree-3 vertices contains an edge of the form $(x_{v,u}, x_{u,v})$. Thus, the following procedure results in a partition (M_1, M_2, M_3) of $E(\widehat{G})$ such that for all $i \in [3]$, $(v,u) \in M_i$ and $(v',u') \in M_i \setminus \{(v,u)\}$, it holds that $\{v,u\} \cap \{v',u'\} = \emptyset$. Initially, $M_1 = M_2 = M_3 = \emptyset$. For each degree-3 vertex v, let (v,x), (v,y) and (v,z) be the edges containing v. We insert (v,x) into M_1, (v,y) into M_2, and (v,z) into M_3 (the choice of which edge is inserted into which set is arbitrary). Finally, we insert each edge of the form $(x_{v,u}, x_{u,v})$ into a set M_i that contains neither $(v, x_{v,u})$ nor $(u, x_{u,v})$.

We are now ready to construct the instance $(G', \mathsf{col}' : E(G') \to 2^{[3]}, k')$. Let $V(G') = V(\widehat{G}) \cup V^\star$, where $V^\star = \{v^\star \mid v \in V(\widehat{G})\}$ contains a copy $v^\star$ of each vertex v in $V(\widehat{G})$. The set $E(G')$ and coloring col' are constructed as follows. For each vertex $v \in V(\widehat{G})$, add an edge $(v, v^\star)$ into $E(G')$ and its color-set is $\{1,2,3\}$. For each $i \in [3]$ and for each $(v,u) \in M_i$, add the edges (v,u) and $(v^\star, u^\star)$ into $E(G')$ and its color-set is $\{i\}$. We set $k' = \widehat{k}$. Clearly, the instance $(G', \mathsf{col}' : E(G') \to 2^{[3]}, k')$ can be constructed in polynomial time, and it holds that $k' = \mathcal{O}(|V(G)| + |E(G)|)$.

Lemma 11.4 proves that $(\widehat{G}, \widehat{k})$ is a *yes* instance of VC if and only if $(G', \mathsf{col}' : E(G') \to 2^{[3]}, k')$ is a *yes* instance of Sim-FES. Observe that because of the above mentioned property of the partition (M_1, M_2, M_3) of $E(\widehat{G})$, we ensure that in G', no vertex participates in two (or more) monochromatic cycles that have the same color. By construction, each monochromatic cycle in G' is of the form $(v, v^\star, u^\star, u, v)$, where $(v,u) \in E(\widehat{G})$, and for each edge $(v,u) \in E(G')$, where either $v, u \in V(\widehat{G})$ or $v, u \in V^\star$, G' contains exactly one monochromatic cycle of this form.

Lemma 11.4. *$(\widehat{G}, \widehat{k})$ is a yes instance of VC if and only if $(G', \mathsf{col}' : E(G') \to 2^{[3]}, k')$ is a yes instance of* Sim-FES.

Proof. In the forward direction, let U be a vertex cover in $\widehat{G}$ of size at most $\widehat{k}$. Define Q as the set of edges $\{(v, v^\star) \mid v \in U\} \subseteq E(G')$. We claim that Q is a solution to $(G', \mathsf{col}' : E(G') \to 2^{[3]}, k')$. Since $|Q| = |U|$, it holds that $|Q| \leq \widehat{k} = k'$. Now, consider a

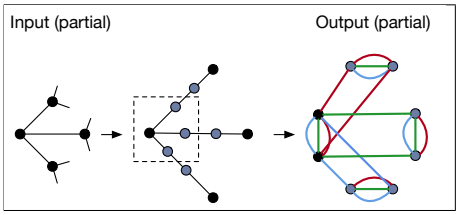

Figure 11.1: The construction given in the proof of Theorem 11.2.

monochromatic cycle in G'. Recall that such a cycle is of the form $(v, v^\star, u^\star, u, v)$, where $(v, u) \in E(\widehat{G})$. Since U is a vertex cover of $\widehat{G}$, it holds that $U \cap \{v, u\} \neq \emptyset$, which implies that $Q \cap \{(v, v^\star), (u, u^\star)\} \neq \emptyset$.

In the reverse direction, let Q be a solution to (G', col', k'). Recall that for each edge $(v, u) \in E(G')$, where either $v, u \in V(\widehat{G})$ or $v, u \in V^\star$, G' contains exactly one monochromatic cycle of this form. Therefore, if Q contains an edge of the form (v, u) or of the form $(v^\star, u^\star)$, such an edge can be replaced by the edge $(v, v^\star)$. Thus, we can assume that Q only contains edges of the form $(v, v^\star)$. Define U as the set of vertices $\{v \mid (v, v^\star) \in Q\} \subseteq V(\widehat{G})$. We claim that U is a vertex cover of $\widehat{G}$ of size at most $\widehat{k}$. Since $|U| \leq |Q|$, it holds that $|U| \leq \widehat{k}$. Now, recall that for each edge $(v, u) \in E(\widehat{G})$, G' contains a monochromatic cycle of the form $(v, v^\star, u^\star, u, v)$. Since Q is a solution to (G', col', k'), it holds that $Q \cap \{(v, v^\star), (u, u^\star)\} \neq \emptyset$, which implies that $U \cap \{v, u\} \neq \emptyset$. □

We get the following theorem and its proof follows from Lemma 11.3 and Lemma 11.4.

Theorem 11.2. Sim-FES *where* $\alpha = 3$ *is* NP-hard. *Furthermore, unless the* ETH *fails,* Sim-FES *when* $\alpha = 3$ *cannot be solved in time* $\mathcal{O}^*(2^{o(k)})$.

11.3 Tight Lower Bounds for Simultaneous Feedback Edge Set

We show that Sim-FES parameterized by k is W[2]-hard hard when $\alpha = \mathcal{O}(|V(G)|)$ and W[1]-hard hard when $\alpha = \mathcal{O}(\log(|V(G)|))$. The approach we follow is similar to the one presented in Chapter 10.

11.3.1 W[2]-hardness of Sim-FES when $\alpha = \mathcal{O}(|V(G)|)$

We give a reduction from Hitting Set (HS) problem to Sim-FES where $\alpha = \mathcal{O}(|V(G)|)$. Let $(U = \{u_1, \ldots, u_{|U|}\}, \mathcal{F} = \{F_1, \ldots, F_{|\mathcal{F}|}\}, k)$ be an instance of HS, where $\mathcal{F} \subseteq 2^U$, which asks whether there exists a subset $S \subseteq U$ of size at most k such that for all $F \in \mathcal{F}$, $S \cap F \neq \emptyset$. It is known that HS parameterized by k is W[2]-hard (see, e.g., [CFK+15]). Thus, to prove the result, it is sufficient to construct (in polynomial time) an instance of the form $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ of Sim-FES that is equivalent to $(U, \mathcal{F}, k)$, where $\alpha = \mathcal{O}(|V(G)|)$. We construct a graph G such that $V(G) = \mathcal{O}(|U||\mathcal{F}|)$ and the number of colors used will be $\alpha = |\mathcal{F}|$. The intuitive idea is to have one edge per element in the universe which is colored with all the indices of sets in the family $\mathcal{F}$ that contains the element and for each $F_i \in \mathcal{F}$ creating a unique monochromatic cycle

with color i which passes through all the edges corresponding to the elements it contain. We explain the reduction formally in the next paragraph.

Without loss of generality we assume that each set in $\mathcal{F}$ contains at least two elements from U. The instance $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ is constructed as follows. Initially, $V(G) = E(G) = \emptyset$. For each element $u_i \in U$, insert two new vertices into $V(G)$, v_i and w_i, add the edge (v_i, w_i) into $E(G)$ and let $\{j \mid F_j \in \mathcal{F}, u_i \in F_j\}$ be its color-set. Now, for all $1 \leq i < j \leq |U|$ and for all $1 \leq t \leq |\mathcal{F}|$ such that $u_i, u_j \in F_t$ and $\{u_{i+1}, \ldots, u_{j-1}\} \cap F_t = \emptyset$, perform the following operation: add a new vertex into $V(G)$, $s_{i,j,t}$, add the edges $(w_i, s_{i,j,t})$ and $(s_{i,j,t}, v_j)$ into $E(G)$ and let their color-set be $\{t\}$. Moreover, for each $1 \leq t \leq |\mathcal{F}|$, let u_i and u_j be the elements with the largest and smallest index contained in F_t, respectively, and perform the following operation: add a new vertex into $V(G)$, $s_{i,j,t}$, add the edges $(w_i, s_{i,j,t})$ and $(s_{i,j,t}, v_j)$ into $E(G)$, and let their color-set be $\{t\}$. Observe that $|V(G)| = \mathcal{O}(|U||\mathcal{F}|)$ and that $\alpha = |\mathcal{F}|$. Therefore, $\alpha = \mathcal{O}(|V(G)|)$. It remains to show that the instances (G, col, k) and $(U, \mathcal{F}, k)$ are equivalent. By construction, each monochromatic cycle in G is of the form $(v_{i_1}, w_{i_1}, s_{i_1,i_2,t}, v_{i_2}, w_{i_2}, s_{i_2,i_3,t}, \ldots, v_{i_{|F_t|}}, w_{i_{|F_t|}}, s_{i_{|F_t|},i_1,t}, v_{i_1})$, where $\{u_{i_1}, u_{i_2}, \ldots, u_{i_{|F_t|}}\} = F_t \in \mathcal{F}$, and for each set $F_t \in \mathcal{F}$, G contains exactly one such monochromatic cycle.

Lemma 11.5. *$(U, \mathcal{F}, k)$ is a yes instance of* HS *if and only if $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ is a yes instance of* Sim-FES.

Proof. In the forward direction, let S be a solution to $(U, \mathcal{F}, k)$. Define Q as the set of edges $\{\{v_i, w_i\} : u_i \in S\} \subseteq E(G)$. We claim that Q is a solution to (G, k, col). Since $|Q| = |S|$, it holds that $|Q| \leq k$. Now, consider a monochromatic cycle C in G. Recall that this cycle is of the form $v_{i_1} - w_{i_1} - s_{i_1,i_2,t} - v_{i_2} - w_{i_2} - s_{i_2,i_3,t} - \cdots - v_{i_{|F_t|}} - w_{i_{|F_t|}} - s_{i_{|F_t|},i_1,t} - v_{i_1}$, where $\{u_{i_1}, u_{i_2}, \ldots, u_{i_{|F_t|}}\} = F_t \in \mathcal{F}$. In particular, observe that $\{\{v_i, w_i\} : u_i \in F_t\} \subseteq E(C)$. Since S is is a hitting set of $\mathcal{F}$, it holds that $S \cap F_t \neq \emptyset$. This implies that $Q \cap \{\{v_i, w_i\} : u_i \in F_t\} \neq \emptyset$, and therefore Q is a solution of $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$.

In the reverse direction, let Q be a solution to $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$. By the form of each monochromatic cycle in G, if Q contains an edge that includes a vertex of the form $s_{i,j,t}$, such an edge can be replaced by the edge $\{v_i, w_i\}$. Thus, we can assume that Q only contains edges of the form $\{v_i, w_i\}$. Define S as the set of elements $\{u_i : \{v_i, w_i\} \in Q\} \subseteq U$. We claim that S is a solution to $(U, \mathcal{F}, k)$. Since $|S| \leq |Q|$, it holds that $|S| \leq k$. Now, recall that for each set $\{u_{i_1}, u_{i_2}, \ldots, u_{i_{|F_t|}}\} = F_t \in \mathcal{F}$, G contains a monochromatic cycle of the form $v_{i_1} - w_{i_1} - s_{i_1,i_2,t} - v_{i_2} - w_{i_2} - s_{i_2,i_3,t} - \cdots - v_{i_{|F_t|}} - w_{i_{|F_t|}} - s_{i_{|F_t|},i_1,t} - v_{i_1}$. Since Q is a solution of $(G, k, \mathsf{col} : E(G) \to 2^{[\alpha]})$, it holds that $Q \cap \{\{v_i, w_i\} : u_i \in F_t\} \neq \emptyset$. This implies that $S \cap F_t \neq \emptyset$. □

Theorem 11.3. Sim-FES *parameterized by k, when $\alpha = \mathcal{O}(|V(G)|)$, is* W[2]-hard.

11.3.2 W[1]-hardness of Sim-FES when $\alpha = \mathcal{O}(\log |V(G)|)$

We modify the reduction given in the proof of Theorem 11.3 to show that when $\alpha = \mathcal{O}(\log |V(G)|)$, Sim-FES is W[1]-hard with respect to the parameter k. This result implies that the dependency on α of our $\mathcal{O}((2^{\mathcal{O}(\alpha)})^k n^{\mathcal{O}(1)})$-time algorithm for Sim-FES

is optimal in the sense that it is unlikely that there exists an $\mathcal{O}((2^{o(\alpha)})^k n^{\mathcal{O}(1)})$-time algorithm for this problem.

We give a reduction from Partitioned Hitting Set (PHS), to Sim-FES where $\alpha = \mathcal{O}(\log |V(G)|)$. Recall that an input of PHS consists of a universe U, a collection $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{|\mathcal{F}|}\}$, where each $\mathcal{F}_i$ is a family of *disjoint* subsets of U, and an integer k. The goal is to decide the existence of a set $S \subseteq U$ of size at most k such that for all $f \in (\bigcup_{i \in [|\mathcal{F}|]} \mathcal{F}_i)$, $S \cap f \neq \emptyset$. The special case of PHS where $|\mathcal{F}| = \mathcal{O}(\log(|U| \cdot |(\cup_{i \in [|\mathcal{F}|]} \mathcal{F}_i)|))$ is W[1]-hard when parameterized by k (see Chapter 10, Section 10.3). Thus, to prove the theorem, it is sufficient to construct (in polynomial time) an instance of the form $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ of Sim-FES that is equivalent to $(U, \mathcal{F}, k)$, where $\alpha = \mathcal{O}(\log |V(G)|)$. The construction of the graph G is exactly similar to the one in Theorem 11.3. But instead of creating a unique monochromatic cycle with a color i for each $f_i \in \cup_{i \in [|\mathcal{F}|]} \mathcal{F}_i$, for each $\mathcal{F}_i \in \mathcal{F}$ we create $|\mathcal{F}_i|$ vertex disjoint cycles of same color i. Since for each $\mathcal{F}_i \in \mathcal{F}$ the sets in $\mathcal{F}_i$ are pairwise disjoint, the correctness is guaranteed. Formal description of the reduction is given below.

Without loss of generality we assume that each set in $\cup_{i \in [|\mathcal{F}|]} \mathcal{F}_i$ contains at least two elements from U. The instance $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ is constructed as follows. Initially, $V(G) = E(G) = \emptyset$. For each element $u_i \in U$, insert two new vertices v_i and w_i into $V(G)$, and add the edge (v_i, w_i) to $E(G)$ with its color-set being $\{j \mid \mathcal{F}_j \in \mathcal{F}, u_i \in (\cup_{F \in \mathcal{F}_j} F)\}$. Now, for all $1 \leq i < j \leq |U|$ and for all $1 \leq t \leq |\mathcal{F}|$ such that there exists $f \in \mathcal{F}_t$ satisfying $u_i, u_j \in f$ and $\{u_{i+1}, \ldots, u_{j-1}\} \cap f = \emptyset$, perform the following operation: add a new vertex $s_{i,j,t}$ into $V(G)$, add the edges $\{w_i, s_{i,j,t}\}$ and $\{s_{i,j,t}, v_j\}$ into $E(G)$ with both of its color-set being $\{t\}$. Moreover, for each $1 \leq t \leq |\mathcal{F}|$ and $f \in F_t$, let u_i and u_j be the elements with the largest and smallest index contained in f, respectively, we perform the following operation: add a new vertex into $V(G)$, $s_{i,j,t}$, add the edges $\{w_i, s_{i,j,t}\}$ and $\{s_{i,j,t}, v_j\}$ into $E(G)$, and let their color-set be $\{t\}$. Observe that $|V(G)| = \mathcal{O}(|U||(\cup_{i \in [|\mathcal{F}|]} \mathcal{F}_i)|)$ and that $\alpha = |\mathcal{F}|$. Since $|\mathcal{F}| = \mathcal{O}(\log(|U| \cdot |(\cup_{i \in [|\mathcal{F}|]} \mathcal{F}_i)|))$, we have that $\alpha = \mathcal{O}(\log |V(G)|)$. Since the sets in each family F_i are disjoint, the construction implies that each monochromatic cycle in G is of the form $v_{i_1} - w_{i_1} - s_{i_1,i_2,t} - v_{i_2} - w_{i_2} - s_{i_2,i_3,t} - \cdots - v_{i_{|f|}} - w_{i_{|f|}} - s_{i_{|f|},i_1,t} - v_{i_1}$, where $\{u_{i_1}, u_{i_2}, \ldots, u_{i_{|F_t|}}\} = f$ for a set $f \in \mathcal{F}_t \in \mathcal{F}$, and for each set $f \in \mathcal{F}_t \in \mathcal{F}$, G contains a monochromatic cycle of this form. By using the arguments similar to one in the proof of Lemma 11.5, we get that the instances $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ and $(U, \mathcal{F}, k)$ are equivalent. Hence we get the following theorem.

Theorem 11.4. Sim-FES *parameterized by* k, *when* $\alpha = \mathcal{O}(\log |V(G)|)$ *is* W[1]-hard.

11.4 Kernel for Simultaneous Feedback Edge Set

In this section, we give a kernel for Sim-FES with $\mathcal{O}((k\alpha)^{\mathcal{O}(\alpha)})$ vertices. We start by applying preprocessing rules similar in spirit to the ones used to obtain a kernel for Feedback Vertex Set, but it requires subtle differences due to the fact that we handle a problem where edges rather than vertices are deleted, as well as the fact that the edges are colored (in particular, each edge in Sim-FES has a color-set, while each vertex in Sim-FVS is uncolored). We obtain an approximate solution by computing a spanning tree per color. We rely on the approximate solution to bound the number of vertices whose degree in certain subgraphs of G is not equal to 2. Then, the number of the

remaining vertices is bounded by adapting the approach we followed for the Sim-FVS problem.

Let $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ be an instance of Sim-FES. It is easy to verify that the following reduction rules are correct when applied exhaustively in the order in which they are listed. We note that the resulting instance can contain multiple edges.

Reduction Rule 11.1. *If $k < 0$, return that $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ is a no instance.*

Reduction Rule 11.2. *If for all $i \in [\alpha]$, G_i is acyclic, return that $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ is a yes instance.*

Reduction Rule 11.3. *If there is a self-loop at a vertex $v \in V(G)$, then remove v from G and decrement k by 1.*

Reduction Rule 11.4. *If there exists an isolated vertex in G, then remove it.*

Reduction Rule 11.5. *If there exists $i \in [\alpha]$ and an edge whose color-set contains i but it does not participate in any cycle in G_i, remove i from its color-set. If the color-set becomes empty, remove the edge.*

Reduction Rule 11.6. *If there exists $i \in [\alpha]$ and a vertex v of degree exactly two in G, remove v and connect its two neighbors by an edge whose color-set is the same as the color-set of the two edges incident to v (we prove in Lemma 11.6 that the color set of two edges are same).*

Lemma 11.6. *Reduction Rule 11.6 is safe.*

Proof. Let G be a graph with coloring function $\mathsf{col} : E(G) \to 2^{[\alpha]}$. Let v be a vertex in $V(G)$ such that v has total degree 2 in G. We have applied Reduction Rules 11.1 to 11.5 exhaustively (in that order). Therefore, when Reduction Rule 11.6 is applied, the edges incident to v have the same color-set say i, since otherwise Reduction Rule 11.5 would be applicable. Let u, w be the neighbors of v in G_i, where $i \in [\alpha]$. Consider the graph G' with vertex set as $V(G) \setminus \{v\}$ and edge set as $E(G') = (E(G) \setminus \{(v,u),(v,w)\}) \cup \{(u,w)\}$ and coloring function col' such that $\mathsf{col}'(u,w) = \mathsf{col}(u,w) \cup \{i\}$ and for all other edges $e \in E(G') \setminus \{(u,w)\}$, $\mathsf{col}'(e) = \mathsf{col}(e)$. We show that (G, col, k) is a *yes* instance of Sim-FES if and only if (G', col', k) is a *yes* instance of Sim-FES.

In the forward direction, let S be a solution to Sim-FES in G of size at most k. Suppose S is not a solution in G'. Then, there is a cycle C in G'_t, for some $t \in [\alpha]$. Note that C cannot be a cycle in G'_j as $G'_j = G_j$, for $j \in [\alpha] \setminus \{i\}$. Therefore C must be a cycle in G'_i. All the cycles C' not containing the edge (u,w) are also cycles in G_i and therefore S must contain some edge from C'. It follows that C must contain the edge (u,w). Note that the edges $(E(C) \setminus \{(u,w)\}) \cup \{(v,u),(w,v)\}$ form a cycle in G_i. Therefore S must contain an edge from $E(C) \cup \{(v,u),(w,v)\}$. We consider the following cases:

- Case 1: $(v,u),(w,v) \notin S$. In this case, S must contains an edge from $E(C) \setminus \{(u,w)\}$. Hence, S is a solution in G'.
- Case 2: At least one of $(v,u),(w,v)$ belongs to S, say $(v,u) \in S$. Let $S' = (S \setminus \{(v,u)\}) \cup \{(u,w)\}$. Observe that S' intersects all cycles in G'_i. Therefore S' is a solution in G' of size at most k.

In the reverse direction, consider a solution S to Sim-FES in G'. If S is a solution in G we have a proof of the claim. Therefore, we assume that S is not a solution in G. S intersects all cycles in G_j, since $G_j = G'_j$, for all $j \in [\alpha] \setminus \{i\}$. All cycles in G_i not containing v are also cycles in G'_i and therefore S intersects all such cycles.

A cycle in G_i containing v must contain u and w (v is a degree-two vertex in G_i). We assume that there is a cycle C containing v in G such that S does not intersect C (otherwise S is a solution in G). Note that in G'_i we added an edge (u, w) and we keep multi-edges. Corresponding to each copy of (u, w) we have a cycle in G' with edges $(E(C) \setminus \{(v, u), (v, w)\}) \cup \{(u, w)\}$. Therefore, S must contain all copies of (u, w). We create a solution S' by replacing a copy (with same color as (v, u)) of $(u, w) \in S$ by (v, u). We claim that S' is a solution in G. S' intersects all the cycles in G containing v. Observe that all cycles in G not containing v are also cycles in G' and they do not contain the deleted copy of the edge (u, w) from S. Therefore, they are intersected by S'. Therefore, S' is a solution in G of size at most k. □

We apply Reduction Rule 11.1 to 11.6 exhaustively (in that order). The safeness of Reduction Rules 11.1 to 11.5 are easy to see. Lemma 11.6 proves the safeness Reduction Rule 11.6. After this, we follow the approach similar to that in Sim-FVS to bound the size of the instance, which gives us the following theorem.

Theorem 11.5. Sim-FES *admits a kernel with* $(k\alpha)^{\mathcal{O}(\alpha)}$ *vertices.*

Proof. Let $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, k)$ be an instance of Sim-FES where none of the reduction rules are applicable. For each graph G_i, we compute a spanning forest, F_i, maximizing $|E(F_i)|$. Let $X_i = E(G_i) \setminus E(F_i)$. If $|X_i| > k$, the instance is a no-instance. Thus, we can assume that for each $i \in [\alpha]$, X_i contains at most k edges. Let $X = \cup_{i=1}^{\alpha} X_i$ denote the union of the sets X_i. Clearly, $|X| \leq k\alpha$. Let U denote the subset of $V(G)$ that contains the vertices incident to at least one edge in X. Since Reduction Rule 11.5 is not applicable, therefore $|U| \leq 2k\alpha$. Thus, the number of leaves in each $G_i - X$ is bounded by $2k\alpha$. Accordingly, the number of vertices in each $G_i - X$ whose degree is at least 3 is bounded by $2k\alpha$. It remains to bound the number of vertices that are not incident to any edge in X and whose degree in each G_i is 0 or 2 (their degree in G is at least 3). Let T be the set of vertices in G which is either a leaf or a degree 3 vertex in some G_i, for $i \in [\alpha]$. Denote the set of vertices which are not in T, not incident to any edge in X and whose degree in G_i is 2 by D_i. Let $\mathcal{P}_i$ denote the set of paths in G_i, for $i \in [\alpha]$, whose internal vertices belong to D_i and whose first and last vertices do not belong to D_i. Moreover, let $D = \cup_{i=1}^{\alpha} D_i$ and $\mathcal{P} = \cup_{i=1}^{\alpha} \mathcal{P}_i$. Observe that for $i \in [\alpha]$, $|\mathcal{P}_i| \leq 4k\alpha$ and $|\mathcal{P}| \leq 4k\alpha^2$.

To obtain the desired kernel, it remains to show that $|D| = \mathcal{O}((k\alpha)^{\mathcal{O}(\alpha)})$. For each edge $e \in E(G)$, let $\mathcal{P}[e]$ be the set of paths in $\mathcal{P}$ to which e belongs. Each edge belongs to at most one path in each $\mathcal{P}_i$, for any $i \in [\alpha]$. For each $v \in D$, by $E(v)$ we denote the set of edges incident to v in G. Observe that each vertex in D is incident to at most 2α edges. For each vertex $v \in D$, there are at most $(4k\alpha + 1)^{\alpha}$ options of choosing to which paths in $\mathcal{P}$ the vertex v belongs. Note that here the extra additive one is to include the case when a vertex does not belong to any path in a color class. the edges incident to v. Thus, if $|D| \geq 3(4k\alpha + 1)^{\alpha}$, Thus, there exists a constant c such that if $|D| > (k\alpha)^{c\alpha}$, then D contains (at least) three vertices, r, s and t, such that for all $q, p \in \{r, s, t\}$, there is a bijection $f : E(q) \to E(p)$ such that $\mathcal{P}[e] = \mathcal{P}[f(e)]$ for all $e \in E(q)$. In particular,

if $|D| > (k\alpha)^{c\alpha}$, then D contains two non-adjacent vertices, v and u, such that there is a bijection $f : E(v) \to E(u)$ satisfying $\mathcal{P}[e] = \mathcal{P}[f(e)]$ for all $e \in E(v)$. In this case, it is not necessary insert any edge $e \in E(v)$ into a solution, since it has the same affect as inserting the edge $f(e)$. Thus, we can remove the vertex v, and for each two neighbors of v, x and y, and for each color $i \in [\alpha]$ such that $i \in \mathsf{col}(\{v, x\}) \cap \mathsf{col}(\{v, y\})$, we insert an edge (x, y) whose color-set is $\{i\}$. After an exhaustive application of this operation (as well as Reduction Rules 11.1 to 11.6), we obtain the desired bound on $|D|$, which concludes the proof of Theorem 11.5. □

11.5 Max-Sim Acyclic-Subgraph

In this section, we design an algorithm for Max-Sim Acyclic-Subgraph. Let $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, q)$ be an input to Max-Sim-Subgraph. A set $F \subseteq E(G)$ such that for all $i \in [\alpha]$, $G[F_i]$ is acyclic is called *simultaneous forest*. Here, $F_i = \{e \in F \mid i \in \mathsf{col}(e)\}$, denotes the subset of edges of F which has the integer i in its image when the function col is applied to it. We will solve Max-Sim-Subgraph by reducing to an equivalent instance of the α-Matroid Parity problem and then using the algorithm for the same.

We start by giving a construction that reduces the Max-Sim-Subgraph to α-Matroid Parity. Let $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, q)$ be an input to Max-Sim-Subgraph. Given, $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, q)$, for $i \in [\alpha]$, recall that by G_i we denote the graph with the vertex set $V(G_i) = V(G)$ and the edge set $E(G_i) = \{e^i \mid e \in E(G) \text{ and } i \in \mathsf{col}(e)\}$. For each edge $e \in E(G)$, we will have its distinct copy in G_i if $i \in \mathsf{col}(e)$. Thus, for each edge $e \in E(G)$, by $\mathsf{Original}(e)$ we denote the set of edges $\{e^j \mid j \in \mathsf{col}(e)\}$. On the other hand for each edge $e \in E(G)$, by $\mathsf{Fake}(e)$ we denote the set of edges $\{e^j \mid j \in [\alpha] - \mathsf{col}(e)\}$. Finally, for each edge $e \in E(G)$, by $\mathsf{Copies}(e)$ we denote the set $\mathsf{Original}(e) \cup \mathsf{Fake}(e)$. Let $M_i = (E_i, \mathcal{I}_i)$ denote the graphic matroid on G_i. That is, edges of G_i forms the universe E_i and $\mathcal{I}_i$ contains, $S \subseteq E(G_i)$ such that $G_i[S]$ forms a forest. By Proposition 2.5 we know that graphic matroids are representable over any field and given a graph G one can find the corresponding representation matrix in time polynomial in $|V(G)|$. Let A_i denote the linear representation of M_i. That is, A_i is a matrix over $\mathbb{F}_2$, where the set of columns of A_i are denoted by $E(G_i)$. In particular, A_i has dimension $d \times |E(G_i)|$, where $d = \mathsf{rank}(M_i)$. A set $X \subseteq E(G_i)$ is independent (that is $X \in \mathcal{I}_i$) if and only if the corresponding columns are linearly independent over $\mathbb{F}_2$. Let $\mathsf{Fake}(G)$ denote the set of edges in $\bigcup_{e \in E(G)} \mathsf{Fake}(e)$. Furthermore, let $\tau = |\mathsf{Fake}(G)| = \sum_{e \in E(G)} |\mathsf{Fake}(e)|$. Let $M_{\alpha+1}$ be the uniform matroid over $\mathsf{Fake}(G)$ of rank τ. That is, $E_{\alpha+1} = \mathsf{Fake}(G)$ and $M_{\alpha+1} = U_{\tau,\tau}$. Let I_τ denote the identity matrix of dimension $\tau \times \tau$. Observe that, $A_{\alpha+1} = I_\tau$ denotes the linear representation of $M_{\alpha+1}$ over $\mathbb{F}_2$. Notice that $E_i \cap E_j = \emptyset$ for all $1 \leq i \neq j \leq \alpha + 1$. Let M denote the direct sum of $M_1 \oplus \cdots \oplus M_{\alpha+1}$ with its representation matrix being A_M.

Now we are ready to define an instance of α-Matroid Parity. The ground set is the columns of A_M, which is indexed by edges in $\bigcup_{e \in E(G)} \mathsf{Copies}(e)$. Furthermore, the ground set is partitioned into $\mathsf{Copies}(e)$, $e \in E(G)$, which are called blocks. The main technical lemma of this section on which the whole algorithm is based is the following.

Lemma 11.7. *Let $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, q)$ be an instance of* Max-Sim-Subgraph. *Then G has a simultaneous forest of size q if and only if $(A_M, \biguplus_{e \in E(G)} \mathsf{Copies}(e), q)$ is a*

yes instance of α-Matroid Parity. *Furthermore, given* $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, q)$ *we can obtain an instance* $(A_M, \biguplus_{e \in E(G)} \mathsf{Copies}(e), q)$ *in polynomial time.*

Proof. We first show the forward direction of the proof. Let F be a simultaneous forest of size q. Then we claim that the columns corresponding to $S = \bigcup_{e \in F} \mathsf{Copies}(e)$ form an independent set in M and furthermore, it is the union of q blocks. That is, we need to show that the columns corresponding to $S = \bigcup_{e \in F} \mathsf{Copies}(e)$ are linearly independent in A_M over $\mathbb{F}_2$. By the definition of direct sum and its linear representation, it reduces to showing that F is linearly independent if and only if $F \cap E_i \in \mathcal{I}_i$ for all $i \leq \alpha + 1$. Since F is a simultaneous forest of size q, we have that $G[F_i]$, $F_i = \{e \in F \mid i \in \mathsf{col}(e)\}$, is a forest. Hence, this implies that $F^i = \{e^i \mid e \in F_i\}$ forms a forest in G_i. This takes care of all the edges in $\cup_{e \in F} \mathsf{Original}(e)$. Now let $S^* = S - \cup_{i \in [\alpha]} F^i = \cup_{e \in F} \mathsf{Fake}(e) = F^{\alpha+1}$. However, S^* is a subset of $U_{\tau,\tau}$ and thus is an independent set since $|S^*| \leq \tau$. This completes the proof of the forward direction.

Now we show the reverse direction of the proof. Since, $(A_M, \biguplus_{e \in E(G)} \mathsf{Copies}(e), q)$ is a *yes* instance of α-Matroid Parity, there exists an independent set, say S, that is the union of q blocks. By construction S corresponds to union of the sets $\mathsf{Copies}(e)$ for some q edges in G. Let these edges be $F = \{e_1, \ldots, e_q\}$. We claim that F is a simultaneous forest of size q. Towards this, we need to show that $G[F_i]$, where $F_i = \{e \in F \mid i \in \mathsf{col}(e)\}$, is a forest. This happens if and only if $F^i = \{e^i \mid e \in F_i\}$ forms a forest in G_i. However, we know that the columns corresponding to F_i are linearly independent in M_i and in particular in A_i – the linear representation of graphic matroid of G_i. This shows that F_i forms a forest in G_i and hence $G[F_i]$ is a forest. This completes the equivalence proof.

Finally, it easily follows from the discussion preceding the lemma that given $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, q)$ we can obtain an instance $(A_M, \biguplus_{e \in E(G)} \mathsf{Copies}(e), q)$ in time polynomial in $|V(G)|$. This completes the proof of the lemma. □

We will use the polynomial time reduction provided in Lemma 11.7 to get the desired FPT algorithm for Max-Sim-Subgraph. Towards this will use the following FPT result regarding α-Matroid Parity for our FPT as well as for an exact exponential time algorithm.

Given an instance $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, q)$ of Max-Sim-Subgraph we first apply Lemma 11.7 and obtain an instance $(A_M, \biguplus_{e \in E(G)} \mathsf{Copies}(e), q)$ of α-Matroid Parity and then apply Proposition 2.7 to obtain the following result.

Theorem 11.6. Max-Sim-Subgraph *can be solved in time* $\mathcal{O}(2^{\omega q \alpha} |V(G)|^{\mathcal{O}(1)})$.

Let $(G, \mathsf{col} : E(G) \to 2^{[\alpha]}, q)$ be an instance of Max-Sim-Subgraph. Observe that q is upper bounded by $\alpha(|V(G)| - 1)$. Thus, as a corollary to Theorem 11.6 we get an exact algorithm for finding the largest sized simultaneous acyclic subgraph, running in time $\mathcal{O}(2^{\omega n \alpha^2} |V(G)|^{\mathcal{O}(1)})$.

Chapter 12

Simultaneous FVS/OCT

In this chapter, we investigate the complexity of Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion in two settings. First, we consider the problem with $\mathcal{F}_1$ being the family of all bipartite graphs and $\mathcal{F}_2 = \mathcal{F}_3 = \ldots = \mathcal{F}_\alpha$ being the family of all forests. We call this problem Simultaneous FVS/OCT (or Sim-FVS/OCT for short) and define it as follows.

Simultaneous FVS/OCT (Sim-FVS/OCT)	**Parameter:** k and α
Input: An α-edge-colored graph $G = (V, E_1, \ldots, E_\alpha)$ and an integer k.	
Question: Is there a set $S \subseteq V(G)$ of size at most k such that $G_1 - S$ is a bipartite graph and $G_2 - S$, …, $G_\alpha - S$ are acyclic?	

We call a solution S to the Sim-FVS/OCT problem a *sim-fvs-oct*. We design an algorithm that, given an instance $(G = (V, E_1, \ldots, E_\alpha), k)$ of Sim-FVS/OCT, runs in time $\mathcal{O}^\star(k^{\mathsf{poly}(\alpha,k)})$ and either computes a sim-fvs-oct in G of size at most k or correctly concludes that such a set does not exist.

In the second setting, we consider the Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion problem where $\mathcal{F}_1 = \ldots = \mathcal{F}_\alpha$ is the family of all bipartite graphs. We call this problem Simultaneous OCT (or Sim-OCT for short) and define it as follows.

Simultaneous OCT (Sim-OCT)	**Parameter:** k and α
Input: An α-edge-colored graph $G = (V, E_1, \ldots, E_\alpha)$ and an integer k.	
Question: Is there a set $S \subseteq V(G)$ of size at most k such that $G_i - S$ is bipartite, for each $i \in [\alpha]$?	

We refer to a solution S to the Sim-OCT problem as a *sim-oct*. In this chapter, our second (and rather surprising) result is a negative answer to an open question of Agrawal et al. [ALMS16]. We show that, even for $\alpha = 2$, the problem Sim-OCT is W[1]-hard. To prove this result, we first reduce the well-known Multi-Colored Clique problem [FHRV09] to an auxiliary problem we call Simultaneous Cut (or Sim-Cut). Sim-Cut is a natural generalization of the classical (s,t)-Cut problem to edge-colored graphs. Finally, we show that Sim-Cut can be reduced to Sim-OCT. Notice that W[1]-hardness of Sim-OCT implies that Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion problem with at least two of the families being the family of bipartite graphs is W[1]-hard.

Overview of the algorithm. Note that for any fixed k and α, our algorithm for solving the Sim-FVS/OCT problem runs in polynomial time. The said algorithm can be broken down into four stages, three of which are reductions to auxiliary problems.

Initially, as was first proposed by Ye [Ye15], we use the notion of compact representations of feedback vertex sets (see Section 12.1 for formal definitions) to reduce Sim-FVS/OCT into $2^{\mathcal{O}(\alpha k)}$ instances of the Colorful OCT problem, which is formally defined as follows. We note that, in any reduced instance, ℓ will be bounded above by αk.

Colorful OCT	**Parameter:** k and ℓ
Input: A graph G, integers k and ℓ, and a grouping $\mathcal{P}$ of the vertices of G into (not necessarily distinct) sets $\{P_1, \ldots, P_\ell\}$.	
Question: Is there a set $S \subseteq V(G)$ of size at most k such that $G - S$ is a bipartite graph and $S \cap P_i \neq \emptyset$, for each $i \in [\ell]$?	

Intuitively, compact representations give us a partition of a vertex subset of the graph into sets such that picking one vertex from each part is "guaranteed" to constitute a feedback vertex set of each graph G_i, $2 \leq i \leq \alpha$. As such, we are able to encode the feedback vertex set "side" of the Sim-FVS/OCT problem (via the reduction) as colors on the vertices (i.e. different sets in $\mathcal{P}$ represent different colors for each vertex) and focus on a "colored" variant of Odd Cycle Transversal. Naturally, the second stage is to solve the Colorful OCT problem within the claimed running time. To do so, we reduce an instance of Colorful OCT to an instance of the compression variant of the problem, i.e. Colorful OCT Compression. This problem assumes an odd cycle transversal of size at most k as part of the input. Note that finding an odd cycle transversal of size at most k in a graph can be accomplished using the fixed-parameter tractable algorithms for Odd Cycle Transversal parameterized by the solution size [IOY14, RS14], both of which run in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time.

Colorful OCT Compression	**Parameter:** k and ℓ
Input: A graph G, integers k and ℓ, a grouping $\mathcal{P}$ of the vertices of G into (not necessarily distinct) sets $\{P_1, \ldots, P_\ell\}$, and a set $O \subseteq V(G)$ of size at most k such that $G - O$ is bipartite.	
Question: Is there a set $S \subseteq V$ of size at most k such that $G - S$ is a bipartite graph and $S \cap P_i \neq \emptyset$, for each $i \in [\ell]$?	

Now, to solve an instance of Colorful OCT Compression, we reduce it into $2^{\mathcal{O}(k)}$ instances of yet another problem, namely Colorful Separator. This reduction is in many ways similar to the iterative compression algorithm for solving the Odd Cycle Transversal problem [CFK+15, GGH+06, RSV04].

Colorful Separator	**Parameter:** k and ℓ
Input: A graph G, integers k and ℓ, a grouping $\mathcal{P}$ of the vertices of G into (not necessarily distinct) sets $\{P_1, \ldots, P_\ell\}$, and vertices s and t in $V(G)$.	
Question: Is there an (s,t)-separator $S \subseteq V(G) \setminus \{s,t\}$ such that $\lvert S\rvert \leq k$ and $S \cap P_i \neq \emptyset$, for each $i \in [\ell]$?	

Finally, and arguably the most technical part of the algorithm, is to show how to solve an instance of Colorful Separator. We will in fact solve a much more general problem, which we define in Section 12.3. Our two main ingredients are a dynamic programming routine and a generalization of the concept of important separators, which has been recently defined to design parameterized algorithms for several "cut" problems [GRS17, LR12, LRS16]. We note that an alternative algorithm for solving Col-

orful Separator can be obtained by applying the treewidth reduction result of Marx et al. [MOR13]. However, a "simple" application of this result would give an algorithm with a worse running time (double exponential).

12.1 Preliminaries

For a graph G and set $X \subseteq V(G)$, we refer to a partition (A, B) of X as a *valid bipartition* of $G[X]$ if $G[A]$ and $G[B]$ are both edgeless graphs. We refer to a valid bipartition of $V(G)$ as a valid bipartition of the graph G.

Definition 12.1. Let G be a graph and X and Y be disjoint subsets of $V(G)$. A vertex set S disjoint from $X \cup Y$ is called an *(X,Y)-separator* if there is no (X, Y)-path in $G - S$. We denote by $R_G(X, S)$ the set of vertices of $G - S$ reachable from vertices of X via paths and by $NR_G(X, S)$ the set of vertices of $G - S$ *not* reachable from vertices of X.

We remark that it is not necessary that Y and $N(X)$ be disjoint in the above definition. If these sets do intersect, then there is no (X,Y)-separator in the graph.

Definition 12.2. [GGH$^+$06] A *compact representation* of a set $\mathcal{S}$ of minimal feedback vertex sets of a graph G is a collection $\mathcal{C}$ of pairwise disjoint subsets of $V(G)$ such that choosing exactly one vertex from every set in $\mathcal{C}$ results in a minimal feedback vertex set for G that is in $\mathcal{S}$.

Lemma 12.1. [GGH$^+$06] *The set of all minimal feedback vertex sets of size at most k can be represented by a collection of compact representations of size $2^{\mathcal{O}(k)}$. Furthermore, given a graph $G = (V, E)$ and a feedback vertex set F for G of size $k + 1$, we can enumerate the compact representations of all minimal feedback vertex sets for G having size at most k in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time.*

12.2 From Simultaneous FVS/OCT to Colorful OCT

We first describe how to reduce an instance of Sim-FVS/OCT to $2^{\mathcal{O}(\alpha k)}$ instances of Colorful OCT. Note that since both Feedback Vertex Set [LRS16] and Odd Cycle Transversal [RS14, IOY14] can be solved in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time, we assume that, along with an instance $(G = (V, E_1, \ldots, E_\alpha), k)$, we are given sets $O, F_2, \ldots, F_\alpha \subseteq V(G)$ each of size at most k such that $G_1 - O$ is a bipartite graph and $G_i - F_i$, $2 \leq i \leq \alpha$, is acyclic (as otherwise we can safely conclude that the given instance is a *no*-instance).

Lemma 12.2. *There is an algorithm that, given an instance $(G = (V, E_1, \ldots, E_\alpha), k)$ of* Sim-FVS/OCT*, runs in time $\mathcal{O}^\star(2^{\mathcal{O}(\alpha k)})$ and returns a set of $2^{\mathcal{O}(\alpha k)}$ instances of* Colorful OCT *such that the original instance is a yes-instance if and only if at least one of the returned instances is a yes-instance.*

Proof. Armed with the sets F_i which are of size at most k, we apply the algorithm of Lemma 12.1 to each graph G_i, $2 \leq i \leq \alpha$, to obtain a set of compact representations $\mathbf{C}_i = \{\mathcal{C}_i^1, \mathcal{C}_i^2, \ldots\}$, $2 \leq i \leq \alpha$. Note that each $\mathbf{C}_i$ is of size $2^{\mathcal{O}(k)}$ and each $\mathcal{C}_i^j$ is of size at most k. The said algorithm runs in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time for each graph G_i. For each tuple

$\{\mathcal{C}_2^{j_2}, \ldots, \mathcal{C}_\alpha^{j_\alpha}\} \in \mathbf{C}_2 \times \ldots \times \mathbf{C}_\alpha$, we construct an instance $(G', \mathcal{P}, k', \ell)$ of Colorful OCT as follows. We let $G' = (V, E_1)$, $k' = k$, and $\ell = \sum_{i \in [\alpha]} |\mathcal{C}_i^j| \leq \alpha k$.

For each $\mathcal{C} \in \{\mathcal{C}_2^{j_2}, \ldots, \mathcal{C}_\alpha^{j_\alpha}\}$ and for each set $C \in \mathcal{C}$, we add a set $P \in \mathcal{P}$ and we let $P = C$. In other words, all vertices in C are added to P. Observe that $|\mathcal{C}| \leq k$. Since each $\mathbf{C}_i$ is of size $2^{\mathcal{O}(k)}$, it is easy to verify that the number of instances is in fact $2^{\mathcal{O}(\alpha k)}$. We now prove the correctness of the algorithm.

Assume that $(G = (V, E_1, \ldots, E_\alpha), k)$ is a *yes*-instance, and let S be a solution of size at most k. Note that S need not be a minimal fvs in G_i, $2 \leq i \leq \alpha$. However, for each $i \in \{2, \ldots, \alpha\}$, there exists a set $S' \subseteq S$ such that S' is a minimal fvs for G_i. Hence, by Definition 12.2 and Lemma 12.1, for every $i \in \{2, \ldots, \alpha\}$, there exists a $\mathcal{C}_i^j \in \mathbf{C}_i$ such that for all $C \in \mathcal{C}_i^j$ we have $S' \cap C \neq \emptyset$. Since we enumerate all compact representations and create one instance for each, we know that at least one instance $(G', \mathcal{P}, k', \ell)$ of Colorful OCT will correspond to the correct choice. The fact that S is a solution for $(G', \mathcal{P}, k', \ell)$ follows from the fact that S contains a minimal oct for G_1.

For the other direction, let S' be a solution for an instance $(G', \mathcal{P}, k', \ell)$ of Colorful OCT. Since S' is of size at most k, it is clearly an oct for G_1. Moreover, since S' must intersect every $P \in \mathcal{P}$, it follows from the definition of compact representations and our construction that S' is an fvs for G_i, $2 \leq i \leq \alpha$, as needed. □

We now focus on solving an instance $(G, \mathcal{P}, k, \ell)$ of Colorful OCT. Recall that we also have access to the set O which is an oct of G of size at most k. Our next step is to reduce $(G, \mathcal{P}, k, \ell)$ to an instance $(G, \mathcal{P}, O, k, \ell)$ of Colorful OCT Compression. The correctness of this reduction is immediate. The final piece in our sequence of reductions is to reduce $(G, \mathcal{P}, O, k, \ell)$ to $2^{\mathcal{O}(k)}$ instances of Colorful Separator. Before we state our final reduction, we need the following.

Definition 12.3. Let G be a graph, let O be an oct of G, let $X \subseteq O$, let $\mathcal{Q} = (L, R)$ be a valid bipartition of $G - O$, and let $\mathcal{W} = (A, B)$ be a partition of X. We define the graph $G^X_{\mathcal{Q},\mathcal{W}}$ as the graph obtained from G as follows. Add two new vertices s and t, make s adjacent to all vertices in $(N(A) \cap L) \cup (N(B) \cap R)$, and make t adjacent to all vertices in $(N(A) \cap R) \cup (N(B) \cap L)$. Finally, delete X.

Proposition 12.1. [CFK+15] *Let G be a graph, let X be an oct of G, and let $\mathcal{Q}$ be a valid bipartition of $G - X$. Then the following statements hold.*

(i) Let Y be an oct of G, $Z = X \setminus Y$, let $G' = G - (X \cap Y)$, and let $\mathcal{Q}' = \mathcal{Q}\big|_{V(G')}$. Then, there is a partition $\mathcal{Z} = (Z_1, Z_2)$ of Z such that $Y \setminus X$ is an (s,t)-separator in $G'^Z_{\mathcal{Q}',\mathcal{Z}}$.

(ii) Let $Z \subseteq X$, let $G' = G - Z$, and let $\mathcal{Q}' = \mathcal{Q}\big|_{V(G')}$. Let $\mathcal{B}$ be a valid bipartition of $X \setminus Z$. If Y is an (s,t)-separator in the graph $G'^{X \setminus Z}_{\mathcal{Q}',\mathcal{B}}$, then $Y \cup Z$ is an oct of G.

Lemma 12.3. *There is an algorithm that, given an instance $(G, \mathcal{P}, O, k, \ell)$ of* Colorful OCT Compression*, runs in time $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ and returns a set of $2^{\mathcal{O}(k)}$ instances of* Colorful Separator *such that the original instance is a yes-instance if and only if at least one of the returned instances is a yes-instance.*

Proof. First, we let $\mathcal{Q}$ be an arbitrary valid bipartition of $G - O$. Next, for each of the at most 3^k partitions of O into three sets X, Y, and Z, we make sure that the following conditions hold. We check that $O = X \cup Y \cup Z$, $G[X]$ and $G[Y]$ are edgeless ((X, Y) is a valid bipartition of $O \setminus Z$), and $P \not\subseteq X \cup Y$, for any $P \in \mathcal{P}$. If all conditions hold, we construct an instance $(G'^{X\cup Y}_{\mathcal{Q}',\mathcal{W}}, \mathcal{P}', s, t, k', \ell')$ of Colorful Separator where $G' = G - Z$, $\mathcal{Q}'$ is the bipartition of $G - O$ restricted to $V(G')$, $\mathcal{W}$ is the partition (X, Y), and the graph $G'^{X\cup Y}_{\mathcal{Q},\mathcal{W}}$ is obtained from G' by adding vertices s and t and deleting $X \cup Y$ (see Definition 12.3). We set $k' = k - |Z|$. It remains to show how to construct $\mathcal{P}' = \{P'_1, \ldots, P'_{\ell'}\}$ from $\mathcal{P} = \{P_1, \ldots, P_\ell\}$. Let $\mathcal{P}''$ be the subset of $\mathcal{P}$ consisting of all sets $P \in \mathcal{P}$ such that $P \cap Z \neq \emptyset$. For every $P \in \mathcal{P} \setminus \mathcal{P}''$, we assign a unique index $j \in [|\mathcal{P} \setminus \mathcal{P}''|]$ and set $P'_j = P$. It follows that $\ell' \leq \ell$.

We now claim that the given instance $(G, \mathcal{P}, O, k, \ell)$ is a *yes*-instance of Colorful OCT Compression if and only if at least one of the constructed 3^k instances $(G'^{X\cup Y}_{\mathcal{Q}',\mathcal{W}}, \mathcal{P}', s, t, k - |Z|, \ell')$ is a *yes*-instance of Colorful Separator. We first argue the forward direction. Suppose that $(G, \mathcal{P}, O, k, \ell)$ is a *yes*-instance of Colorful OCT Compression. Let $S \subseteq V(G)$ denote a set of vertices of size at most k such that $G - S$ is bipartite and $S \cap P \neq \emptyset$, for each $P \in \mathcal{P}$. Let $Z = S \cap O$ and $\mathcal{P}'' = \{P \in \mathcal{P} \mid P \cap Z \neq \emptyset\}$. Then, from statement (i) of Proposition 12.1, there is a partition $\mathcal{R}$ of $O \setminus S$ into sets X and Y such that $S \setminus O$ is an (s, t)-separator in the graph $G'^{O\setminus S}_{\mathcal{Q}',\mathcal{R}}$, where $G' = G - Z$. Moreover, for each $P \in \mathcal{P} \setminus \mathcal{P}''$, $(S \setminus O) \cap P \neq \emptyset$. Therefore, $(G'^{O\setminus S}_{\mathcal{Q}',\mathcal{R}}, \mathcal{P}', s, t, k - |Z|, \ell')$ is a *yes*-instance of Colorful Separator.

For the reverse direction, suppose that there is a partition of O into sets X, Y, and Z such that $(G'^{X\cup Y}_{\mathcal{Q}',\mathcal{W}}, \mathcal{P}', s, t, k - |Z|, \ell')$ is a *yes*-instance of Colorful Separator. Then, let Y be a corresponding solution. That is, Y is an (s, t)-separator in $G'^{X\cup Y}_{\mathcal{Q}',\mathcal{W}}$ of size at most $k - |Z|$, where $G' = G - Z$. By the second statement of Proposition 12.1, $Y \cup Z$ is an oct of G of size at most k. Moreover, for each $P \in \mathcal{P}$, either $Z \cap P \neq \emptyset$ or $Y \cap P \neq \emptyset$. Hence, $(G, \mathcal{P}, O, k, \ell)$ is a *yes*-instance of Colorful OCT Compression. □

To summarize, given an instance $(G = (V, E_1, \ldots, E_\alpha), k)$ of Sim-FVS/OCT, we first compute an odd cycle transversal of G_1 and a feedback vertex set of G_i, $i \in [\alpha] \setminus \{1\}$, in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time. Then, we generate $2^{\mathcal{O}(\alpha k)}$ instances of Colorful OCT, of the form $(G, \mathcal{P}, k, \ell \leq \alpha k)$, in $\mathcal{O}^\star(2^{\mathcal{O}(\alpha k)})$ time. Each instance of Colorful OCT is converted into an instance $(G, \mathcal{P}, O, k, \ell)$ of Colorful OCT Compression in polynomial time. Finally, for each instance of Colorful OCT Compression we generate $2^{\mathcal{O}(k)}$ instances of Colorful Separator, with parameters k and $\ell \leq \alpha k$, in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time. Lemmas 12.2 and 12.3 together imply that if we can solve an instance of Colorful Separator in $\mathcal{O}^\star(k^{\mathsf{poly}(\alpha,k)})$ time then the algorithm for Sim-FVS/OCT follows. We describe such an algorithm in the next section.

12.3 FPT Algorithm for Finding Colorful Separators

We in fact give an algorithm for a more general problem, which we call Colorful Multiway Cut (or CMWC for short). Before we proceed, we need a few definitions.

Definition 12.4. Given a graph G, a set $T \subseteq V(G)$, and a partition $\mathcal{T}$ of T into (pairwise disjoint) sets $\{T_1, \ldots, T_r\}$, we say that $S \subseteq V(G) \setminus T$ is a *$\mathcal{T}$-multiway cut* if, in

$G - S$, no vertex in $T_i \setminus S$ can reach a vertex in $T_j \setminus S$, for all $i, j \in [r]$, such that $i \neq j$. We say that $\mathcal{T}$ is an *edge-free partition* of T if there are no edges (u, v) in $G[T]$ where u and v belong to different sets of $\mathcal{T}$.

Given a grouping $\{P_1, \ldots, P_\ell\}$ of the vertices of a graph G, we define a partial coloring function $\mathsf{col} : V(G) \to 2^{[\ell]}$. That is, we have $i \in \mathsf{col}(v)$ if and only if $v \in P_i$, for some $i \in [\ell]$. In this context, for a set $C \subseteq [\ell]$, a subset S of vertices of G is called *C-colorful* if, for each $i \in C$, there is a vertex v in S such that $i \in \mathsf{col}(v)$. For a subset $S \subseteq V(G)$, we denote by $\mathsf{col}(S)$ the set $\{j \mid v \in S \cap (\bigcup_{i=1}^{\ell} P_i) \wedge j \in \mathsf{col}(v)\}$, i.e. the set of colors appearing in S. The CMWC can now be defined as follows.

Colorful Multiway Cut (CMWC)	**Parameter:** k, $\lvert T\rvert$, and ℓ
Input: A graph G, a set $T \subseteq V(G)$, a partition $\mathcal{T}$ of T into (pairwise disjoint) sets $\{T_1, \ldots, T_r\}$, a grouping $\mathcal{P}$ of $V(G)$ into (not necessarily distinct) sets $\{P_1, \ldots, P_\ell\}$, a set $C \subseteq [\ell]$, and an integer k.	
Question: Is there a set $S \subseteq V(G) \setminus T$ such that $\lvert S\rvert \leq k$, S is a $\mathcal{T}$-multiway cut in G, and S is C-colorful?	

12.3.1 Setting up the algorithm

Let $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ be an instance of CMWC. We start by stating a few simple reduction rules (which are applied in the order they are stated).

Reduction Rule 12.1. *If $k < 0$ then return that $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ is a no-instance.*

Reduction Rule 12.2. *If $k = 0$ and $\emptyset$ is a solution to $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ then return that $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ is yes-instance. If $k = 0$ and $\emptyset$ is not a solution then return no.*

Reduction Rule 12.3. *If there exists $i \in C$ such that $P_i \subseteq T$ then return no.*

Reduction Rule 12.4. *If there exists $i \in C$ such that $P_i \cap T \neq \emptyset$ then set $P_i = P_i \setminus T$.*

Reduction Rule 12.5. *If there exists $i \in C$ such that $P_i = \emptyset$ then return no.*

Reduction Rule 12.6. *If $\mathcal{T}$ is not an edge-free partition then return no.*

It is easy to see that Reduction Rules 12.1 to 12.6 are safe and can be applied in polynomial time. When $k > 0$ and $\emptyset$ is a $\mathcal{T}$-multiway cut, we can solve the corresponding instance in time $\mathcal{O}^\star(2^{\mathcal{O}(\ell)})$. The following observation describes how.

Observation 47. *Let $\mathcal{I} = (G, T, \mathcal{T}, \mathcal{P}, C, k)$ be an instance of* Colorful Multiway Cut. *If $k > 0$ and $\emptyset$ is a $\mathcal{T}$-multiway cut then $\mathcal{I}$ can be solved in $\mathcal{O}(2^{\mathcal{O}(\ell)} n^2)$ time, where $n = \lvert V(G)\rvert$.*

Proof. If $k > 0$ and $\emptyset$ is a $\mathcal{T}$-multiway cut then we are left with the problem of finding a set $S \subseteq V(G) \setminus T$ of size at most k such that $S \cap P_i \neq \emptyset$, for each $i \in C$. Hence, we construct a family $\mathcal{F}$ consisting of a set $f_{P_i} = P_i$ for each for each $i \in C$ and we let $\mathcal{U} = \cup_{i \in C} P_i$. Note that $\lvert\mathcal{F}\rvert \leq \ell \leq \alpha k$ and $\lvert\mathcal{U}\rvert \leq \lvert V(G)\rvert$. Since Reduction Rules 12.3 to 12.5 are not applicable, for each $i \in C$, we have $f_{P_i} \neq \emptyset$ and $f_{P_i} \cap T = \emptyset$. If we can find a subset $U \subseteq \mathcal{U}$ which intersects all the sets in $\mathcal{F}$, such that $\lvert U\rvert \leq k$, then U is the required solution. Otherwise, we have a no-instance. It is known that the Hitting

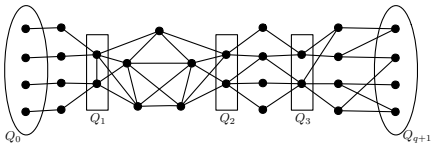

Figure 12.1: An illustration of a tight separator sequence.

Set problem parameterized by the size of the family $\mathcal{F}$ is fixed-parameter tractable and can be solved in $\mathcal{O}(2^{\mathcal{O}(|\mathcal{F}|)}|\mathcal{U}|^2)$ time [CFK+15]. In particular, we can find an optimum hitting set $U \subseteq \mathcal{U}$, hitting all the sets in $\mathcal{F}$. Therefore, we have a subset of vertices that intersects all sets P_i, for $i \in C$. □

Before proceeding with the description of the algorithm, we first recall the notion of tight separator sequences introduced in [LR12]. However, the definition and structural lemmas regarding tight separator sequences used in this paper are from [LRS16]. Note that although [LRS16] contains Definition 12.5 and Lemma 12.4 in terms of directed graphs, the same holds true for undirected graphs because one can represent any undirected graph as a directed graph by adding bidirectional edges between every pair of adjacent vertices.

Definition 12.5. Let X and Y be two subsets of $V(G)$ and let $k \in \mathbb{N}$. A *tight (X,Y)-reachability sequence* of order k is an ordered collection $\mathcal{H} = \{H_0, H_1, \ldots, H_q, H_{q+1}\}$ of sets in $V(G)$ satisfying the following properties:

- $X \subseteq H_i \subseteq V(G) \setminus N[Y]$ for any $0 \leq i \leq q$;
- $X = H_0 \subset H_1 \subset H_2 \subset \ldots \subset H_q \subset H_{q+1} = V(G) \setminus Y$;
- H_i is reachable from X in $G[H_i]$ and every vertex in $N(H_i)$ can reach Y in $G - H_i$ (implying that $N(H_i)$ is a minimal (X,Y)-separator in G);
- $|N(H_i)| \leq k$ for every $1 \leq i \leq q$;
- $N(H_i) \cap N(H_j) = \emptyset$ for all $1 \leq i, j \leq q$ and $i \neq j$;
- For any $0 \leq i \leq q-1$, there is no (X,Y)-separator S of size at most k where $S \subseteq H_{i+1} \setminus N[H_i]$ or $S \cap N[H_q] = \emptyset$ or $S \subseteq H_1$.

We let $Q_0 = X$, $Q_i = N(H_i)$, for $1 \leq i \leq q$, $Q_{q+1} = Y$, and $\mathcal{Q} = \{Q_0, Q_1, \ldots, Q_q, Q_{q+1}\}$. We call $\mathcal{Q}$ a *tight (X,Y)-separator sequence* of order k.

Lemma 12.4. *(see* [LRS16]*) There is an algorithm that, given a graph G on n vertices and m edges, subsets $X, Y \subseteq V(G)$ and $k \in \mathbb{N}$, runs in time $\mathcal{O}(k^2nm)$ and either correctly concludes that there is no (X,Y)-separator of size at most k in G or returns the sets $H_0, H_1, H_2 \setminus H_1, \ldots, H_q \setminus H_{q-1}, H_{q+1} \setminus H_q$ corresponding to a tight (X,Y)-reachability sequence $\mathcal{H} = \{H_0, H_1, \ldots, H_q, H_{q+1}\}$ of order k.*

See Figure 12.1 for an illustration of a tight (X, Y)-separator sequence. Our algorithm will be a combination of dynamic programming over the sets Q_i, $0 \leq i \leq q+1$, and

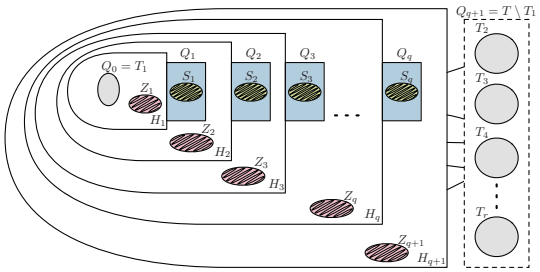

Figure 12.2: An illustration of the division of a solution S into various sets.

recursive calls for solving "smaller" instances of the same problem. Below we state some observations that help understand the structure of a solution and are crucial for achieving the stated running time.

Observation 48. *Let $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ be an instance of* Colorful Multiway Cut *and let T_1 be a set in $\mathcal{T}$ which is linked to some set in $\mathcal{T} \setminus \{T_1\}$. Moreover, let $\mathcal{H} = \{H_0, H_1, \ldots, H_q, H_{q+1}\}$ be a tight $(T_1, T \setminus T_1)$-reachability sequence of order k and let $\mathcal{Q} = \{Q_0, Q_1, \ldots, Q_q, Q_{q+1}\}$ be the corresponding tight separator sequence. Assume $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ is a* yes*-instance and let S be one of its solution. Then, S can be partitioned into the following (pairwise-disjoint) sets (see Figure 12.2).*

- $Z_1 = S \cap (H_1 \setminus Q_0)$.
- $S_i = S \cap Q_i$ *for* $1 \leq i \leq q$.
- $Z_i = (S \cap (H_i \setminus N[H_{i-1}])) \setminus Q_{q+1}$ *for* $2 \leq i \leq q+1$.

We invoke the last property of tight separator sequences to obtain the following bound.

Observation 49. $|Z_i| \leq k - 1$ *for each* $i \in [q+1]$.

Observation 49 is crucial as it allows us to apply our algorithm on sub-instances with a strictly smaller parameter.

To keep the presentation clean, we shall define two routines **ALG1** and **ALG2**. **ALG1** (Algorithm 4) delegates most of the "heavy lifting" to **ALG2**. That is, **ALG1** simply checks if any of the reduction rules are applicable and solves the instance if it corresponds to one of the base cases. When this is not the case, **ALG1** proceeds by computing a tight separator sequence and calls **ALG2**. Note that we can safely return false when the algorithm fails to construct such a sequence (Lines 7 and 8 of Algorithm 4). We now move to the description of **ALG2**, which takes as additional input the newly constructed tight separator sequence. Roughly speaking, **ALG2** will recursively solve a "large" number of instances restricted to graphs that "reside" between two consecutive separators of a separator sequence. The number of instances will be bounded by the number of possible "interactions" between the two consecutive separators and a hypothetical solution. However, due to Observation 49, each one of those recursive calls can be made with a strictly smaller value of k. Having solved all such instances (and stored the outcomes in tables), **ALG2** then proceeds using a dynamic programming

Algorithm 4: Pseudocode for **ALG1**

Input: A graph G, a set $T \subseteq V(G)$, a partition $\mathcal{T}$ of T into (pairwise disjoint) sets $\{T_1, \ldots, T_r\}$, a grouping $\mathcal{P}$ of $V(G)$ into (not necessarily distinct) sets $\{P_1, \ldots, P_\ell\}$, a set $C \subseteq [\ell]$, and an integer k.

Output: *yes* or *no*.

Apply all reduction rules (in order) and return*yes* or *no* appropriately (if applicable).

if $k > 0$ *and* $\emptyset$ *is a* $\mathcal{T}$*-multiway cut* **then**

  return *yes* or *no* appropriately (Observation 47)

Let $T_1 \in \mathcal{T}$ such that T_1 is linked to some $T_j \in \mathcal{T}$, where $j \neq 1$.

Let $\mathcal{H} = \{H_0, H_1, \ldots, H_q, H_{q+1}\}$ be a $(T_1, T \setminus T_1)$-reachability sequence of order k;

Let $\mathcal{Q} = \{Q_0, Q_1, \ldots, Q_q, Q_{q+1}\}$ be the corresponding $(T_1, T \setminus T_1)$-separator sequence;

if $\mathcal{Q} = \emptyset$ **then**

  return *no*;

return ALG2$(G, T, \mathcal{T}, \mathcal{P}, C, k, \mathcal{Q})$;

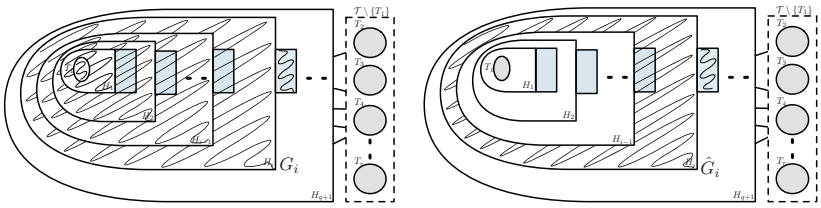

Figure 12.3: An illustration of graphs in Definition 12.6.

routine which computes the answer in a left-to-right manner, i.e. starting from Q_0 all the way to Q_{q+1}. We now give a formal description.

Definition 12.6. For a graph G and a tight separator sequence $\mathcal{Q} = \{Q_0, Q_1, \ldots, Q_q, Q_{q+1}\}$, we let $G_i = G - R_G(Q_{q+1}, Q_i)$, *i.e.* the graph obtained after removing the vertices that are reachable from Q_{q+1} after deleting Q_i, and we let $\widehat{G}_i = G_i - (V(G_{i-1}) \setminus Q_{i-1})$ (see Figure 12.3).

For each graph G_i, $i \in [q+1]$, we maintain a table Γ_i, where each entry is indexed by a tuple $(X, \mathcal{A}, \overline{C}, \overline{p})$. For each graph $\widehat{G}_i$, $i \in [q+1]$, we maintain a table Λ_i, where each entry is indexed by a tuple $(L, R, \mathcal{B}, \widehat{C}, \widehat{p})$. The tuples are described below.

- $X \subseteq Q_i \setminus T$ and $L \subseteq Q_{i-1} \setminus T$ and $R \subseteq Q_i \setminus T$;
- $\mathcal{A}$ is an edge-free partition of $(Q_i \cup Q_0) \setminus X$;
- $\mathcal{B}$ is an edge-free partition of $(Q_{i-1} \cup Q_i) \setminus (L \cup R)$;
- $\overline{C}, \widehat{C} \subseteq [\ell]$ and $\overline{p} \leq k - |X|$ and $\widehat{p} \leq k - |L \cup R|$ if $L \cup R \neq \emptyset$ and $\widehat{p} \leq k - 1$, otherwise.

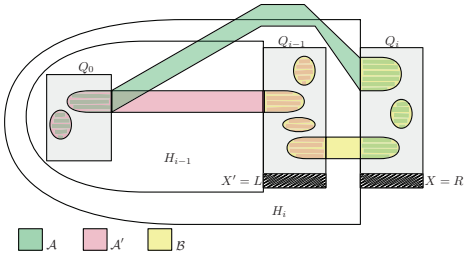

Figure 12.4: An illustration of compatible tuples.

Definition 12.7. For a tuple $\tau = (X, \mathcal{A}, \overline{C}, \overline{p})$, we denote by $\mathbb{I}_\tau$ the instance $(G_i - X, (Q_i \cup Q_0) \setminus X, \mathcal{A}, \mathcal{P}\big|_{V(G_i - X)}, \overline{C}, \overline{p})$ of CMWC. Similarly, for a tuple $\tau = (L, R, \mathcal{B}, \widehat{C}, \widehat{p})$, we denote by $\mathbb{I}_\tau$ the instance $(\widehat{G}_i - (L \cup R), (Q_{i-1} \cup Q_i) \setminus (L \cup R), \mathcal{B}, \mathcal{P}\big|_{V(G_i - (L \cup R))}, \widehat{C}, \widehat{p})$ of CMWC. Finally, we define $\Gamma_i(\tau)$ (or $\Lambda_i(\tau)$)$=$ *yes* if and only if $\mathbb{I}_\tau$ is a *yes*-instance of CMWC.

Definition 12.8. Given three tuples $\tau_1 = (X, \mathcal{A}, \overline{C}, \overline{p})$, $\tau_2 = (L, R, \mathcal{B}, \widehat{C}, \widehat{p})$, and $\tau_3 = (X', \mathcal{A}', \overline{C}', \overline{p}')$, we say that they are *compatible* if all of the following conditions hold (see Figure 12.4).

- $\tau_1 \in \Gamma_i$ and $\tau_2 \in \Lambda_i$ and $\tau_3 \in \Gamma_{i-1}$, where $i \in [q+1]$;
- $X' = L$ and $X = R$;
- $\mathcal{A}\big|_{Q_i \setminus X} = \mathcal{B}\big|_{Q_i \setminus R}$ and $\mathcal{B}\big|_{Q_{i-1} \setminus L} = \mathcal{A}'\big|_{Q_{i-1} \setminus X'}$ and $\mathcal{A}\big|_{Q_0} = \mathcal{A}'\big|_{Q_0}$;
- $\overline{p}' + \widehat{p} + |L| \leq p$ and $\overline{C}' \cup \widehat{C} \cup \mathsf{col}(L) = \overline{C}$.

The complete description of **ALG2** is given in Algorithm 5. Initially, we set all table entries to *no* (Lines 1 and 2). Then, for each $\widehat{G}_i \in \{\widehat{G}_1, \ldots, \widehat{G}_{q+1}\}$ and for each possible tuple $(L, R, \mathcal{B}, \widehat{C}, \widehat{p}) \in \Lambda_i$, we solve the corresponding CMWC instance $\mathcal{I} = (\widehat{G}_i - (L \cup R), (Q_{i-1} \cup Q_i) \setminus (L \cup R), \mathcal{B}, \mathcal{P}\big|_{V(G_i - (L \cup R))}, \widehat{p})$. That is, we set $\Lambda_i(L, R, \mathcal{B}, \widehat{C}, \widehat{p})$ if $\mathcal{I}$ is a *yes*-instance (Lines 3 to 8). Having computed all those values, we then proceed to filling table Γ_1. Since G_0 is a subgraph of G_1, and $G_1 = \widehat{G}_1$, we simply set $\Gamma_1(X, \mathcal{A}, \overline{C}, \overline{p}) = \Lambda_1(\emptyset, X, \mathcal{A}, \overline{C}, \overline{p})$ (for all tuples). This is justified by the fact that a solution is not allowed to delete any vertex in Q_0. To complete table Γ_i, $i > 1$, we simply use the following:

$$\Gamma_i(X, \mathcal{A}, \overline{C}, \overline{p}) = \bigvee [\Gamma_{i-1}(X', \mathcal{A}', \overline{C}', \overline{p}') \wedge \Lambda_i(L, R, \mathcal{B}, \widehat{C}, \widehat{p})],$$

where tuples $(X, \mathcal{A}, \overline{C}, \overline{p})$, $(X', \mathcal{A}', \overline{C}', \overline{p}')$, and $(L, R, \mathcal{B}, \widehat{C}, \widehat{p})$ are compatible. Finally, **ALG2** returns *yes* whenever there exists a tuple $\Gamma_{q+1}(\emptyset, \mathcal{T}, C, p) =$ *yes* (for some $p \leq k$).

12.3.2 Correctness and runtime analysis

We are now ready to prove our main structural lemma which reduces the computation of the entries in Γ_i (when $i > 1$) to those in Γ_{i-1} and Λ_i. The lemma is proved in a purely existential setting and serves as the proof of correctness of the algorithm.

Algorithm 5: Pseudocode for **ALG2**

Input: $(G, T, \mathcal{T}, \mathcal{P}, C, k, \mathcal{Q})$
Output: *yes* or *no*.

Initialize all entries in Γ_i to *no*, for $i \in [q+1]$;
Initialize all entries in Λ_i to *no*, for $i \in [q+1]$;
for *each* $\widehat{G}_i \in \{\widehat{G}_1, \ldots, \widehat{G}_{q+1}\}$ **do**
  for *each* $L \subseteq Q_{i-1} \setminus T$ *and each each* $R \subseteq Q_i \setminus T$ **do**
    for *each edge-free partition* $\mathcal{B}$ *of* $(Q_{i-1} \cup Q_i) \setminus (L \cup R)$ **do**
      for *each* $\widehat{C} \subseteq [\ell]$ *and each* $0 \leq \widehat{p} \leq k - \max\{1, |L \cup R|\}$ **do**
        $\mathbb{I} = (\widehat{G}_i - (L \cup R), (Q_{i-1} \cup Q_i) \setminus (L \cup R), \mathcal{B}, \mathcal{P}\big|_{V(G_i-(L\cup R))}, \widehat{p})$;
        $\Lambda_i(L, R, \mathcal{B}, \widehat{C}, \widehat{p}) = \mathbf{ALG1}(\mathbb{I})$;
Copy table entries for Γ_1, *i.e.* $\Gamma_1(X, \mathcal{A}, \overline{C}, \overline{p}) = \Lambda_1(\emptyset, X, \mathcal{A}, \overline{C}, \overline{p})$;
for *each* $G_i \in \{G_2, \ldots, G_{q+1}\}$ *(in order)* **do**
  for *each* $X \subseteq Q_i \setminus T$ **do**
    for *each edge-free partition* $\mathcal{A}$ *of* $(Q_i \cup Q_0) \setminus X$ **do**
      for *each* $\overline{C} \subseteq [\ell]$ *and each* $0 \leq \overline{p} \leq k - |X|$ **do**
        $\tau_1 = (X, \mathcal{A}, \overline{C}, \overline{p})$;
        for *each tuple* $\tau_2 = (L, R, \mathcal{B}, \widehat{C}, \widehat{p}) \in \Lambda_i$ **do**
          for *each tuple* $\tau_3 = (X', \mathcal{A}', \overline{C}', \overline{p}') \in \Gamma_{i-1}$ **do**
            if τ_1, τ_2, *and* τ_3 *are compatible* **then**
              $\Gamma_i(\tau_1) = \Gamma_i(\tau_1) \vee [\Gamma_{i-1}(\tau_3) \wedge \Lambda_i(\tau_2)]$;
if $\Gamma_{q+1}(\emptyset, \mathcal{T}, C, p) =$ *yes (for some* $p \leq k$*)* **then**
  return *yes*;
return *no*;

Lemma 12.5. *For any* $i \in [q+1]$ *and tuple* $\tau_1 = (X, \mathcal{A}, C_1, p_1) \in \Gamma_i$, $\mathbb{I}_{\tau_1}$ *is a yes-instance if and only if there is a tuple* $\tau_2 = (L, R, \mathcal{B}, C_2, p_2) \in \Lambda_i$ *and a tuple* $\tau_3 = (X', \mathcal{A}', C_3, p_3) \in \Gamma_{i-1}$ *such that* $\mathbb{I}_{\tau_2}$ *and* $\mathbb{I}_{\tau_3}$ *are both yes-instances and all three tuples are compatible.*

Proof. We begin with the forward direction. Suppose that $\mathbb{I}_{\tau_1}$ is a *yes*-instance. Then, it must be the case that there is a C_1-colorful set $S_{\tau_1} \subseteq V(G_i) \setminus X$ of size at most p_1 such that S_{τ_1} is an $\mathcal{A}$-multiway cut in G_i. We define sets $S_{\tau_2} = S_{\tau_1} \cap (V(\widehat{G}_i) \setminus (Q_{i-1} \cup Q_i))$ and $S_{\tau_3} = S_{\tau_1} \cap (V(G_{i-1}) \setminus Q_{i-1})$ (see Figure 12.5). We will now define tuples τ_2 and τ_3. Let $C_2 = \mathsf{col}(S_{\tau_2})$, $C_3 = \mathsf{col}(S_{\tau_3})$, $p_2 = |S_{\tau_2}|$, $p_3 = |S_{\tau_3}|$, $X' = L = S_{\tau_1} \cap Q_{i-1}$ and $R = X$. Furthermore, we let $\mathcal{C}$ be the equivalence class defined on the vertex sets of connected components in $G_i - (S_{\tau_1} \cup X)$. We set $\mathcal{B} = \mathcal{C}|_{(Q_{i-1} \cup Q_i) \setminus L}$ and $\mathcal{A}' = \mathcal{C}|_{Q_0 \cup Q_{i-1}}$. Finally, we set $\tau_2 = (L, R, \mathcal{B}, C_2, p_2)$ and $\tau_3 = (X', \mathcal{A}', C_3, p_3)$. We will now argue that S_{τ_2} and S_{τ_3} are solutions for $\mathbb{I}_{\tau_2}$ and $\mathbb{I}_{\tau_3}$, respectively. For two distinct sets $B, B' \in \mathcal{B}$ we have no path between them in $G_{\tau_2} - S_{\tau_2}$ since they belong to different connected components in $G_i - (S_{\tau_1} \cup X)$, $\widehat{G}_i$ is a subgraph of G_i, Q_{i-1} is a (Q_0, Q_i)-separator, and by the definition of $\mathcal{B}$. This proves that S_{τ_2} is a solution for $\mathbb{I}_{\tau_2}$. An analogous argument can be given for

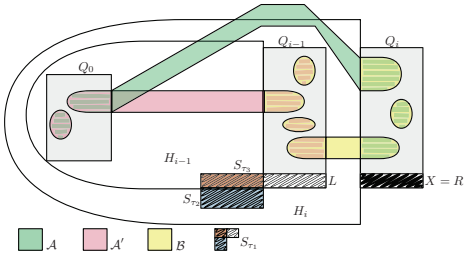

Figure 12.5: An illustration of the proof of Lemma 12.5.

S_{τ_3} being a solution for $\mathbb{I}_{\tau_3}$. This completes the argument in the forward direction.

In the reverse direction, suppose that there are $\tau_2 = (L, R, \mathcal{B}, C_2, p_2) \in \Lambda_i$ and $\tau_3 = (X', \mathcal{A}', C_3, p_3) \in \Gamma_{i-1}$ compatible with $\tau_1 \in \Gamma_i$ such that $\mathbb{I}_{\tau_3}$ and $\mathbb{I}_{\tau_2}$ are *yes*-instances. Let S_{τ_2} and S_{τ_3} be solutions for the respective instances.

We claim that $S_{\tau_1} = S_{\tau_2} \cup S_{\tau_3} \cup L$ is a solution for $\mathbb{I}_{\tau_1}$. Notice that $p_1 \leq p_2 + p_3 + |L|$ and $C_1 = C_2 \cup C_3 \cup \mathsf{col}(L)$ (we have $R = X$). We need to show that S_{τ_1} is an $\mathcal{A}$-multiway cut in $G_i - X$. Targeting a contradiction, suppose there are distinct sets $A_1, A_2 \in \mathcal{A}$ such that A_1 and A_2 are linked. Then, there exists a path P from $a_1 \in A_1$ to a vertex $a_2 \in A_2$. Consider the following ordered sequence of vertices $x_1, x_2, \ldots, x_t$ in $V(P) \cap (Q_0 \cup Q_{i-1} \cup Q_i)$ obtained from P, i.e. by the order in which they appear in P. Notice that each of the subpaths P_j from (x_j, x_{j+1}) of P, for $j \in [t-1]$, is completely contained in one of $\widehat{G}_i - S_{\tau_2}$ or $G_{i-1} - S_{\tau_3}$. But this implies that there exists $B \in \mathcal{B}$ such that $\{x_j \mid j \in [t]\} \cap (Q_{i-1} \cup Q_i) \subseteq B$. Similarly, there exists $A' \in \mathcal{A}'$ such that $\{x_j \mid j \in [t]\} \cap (Q_0 \cup Q_{i-1}) \subseteq A'$. But since τ_1, τ_2, and τ_3 are compatible therefore, a_1 and a_2 must belong to the same set in $\mathcal{A}$, contradicting the choice of a_1 and a_2. This concludes the proof. □

The above lemma proves the correctness of our algorithm since we set $\Gamma_i(\tau_1)$ to be *yes* precisely when there are τ_2 and τ_3 such that $\Gamma_{i-1}(\tau_2) = \Lambda_i(\tau_3) =$ *yes*. Finally, we prove the claimed running time bound.

Theorem 12.1. Colorful Multiway Cut *can be solved in* $\mathcal{O}^\star((k+t)^{\mathcal{O}(kt+k^3)} 2^{\mathcal{O}(\ell k)})$ *time, where* $t = |T|$.

Proof. Let $n = |V(G)|$, $m = |E(G)|$, $t = |T|$, and $\mathbb{T}(n, t, \ell, k)$ denote the "local" time taken by our algorithm to solve an instance $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ of Colorful Multiway Cut. By local, we mean the time taken ignoring all recursive calls. At the base case, the algorithm correctly decides the instance in $\mathcal{O}(2^{\mathcal{O}(\ell)} n^2)$ time (Observation 47). Hence, at the base case we have $\mathbb{T}(n, t, \ell, k) = \mathcal{O}(2^{\mathcal{O}(\ell)} n^2)$. Otherwise, we have $\mathbb{T}(n, t, \ell, k) \leq \mathcal{O}(k^2 nm) + \mathcal{O}(n(t+k)^{t+k} 2^k 2^\ell)$, where $\mathcal{O}(k^2 nm)$ is the time taken to construct a tight separator sequence (Lemma 12.4) and $\mathcal{O}(n(t+k)^{t+k} 2^k 2^\ell)$ is the time taken to compute all table entries Γ_i, for $i \in [q+1] \cup \{0\}$. Stated differently, $\mathcal{O}((t+k)^{t+k} 2^k 2^\ell)$ is the size of the largest table. The correctness of this step follows from Lemma 12.5 and the description of our two subroutines **ALG1** and **ALG2**.

Now consider the recursion tree. We let N_d denote a node in this tree at depth d. Note that the depth of our recursion tree is at most k; since k decreases by at least

one in every recursive call (Observation 49 and Definition 12.7). Consider any node N_d in the recursion tree with associated measures (n', t', ℓ', k'), i.e. $N_d = (n', t', \ell', k')$. We have $n' \leq n$, $k' \leq k$, $\ell' \leq \ell$, and $t' \leq \mathsf{max}(2k, t + k^2) \leq t + 2k + k^2$ (since t is either $2k$ or increases by at most k when computing Γ_{q+1} and the depth of our recursion tree is at most k). Moreover, if we sum n' for all nodes at depth d in the recursion tree we get $\sum_{N_d} n' \leq (\sum_{N_{d-1}} n')\mathcal{O}((t + 3k + k^2)^{t+3k+k^2} 2^k 2^\ell)$ (since $t' \leq t + 2k + k^2$). Therefore, at the deepest level, i.e. level k, we get:

$$\sum_{N_k} n' \leq n \cdot \mathcal{O}(((t + 3k + k^2)^{t+3k+k^2} 2^k 2^\ell)^k) = n \cdot \mathcal{O}((t + 3k + k^2)^{kt+3k^2+k^3} 2^{k^2} 2^{k\ell}).$$

Replacing for n in $\mathbb{T}(n, t, \ell, k)$, we get:

$$\mathbb{T}(n, t, \ell, k) \leq (k + t)^{\mathcal{O}(kt+k^3)} 2^{\mathcal{O}(\ell k)} n^{\mathcal{O}(1)}.$$

Multiplying by the number of nodes in the recursion tree, which is bounded by $\mathcal{O}((k + t)^{\mathcal{O}(kt+k^3)} 2^{\mathcal{O}(\ell k)} n^{\mathcal{O}(1)})$, we get the desired running time. □

Combining Theorem 12.1 with our series of reductions from Section 12.2, we have obtain the following corollary (Corollary 12.1).

Corollary 12.1. Sim-FVS/OCT *can be solved in* $\mathcal{O}^\star(k^{\mathsf{poly}(\alpha,k)})$ *time.*

Proof. Recall that Lemmas 12.2 and 12.3 together imply that if we can solve an instance of Colorful Separator in $\mathcal{O}^\star(k^{\mathsf{poly}(\alpha,k)})$ time then the algorithm for Sim-FVS/OCT follows. Any instance of Colorful Separator can be reduced to an instance of Colorful Multiway Cut with $|T| = 2$. From Theorem 12.1, such an instance can be solved in time $\mathcal{O}^\star(k^{\mathcal{O}(k^3)} 2^{\mathcal{O}(\alpha k)})$. □

12.4 W[1]-hardness of Simultaneous OCT

In this section, we show that Sim-OCT is W[1]-hard. For notational convenience, we shall use a different encoding of α-edge-colored graphs. Given a graph G with vertex set $V(G)$ and edge set $E(G)$, we define a coloring function $\mathsf{col}(e) \subseteq 2^{[\alpha]}$. In particular, when $\alpha = 2$, we have $\mathsf{col}(e) \subseteq 2^{[2]}$. We start by establishing W[1]-hardness of Sim-Cut. In Section 12.4.1 we show that Sim-Cut is W[1]-hard, even for $\alpha = 2$, by giving a parameterized reduction from Multi-Colored Clique. In Section 12.4.2 we give a parameterized reduction from Sim-Cut to Sim-OCT for the same value of α and hence establish the W[1]-hardness of Sim-OCT for $\alpha = 2$. We note that this also implies W[1]-hardness of Sim-OCT for all $\alpha \geq 2$.

12.4.1 W[1]-hardness of Simultaneous Cut

The Simultaneous Cut (or Sim-Cut for short) problem is formally defined below.

Simultaneous Cut (Sim-Cut) **Parameter:** k and α
Input: A graph G, two vertices $s, t \in V(G)$, an integer k, and a coloring function $\mathsf{col} : E(G) \to 2^{[\alpha]}$.
Question: Is there $X \subseteq V(G) \setminus \{s, t\}$ of size at most k such that, for all $i \in [\alpha]$, $G_i - X$ has no (s,t)-paths? Here, for $i \in [\alpha]$, $G_i = (V(G), E_i)$, where $E_i = \{e \in E(G) \mid i \in \mathsf{col}(e)\}$.

We give a parameterized reduction from Multi-Colored Clique which is known to be W[1]-hard [FHRV09]. Given an instance $(G, V_1, V_2, \ldots, V_k)$ of Multi-Colored Clique, we proceed by creating an instance $(G', s, t, k', \mathsf{col}' : E(G') \to 2^{\{1,2\}})$ of Sim-Cut such that $(G, V_1, V_2, \ldots, V_k)$ is a *yes*-instance of Multi-Colored Clique if and only if $(G', s, t, k', \mathsf{col}' : E(G') \to 2^{\{1,2\}})$ is a *yes*-instance of Sim-Cut. Without loss of generality we assume that for each $i, j \in [k]$ we have $|V_i| = |V_j|$, which can easily be achieved (in polynomial time) by adding isolated vertices.

The intuitive description of the parameterized reduction is as follows. Let $(G, V_1, V_2, \ldots, V_k)$ be an instance of Multi-Colored Clique. Since $|V_i| = |V_j|$, for all $i, j \in [k]$, we assume that $|V_i| = |V_j| = n$. Furthermore, we assume that for every $i, j \in [k]$, $i \neq j$, there is at least one edge between V_i and V_j, otherwise, the instance is a trivial no-instance of Multi-Colored Clique and our reduction will simply output a trivial no-instance of Sim-Cut with $\alpha = 2$. For each $i \in [k]$ we assume an arbitrary (but fixed) ordering on the vertices in V_i. For each $i \in [k]$, we will have a vertex selection gadget $\mathcal{S}_i$ that will be responsible for selecting a vertex in V_i. To achieve this, $\mathcal{S}_i$ will have $k-1$ copies of each vertex in V_i, so that each vertex in V_i has a copy corresponding to every $j \in [k] \setminus \{i\}$. For each $j \in [k] \setminus \{i\}$, we have an (s,t)-path with all edges having color 1. Each path contains exactly one copy of every vertex in V_i. Furthermore, these vertices appear in the order given by the ordering we already fixed on the vertices of V_i (see Figure 12.6).

The jth copy of the vertex set V_i will be used to ensure that there is an edge between the selected vertex in V_i and a vertex in V_j. The copies of any single vertex will form an (s,t)-separator of size $k-1$. Furthermore, the size of minimum (s,t)-separator in $\mathcal{S}_i$ will be $k-1$ and there will be exactly n distinct minimum separator each of which will correspond to a set comprising of $k-1$ copies of a vertex in V_i. By construction of the gadget and by setting budget constraints appropriately we will ensure that we must select a vertex from each of the $k-1$ copies of V_i, for each $i \in [k]$ and the selected $k-1$ vertices correspond to copies of the same vertex, i.e. we select a minimum separator. This will ensure that we have selected exactly one vertex from each V_i, for $i \in [k]$.

For $i, j \in [k]$, $i \neq j$, we will have edge selection gadgets E_{ij} which will ensure that there is an edge selected between V_i and V_j, and the selected edge is incident to the vertex selected from the vertex selection gadget. Finally, we will have a compatibility gadget which will ensure that the edges selected by E_{ij} and E_{ji} correspond to the same edge in G. We need to differentiate between gadgets E_{ij} and E_{ji} for technical reasons that will become clear later. We will now move to the formal description of the reduction.

Construction. Initially, $V(G') = \emptyset$ and $E(G') = \emptyset$. We add two special vertices s and t to $V(G')$, which are the vertices we want to separate, and which will be common to all the gadgets. For $i \in [k]$ we let v^i_j be the jth vertex in V_i. We now formally describe the construction of the various gadgets. We note that the gadgets are not necessarily vertex or edge disjoint (in addition to intersection with $\{s, t\}$).

Vertex Selection Gadget. For each $i \in [k]$ we have a vertex selection gadget $\mathcal{S}_i$ defined as follows. For each $j \in [k] \setminus \{i\}$, $\mathcal{S}_i$ contains vertices in $V_{ij} = \{v^i_{j1}, v^i_{j2}, \ldots, v^i_{jn}\}$ (refer to Figure 12.6). Here, the vertices $v^i_{j1}, v^i_{j2}, \ldots, v^i_{jn}$ corresponds to one copy of the vertices $v^i_1, v^i_2, \ldots, v^i_n$ in V_i. Note that for $j, j' \in [k] \setminus \{i\}$ vertices $v^i_{j\ell}, v^i_{j'\ell}$ correspond to

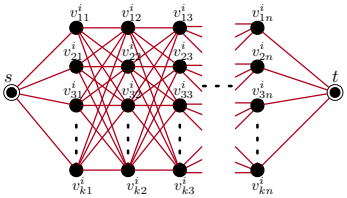

Figure 12.6: Vertex selection gadget with red color denoting color 1.

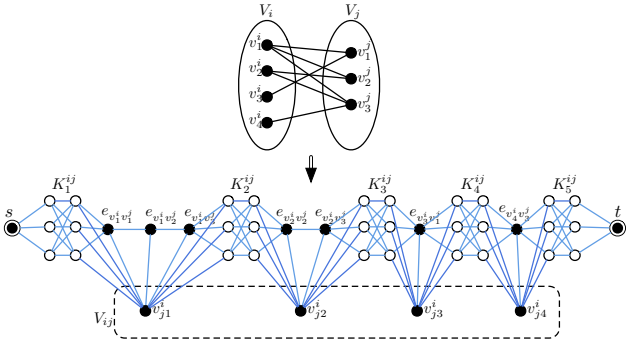

Figure 12.7: An illustration of an edge selection gadget with blue color denoting color 2.

copies of the same vertex, namely $v^i_\ell \in V_i$. For $i \in [k]$ and $\ell \in [n]$, we let $V^i_\ell = \{v^i_{j\ell} \mid j \in [k] \setminus \{i\}\}$, i.e. V^i_ℓ denotes the set comprising of $k-1$ copies of the vertex $v^i_\ell \in V_i$. For $i \in [k]$, $\ell \in [n-1]$, and for each $u \in V^i_\ell$ and $u' \in V^i_{\ell+1}$ we add the edge $(u,u') \in E(G')$ and set $\mathsf{col}'((u,u')) = \{1\}$. Note that $G'[V^i_\ell \cup V^i_{\ell+1}]$ is a complete bipartite graph with all edges having the color 1 in their color set. For $i \in [k]$, $u \in V^i_1$ we add the edge $(s,u) \in E(G')$ and set $\mathsf{col}'((s,u)) = \{1\}$. Similarly, for $i \in [k]$, $u \in V^i_n$ we add the edge $(u,t) \in E(G')$ and set $\mathsf{col}'((u,t)) = \{1\}$.

Edge Selection Gadget. For $i \in [k]$ and $j \in [k] \setminus \{i\}$ the edge selection gadget $\mathcal{E}_{ij}$ is constructed as follows. The vertex set of $\mathcal{E}_{ij}$ contains a vertex $e_{uu'}$, for each edge $(u,u') \in E(G)$ with $u \in V_i$ and $u' \in V_j$. We refer the reader to Figure 12.7 for an illustration. We note here that $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$ denote distinct gadgets. For $\ell \in [n]$, we let $E^{ij}_\ell = \{e_{v^i_\ell u'} \mid u' \in V_j, (v^i_\ell, u) \in E(G)\}$, i.e. E^{ij}_ℓ contains vertices corresponding to those edges between V_i and V_j that are incident to the vertex $v^i_\ell \in V_i$. We let $E_{ij} = \cup_{\ell \in [n]} E^{ij}_\ell$. For $\ell \in [n]$ and each $u \in E^{ij}_\ell$, we add the edge $(u, v^i_{j\ell})$ to $\mathcal{E}_{ij}$. We add an induced path P^{ij}_ℓ on the vertices in E^{ij}_ℓ (where the vertices appear in the natural order implied by the ordering of the vertices in V_j) and add these edges to $\mathcal{E}^{ij}_\ell$. For each edge $e \in E(P^{ij}_\ell)$, we let $\mathsf{col}'(e) = \{2\}$. For $\ell \in [n+1]$, we let $\mathbf{K}^{ij}_\ell$ denote a $K_{3,3}$ (complete bipartite graph with 3 vertices on both side) with vertex bipartition $(\{p^{ij}_\ell, q^{ij}_\ell, r^{ij}_\ell\}, \{\bar{p}^{ij}_\ell, \bar{q}^{ij}_\ell, \bar{r}^{ij}_\ell\})$ and add it to $\mathcal{E}_{ij}$. We will refer to $\mathbf{K}^{ij}_\ell$s as *barrier blocks* of $\mathcal{E}_{ij}$. Finally, we join s, t and E^{ij}_ℓ, for $\ell \in [n]$ using the barrier blocks. This is done as follows.

For $\ell \in [n]$, let a^{ij}_ℓ, b^{ij}_ℓ be the first and the last vertex respectively, in the path

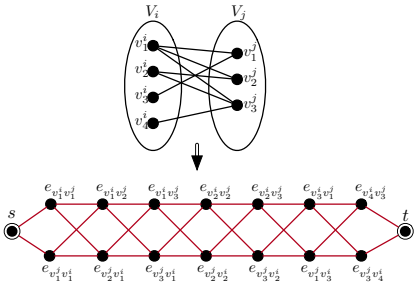

Figure 12.8: An illustration of edge compatibility gadget with red color denoting color 1 and $i < j$.

P_ℓ^{ij}. We add the edges $(a_\ell^{ij}, \bar{p}_\ell^{ij}), (a_\ell^{ij}, \bar{q}_\ell^{ij}), (a_\ell^{ij}, \bar{r}_\ell^{ij})$ and $(b_\ell^{ij}, p_{\ell+1}^{ij}), (b_\ell^{ij}, q_{\ell+1}^{ij}), (b_\ell^{ij}, r_{\ell+1}^{ij})$ to $E(\mathcal{E}_{ij})$. Also, for $\ell \in [n]$, we add the edges $(v_{j\ell}^i, \bar{p}_\ell^{ij}), (v_{j\ell}^i, \bar{q}_\ell^{ij}), (v_{j\ell}^i, \bar{r}_\ell^{ij})$ and $(v_{j\ell}^i, p_{\ell+1}^{ij}), (v_{j\ell}^i, q_{\ell+1}^{ij}), (v_{j\ell}^i, r_{\ell+1}^{ij})$ to $E(\mathcal{E}_{ij})$. In addition, we add the edges $(s, p_1^{ij}), (s, q_1^{ij})$, $(s, \bar{r}_1^{ij})$, $(\bar{p}_{n+1}^{ij}, t), (\bar{q}_{n+1}^{ij}, t), (\bar{r}_{n+1}^{ij}, t)$ to $\mathcal{E}_{ij}$. For each $e \in E(\mathcal{E}_{ij})$, we set $\mathsf{col}'(e) = \{2\}$. This completes the description of the edge selection gadget.

Edge Compatibility Gadget. This gadget is used to ensure that the edge selected by $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$ corresponds to the same edge of G. For $i, j \in [k]$, $i < j$, the edge compatibility gadget $\mathcal{C}_{ij}$ is constructed as described below. Basically, $\mathcal{C}_{ij}$ comprises of a set of edges between vertices in E_{ij} and vertices in E_{ji}. Recall that E_{ij} and E_{ji} contains vertices corresponding to the same edges, namely the edges between V_i and V_j in G. Hence, we can think of E_{ji} as a set comprising of a copy of the vertices in E_{ij}. We fix a lexicographic ordering on vertices in E_{ij} which we obtain as follows. For $e_{v_a^i, v_x^j}, e_{v_b^i, v_y^j} \in E_{ij}$, $e_{v_a^i, v_x^j} < e_{v_b^i, v_y^j}$ if (i) $a < b$ or (ii) $a = b$ and $x < y$. We denote the ordering of vertices in E_{ij} by $e_1^{ij}, e_2^{ij}, \ldots, e_m^{ij}$. Refer to Figure 12.8 for an illustration. Note this also fixes an ordering of vertices in E_{ji} which we denote by $e_1^{ji}, e_2^{ji}, \ldots, e_m^{ji}$. Here, m is the number of edges between V_i and V_j in G. For $\ell \in [m-1]$, we add the edges $(e_\ell^{ij}, e_{\ell+1}^{ij}), (e_\ell^{ij}, e_{\ell+1}^{ji}), (e_\ell^{ji}, e_{\ell+1}^{ij}), (e_\ell^{ji}, e_{\ell+1}^{ji})$ to $\mathcal{C}_{ij}$. That is we add all the edges in the bipartition between each consecutive pair of vertices in the ordered sets E_{ij} and E_{ji}. We add edges $(s, e_1^{ij}), (s, e_1^{ji})(e_m^{ij}, t), (e_m^{ji}, t)$ to $\mathcal{C}_{ij}$. For each edge $e \in \mathcal{C}_{ij}$, we set $\mathsf{col}'(e) = \{1\}$. We note here that in case we have created multiple edges say e, e' between vertices u, v then we delete e' and set $\mathsf{col}'(e) := \mathsf{col}'(e) \cup \mathsf{col}'(e')$.

We finally set $k' = k(k-1) + 2\binom{k}{2}$. In the following we prove certain lemmata which will be helpful in establishing the equivalence between the given instance of Multi-Colored Clique and the created instance of Sim-Cut. We denote the graph constructed above as G' with the coloring function on the edge set being col'. For $i \in [2]$, by G'_i we denote the graph with vertex set $V(G')$ and edge set $E_i = \{e \in E(G') \mid i \in \mathsf{col}'(e)\}$.

Lemma 12.6. *For $i \in [n]$, consider the graph $\hat{G}_i = G'_1[V(\mathcal{S}_i)]$. The minimum (s,t)-separator in $\hat{G}_i$ has size $k-1$. Furthermore, $\mathcal{F} = \{V_\ell^i \mid \ell \in [n]\}$ is the set of all minimum sized (s,t)-separators in $\hat{G}_i$.*

Proof. We start by showing that for any (s,t)-separator X' in $\hat{G}_i$, there exists $\ell \in [n]$ such that $V_\ell^i \subseteq X'$. Suppose not, then there exists an (s,t)-separator X, in $\hat{G}_i$ such

that for all $\ell \in [n]$, $V^i_\ell \not\subseteq X$. This in turn implies that for all $\ell \in [n]$, there exists $v^*_\ell \in V^i_\ell$ such that $v^*_\ell \notin X$. Recall that from construction $(s, v^*_1), (v^*_n, t) \in E(\hat{G}_i)$ and for all $\ell \in [n-1]$, $(v^*_\ell, v^*_{\ell+1}) \in E(\hat{G}_i)$. This implies that there is an (s,t)-path in $\hat{G}_i - X$, namely $P = s, v^*_1, v^*_2, \ldots, v^*_n, t$, contradicting that X is an (s,t)-separator in $\hat{G}_i$. Since for any (s,t)-separator X' in $\hat{G}_i$, there exists $\ell \in [n]$ such that $V^i_\ell \subseteq X'$, this implies that the size of minimum (s,t)-separator in $\hat{G}_i$ is at least $k-1$ and $\mathcal{F} = \{V^i_\ell \mid \ell \in [n]\}$ is the set of all minimum sized (s,t)-separators in $\hat{G}_i$. □

Lemma 12.7. *For $i, j \in [n]$ and $i \neq j$, consider the graph $\hat{G}_{ij} = G'_1[V(E_{ij}) \cup V(E_{ji}) \cup \{s,t\}]$. The minimum (s,t)-separator in $\hat{G}_{ij}$ has size 2. Furthermore, $\mathcal{F} = \{\{e^{ij}_\ell, e^{ji}_\ell\} \mid \ell \in [m]\}$ is the set of all minimum sized (s,t)-separators in $\hat{G}_{ij}$. Here, m is the number of edges between V_i and V_j in G.*

Proof. We start by showing that for any (s,t)-separator X' in $\hat{G}_{ij}$, there exists $\ell \in [m]$ such that $e^{ij}_\ell, e^{ji}_\ell \in X'$. Suppose not, then there exists an (s,t)-separator X, in $\hat{G}_{ij}$ such that for all $\ell \in [m]$ there exits $\hat{e}_\ell \in \{e^{ij}_\ell, e^{ji}_\ell\}$ such that $\hat{e}_\ell \notin X'$. Recall that from construction $(s, \hat{e}_1), (\hat{e}_m, t) \in E(\hat{G}_{ij})$ and for all $\ell \in [m-1]$, $(\hat{e}_\ell, \hat{e}_{\ell+1}) \in E(\hat{G}_{ij})$. This implies that there is an (s,t)-path in $\hat{G}_{ij} - X$, namely $P = s, \hat{e}_1, \hat{e}_2, \ldots, \hat{e}_n, t$, contradicting that X is an (s,t)-separator in $\hat{G}_{ij}$. Since for any (s,t)-separator X' in $\hat{G}_{ij}$, there exists $\ell \in [m]$ such that $\{e^{ij}_\ell, e^{ji}_\ell\} \subseteq X'$, this implies that the size of minimum (s,t)-separator in $\hat{G}_{ij}$ is at least 2 and $\mathcal{F} = \{\{e^{ij}_\ell, e^{ji}_\ell\} \mid \ell \in [m]\}$ is the set of all minimum sized (s,t)-separators in $\hat{G}_{ij}$. □

Lemma 12.8. *For $i, j \in [n]$ and $i \neq j$, consider the graph $\hat{G}_{ij} = G'_2[V(\mathcal{E}_{ij})]$. The minimum (s,t)-separator in $\hat{G}_{ij}$ has size 2. For $\ell \in [n]$, let $\mathcal{F}_\ell = \{\{e, v^i_{j\ell}\} \mid e \in E^{ij}_\ell\}$. Then, $\mathcal{F} = \cup_{\ell \in [n]} \mathcal{F}_\ell$ is the set of all minimum sized (s,t)-separators in $\hat{G}_{ij}$.*

Proof. We start by showing that for any (s,t)-separator X' of size at most 2 in $\hat{G}_{ij}$, there exists $\ell \in [n]$, $e \in E^{ij}_\ell$ such that $v^i_{j\ell}, e \in X'$. Suppose not. Then there exists an (s,t)-separator X, in $\hat{G}_{ij}$ of size at most 2, such that for all $\ell \in [n]$, either $v^i_{j\ell} \notin X$ or $E^{ij}_\ell \cap X = \emptyset$. Since $|X| \leq 2$, for every barrier block $\mathbf{K}^{ij}_\ell$, for $\ell \in [n+1]$, there exists $p^*_\ell \in \{p^{ij}_\ell, q^{ij}_\ell, r^{ij}_\ell\} \setminus X$ and $\bar{p}^*_\ell \in \{\bar{p}^{ij}_\ell, \bar{q}^{ij}_\ell, \bar{r}^{ij}_\ell\} \setminus X$. For each $\ell \in [n]$, we now define a path P_ℓ as follows. If $v^i_{j\ell} \notin X$ then $P_\ell = \bar{p}^*_\ell, v^i_{j\ell}, p^*_{\ell+1}$, otherwise $P_\ell = \bar{p}^*_\ell, E^{ij}_\ell, p^*_{\ell+1}$, where the path is defined with respect to the ordering of vertices of E^{ij}_ℓ used in the construction of the edge selection gadget. But then, $P = s, p^*_1, \bar{p}^*_1, P^{ij}_1, p^*_2, P^{ij}_2, \ldots, p^*_n P^{ij}_n, \bar{p}^*_{n+1}, t$ is an (s,t)-path in $\hat{G}_{ij} - X$, contradicting the assumption that X is an (s,t)-separator in $\hat{G}_{ij}$. Furthermore, it follows from the construction that for all $\ell \in [n]$, $e \in E^{ij}_\ell$, $\{v^i_{j\ell}, e\}$ is an (s,t)-separator in $\hat{G}_{ij}$. This concludes the proof. □

Lemma 12.9. *$(G, V_1, V_2, \ldots, V_k)$ is a yes-instance of* Multi-Colored Clique *if and only if $(G', s, t, k', \mathsf{col}' : E(G') \to 2^{\{1,2\}})$ is a yes-instance of* Sim-OCT.

Proof. In the forward direction suppose that $(G, V_1, V_2, \ldots, V_k)$ is a *yes*-instance of Multi-Colored Clique and let $X = \{v^i_{\ell_i} \in V_i \mid i \in [k]\}$ be a set such that $G[X]$ is a clique. We note here that for $i \in [k]$, $v^i_{\ell_i}$ is the ℓ_ith vertex in V_i. Let $Y = \{e_{uv}, e_{vu} \mid u, v \in X\}$ and $X' = (\cup_{i \in [k]} V^i_{\ell_i}) \cup Y$. We will show that X' is a solution

to Sim-Cut in $(G', s, t, k', \mathsf{col}' : E(G') \to 2^{\{1,2\}})$. Note that $|X'| = k' = k(k-1) + 2\binom{k}{2}$. Therefore, we only need to show that $G'_1 - X'$ and $G'_2 - X'$ have no (s,t)-paths, respectively.

We first show that $G'_1 - X'$ has no (s,t)-paths. Consider the set $\mathcal{Z} = \{V(\mathcal{S}_i) \setminus \{s,t\} \mid i \in [k]\} \cup \{E_{ij} \mid i,j \in [k], i \neq j\}$. Observe that for any two distinct sets $A, B \in \mathcal{Z}$, $A \cap B = \emptyset$, and for any $a \in A$ and $b \in B$, $(a,b) \notin E(G'_1)$. Hence any (s,t)-separator in G'_1 is the union of (s,t)-separators in $G'_1[A \cup \{s,t\}]$, for $A \in \mathcal{Z}$. But then from the construction of the set X', Lemma 12.6 and Lemma 12.7 it follows that X' is an (s,t)-separator in G'_1.

We will now show that $G'_2 - X'$ has no (s,t)-paths. Consider the set $\mathcal{Z}' = \{V(\mathcal{E}_{ij}) \setminus \{s,t\} \mid i,j \in [k], i \neq j\}$. For any two distinct sets $A', B' \in \mathcal{Z}'$, $A' \cap B' = \emptyset$, and for any $a' \in A'$ and $b' \in B'$, $(a', b') \notin E(G'_2)$. Hence any (s,t)-separator in G'_2 is the union of (s,t)-separators in $G'_2[A' \cup \{s,t\}]$, for $A' \in \mathcal{Z}'$. But then from the construction of the set X' and Lemma 12.8 it follows that X' is an (s,t)-separator in G'_2. This concludes the proof in the forward direction.

In the reverse direction, let $(G', s, t, k', \mathsf{col}' : E(G') \to 2^{\{1,2\}})$ be a *yes*-instance of Sim-OCT and X' be one of its solution. Since X' is a solution, therefore, $G'_1 - X'$ and $G'_2 - X'$ have no (s,t)-paths. Consider the set $\mathcal{Z} = \{V(\mathcal{S}_i) \setminus \{s,t\} \mid i \in [k]\} \cup \{E_{ij} \mid i,j \in [k], i \neq j\}$. Observe that for any two distinct sets $A, B \in \mathcal{Z}$, $A \cap B = \emptyset$, and for any $a \in A$ and $b \in B$, $(a,b) \notin E(G'_1)$. Hence, any (s,t)-separator in G'_1 is the union of (s,t)-separator in $G'_1[A \cup \{s,t\}]$, for $A \in \mathcal{Z}$. This together with Lemma 12.6, Lemma 12.7 and the definition of k' implies that for each $A \in \mathcal{Z}$, we must pick a minimum sized separator (s,t)-separator in $G'_1[A \cup \{s,t\}]$. Any minimum sized (s,t)-separator in $G'_1[V(\mathcal{S}_i)]$ must belong to $\mathcal{F}_v = \{V^i_\ell \mid \ell \in [n]\}$. We let $X = \{v^i_{\ell_i} \mid i \in [k], V^i_{\ell_i} \subseteq X'\}$. We will show that X is a solution to Multi-Colored Clique in $(G, V_1, V_2, \ldots, V_k)$. It is easy to see that $|X \cap V_i| = 1$. We need to show that $G[X]$ is a clique. Consider $v^i_{\ell_i}, v^j_{\ell_j} \in X$, where $i, j \in [k]$, $i < j$ and suppose $(v^i_{\ell_i}, v^j_{\ell_j}) \notin E(G)$. Lemma 12.8 (together with construction of k') implies that for some $e \in E^{ij}_{\ell_i}$, $e \in X'$ and for some $e' \in E^{ji}_{\ell_j}$, $e' \in X'$. Moreover, Lemma 12.8 implies that $\{e, e'\} \in \mathcal{F} = \{\{e^{ij}_\ell, e^{ji}_\ell\} \mid \ell \in [m]\}$. But this implies that $(v^i_{\ell_i}, v^j_{\ell_j}) \in E(G)$. This concludes the proof. □

Theorem 12.2 follows from combining Lemma 12.9 and the W[1]-hardness of Multi-Colored Clique.

Theorem 12.2. *For all $\alpha \geq 2$,* Sim-Cut *is* W[1]-hard *when parameterized by k. Here, α is the number of colors in the coloring function of the edge set.*

12.4.2 From Simultaneous Cut to Simultaneous OCT

In this section, we give a parameterized reduction from Sim-Cut to Sim-OCT. Roughly speaking, given an instance $(G, s, t, k, \mathsf{col} : E(G) \to 2^{[\alpha]})$ of Sim-Cut we create an instance $(G', k', \mathsf{col}' : E(G') \to 2^{[\alpha]})$ of Sim-OCT by *subdividing* edges in G and adding $k+1$ vertex disjoint (s,t)-paths on 2 vertices. Furthermore, we create $k+1$ duplicates (also known as *false twins*) of s and t. The objective behind these operations is the conversion of (s,t)-paths in G to odd-cycles in G'. Moreover, all the odd cycles in G' will be shown to correspond to an (s', t')-path (subdivided), where s' and t' are false twins of s and t, respectively, along with one of the newly added paths on 2 vertices. Along

the way, we will also duplicate certain vertices $k+1$ times simply to ensure that a copy of these vertices always remains in the graph resulting from deleting a set of at most k vertices. We now move to the formal description of the reduction.

Let $(G, s, t, k, \mathsf{col} : E(G) \to 2^{[\alpha]})$ be an instance of Sim-Cut. For technical reasons we will assume that $(s,t) \notin E(G)$. Such an assumption is legitimate because otherwise, either we have a trivial no-instance of Sim-Cut which is the case when $\mathsf{col}((s,t)) \neq \emptyset$ or we can delete the edge (s,t) which is the case when $\mathsf{col}((s,t)) = \emptyset$. We create an instance $(G', k', \mathsf{col}' : E(G') \to 2^{[\alpha]})$ of Sim-OCT as follows. Initially, $V(G') = V(G)$ and $E(G') = \emptyset$. For each edge $(u,v) \in E(G)$ we add a vertex e_{uv} to $V(G')$, add the edges $(u, e_{uv}), (v, e_{uv})$ to $E(G')$, and set $\mathsf{col}'((u, e_{uv})) = \mathsf{col}((u,v))$ and $\mathsf{col}'((v, e_{uv})) = \mathsf{col}((u,v))$. For $i \in [k+1]$, we add vertices w_i, w_i' to $V(G')$ and add the edges $(s, w_i), (w_i, w_i'), (w_i', t)$ to $E(G')$. In addition, we set $\mathsf{col}'((s, w_i)) = \mathsf{col}'((w_i, w_i')) = \mathsf{col}'((w_i', t)) = [\alpha]$. We create k chromatic false twins, i.e. false twins with the color sets on edges duplicated appropriately, of vertices s and t respectively in G' and let $S_f = \{s_i \mid i \in [k]\} \cup \{s\}$ and $T_f = \{t_i \mid i \in [k]\} \cup \{t\}$. Finally, we set $k' = k$.

Proposition 12.2. *Let H be a graph containing a cycle C with an odd number of vertices. Then H contains an induced cycle C' with an odd number of vertices.*

In the following lemmata we establish some of the properties of the instance $(G', k', \mathsf{col}' : E(G') \to 2^{[\alpha]})$ of Sim-OCT that will be helpful in establishing its equivalence with the instance $(G, s, t, k, \mathsf{col} : E(G) \to 2^{[\alpha]})$ of Sim-Cut. We let G' to be the graph constructed as described above from G. For $i \in [\alpha]$, by G_i we denote the graph $G[E_i]$, where $E_i = \{e \in E(G) \mid i \in \mathsf{col}(e)\}$. Analogously, we define G_i', for $i \in [\alpha]$.

Lemma 12.10. *For $i \in [\alpha]$, let C be an induced cycle in G_i' such that $|V(C)| \neq 4$. Then, $|V(C) \cap S_f| \leq 1$ and $|V(C) \cap T_f| \leq 1$.*

Proof. Consider an induced cycle C in G_i' such that $|V(C)| \neq 4$. We will only argue that $|V(C) \cap S_f| \leq 1$ and $|V(C) \cap T_f| \leq 1$ will follow from a symmetric argument. If C contains a vertex from S_f, say s', then C must contain at least 2 vertices from $\{w_i, w_i' \mid i \in [k+1]\} \cup \{e_{sv} \mid v \in N_{G_i}(s)\}$ since they are the only neighbors of s' in G'. But then C cannot contain any other vertex from S_f since vertices in S_f are chromatic false twins of s' in G' and $|V(C)| \neq 4$. This concludes the proof. □

Lemma 12.11. *For $i \in [\alpha]$, let C be an induced cycle in G_i' such that $V(C) \cap \{w_i, w_i' \mid i \in [k+1]\} = \emptyset$. Then, C is a cycle with an even number of vertices.*

Proof. Consider an induced cycle C in G_i' such that $|V(C)| \cap \{w_i, w_i' \mid i \in [k+1]\} = \emptyset$. If $|V(C)| = 4$ then the claim trivially holds. Otherwise, Lemma 12.10 implies that $|V(C) \cap S_f| \leq 1$ and $|V(C) \cap T_f| \leq 1$. Since vertices in S_f and T_f are chromatic false twins, we can find a cycle C' with $|V(C)|$ vertices by replacing vertex $s' \in V(C) \cap S_f$ (if it exists) by s and vertex $t' \in V(C) \cap S_f$ (if it exists) by t. Recall that for $X = (V(G') \setminus (S_f \cup T_f \cup \{w_i, w_i' \mid i \in [k+1]\})) \cup \{s,t\}$, $G'[X]$ is a graph obtained by subdivision of edges in G. But C' is a cycle in $G_i'[X]$ and hence it follows that $|V(C')| = |V(C)|$ is an even number. □

Lemma 12.12. *For $i \in [\alpha]$, let C be an induced cycle in G_i' such that $|V(C)| \neq 4$. Then for $\ell \in [k+1]$, $w_\ell \in V(C)$ if and only if $w_\ell' \in V(C)$. Furthermore, if $|V(C)| \neq 6$ then $|V(C) \cap \{w_j \mid j \in [k+1]\}| \leq 1$.*

Proof. For $\ell \in [k+1]$, let C be an induced cycle in G'_i such that $w_\ell \in V(C)$. Recall that $N_{G'_i}(w_\ell) = S_f \cup \{w'_\ell\}$. Therefore, C must contain two vertices from $S_f \cup \{w'_\ell\}$. From Lemma 12.10 it follows that $|V(C) \cap S_f| \leq 1$. This implies that $w'_\ell \in V(C)$. An analogous argument can be given for the reverse direction.

For the second part of the lemma, suppose there exists distinct $\ell, \ell' \in [k+1]$ such that $w_\ell, w_{\ell'} \in V(C)$. First part of the lemma implies that $w'_\ell, w'_{\ell'} \in V(C)$. But then C must contain a vertex $s' \in S_f$ and a vertex $t' \in T_f$ as C must contain two neighbors of w_ℓ and two neighbors of w'_ℓ. But $s' \in N_{G'_i}(w_{\ell'})$ and $t' \in N_{G'_i}(w'_{\ell'})$. This contradicts the assumption that C is an induced cycle such that $|V(C)| \neq 6$. □

Lemma 12.13. *$(G, s, t, k, \mathsf{col} : E(G) \to 2^{[\alpha]})$ is a yes-instance of* Sim-Cut *if and only if $(G', k', \mathsf{col}' : E(G') \to 2^{[\alpha]})$ is a yes-instance of* Sim-OCT.

Proof. In the forward direction let $(G, s, t, k, \mathsf{col} : E(G) \to 2^{[\alpha]})$ be a *yes*-instance of Sim-Cut and $S \subseteq V(G) \setminus \{s, t\}$ be one of its solutions. We will show that S is a solution to the instance $(G', k', \mathsf{col}' : E(G') \to 2^{[\alpha]})$ of Sim-OCT. Suppose not. Then, there is an odd cycle $\hat{C}$ in $G'_i - S$, for some $i \in [\alpha]$. Since $G'_i - S$ has an odd-cycle $\hat{C}$, Proposition 12.2 implies that $G'_i - S$ has an induced odd-cycle C. As C is an odd-cycle, Lemma 12.11 and Lemma 12.12 imply that there exists a unique $\ell \in [k+1]$ such that $w_\ell, w'_\ell \in V(C)$. But then C must contain a vertex in S_f and a vertex in T_f. This together with Lemma 12.10 implies that there exists a unique $s' \in S_f$ and $t' \in T_f$ such that $s', t' \in V(C)$. Let P' be the path from s' to t' obtained from C by deleting w_ℓ and w'_ℓ. Since $V(P') \cap (S_f \setminus \{s'\}) = \emptyset$ and $V(P') \cap (T_f \setminus \{t'\}) = \emptyset$ it must be that all the internal vertices in P' are in $X = V(G') \setminus (S_f \cup T_f \cup \{w_j, w'_j \mid j \in [k+1]\} \cup S)$. Recall that $G'[X]$ is obtained from G by subdividing edges in G. But then we can obtain an (s,t)-path in $G_i - S$ from P' by replacing s' by s, t' by t and edges $(u, e_{uv})(e_{uv}, v)$ by (u, v) in $G - S$ contradicting that S is a solution to Sim-Cut.

In the reverse direction, let $(G', k', \mathsf{col}' : E(G') \to 2^{[\alpha]})$ be a *yes*-instance of Sim-OCT and $S' \subseteq V(G')$ be a solution. Let $\hat{S} = S' \setminus (S_f \cup T_f \cup \{w_i, w'_i \mid i \in [k+1]\})$. We obtain S from $\hat{S}$ by replacing each $e_{uv} \in \hat{S}$ (if any) by either of u or v. Here, in making the choice we give preference to one that is not s nor t and since $(s, t) \notin E(G)$ such a choice always exists. We will show that S is a solution to the instance $(G, s, t, k, \mathsf{col} : E(G) \to 2^{[\alpha]})$ of Sim-Cut. Note that $|S| \leq k$, therefore it is enough to show that for each $i \in [\alpha]$, $G_i - S$ has no (s,t)-path. Aiming for a contradiction, suppose for some $i \in [\alpha]$, $G_i - S$ has an (s,t)-path P. Since $|S'| \leq k$, there exists $j \in [k+1]$ such that $w_j, w'_j \notin S'$, $s' \in S_f \setminus S'$ and $t' \in T_f \setminus S'$. Let P'_1 be the (s',t')-path in G'_i obtained from P by replacing each edge (u, v) by (u, e_{uv}) and (e_{uv}, v), replacing s by s' and t by t'. Also, let $P'_2 = s', w_j, w'_j, t'$ be another (s',t')-path in G'_i. Recall that by construction, for each edge $e \in E(P'_1) \cup E(P'_2)$, $i \in \mathsf{col}'(e)$. Furthermore, $S' \cap (V(P'_1) \cup V(P'_2)) = \emptyset$, which follows from our construction of the paths P'_1 and P'_2. But then we have two (s',t')-paths P'_1 and P'_2 (internally vertex disjoint). Therefore, we obtain a cycle C containing s' and t' with paths P'_1 and P'_2 between them. Notice that C has an odd number of vertices since P'_2 has an even number of vertices and P'_1 has odd number of vertices. This contradicts the fact that S' is a solution to Sim-OCT, as needed. □

As a consequence of the reduction presented above, we obtain the following theorem.

Theorem 12.3. *For all $\alpha \geq 2$,* Sim-OCT *is* W[1]-hard *when parameterized by k. Here, α is the number of colors in the coloring function of the edge set.*

Part IV

Conclusions

Chapter 13

Discussions and Open Problems

In Part II of the thesis we looked at $\mathcal{F}$-Editing problems where the edit operations were restricted to one of vertex deletion and edge contraction for various classes of chordal graphs. Below we briefly discuss these results, and some future directions.

- In Chapter 5, we studied the problem Block Graph Vertex Deletion. We designed an FPT algorithm running in time $\mathcal{O}(4^k|V(G)|^{\mathcal{O}(1)})$ and a polynomial kernel with $\mathcal{O}(k^4)$ vertices for the problem. Improving either the FPT algorithm or the kernelization algorithm for the problem remains as an interesting future direction.

- In Chapter 6, we designed $\mathcal{O}(\log^{\mathcal{O}(1)} n)$-approximation algorithms for Weighted Planar $\mathscr{F}$-Minor-Free Deletion, Weighted Chordal Vertex Deletion and Weighted Distance Hereditary Vertex Deletion. These algorithms are the first ones for these problems whose approximation factors are bounded by $\mathcal{O}(\log^{\mathcal{O}(1)} n)$. Along the way, we also obtained a constant-factor approximation algorithm for Weighted Multicut on chordal graphs. All these algorithms are based on the same recursive scheme. We believe that the scope of applicability of the approach used in this chapter is very wide. The following are some of the natural open problems that arise from this chapter.

  - Does Weighted Planar $\mathscr{F}$-Minor-Free Deletion admit a constant-factor approximation algorithm? Furthermore, studying the problem for families $\mathscr{F}$ that do not necessarily contain a planar graph is another direction for further research.
  - Does Weighted Chordal Vertex Deletion admit a constant-factor approximation algorithm?
  - Does Weighted Rankwidth-η Vertex Deletion admit a $\mathcal{O}(\log^{\mathcal{O}(1)} n)$-factor approximation algorithm?
  - On which other graph classes Weighted Multicut admits a constant-factor approximation?

- In Chapter 7, we designed a polynomial kernel for Chordal Vertex Deletion of size $\mathcal{O}(k^{12}\log^{10} k)$. This kernel significantly improves over the previously known $\mathcal{O}(k^{161}\log^{58} k)$ sized kernel. We believe that the notion of independence degree and the bootstrapping trick used in the kernelization procedure could be useful in designing polynomial kernels for other $\mathcal{F}$-Vertex (Edge) Deletion problems,

where $\mathcal{F}$ is characterized by an infinite set of forbidden induced graphs. The following are some of the natural open problems.

  - Design a polynomial kernel for Chordal Vertex Deletion of size $\mathcal{O}(k^c)$ for some fixed constant $c \leq 5$.
  - Does there exist an FPT algorithm for Chordal Vertex Deletion with running time $c^k n^{\mathcal{O}(1)}$, for some fixed constant c? We note that Marx [Mar10] designed an FPT algorithm for Chordal Vertex Deletion, which resolved an open problem of Cai [Cai03].

- In Chapter 8, we designed a polynomial kernel for Interval Vertex Deletion with $\mathcal{O}(k^{1740})$ vertices. We believe that the kernel size can be improved at the cost of significantly more involved arguments and by considering a solution of lower redundancy. But it seems like obtaining a kernel of size around $O(k^{10})$ would require new ideas. Designing a kernel with better size bound remains an interesting research direction. We believe that the idea of using solutions of higher redundancy is extendable in designing algorithms for other problems as well. Also, the technique of linear algebra that we use for bounding (for instance) the number of module components seems to have wider applicability for designing polynomial kernels for other problems.

- In Chapter 9, we considered the problem Split Contraction. We looked at two important results regarding the complexity of the problem. First, we proved that under the ETH, the problem cannot be solved in time $2^{o(\ell^2)} \cdot n^{\mathcal{O}(1)}$ where ℓ is the vertex cover number of the input graph, and this lower bound is tight. To the best of our knowledge, this is the first tight lower bound of the form $2^{o(\ell^2)} \cdot n^{\mathcal{O}(1)}$ for problems parameterized by the vertex cover number of the input graph. Second, we have proved that Split Contraction, despite its deceptive simplicity, is actually W[1]-hard with respect to the solution size. We believe that techniques integrated in the constructions can be used to derive conditional lower bounds and W[1]-hardness results in the context of other graph editing problems. In the exact setting, it is easy to see that Split Contraction can be solved in time $2^{\mathcal{O}(n \log n)}$. Can it be solved in time $2^{o(n \log n)}$? A negative answer would imply, for instance, that it is neither possible to find a topological clique minor in a given graph in time $2^{o(n \log n)}$, which is an interesting open problem [CFG+17].

In Part III of the thesis we looked at Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Editing problems. We restricted ourselves to edit operations being vertex or edge deletions. As mentioned by Cai and Ye [CY14], we believe that studying generalizations of other classical problems to edge-colored graphs might lead to interesting new insights about combinatorial and structural properties of such problems. Next, we briefly look at the results we obtained and some open problems.

- In Chapter 10, we studied the problem Simultaneous Feedback Vertex Set. We showed that the problem admits an FPT algorithm running in $\mathcal{O}^\star(23^{\alpha k})$ time, for any constant α. For the special case of $\alpha = 2$, we designed a faster $\mathcal{O}^\star(81^k)$ time algorithm which follows from the observation that the base case of the general algorithm can be solved in polynomial time when $\alpha = 2$. Moreover, for constant α,

we presented a kernel for the problem with $\mathcal{O}(\alpha k^{3(\alpha+1)})$ vertices. It is interesting to note that our algorithm implies that Simultaneous Feedback Vertex Set can be solved in $(2^{\mathcal{O}(\alpha)})^k n^{\mathcal{O}(1)}$ time. However, we have also seen that Simultaneous Feedback Vertex Set becomes W[1]-hard when $\alpha \in \mathcal{O}(\log n)$. This implies that (under plausible complexity assumptions) an algorithm running in $(2^{o(\alpha)})^k n^{\mathcal{O}(1)}$ time cannot exist. In other words, the running time cannot be subexponential in either k or α.

- In Chapter 11, we look at the problem Simultaneous Feedback Edge Set. We show that for $\alpha = 3$, Simultaneous Feedback Edge Set is NP-hard by giving a reduction from Vertex Cover on cubic graphs. The same reduction shows that the problem does not admit an algorithm running in time $\mathcal{O}(2^{o(k)} n^{\mathcal{O}(1)})$ unless ETHfails. This hardness result is complimented by an FPT algorithm for Simultaneous Feedback Edge Set running in time $\mathcal{O}(2^{\omega k\alpha+\alpha \log k} n^{\mathcal{O}(1)})$, where ω is the exponent in the running time of matrix multiplication. The same algorithm gives a polynomial time algorithm for the case when $\alpha = 2$. We also give a kernel for the problem with $(k\alpha)^{\mathcal{O}(\alpha)}$ vertices. Finally, we consider the problem Max-Sim Acyclic-Subgraph. We give an FPT algorithm for Max-Sim Acyclic-Subgraph running in time $\mathcal{O}(2^{\omega q\alpha} n^{\mathcal{O}(1)})$.

- In Chapter 12, we studied the problem Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion problem, where each $\mathcal{F}_i$ is either the family of forests or the family of bipartite graphs. The work upon which the chapter is based initiated the investigation of the complexity of Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion with different families of graphs. We obtained a complete characterization of the Parameterized Complexity of the problem when one or more of the $\mathcal{F}_i's$ is the class of bipartite graphs and the rest (if any) are forests. We showed that if $\mathcal{F}_1$ is the family of bipartite graphs and each of $\mathcal{F}_2 = \mathcal{F}_3 = \ldots = \mathcal{F}_\alpha$ is the family of forests then the problem is FPT when parameterized by k and α. However, even when $\mathcal{F}_1$ and $\mathcal{F}_2$ are both the family of bipartite graphs, then the Simultaneous $(\mathcal{F}_1, \mathcal{F}_2)$-Deletion problem itself is already W[1]-hard. The following are some of the open problems arising from the chapter.

  - Towards designing the FPT algorithm we designed an FPT algorithm for the problem Colorful Multiway Cut. It is natural to ask whether one can improve the running time of the algorithm for Colorful Multiway Cut. In particular, is it possible to solve the problem in $\mathcal{O}^\star(k^{\mathcal{O}(k)})$ time when the number of terminals is constant and the number of colors is at most k?
  - Another interesting question which remains open is whether the Simultaneous FVS/OCT problem admits a (randomized) polynomial kernel.

Bibliography

[ACF+04] Faisal N. Abu-Khzam, Rebecca L. Collins, Michael R. Fellows, Michael A. Langston, W. Henry Suters, and Christopher T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In *Proceedings of the Sixth Workshop on Algorithm Engineering and Experiments and the First Workshop on Analytic Algorithmics and Combinatorics (ALENEX/ANALC)*, pages 62–69, 2004. 1.1

[AGK+11] Noga Alon, Gregory Gutin, Eun Jung Kim, Stefan Szeider, and Anders Yeo. Solving MAX-r-SAT above a tight lower bound. *Algorithmica*, 61(3):638–655, 2011. 1.1

[Agr17a] Akanksha Agrawal. Fine-grained complexity of rainbow coloring and its variants. In *(to appear) 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2017. 1

[Agr17b] Akanksha Agrawal. On the parameterized complexity of happy vertex coloring. In *(to appear) 28th International Workshop on Combinatorial Algorithms (IWOCA)*, 2017. 2

[AH83] Takao Asano and Tomio Hirata. Edge-contraction problems. *Journal of Computer and System Sciences*, 26(2):197–208, 1983. 1.1

[AK10] Faisal N. Abu-Khzam. A kernelization algorithm for d-Hitting Set. *Journal of Computer and System Sciences*, 76(7):524–531, 2010. 1.1

[AKL+17] Akanksha Agrawal, R. Krithika, Daniel Lokshtanov, Amer E. Mouawad, and M.S. Ramanujan. On the parameterized complexity of simultaneous deletion problems. In *Manuscript*, 2017. 3, 1.3

[AKLS16] Akanksha Agrawal, Sudeshna Kolay, Daniel Lokshtanov, and Saket Saurabh. A faster FPT algorithm and a smaller kernel for block graph vertex deletion. In *Theoretical Informatics - 12th Latin American Symposium (LATIN)*, pages 1–13, 2016. 7, 1.3

[AKST17] Akanksha Agrawal, Lawqueen Kanesh, Saket Saurabh, and Prafullkumar Tale. Paths to trees and cacti. In *Algorithms and Complexity - 10th International Conference (CIAC)*, pages 31–42, 2017. 1.1

[ALM+17a] Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Feedback vertex set inspired kernel for chordal vertex deletion. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1383–1398, 2017. 5, 1.3, 6.2, 8.4

[ALM+17b] Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Interval vertex deletion admits a polynomial kernel. In *Manuscript*, 2017. 1, 1.3

[ALM+17c] Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Polylogarithmic approximation algorithms for weighted-$\mathcal{F}$-deletion problems. In *Manuscript*, 2017. 2, 1.3

[ALMS16] Akanksha Agrawal, Daniel Lokshtanov, Amer E. Mouawad, and Saket Saurabh. Simultaneous feedback vertex set: A parameterized perspective. In *33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 7:1–7:15, 2016. 8, 1.3, 12

[ALSZ17] Akanksha Agrawal, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Split contraction: The untold story. In *34th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 5:1–5:14, 2017. 4, 1.3

[APSZ16] Akanksha Agrawal, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Simultaneous feedback edge set: A parameterized perspective. In *27th International Symposium on Algorithms and Computation (ISAAC)*, pages 5:1–5:13, 2016. 6, 1.3

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. 3.2.3

[BBF99] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999. 1.1, 5.2, 6.1, 10.2, 10.2.2

[BBF+01] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001. 8

[BBL13] Hans L. Bodlaender, Paul S. Bonsma, and Daniel Lokshtanov. The fine details of fast dynamic programming over tree decompositions. In *Parameterized and Exact Computation - 8th International Symposium (IPEC)*, pages 41–53, 2013. 3.2.2

[BCK+16] Ivan Bliznets, Marek Cygan, Pawel Komosa, Lukás Mach, and Michal Pilipczuk. Lower bounds for the parameterized complexity of minimum fill-in and other completion problems. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1132–1151, 2016. 1.1

[BDD+16] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A c^k n 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016. 3.2.2

[BDFH09] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. 1.1

[BFL+16] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (meta) kernelization. *Journal of the ACM*, 63(5):44:1–44:69, 2016. 1.1

[BJK13] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Preprocessing for treewidth: A combinatorial analysis through kernelization. *SIAM Journal on Discrete Mathematics*, 27(4):2108–2142, 2013. 1.1

[BLS99] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999. 5, 8, 8.4

[Bol65] Béla Bollobás. On generalized graphs. *Acta Mathematica Hungarica*, 16(3-4):447–452, 1965. 8

[BPT92] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(5&6):555–581, 1992. 6.1.1

[BRU17] Nikhil Bansal, Daniel Reichman, and Seeun William Umboh. LP-based robust algorithms for noisy minor-free and bounded treewidth graphs. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1964–1979, 2017. 1.3, 6, 6.4

[BYE81] Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981. 1.1, 6.1

[BYGNR98] Reuven Bar-Yehuda, Dan Geiger, Joseph Naor, and Ron M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM Journal on Computing*, 27(4):942–959, 1998. 1.1, 6.1

[Cai96] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996. 1.1, 1.1

[Cai03] Leizhen Cai. Parameterized complexity of vertex colouring. *Discrete Applied Mathematics*, 127(3):415–429, 2003. 13

[Cao16] Yixin Cao. Linear recognition of almost interval graphs. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1096–1115, 2016. 1.1, 8, 8, 8.1

[Cao17] Yixin Cao. Unit interval editing is fixed-parameter tractable. *Information and Computation*, 253:109–126, 2017. 1.1

[CFG+17] Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socala. Tight lower bounds on graph embedding problems. *Journal of the ACM*, 64(3):18:1–18:22, 2017. 2.1, 4.2.1, 13

[CFK+15] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. 1.1, 2.1, 2.4, 3.2.1, 1, 3.2.1, 3.6, 3.2.2, 3.2.3, 4.1.2, 9, 10, 10.1, 26, 10.2, 11, 11.3.1, 12, 12.1, 12.3.1

[CFL+08] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188 – 1198, 2008. 3.2.1

[CG13] Leizhen Cai and Chengwei Guo. Contracting few edges to remove forbidden induced subgraphs. In *Parameterized and Exact Computation - 8th International Symposium (IPEC)*, pages 97–109, 2013. 1.1

[CKJ01] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001. 1.1

[CKP13] Marek Cygan, Łukasz Kowalik, and Marcin Pilipczuk. Open problems from workshop on kernels. *URL: http://worker2013.mimuw.edu.pl/slides/worker-opl.pdf*, 2013. 1.1, 8

[CKX10] Jianer Chen, Iyad A Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40):3736–3756, 2010. 9.3.1, 9.3.1

[CM15a] Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(3):21, 2015. 1.1, 8

[CM15b] Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(3):21:1–21:35, 2015. 8.2, 8.2

[CM16] Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016. 1.1, 7, 7.1

[CPP16] Marek Cygan, Marcin Pilipczuk, and Michal Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM Journal on Computing*, 45(1):67–83, 2016. 9

[CSRL01] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. 2

[CY14] Leizhen Cai and Junjie Ye. Dual connectedness of edge-bicolored graphs and beyond. In *39th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 8635, pages 141–152, 2014. 1.2, 13

[DDLS15] Pål Grønås Drange, Markus Sortland Dregi, Daniel Lokshtanov, and Blair D. Sullivan. On the threshold of intractability. In *Algorithms - 23rd Annual European Symposium (ESA)*, pages 411–423, 2015. 1.1

[DF92] R. G. Downey and M. R. Fellows. Fixed-parameter intractability. In *Proceedings of the 7th Annual Structure in Complexity Theory Conference*, pages 36–49, 1992. 4.1.2

[DF95] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1&2):109–131, 1995. 9.2

[DF13] Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized complexity*. Springer-Verlag, 2013. 1.1

[DFPV15] Pål Grønås Drange, Fedor V. Fomin, Michal Pilipczuk, and Yngve Villanger. Exploring the subexponential complexity of completion problems. *ACM Transactions on Computation Theory*, 7(4):14:1–14:38, 2015. 1.1

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. 2

[DP15] Pål Grønås Drange and Michal Pilipczuk. A polynomial kernel for trivially perfect editing. In *Algorithms - 23rd Annual European Symposium (ESA)*, pages 424–436, 2015. 1.1

[DvM14] Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *Journal of the ACM*, 61(4):23:1–23:27, 2014. 1.1

[Edw97] Keith Edwards. The harmonious chromatic number and the achromatic number. *Surveys in Combinatorics*, pages 13–48, 1997. 2.1, 4.2.1

[Far89] M Farber. On diameters and radii of bridged graphs. *Discrete Mathematics*, 73:249–260, 1989. 6, 6.12

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. 1.1

[FHL08] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008. 2, 6.1.2, 6.2.2, 6.4.2

[FHRV09] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical computer science*, 410(1):53–61, 2009. 4.1.3, 9.2.2, 12, 12.4.1

[FJP10] Samuel Fiorini, Gwenaël Joret, and Ugo Pietropaoli. Hitting diamonds and growing cacti. In *Proceedings of the 14th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 6080, pages 191–204, 2010. 6.1

[FJP14] Fedor V. Fomin, Bart M. P. Jansen, and Michal Pilipczuk. Preprocessing subgraph and minor problems: When does a small vertex cover help? *Journal of Computer and System Sciences*, 80(2):468–495, 2014. 8

[FKP+14] Fedor V Fomin, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *Journal of Computer and System Sciences*, 80(7):1430–1447, 2014. 1.1

[FLM+16] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. Hitting forbidden minors: Approximation and kernelization. *SIAM Journal on Discrete Mathematics*, 30(1):383–410, 2016. 6.1, 6.4

[FLMS12] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-Deletion: Approximation, Kernelization and Optimal FPT algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, (FOCS)*, pages 470–479, 2012. 1.1, 6.1, 6.1.1

[FLPS16] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *Journal of the ACM*, 63(4):29:1–29:60, 2016. 2.1, 8

[FLRS11] Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 748–759, 2011. 6.1, 6.4

[FLS12] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Bidimensionality and geometric graphs. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1563–1575, 2012. 6.1, 6.4

[FLST10] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 503–510, 2010. 1.1, 6.4

[FS11] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *Journal of Computer and System Sciences*, 77(1):91–106, 2011. 1.1

[FSV13] Fedor V. Fomin, Saket Saurabh, and Yngve Villanger. A polynomial kernel for proper interval vertex deletion. *SIAM Journal on Discrete Mathematics*, 27(4):1964–1976, 2013. 1.1

[Fuj98] Toshihiro Fujito. A unified approximation algorithm for node-deletion problems. *Discrete Applied Mathematics*, 86:213–231, 1998. 1.1

[GC15] Chengwei Guo and Leizhen Cai. Obtaining split graphs by edge contraction. *Theoretical Computer Science*, 607:60–67, 2015. 9

[GGH+06] Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386 – 1396, 2006. 12, 12.2, 12.1

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979. 4.2.1

[GJLS15] Archontia C. Giannopoulou, Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. Uniform kernelization complexity of hitting forbidden minors. In *Automata, Languages, and Programming - 42nd International Colloquium, (ICALP)*, pages 629–641, 2015. 1.1

[GKK+15] Esha Ghosh, Sudeshna Kolay, Mrinal Kumar, Pranabendu Misra, Fahad Panolan, Ashutosh Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015. 1.1

[GM09] Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *Journal of Combinatorial Theory, Series B*, 99(1):218 – 228, 2009. 10.5

[GM13] Sylvain Guillemot and Dániel Marx. A faster FPT algorithm for bipartite contraction. *Information Processing Letters*, 113(22–24):906–912, 2013. 1.1

[GNS06] Daniel Golovin, Viswanath Nagarajan, and Mohit Singh. Approximating the k-multicut problem. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 621–630, 2006. 6.3

[Gol04] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004. 2.2, 2.3, 6.2.1, 6.2.1, 7.3, 7.4.1, 8, 8.4, 8.4.1, 8.4.1, 9.3.1

[GRS17] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. *ACM Transactions on Algorithms*, 13(2):29:1–29:32, 2017. 12

[GvtHP13] Petr A. Golovach, Pim van 't Hof, and Daniel Paulusma. Obtaining planarity by contracting few edges. *Theoretical Computer Science*, 476:38–46, 2013. 1.1

[GVY96] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996. 6.3, 6.3, 6.5

[HM90] Peter L Hammer and Frédéric Maffray. Completely separable graphs. *Discrete applied mathematics*, 27(1):85–99, 1990. 6.4

[How77] Edward Howorka. A characterization of distance-hereditary graphs. *The quarterly journal of mathematics*, 28(4):417–420, 1977. 6.4

[How81] Edward Howorka. A characterization of ptolemaic graphs. *Journal of Graph Theory*, 5(3):323–331, 1981. 5

[HvtHJ+13] Pinar Heggernes, Pim van 't Hof, Bart M. P. Jansen, Stefan Kratsch, and Yngve Villanger. Parameterized complexity of vertex deletion into perfect graph classes. *Theoretical Computer Science*, 511:172–180, 2013. 1.1

[HvtHL+14] Pinar Heggernes, Pim van 't Hof, Benjamin Lévêque, Daniel Lokshtanov, and Christophe Paul. Contracting graphs to paths and trees. *Algorithmica*, 68(1):109–132, 2014. 1.1

[HvtHLP13] Pinar Heggernes, Pim van 't Hof, Daniel Lokshtanov, and Christophe Paul. Obtaining a bipartite graph by contracting few edges. *SIAM Journal on Discrete Mathematics*, 27(4):2143–2156, 2013. 1.1

[IK17] Satoru Iwata and Yusuke Kobayashi. A weighted linear matroid parity algorithm. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 264–276, 2017. 2, 3.2.1, 3.2.1

[IOY14] Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-time FPT algorithms via network flow. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1749–1761, 2014. 12, 12.2

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. 4.1, 4.2, 4.2, 4.2.1, 9, 9.1

[Jan13] Bart MP Jansen. *The power of data reduction: Kernels for Fundamental Graph Problems*. PhD thesis, Utrecht University, 2013. 8

[Jan17] Bart M. P. Jansen. Turing kernelization for finding long paths and cycles in restricted graph classes. *Journal of Computer and System Sciences*, 85:18–37, 2017. 1.1

[JP16] Bart M. P. Jansen and Marcin Pilipczuk. Approximation and kernelization for chordal vertex deletion. *CoRR*, abs/1605.03001, 2016. 8

[JP17] Bart M. P. Jansen and Marcin Pilipczuk. Approximation and kernelization for chordal vertex deletion. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1399–1418, 2017. 1.1, 1.3, 6.2, 7, 7.1, 7.1, 7.4, 7.20, 8, 8.4

[KCOW16] Yuping Ke, Yixin Cao, Xiating Ouyang, and Jianxin Wang. Unit interval vertex deletion: Fewer vertices are relevant. *arXiv preprint arXiv:1607.01162*, 2016. 1.1

[Ken69] David Kendall. Incidence matrices, interval graphs and seriation in archeology. *Pacific Journal of mathematics*, 28(3):565–570, 1969. 8

[KK15] Eun Jung Kim and O-joung Kwon. A polynomial kernel for block graph deletion. In *10th International Symposium on Parameterized and Exact Computation, (IPEC)*, pages 270–281, 2015. 5

[KK16] E. J. Kim and O. Kwon. A Polynomial Kernel for Distance-Hereditary Vertex Deletion. *ArXiv e-prints*, 2016. 6.4, 6.30, 6.4.2, 6.32

[KK17] Eun Jung Kim and O-joung Kwon. A polynomial kernel for block graph deletion. *Algorithmica*, 79(1):251–270, 2017. 1.3, 5, 5, 5.3, 5.1, 5.3, 5.3

[KL16] Lukasz Kowalik and Juho Lauri. On finding rainbow and colorful paths. *Theoretical Computer Science*, 628(C):110–114, 2016. 3.2.3, 3.2.3, 3.2.3, 3.2.3, 3.2.3

[Klo94] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. 3.6

[KLP+15] Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Transactions on Algorithms*, 12(2):21:1–21:41, 2015. 1.1

[Kom15] Christian Komusiewicz. Tight running time lower bounds for vertex deletion problems. *arXiv preprint arXiv:1511.05449*, 2015. 4.2.1, 9.1, 11.2

[KP14] Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556–560, 2014. 3.2.1, 3.2.1

[Kra12] Stefan Kratsch. Polynomial kernelizations for MIN $F^+\Pi_1$ and MAX NP. *Algorithmica*, 63(1-2):532–550, 2012. 1.1

[Kra14] Stefan Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113, 2014. 1.1

[KT05] J. Kleinberg and E. Tardos. *Algorithm design*. Addison-Wesley, 2005. 6.2.2, 6.4.2

[KW12] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 450–459, 2012. 1.1, 8

[KW14] Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms*, 10(4):20:1–20:15, 2014. 1.1

[Lay06] David C. Lay. *Linear algebra and its applications*. Linear Algebra and Its Applications. Pearson/Addison-Wesley, 2006. 2

[LB62] C. G. Lekkeikerker and J. Ch. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962. 8, 8

[LM87] Sin-Min Lee and John Mitchem. An upper bound for the harmonious chromatic number. *Journal of Graph Theory*, 11(4):565–567, 1987. 2.1, 4.2.1

[LMPS15] Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP*, pages 922–934, 2015. 2.6, 2.7

[LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 760–776, 2011. 9

[LMS12] Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization - preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, pages 129–161, 2012. 1.1

[LMS13] Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. On the hardness of eliminating small induced subgraphs by contracting edges. In *Parameterized and Exact Computation - 8th International Symposium (IPEC)*, pages 243–254, 2013. 1.1

[Lok08] Daniel Lokshtanov. Wheel-free deletion is w[2]-hard. In *Parameterized and Exact Computation, Third International Workshop, IWPEC*, pages 141–147, 2008. 10.3

[Lov72] László Lovász. A characterization of perfect graphs. *Journal of Combinatorial Theory, Series B*, 13(2):95–98, 1972. 8.4.1

[Lov80] László Lovász. Matroid matching and some applications. *Journal of Combinatorial Theory, Series B*, 28(2):208–236, 1980. 12

[LR99] Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46:787—-832, 1999. 6, 2, 6.1.2, 6.2.2, 6.4.2

[LR12] Daniel Lokshtanov and M. S. Ramanujan. Parameterized Tractability of Multiway Cut with Parity Constraints. In *39th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 750–761, 2012. 12, 12.3.1

[LRS16] Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. A linear time parameterized algorithm for directed feedback vertex set. *ArXiv e-prints*, 2016. 12, 12.2, 12.3.1, 12.4

[LY80] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *Journal of Computer and System Sciences*, 20(2):219 – 230, 1980. 1.1, 5

[LY93] Carsten Lund and Mihalis Yannakakis. The approximation of maximum subgraph problems. In *Proceedings of the 20nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 700, pages 40–51, 1993. 1.1

[LY94] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994. 1.1

[Mar07] Dániel Marx. Can you beat treewidth? In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 169–179, 2007. 10.5

[Mar09] Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009. 2, 2.7, 8

[Mar10] Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010. 1.1, 7, 7.4, 7.4, 7.4.1, 7.21, 7.4.1, 7.22, 7.23, 7.4.1, 7.24, 7.4.1, 7.25, 7.26, 7.4.2, 7.27, 7.4.2, 7.28, 7.4.2, 8, 13

[MM65] John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965. 6

[MM16] Dániel Marx and Valia Mitsou. Double-exponential and triple-exponential bounds for choosability problems parameterized by treewidth. In *43rd International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 28:1–28:15, 2016. 9

[Moh01] Bojan Mohar. Face covers and the genus problem for apex graphs. *Journal of Combinatorial Theory, Series B*, 82(1):102–117, 2001. 11, 11.2

[MOR13] Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30:1–30:35, 2013. 1.1, 12

[MRRS12] Pranabendu Misra, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Parameterized algorithms for even cycle transversal. In *Graph-Theoretic Concepts in Computer Science - 38th International Workshop, (WG)*, pages 172–183, 2012. 10.7

[MV80] S. Micali and V. V. Vazirani. An $\mathcal{O}(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science*, 1980. 2

[MX91] Colin McDiarmid and Luo Xinhua. Upper bounds for harmonious coloring. *Journal of Graph Theory*, 15(6):629–636, 1991. 2.1, 4.2.1

[Nie06] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006. 1.1

[NJ74] George L. Nemhauser and Leslie E. Trotter Jr. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6:48–61, 1974. 1.1, 6.1

[NSS95] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 182–191, 1995. 3.2.3

[OS06] Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006. 6.4

[Oum05] Sang-il Oum. Rank-width and vertex-minors. *Journal of Combinatorial Theory, Series B*, 95(1):79–100, 2005. 6.4

[Oxl06] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2006. 2, 2, 2.4, 2.5, 8.4.2

[PPSvL14] Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for steiner problems on planar and bounded-genus graphs. In *55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 276–285, 2014. 1.1

[RS86] Neil Robertson and Paul D Seymour. Graph minors. v. excluding a planar graph. *Journal of Combinatorial Theory Series B*, 41(1):92–114, 1986. 6.1, 6.1

[RS95] Neil Robertson and Paul D. Seymour. Graph minors .xiii. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. 6.1.2

[RS04] Neil Robertson and Paul D. Seymour. Graph minors. XX. wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004. 6.1

[RS14] M. S. Ramanujan and Saket Saurabh. Linear time parameterized algorithms via skew-symmetric multicuts. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1739–1748, 2014. 12, 12.2

[RSV04] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004. 3.2, 10, 10.1, 12

[Sac70] Horst Sachs. On the berge conjecture concerning perfect graphs. *Combinatorial Structures and their Applications*, 37:384, 1970. 6.4

[Sav82] Carla D. Savage. Depth-first search and the vertex cover problem. *Information Processing Letters*, 14(5):233–237, 1982. 9.3.1

[Tho10] Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2):32:1–32:8, 2010. 1.1, 3.2.1, 7

[TIAS77] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977. 6, 6.12

[UAI+13] Kei Uchizawa, Takanori Aoki, Takehiro Ito, Akira Suzuki, and Xiao Zhou. On the rainbow connectivity of graphs: Complexity and fpt algorithms. *Algorithmica*, 67(2):161–179, 2013. 3.2.3

[Viz64] Vadim G Vizing. On an estimate of the chromatic class of a p-graph. *Diskret Analiz*, 3:25–30, 1964. 10.3

[WAN81] Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the removal of forbidden graphs by edge-deletion or by edge-contraction. *Discrete Applied Mathematics*, 3(2):151–153, 1981. 1.1

[WAN83] Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the NP-hardness of edge-deletion and-contraction problems. *Discrete Applied Mathematics*, 6(1):63–78, 1983. 1.1

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the 44th annual ACM symposium on Theory of computing*, pages 887–898, 2012. 2

[Yan78] Mihalis Yannakakis. Node- and edge-deletion np-complete problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, pages 253–264, 1978. 1.1

[Yan79] Mihalis Yannakakis. The effect of a connectivity requirement on the complexity of maximum subgraph problems. *Journal of the ACM*, 26(4):618–630, 1979. 1.1

[Yan94] Mihalis Yannakakis. Some open problems in approximation. In *Proceedings of 2nd Italian Conference on Algorithms and Complexity, Second (CIAC)*, pages 33–39, 1994. 1.1

[Ye15] Junjie Ye. A note on finding dual feedback vertex set. *CoRR*, abs/1510.00773, 2015. 10, 12

[ZL15] Peng Zhang and Angsheng Li. Algorithmic aspects of homophyly of networks. *Theoretical Computer Science*, 593:117–131, 2015. 3