Technische Universität Berlin
Electrical Engineering and Computer Science
Institute of Software Engineering and Theoretical Computer Science
Algorithmics and Computational Complexity (AKT)

Parameterized Complexity of Modifying Graphs to be Biclique-free

Lito Julius Goldmann

Thesis submitted in fulfillment of the requirements for the degree
"Master of Science" (M. Sc.) in the field of Computer Science

September 2021

Supervisor and first reviewer:	Prof. Dr. Rolf Niedermeier
Second reviewer:	Prof. Dr. Markus Brill
Co-Supervisors:	Leon Kellerhals and Tomohiro Koana

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

The independent and unaided completion of the thesis is affirmed by affidavit:

Berlin,	Date	Signature

Abstract

We study the parameterized complexity of the two NP-hard graph modification problems Biclique-Free Vertex Deletion and Biclique-Free Edge Deletion parameterized by different structural graph parameters. The task in those two problems is to determine if there exists a size-k subset of either vertices or, depending on the problem variant, edges whose deletion from the input graph yields a graph that does not contain any biclique with i vertices on one side and j vertices on the other side as a (not necessarily induced) subgraph. Both variants generalize well-studied NP-hard problems, e.g. Vertex Cover, Bounded-Degree Deletion and k-Biclique. We prove that Biclique-Free Vertex Deletion and Biclique-Free Edge Deletion parameterized by the vertex cover number are fixed-parameter tractable (FPT). Furthermore, we provide a linear-size problem kernel for both problems parameterized by the feedback edge set number. To the best of our knowledge, no polynomial- or linear-size problem kernel for the special case Bounded-Degree Deletion parameterized by the feedback edge set number was known until now. This was one of the few remaining open questions regarding the parameterized complexity of Bounded-Degree Deletion with respect to structural graph parameters. We also show that Biclique-Free Vertex Deletion is FPT when parameterized by the distance to disjoint paths and that Biclique-Free Edge Deletion is W[1]-hard when parameterized by the feedback vertex set number or the treedepth. We conclude that the parameterized complexity of Biclique-Free Vertex Deletion and Biclique-Free Edge Deletion is very similar with respect to the studied structural graph parameters. While the parameterized complexity of Biclique-Free Vertex Deletion with $i = 1$ does not differ from the parameterized complexity of Biclique-Free Vertex Deletion with $i \geq 2$ for the studied parameters, the results for $i = 1$ are significantly more involved.

Zusammenfassung

Abstract in German Language

Wir untersuchen die parametrisierte Komplexität der NP-schweren Graphenprobleme Biclique-Free Vertex Deletion und Biclique-Free Edge Deletion für verschiedene strukturelle Graphparameter. Die Aufgabenstellung beider Probleme ist, zu entscheiden, ob eine Teilmenge von höchstens k Knoten bzw. Kanten existiert, welche nach ihrer Löschung vom Eingabegraphen einen Graphen hinterlässt, welcher keinen vollständigen bipartiten Graphen mit Partitionen der Größe i und j als (beliebigen) Teilgraphen enthält. Beide Probleme verallgemeinern bekannte NP-schwere Probleme, wie zum Beispiel Vertex Cover, Bounded-Degree Deletion und k-Biclique. Wir zeigen, dass Biclique-Free Vertex Deletion und Biclique-Free Edge Deletion parametrisierbar (FPT) in der Vertex Cover-Größe sind. Außerdem präsentieren wir einen Problemkern linearer Größe für beide Probleme mit dem Parameter Feedback Edge Set-Größe. Nach unserem Kenntnisstand war bis jetzt kein Problemkern polynomieller oder linearer Größe für den Spezialfall Bounded-Degree Deletion mit dem Parameter Feedback Edge Set-Größe bekannt. Dies war eine der wenigen offenen Fragen bezüglich der parametrisierten Komplexität von Bounded-Degree Deletion mit strukturellen Graphparametern. Des Weiteren zeigen wir, dass Biclique-Free Vertex Deletion parametrisierbar für den Parameter Distanz zu disjunkten Pfaden (distance to disjoint paths) ist, und Biclique-Free Edge Deletion für die Parameter Feedback Vertex Set-Größe und Baumtiefe (treedepth) W[1]-schwer ist. Wir schlussfolgern, dass die parametrisierte Komplexität von Biclique-Free Vertex Deletion und Biclique-Free Edge Deletion, zumindest bei den untersuchten Parametern, sehr ähnlich ist. Obwohl die parametrisierte Komplexität von Biclique-Free Vertex Deletion mit $i = 1$ sich für die untersuchten Parameter nicht von der parametrisierten Komplexität von Biclique-Free Vertex Deletion mit $i \geq 2$ unterscheidet, sind die Ergebnisse für $i = 1$ erheblich komplizierter.

Contents

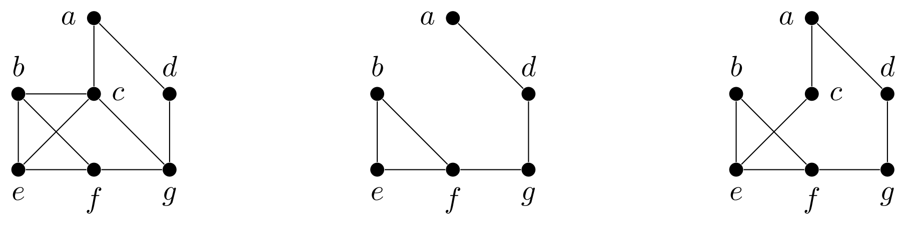
Chapter 1

Introduction

Graph modification problems received a lot of attention in the last decades, as many practical problems can be transformed into some kind of graph modification problem. The task in such a problem is to modify the input graph, for example by adding or deleting vertices or edges, such that it fulfills some problem-specific property and was modified as little as possible. The graph modification problems where one deletes vertices and edges are known as vertex deletion problems and edge deletion problems, respectively. Some of the most studied problems are in fact graph modification problems, e.g. the NP-hard Vertex Cover asks to delete a minimum number of vertices to obtain an edge-free graph.

In this thesis, we study the vertex and edge deletion problems where one wants to obtain a biclique-free graph, that is, a graph that does not contain bicliques of given size as a (not necessarily induced) subgraph. A biclique is a complete bipartite graph and usually denoted by $K_{i,j}$, where i and j are the sizes of its two sides. An example instance of the two problems Biclique-Free Vertex Deletion and Biclique-Free Edge Deletion is shown in Figure 1.1. Note that the problem of adding vertices or edges to obtain a biclique-free graph is not meaningful, because modifying the input graph by adding vertices or edges will not yield a biclique-free graph. The class of biclique-free graphs contains other important graph classes like the class of nowhere dense graphs and the class of degenerate graphs [PRS12; TV19]. Utilizing the fact that a graph is biclique-free brings certain benefits, e.g. Dominating Set can be approximated better and solved more efficiently on biclique-free graphs than on general graphs [Sie19; TV19].

Both Biclique-Free Vertex Deletion and Biclique-Free Edge Deletion are known to be NP-hard [LY80; Yan78]. Studying ways to obtain efficient algorithms for NP-hard problems utilizing certain structures in the input lead to the topic of parameterized algorithmics and complexity. Hence it is rather natural to study the parameterized complexity of the two Biclique-Free Deletion variants using well-established methods from this area that were already successful for other problems. To the best of our knowledge, however, graph modification into biclique-free graphs has not been studied until now, even though it generalizes many other important and well-studied graph modification problems like Vertex Cover, Bounded-Degree Deletion, and C_4-Free Deletion, and is very similar to e.g. Induced-Claw-Free Deletion and Cluster Deletion. We will discuss related work including results regarding those



(a) Input graph containing the $K_{2,2}$ as a subgraph. (b) Optimal output graph for vertex deletion. (c) Optimal output graph for edge deletion.

Figure 1.1: Example instance of $K_{2,2}$-Free Deletion. The input graph in (a) contains the $K_{2,2}$ as a subgraph in the following four vertex subsets: $\{a, c, g, d\}$, $\{b, c, g, f\}$, $\{b, c, e, f\}$ and $\{c, e, f, g\}$. In (b) the vertex c was deleted. In (c) the edges $\{b, c\}$ and $\{c, g\}$ were deleted.

well-studied special cases in the next paragraph. We believe it is of strong theoretical interest to find out which of those results also apply for the two more general Biclique-Free Deletion variants or under which circumstances making graphs biclique-free is a more complex problem.

Related work. There are many general results about graph modification problems that also apply for the two variants of Biclique-Free Deletion. Lewis and Yannakakis [LY80] showed that all nontrivial vertex deletion problems into graphs fulfilling any hereditary property[1] are NP-hard. Hence Biclique-Free Vertex Deletion is NP-hard. Yannakakis [Yan78] showed that the problem of deleting edges such that the resulting graph is without any cycles of fixed length ℓ, with $\ell \geq 3$, is NP-complete. As a cycle of length four $C_4 = K_{2,2}$, this special case of Biclique-Free Edge Deletion is NP-complete. Cai [Cai96] showed that vertex and edge deletion into a graph fulfilling a property with a finite forbidden set characterization is fixed-parameter tractable (FPT) when parameterized by the budget of allowed vertex and edge deletions. Hence both variants of Biclique-Free Deletion with a fixed biclique size are FPT when parameterized by the budget. Some graph modification problems without finite forbidden set characterization are known to be FPT when parameterized by the budget, e.g. Chordal Deletion [Mar10], but some are presumably not in FPT when parameterized by the budget, e.g. Wheel-Free Deletion is W[2]-hard [Lok08]. A recent survey by Crespelle et al. [Cre+20] overviews the parameterized complexity of many different edge deletion problems when parameterized by the budget.

Existing work regarding bicliques mainly focuses on the detection of bicliques of certain sizes in graphs. Deciding whether a graph contains the biclique $K_{k,k}$ as a subgraph is W[1]-hard when parameterized by k [Lin18]. Testing if a bipartite graph contains a biclique with at least k edges, known as the Maximum Edge Biclique Problem in bipartite graphs, is NP-complete [Pee03]. Alexe et al. [Ale+04] presented algorithms for generating all maximal bicliques in a graph and Alzahrani and Horadam [AH19] used maximal bicliques in bipartite graphs to detect communities.

[1]A property is hereditary if all induced subgraphs of a graph satisfying the property also satisfy the property [LY80].

Further research had been done regarding bicliques restricted in size. Cygan et al. [Cyg+14] studied H-Free Vertex Deletion for fixed, general graphs H with respect to treewidth and showed that $K_{2,j}$-Free Vertex Deletion for fixed j can be solved in $\mathcal{O}^*(2^{\mathcal{O}(\omega^2 \log \omega)})$ time[2], where ω denotes the treewidth of the input graph. Various related problems regarding the deletion of certain *induced* subgraphs have been studied. To clearly distinguish those problems we add the prefix Induced to their names. Recall that in Biclique-Free Deletion one is asked to delete *all* subgraphs that are bicliques, i.e. not only the induced ones. Cygan et al. [Cyg+17] showed that Induced-{Claw, Diamond}-Free Edge Deletion admits a polynomial-size problem kernel when parameterized by the budget. The claw is the $K_{1,3}$ and the diamond contains the $K_{1,3}$ as a subgraph. Bonomo-Braberman et al. [BB+20] studied Induced-Claw-Free Vertex Deletion and showed that it is FPT when parameterized by treewidth. Sau and Souza [SS20] generalized this result by showing that Induced-H-Free Vertex Deletion for any fixed graph H containing at least three vertices is FPT parameterized by treewidth. Böcker and Damaschke [BD11] improved existing algorithms for Cluster Deletion, that is, Induced-$K_{1,2}$-Free Edge Deletion, parameterized by the budget.

The problem of deleting at most k vertices to obtain a graph where every vertex has degree at most d is known as the Bounded-Degree Deletion (BDD) problem, which is the special case $K_{1,d+1}$-Free Vertex Deletion of Biclique-Free Vertex Deletion. It is NP-complete in general [LY80], but FPT for fixed d when parameterized by k [Cai96]. If d is not a constant, then it is W[2]-complete when parameterized by k [Fel+11]. Regarding structural graph parameters, it is known that BDD is FPT when parameterized by the feedback edge set number [Bet+12], the vertex cover number [GKO21], the combined parameter treewidth plus d [Cou90], or the combined parameter treewidth plus k [Bet+12]. When parameterized by treewidth alone it is in XP [DJL93] and W[1]-hard [Bet+12]. It is also W[1]-hard when parameterized by the feedback vertex set number or the treedepth [GKO21].

Our contributions. We study the parameterized complexity for both variants of Biclique-Free Deletion with respect to structural graph parameters. We start with Biclique-Free Vertex Deletion. As we do not exclude the special case BDD, we can only hope to find FPT algorithms for parameters that are larger than (or incomparable to) the feedback vertex set number and the treedepth, because BDD parameterized by these parameters is known to be W[1]-hard [GKO21]. Hence we analyze the parameterized complexity with respect to the vertex cover number and present an FPT algorithm. Furthermore, we provide a linear-size problem kernel for the parameter feedback edge set number, which is also a parameter larger than the feedback vertex set number. We thereby extend a result from Betzler et al. [Bet+12], who showed an FPT algorithm for BDD parameterized by the feedback edge set number. Our problem kernel for the special case BDD relies on a rather large case distinction. Due to its size we prove the correctness of the problem kernel using a non-constructive argument. Interestingly the problem kernel for the other cases of Biclique-Free Vertex Deletion is relatively simple. We fill the "gap" between the fixed-parameter tractability for the parameter vertex cover number and the W[1]-hardness for the parameter feedback ver-

[2]We use the $\mathcal{O}^*$-notation to hide polynomial factors of the running time.

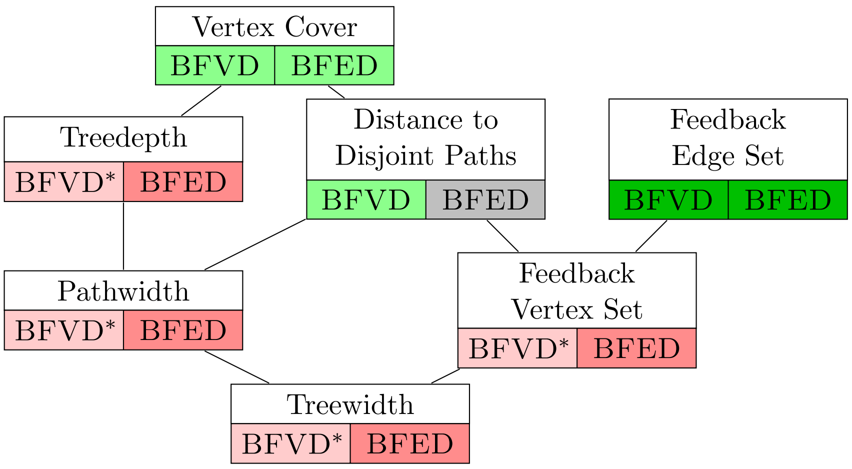


Figure 1.2: Overview of the parameterized complexity of Biclique-Free Vertex Deletion (BFVD) and Biclique-Free Edge Deletion (BFED). An edge between two parameters indicates that, based on well-known results, the lower parameter can be upper-bounded by a function of the upper parameter. The color indicates the complexity for the different parameters and the two variants. Green indicates fixed-parameter tractability with dark green indicating the existence of a linear-size problem kernel, red indicates W[1]-hardness with light red indicating that the W[1]-hardness is only known for $i = 1$, i.e. BDD, and grey indicates that the complexity remains open. The results marked with * are from Ganian, Klute, and Ordyniak [GKO21].

tex set number by presenting an FPT algorithm for the parameter distance to disjoint paths, which lies between the vertex cover number and the feedback vertex set number.

Regarding Biclique-Free Edge Deletion we prove W[1]-hardness with respect to the feedback vertex set number and the treedepth by using the same underlying idea that Ganian, Klute, and Ordyniak [GKO21] used for showing W[1]-hardness for BDD with these parameters. In contrast to Biclique-Free Vertex Deletion, the special case that one side of the biclique has size one of Biclique-Free Edge Deletion is solvable in polynomial time. Using this result we present a linear-size problem kernel with respect to the feedback edge set number. Furthermore, we present an FPT algorithm for the parameter vertex cover number using an integer linear programming formulation. See Figure 1.2 for an overview of the complexity of Biclique-Free Deletion parameterized by different structural graph parameters.

Structure of the work. In Chapter 2, we introduce the notation and concepts used in the remaining thesis, formally define both variants of Biclique-Free Deletion, and prove that finding bicliques is FPT when parameterized by treewidth. In Chapters 3 and 4, we study the parameterized complexity of Biclique-Free Vertex Deletion and Biclique-Free Edge Deletion, respectively, parameterized by different structural graph parameters. We conclude our work in Chapter 5.

Chapter 2

Preliminaries

We introduce the notation and concepts used in the following chapters in Section 2.1, and provide a formal definition of the problems of interest in Section 2.2. In Section 2.3 we present an algorithm parameterized by treewidth that finds bicliques in graphs. We will use it later as a tool in our more sophisticated algorithms.

2.1 Notation and Definitions

In this section, we provide basic notations and definitions that will be used later in this thesis. We include zero in the natural numbers, i.e. $\mathbb{N} := \{0, 1, 2, \dots\}$. We denote by $\mathbb{N}^+ := \mathbb{N} \setminus \{0\}$ all positive natural numbers. For a number $x \in \mathbb{N}^+$, let $[x] := \{1, 2, \dots, x\}$.

Graph Theory. We use the following well-known concepts and definitions from graph theory as introduced by e.g. Diestel [Die17]. Let $G = (V, E)$ denote an (undirected) graph, where V denotes the set of vertices and $E \subseteq \{\{v, w\} \mid v, w \in V,\, v \neq w\}$ denotes the set of edges. For a graph G, we denote by

$V(G)$	the *vertex set* of G;
$E(G)$	the *edge set* of G with $E(G) \subseteq \binom{V(G)}{2}$;
n_G	the number $\lvert V(G)\rvert$ of *vertices*;
m_G	the number $\lvert E(G)\rvert$ of *edges*;
$N_G(v)$	the (open) *neighborhood* of v, formally, $N_G(v) := \{u \in V \mid \{u, v\} \in E(G)\}$;
$\deg_G(v)$	the *degree* of v, formally, $\deg_G(v) := \lvert N_G(v)\rvert$;
$G[V']$	the *induced subgraph* of G on $V' \subseteq V$, formally, $G[V'] := \Big(V',\, E(G) \cap \binom{V'}{2}\Big)$;
$G - V'$	the graph obtained from G by deleting the vertices $V' \subseteq V(G)$, formally, $G - V' := G[V(G) \setminus V']$;

$G - E'$ the graph obtained from G by deleting the edges $E' \subseteq E(G)$, formally, $G - E' := (V(G), E(G) \setminus E')$.

We may omit the subscript G, if the graph G is clear from the context. For an edge $e = \{u, v\} \in E(G)$ the two vertices u and v are called *endpoints* of e. Two vertices $v, w \in V(G)$ in a graph G are *adjacent*, or *neighbors*, if $\{v, w\} \in E(G)$. An edge $e \in E(G)$ and a vertex v are *incident*, if v is an endpoint of e. A graph $G' = (V', E')$ is a *subgraph* of a graph $G = (V, E)$, written as $G' \subseteq G$, if $V' \subseteq V$ and $E' \subseteq E$. We also say that G *contains* G'. Note that a subgraph is not necessarily an induced subgraph. The *common neighborhood* of a vertex subset $V' \subseteq V$ in a graph $G = (V, E)$ is $\bigcap_{v \in V'} N_G(v)$.

A *path* is a graph $P = (V, E)$ with $n \geq 1$, the vertices $V = \{v_1, v_2, \dots, v_n\}$ and the edges $E = \{\{v_i, v_{i+1}\} \mid i \in [n-1]\}$. Sometimes we also denote a path only by its vertex sequence, e.g. $P = v_1 v_2 \dots v_n$, and omit listing its edges explicitly. The two vertices $v_1, v_n \in V$ are the *endpoints* of the path and the vertices $\{v_2, \dots, v_{n-1}\} \subseteq V$ are its *inner vertices*. The *length* of the path is $n - 1$, i.e. the number of edges. In a graph $G = (V, E)$ two vertices $v, w \in V$ are *connected*, if a path that contains both v and w is a subgraph of G. A graph $G = (V, E)$ is connected, if every pair of its vertices is connected. A *cycle* is a graph $C = (V, E)$ with $n \geq 3$, the vertices $V = \{v_1, v_2, \dots, v_n\}$ and the edges $E = \{\{v_i, v_{i+1}\} \mid i \in [n-1]\} \cup \{\{v_1, v_n\}\}$. Sometimes we also denote a cycle only by its vertex sequence, e.g. $C = v_1 v_2 \dots v_n v_1$, and omit listing its edges explicitly. A *forest* is a graph that does not contain any cycle. A *tree* is a connected graph that does not contain any cycle. A *biclique* is a graph $G = (L \uplus R, E)$ with $|L|, |R| \geq 1$ and $E = \{\{v, w\} \mid v \in L, w \in R\}$. The two sets L and R are the *sides* of the biclique G. We denote by $K_{i,j}$ with $i, j \in \mathbb{N}$ and $i \leq j$ the biclique with i vertices on one side and j vertices on the other side. A *connected component* of a graph $G = (V, E)$ is a connected maximal subgraph of G. A *rooted* forest contains one vertex fixed as the *root*. The *height* of a rooted forest is the length of a longest path between the root and some vertex plus one. A vertex x is an *ancestor* of another vertex y in a rooted forest F, if x is on the path between the root and y. The closure $\operatorname{clos}(F)$ of a rooted forest $F = (V, E)$ is the graph $(V, \{\{u, v\} \mid u \neq v,\ v \text{ is an ancestor of } u \text{ in } F\})$. For a graph H, we say a graph G is H-free, if G does not contain H.

Furthermore, we denote the following structural graph parameters for a graph $G = (V, E)$ by

$\mathrm{VCN}(G)$ the *vertex cover number* of G, that is, the size of a minimum subset $V' \subseteq V$ such that $G - V'$ is edge-free;

$\mathrm{DDP}(G)$ the *distance to disjoint paths* of G, that is, the size of a minimum subset $V' \subseteq V$ such that every connected component in $G - V'$ is a path;

$\mathrm{FVN}(G)$ the *feedback vertex set number* of G, that is, the size of a minimum subset $V' \subseteq V$ such that $G - V'$ is cycle-free;

$\mathrm{FEN}(G)$ the *feedback edge set number* of G, that is, the size of a minimum subset $E' \subseteq E$ such that $G - E'$ is cycle-free;

$\mathrm{td}(G)$ the *treedepth* of G, that is, the minimum height of a rooted forest F such that $G \subseteq \operatorname{clos}(F)$ [NM06].

It holds that $\mathrm{FVN}(G) \leq \mathrm{DDP}(G) \leq \mathrm{VCN}(G)$, because in any edge-free graph every connected component is a path of length zero and every graph where all connected components are paths is cycle-free. Furthermore, it holds that $\mathrm{FVN}(G) \leq \mathrm{FEN}(G)$, because if there exists a subset $E' \subseteq E(G)$ such that $G - E'$ is cycle-free, then one may obtain a subset $V' \subseteq V(G)$ with $|V'| \leq |E'|$ such that $G - V'$ is cycle-free by picking an arbitrary endpoint of each edge in E' into V'.

We define for a given graph $G = (V, E)$ the following vertex subsets: $V_{\geq 3}(G) := \{v \in V(G) \mid \deg(v) \geq 3\}$; $V_2(G) := \{v \in V(G) \mid \deg(v) = 2\}$; $V_1(G) := \{v \in V(G) \mid \deg(v) = 1\}$; $V_0(G) := \{v \in V(G) \mid \deg(v) = 0\}$.

Parameterized Complexity. We use the following well-known concepts and definitions from parameterized algorithmics and complexity theory as introduced by e.g. Cygan et al. [Cyg+15]. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed and finite alphabet, e.g. $\Sigma := \{0, 1\}$. An instance $(I, k) \in \Sigma^* \times \mathbb{N}$ consists of the *problem input* I and the *parameter* k. If $(I, k) \in L$, then (I, k) is a yes-instance, otherwise it is a no-instance.

A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is *fixed-parameter tractable* (FPT), if there exists an algorithm $\mathcal{A}$ that takes an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ as input and decides whether $(I, k) \in L$ in time bounded by $f(k) \cdot |(I, k)|^c$ for some computable function f and constant c. We say that $\mathcal{A}$ runs in *FPT time* and sometimes use the notation $\mathcal{O}^*(f(k))$ to simplify the running time statement by hiding the polynomial $|(I, k)|^c$ of the running time, e.g. $f(k) \cdot |(I, k)|^c \in \mathcal{O}^*(f(k))$.

A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ admits a *problem kernel*, if there exists an algorithm $\mathcal{B}$ that takes an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ as input and outputs an equivalent instance $(I', k') \in \Sigma^* \times \mathbb{N}$ with $|I'| + k' \leq g(k)$ in polynomial time for some computable function g. If g is polynomial, then L admits a *polynomial-size problem kernel*. If g is linear, then L admits a *linear-size problem kernel*. Algorithms that show the existence of a problem kernel sometimes consist of *data reduction rules*, which e.g. solve subproblems until the remaining instance is of bounded size.

Using *parameterized reductions* between different problems one can show that certain parameterized problems are at least as hard as other parameterized problems. Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *parameterized reduction* from A to B is an algorithm that takes an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ as input and outputs an equivalent instance $(I', k') \in \Sigma^* \times \mathbb{N}$ with $k' \leq g(k)$ in FPT time for some computable function g.

If there is a parameterized reduction from a parameterized problem A to another parameterized problem B that is FPT, then A is also FPT. It is conjectured and widely believed that all *W[1]-hard* parameterized problems are not FPT. For example Clique and Independent Set parameterized by the solution size are W[1]-hard. If there is a parameterized reduction from a parameterized problem A that is W[1]-hard to another parameterized problem B, then B is also W[1]-hard.

Integer Linear Programming. The following well-studied problem can be useful for showing fixed-parameter tractability using parameterized reductions.

INTEGER LINEAR PROGRAMMING FEASIBILITY (ILP FEASIBILITY)
Input: The number of variables $p \in \mathbb{N}$ and inequalities $m \in \mathbb{N}$, a matrix $A \in \mathbb{Z}^{m \times p}$ and a vector $b \in \mathbb{Z}^m$.
Question: Does there exist a vector $x \in \mathbb{Z}^p$ such that $Ax \leq b$?

It is known that ILP FEASIBILITY with $p \in \mathbb{N}$ variables can be solved in $\mathcal{O}^*(p^{\mathcal{O}(p)})$ time [FT87; Kan87; Len83] and is therefore FPT when parameterized by the number of variables.

Tree Decompositions. Following Cygan et al. [Cyg+15], a *tree decomposition* of a graph $G = (V, E)$ is a pair $(T = (I, F), \{X_t \subseteq V \mid t \in I\})$ consisting of a tree T and a set of *bags* containing vertices of G such that $\bigcup_{t \in I} X_t = V$, for each $\{v, w\} \in E$ there exists a bag $X_t \ni v, w$, and for each vertex $v \in V$ the subgraph $T[\{t \in I \mid v \in X_t\}]$ is still a tree. To better distinguish the vertices of G from the vertices of T we call the vertices of T *nodes*. The *width* of a tree decomposition is the size of a largest bag minus one. The *treewidth* of a graph G, denoted by $\text{tw}(G)$, is the minimum width of a tree decomposition of G. We can assume that any tree decomposition of width ω contains at most $\mathcal{O}(\omega \cdot n)$ nodes. Note that $\text{tw}(G) \leq \text{FVN}(G) + 1$ holds, because the treewidth of a forest is at most one and adding all feedback vertices increases the treewidth by at most $\text{FVN}(G)$, e.g. by adding all feedback vertices to every bag of the tree decomposition of the forest. There exist many different algorithms for computing a tree decomposition of a graph G either of minimum width or with linear approximations, e.g. an algorithm returning a tree decomposition of width $2\,\text{tw}(G) + 1$ in $2^{\mathcal{O}(\text{tw}(G))} \cdot n$ time by Korhonen [Kor21]. Note that this is an FPT algorithm when parameterized by treewidth.

2.2 Problem Definitions

In this section, we formally define the problems studied in this thesis.

(i, j)-BICLIQUE
Input: An undirected graph $G = (V, E)$ and the size of the biclique $i, j \in \mathbb{N}^+$ with $i \leq j$.
Question: Does G contain any $K_{i,j}$ as a (not necessarily induced) subgraph?

The following problems can be viewed as generalizations of the (i, j)-BICLIQUE problem. For an example instance of the two problems, see again Figure 1.1.

BICLIQUE-FREE VERTEX DELETION
Input: An undirected graph $G = (V, E)$, the size of the bicliques $i, j \in \mathbb{N}^+$ with $i \leq j$ and a budget $k \in \mathbb{N}$.
Question: Does there exist a subset $V' \subseteq V$ with $|V'| \leq k$ such that $G - V'$ does not contain any $K_{i,j}$ as a (not necessarily induced) subgraph?

BICLIQUE-FREE EDGE DELETION
Input: An undirected graph $G = (V, E)$, the size of the bicliques $i, j \in \mathbb{N}^+$ with $i \leq j$ and a budget $k \in \mathbb{N}$.
Question: Does there exist a subset $E' \subseteq E$ with $|E'| \leq k$ such that $G - E'$ does not contain any $K_{i,j}$ as a (not necessarily induced) subgraph?

The special case $i = 1$ of Biclique-Free Vertex Deletion is commonly known as Bounded-Degree Deletion or Bounded-Degree Vertex Deletion. We define it as follows.

Bounded-Degree Deletion (BDD)

Input:	An undirected graph $G = (V, E)$, a budget $k \in \mathbb{N}$ and a degree upper bound $d \in \mathbb{N}$.
Question:	Does there exist a subset $V' \subseteq V$ with $\lvert V' \rvert \leq k$ such that each vertex in $G - V'$ has degree at most d?

Note that an instance (G, k, d) of Bounded-Degree Deletion is a yes-instance, if and only if the instance $(G, 1, d + 1, k)$ of Biclique-Free Vertex Deletion is a yes-instance. Furthermore, note that the special case $d = 0$ of Bounded-Degree Deletion, that is, Biclique-Free Vertex Deletion with $i = j = 1$, is Vertex Cover.

2.3 Finding Bicliques

In this section, we show that (i, j)-Biclique is FPT when parameterized by treewidth. Furthermore, we will discuss how this algorithm can be used to detect all vertices and edges that are part of at least one biclique and turn it into a tool used later as a data reduction rule in more sophisticated algorithms.

We assume that a tree decomposition $\mathcal{T}$ of G is given alongside the problem input such that $\mathcal{T}$ is of width at most $\mathcal{O}(\mathrm{tw}(G)^c)$ and contains at most $\mathcal{O}(\mathrm{tw}(G)^c \cdot n)$ nodes for some constant $c \in \mathbb{N}$. Such a tree decomposition can be computed using one of many available algorithms, e.g. the algorithm from Korhonen [Kor21], in FPT time with respect to the treewidth of the input graph. Our algorithm solving (i, j)-Biclique in FPT time parameterized by treewidth is based on the following lemma.

Lemma 2.1. *Let $G = (V, E)$ be a graph and let $(T = (I, F), \{X_t \mid t \in I\})$ be a tree decomposition of G. If G contains any biclique $(L \uplus R, E' = \{\{u, v\} \mid u \in L, v \in R\})$ as a subgraph, then there is at least one bag X_t with $L \subseteq X_t$ or $R \subseteq X_t$.*

Proof. Assume towards contradiction that there is no bag X_t such that $L \subseteq X_t$ or $R \subseteq X_t$. Then there exist two vertices $r_1, r_2 \in R$ such that no bag contains both r_1 and r_2. Note that otherwise there must exist a vertex $r_3 \in R$ and a bag $X_{t'}$ with $r_1, r_2 \in X_{t'}$ and $r_3 \notin X_{t'}$, but there must exist at least one bag containing both r_1 and r_3, and at least one bag containing both r_2 and r_3, which cannot occur in a correct tree decomposition. So let $T_{r_1} := T[\{i \in I \mid r_1 \in X_i\}]$ and $T_{r_2} := T[\{i \in I \mid r_2 \in X_i\}]$ be the nonempty subtree with all nodes where the corresponding bags contain r_1 (but not r_2) and r_2 (but not r_1), respectively. Analogously, let $\ell_1, \ell_2 \in L$ be two vertices such that no bag contains both ℓ_1 and ℓ_2, and let T_{ℓ_1} and T_{ℓ_2} be the corresponding nonempty subtrees.

For all $i, j \in \{1, 2\}$, there exists a bag $X_{\ell_i, r_j} \ni \ell_i, r_j$, because $\{\ell_i, r_j\} \in E'$. There must be a path in T_{ℓ_1} between the nodes corresponding to X_{ℓ_1, r_1} and X_{ℓ_1, r_2}, a path in T_{ℓ_2} between the nodes corresponding to X_{ℓ_2, r_1} and X_{ℓ_2, r_2}, and a path in T_{r_1} between the nodes corresponding to X_{ℓ_1, r_1} and X_{ℓ_2, r_1}. Finally, there must be a path in T_{r_2} between the nodes corresponding to X_{ℓ_1, r_2} and X_{ℓ_2, r_2}, but such a path cannot exist, because it would destroy the tree decomposition by closing a cycle in T. □

Using this lemma we can now show that (i, j)-Biclique is FPT when parameterized by treewidth by providing an algorithm.

Theorem 2.2. *Given a tree decomposition of width at most ω of G with at most $\mathcal{O}(\omega \cdot n)$ nodes, (i, j)-Biclique can be solved in $\mathcal{O}(2^\omega \cdot \omega^2 \cdot n^2)$ time.*

Proof. Let $(T = (I, F), \{X_t \mid t \in I\})$ be a tree decomposition of width ω of G with $\mathcal{O}(\omega \cdot n)$ nodes. For each bag X_t and for each subset $X'_t \subseteq X_t$ with $|X'_t| = i$ or $|X'_t| = j$ check the size of its common neighborhood. If $|X'_t| = i$ and the size of its common neighborhood is at least j, or if $|X'_t| = j$ and the size of its common neighborhood is at least i, then output *yes*, because X'_t and the vertices in its common neighborhood form a biclique. If each bag X_t was processed without finding any biclique, then output *no*.

This algorithm is correct, as it will find all bicliques contained in G as a subgraph, because for each biclique there is at least one bag X_t such that a subset $X'_t \subseteq X_t$ will be exactly one side of the biclique, as shown in Lemma 2.1. Checking the common neighborhood for $\mathcal{O}(2^\omega)$ many subsets of size at most ω requires $\mathcal{O}(2^\omega \cdot \omega \cdot n)$ time. This step is repeated for each of the $\mathcal{O}(\omega \cdot n)$ nodes. Hence the total running time of the algorithm is $\mathcal{O}(2^\omega \cdot \omega^2 \cdot n^2)$. ☐

It is also possible to use this algorithm to detect which vertices and edges from the input graph are part of at least one biclique. If the algorithm finds a subset $X'_t \subseteq X_t$ of size i, respectively j, with a common neighborhood of size at least j, respectively i, then the vertices from X'_t and the common neighborhood, and the edges between these vertices are part of at least one biclique. Moreover, it is possible to store such X'_t and its common neighborhood as a pair to build a list of biclique parts that can later be used to obtain a specific biclique: For a set X'_t of size i, respectively j, pick any subset of its common neighborhood of size j, respectively i, to obtain a $K_{i,j}$.

We can use this extension of the algorithm from Theorem 2.2 to obtain the following simple, general data reduction rule applicable in FPT time with respect to treewidth.

Reduction Rule 2.3.1. *Let (G, i, j, k) be an instance of* Biclique-Free Vertex Deletion *or* Biclique-Free Edge Deletion. *Delete all vertices and edges from G that are not part of at least one $K_{i,j}$ contained in G as a subgraph.*

Lemma 2.3. *Reduction Rule 2.3.1 is correct. Given a tree decomposition of width at most ω of G and with at most $\mathcal{O}(\omega \cdot n)$ nodes, it can be applied exhaustively in $\mathcal{O}(2^\omega \cdot \omega^2 \cdot n^2)$ time.*

Proof. The reduction rule is correct, as picking vertices or edges that are not part of any biclique into the solution set does not resolve any existing bicliques and on the other hand not picking them into the solution set does not lead to any bicliques that need to be resolved.

Solve (i, j)-Biclique on G in $\mathcal{O}(2^\omega \cdot \omega^2 \cdot n^2)$ time (see Theorem 2.2) to detect all vertices and edges that are not part of at least one $K_{i,j}$ contained in G as a subgraph and then delete them. Deleting the found vertices and edges requires $\mathcal{O}(n+m) \subseteq \mathcal{O}(n^2)$ time, as the deletion of all vertices and edges while maintaining a correct graph representation is possible in this time. Hence the reduction rule can be applied in $\mathcal{O}(2^\omega \cdot \omega^2 \cdot n^2)$ time. ☐

Chapter 3

Biclique-Free Vertex Deletion

In this chapter, we analyze the parameterized complexity of the Biclique-Free Vertex Deletion problem. It is known to be NP-hard [LY80] in general, and some special cases are W[1]-hard for the natural parameters j and k [Fel+11; Lin18], and for the structural graph parameters feedback vertex set number and treedepth [GKO21]. See Chapter 1 for detailed information on known results.

With little hope for FPT algorithms with respect to natural parameters or structural graph parameters that are at most the feedback vertex set number or the treedepth, we try to find FPT algorithms for structural graph parameters that are larger than the feedback vertex set number or the treedepth. We show in Section 3.1 that Biclique-Free Vertex Deletion is FPT when parameterized by the vertex cover number. In Section 3.2 we show that it admits a linear-size problem kernel for the parameter feedback edge set number, which is also larger than the feedback vertex set number. To the best of our knowledge, it was not known before if BDD admits a linear-size problem kernel when parameterized by the feedback edge set number. We answer this question positively. In Section 3.3 we show that Biclique-Free Vertex Deletion is also FPT when parameterized by a parameter lying directly in between the vertex cover number and the feedback vertex set number, namely, the distance to disjoint paths.

3.1 Parameterization by Vertex Cover Number

In this section, we present an algorithm running in FPT time for solving Biclique-Free Vertex Deletion parameterized by the size of the minimum vertex cover. We assume that a minimum vertex cover S is given along with the problem input. Note that one can compute a minimum vertex cover for any graph in FPT time with respect to the size of a minimum vertex cover.

Theorem 3.1. *Let S be a vertex cover of G. Then* Biclique-Free Vertex Deletion *can be solved in $\mathcal{O}(2^{2|S|+|S|^2} \cdot |S| \cdot n)$ time.*

Proof. If the budget $k \geq |S|$, then we can output *yes* and are done, because deleting all vertices in S would be covered by the budget and the remaining independent set would be biclique-free. Therefore we can assume $k < |S|$ for the rest of the algorithm.

We first guess which vertices in the vertex cover should be deleted from the input graph G. Let $S^* := S \backslash S'$ be the remaining vertex cover after we deleted the subset $S' \subseteq S$ from G and let $I := V \setminus S$ be the independent set corresponding to the vertex cover. Two vertices in I are of the same *type*, if they have the same neighborhood. Because all neighbors of the vertices in I are by definition part of S^*, there are at most $2^{|S^*|}$ different types of vertices in I. As a second step, we guess for each type how many vertices of this type we delete from G. We do not need to distinguish between vertices of the same type, because if an instance of BICLIQUE-FREE VERTEX DELETION has a solution containing a specific subset of vertices all of the same type, then it also has a solution that instead contains any other equally sized subset of vertices all of this type. If the remaining graph does not contain the $K_{i,j}$ as a subgraph and we deleted at most k vertices, then output *yes*. Otherwise output *no*.

The algorithm is correct, because we guess which vertices from both S and I should be deleted and only output *yes*, if the remaining graph is indeed $K_{i,j}$-free and the budget is sufficient.

There are $2^{|S|}$ possible subsets S' and $\mathcal{O}(2^{|S| \cdot k}) \subseteq \mathcal{O}(2^{|S|^2})$ possible combinations of picking the number of vertices to delete for each of the $2^{|S^*|}$ types, which is in total at most k. Checking if the graph contains the $K_{i,j}$ as a subgraph can be done in $\mathcal{O}(2^{|S|} \cdot |S| \cdot n)$ time by checking the size of the common neighborhood in $\mathcal{O}(|S| \cdot n)$ time for all subsets of S. This sums up in total to $\mathcal{O}(2^{2|S|+|S|^2} \cdot |S| \cdot n)$, which is FPT time. Reading the input and checking which types exist is also possible in this time. □

3.2 Parameterization by Feedback Edge Set Number

After we settled that BICLIQUE-FREE VERTEX DELETION is FPT when parameterized by the vertex cover number in Section 3.1, we now show that BICLIQUE-FREE VERTEX DELETION admits a linear-size problem kernel computable in $\mathcal{O}(n + m)$ time when parameterized by feedback edge set number. We base our problem kernel on the observation that the number of vertices with degree at least three in a graph with minimum degree two can be bounded linearly by the feedback edge set number. We prove this and provide the linear-size problem kernel for the case $i \geq 2$ in Section 3.2.1 and then cover the seemingly more complicated case of $i = 1$ commonly known as BOUNDED-DEGREE DELETION in Section 3.2.2. It was already known before that BDD is FPT when parameterized by the feedback edge set number [Bet+12]. We extend this result by providing a linear-size problem kernel.

3.2.1 The Case of $i \geq 2$

In this subsection, we show that BICLIQUE-FREE VERTEX DELETION with $i \geq 2$ admits a linear-size problem kernel computable in $\mathcal{O}(n + m)$ time when parameterized by the feedback edge set number using the following data reduction rules and lemmata.

Reduction Rule 3.2.1. *Let (G, i, j, k) be an instance of* BICLIQUE-FREE VERTEX DELETION *or* BICLIQUE-FREE EDGE DELETION. *Delete all vertices with degree less than i from G.*

Lemma 3.2. *Reduction Rule 3.2.1 is correct and can be applied exhaustively in $\mathcal{O}(n+m)$ time.*

Proof. The reduction rule is correct, as vertices with degree less than i cannot be part of any subgraph of G that is the $K_{i,j}$, because the minimum degree in the $K_{i,j}$ is i.

The exhaustive application of the rule may delete all vertices of G, which takes $\mathcal{O}(n+m)$ time. Note that if we deleted a vertex, then the rule might become applicable for each of its neighbors. Therefore we have to check the degree for all vertices once and repeat the check for all neighbors of deleted vertices. This can be done in $\mathcal{O}(n+m)$ time by maintaining a list of vertices that still need to be tested. Hence the rule can be applied in $\mathcal{O}(n+m)$ time in total. □

The following two lemmata can be considered well-known and appear in a similar manner for example in Epstein, Levin, and Woeginger [ELW15] and Kellerhals and Koana [KK20]. For completeness we still provide a full proof.

Lemma 3.3. *Let $G = (V, E)$ be a graph with feedback edge set $F \subseteq E$. Then it holds that $|V_{\geq 3}(G)| - |V_1(G)| \leq 2\,|F|$.*

Proof. Removing all feedback edges from G would result by definition in a forest with c connected components and $n-c$ edges. Therefore $m = n+|F|-c$. Inserting $3\,|V_{\geq 3}(G)|+2\,|V_2(G)|+|V_1(G)| \leq \sum_{v\in V} \deg(v) = 2m$ yields $|V_{\geq 3}(G)|-|V_1(G)| \leq 2\,|F|-2c+2\,|V_0(G)|$. Since every degree-zero vertex is a connected component, it holds that $|V_0(G)| \leq c$ and we get $|V_{\geq 3}(G)| - |V_1(G)| \leq 2\,|F|$. □

Lemma 3.4. *Let $G = (V, E)$ be a graph with feedback edge set $F \subseteq E$. Let $c \in \mathbb{N}$ be the number of connected components in $G[V_2(G)]$. If every vertex in G has degree at least two, then $c \leq 3\,|F| - 1$.*

Proof. Since every vertex in G has degree at least two, every degree-two vertex in G is part of exactly one of the following three types of subgraphs of G: a cycle consisting only of degree-two vertices, a path consisting only of degree-two vertices whose two endpoints are adjacent to either the same vertex of higher degree or two distinct vertices of higher degree. For each of those three types it holds, that the degree-two vertices in such a subgraph form one connected component in $G[V_2(G)]$. Note that subgraphs of one of the first two types contain at least one edge of F. For the last type note that adding a path between two vertices $u, v \in V$ increases the feedback edge set number by one, if u and v are already connected beforehand.

We know from Lemma 3.3 that $|V_{\geq 3}(G)| \leq 2\,|F|$ holds, as $|V_1(G)| = 0$. Therefore there can be at most $2\,|F|-1$ subgraphs of the last type without increasing the feedback edge set number of G, because a forest with $2\,|F|$ vertices contains at most $2\,|F| - 1$ edges. Every additional connected component in $G[V_2(G)]$ increases the feedback edge set number of G by exactly one. This yields $c \leq |F| + 2\,|F| - 1 = 3\,|F| - 1$. □

We now use the data reduction rules and lemmata to prove the existence of a linear-size problem kernel computable in $\mathcal{O}(n+m)$ time for Biclique-Free Vertex Deletion with $i \geq 2$ parameterized by the feedback edge set number. It is based on the observation that large subgraphs containing only degree-two vertices cannot contain any biclique with $i \geq 2$ and therefore can be deleted.

Theorem 3.5. Biclique-Free Vertex Deletion *with $i \geq 2$ admits a linear-size problem kernel computable in $\mathcal{O}(n+m)$ time when parameterized by the feedback edge set number.*

Proof. Let $F \subseteq E$ be a minimum feedback edge set for the input graph G computable in $\mathcal{O}(n+m)$ time using depth-first search. If the budget $k \geq |F|$, then we can output a trivial yes-instance and are done. This is correct, because deleting an arbitrary endpoint for each feedback edge is covered by the budget and results in a forest. Forests do not contain any $K_{i,j}$ with $i, j \geq 2$ as a subgraph. We can therefore assume that $k < |F|$.

Apply Reduction Rule 3.2.1 exhaustively in $\mathcal{O}(n+m)$ time (see Lemma 3.2) to remove all vertices with degree at most one. By Lemma 3.3 the remaining graph G now contains at most $2|F|$ vertices of degree at least three. Also delete all vertices from G that are contained in a connected component of $G[V_2(G)]$ with more than four vertices, because they cannot be contained in any biclique of sufficient size. This is also possible in $\mathcal{O}(n+m)$ time using depth-first search. Then every connected component in $G[V_2(G)]$ contains at most four vertices. Using Lemma 3.4 we can conclude that there exist at most $12|F|-4$ vertices of degree two in G. By combining both upper bounds we achieve $n \leq 14|F|-4$ in total. Using $m \leq n+|F|-1$ yields $m \leq 15|F|-5$. If $j \geq n$ holds, then output a trivial yes-instance, as the graph cannot contain the $K_{i,j}$, so assume $i \leq j < n$. Now it holds that $k < |F|$, $i \leq j < n \leq 14|F|-4$, and $m \leq 15|F|-5$. Hence the size of the remaining instance is bounded linearly in $|F|$. □

3.2.2 The Case of Bounded-Degree Deletion

In this subsection, we show the existence of a linear-size problem kernel computable in $\mathcal{O}(n+m)$ time for the case $i = 1$ of Biclique-Free Vertex Deletion parameterized by the feedback edge set number. See Section 2.2 for a formal problem definition of this special case of Biclique-Free Vertex Deletion known as Bounded-Degree Deletion.

It was already known that Bounded-Degree Deletion is FPT when parameterized by the feedback edge set number [Bet+12], but to the best of our knowledge no linear or polynomial-size problem kernel was known until now.

Recall that we denote by $G[V_2(G)]$ the subgraph induced by all degree-two vertices, i.e. $V_2(G) := \{v \in V(G) \mid \deg(v) = 2\}$. We follow the same strategy we used in Section 3.2.1 to show the existence of a problem kernel for the case $i \geq 2$: We delete vertices of degree at most one and upper-bound the number of degree-two vertices in each connected component in $G[V_2(G)]$ by applying multiple data reduction rules. Then the linear-size problem kernel follows from Lemmas 3.3 and 3.4. Since $i = 1$, in contrast to Section 3.2.1 where $i \geq 2$ was given, we cannot delete vertices with degree at most one by using the relatively straightforward Reduction Rule 3.2.1 again. Instead we introduce vertex weights representing additional degree-one neighbors that cannot be picked into the solution set. We arrive at the following generalization of Bounded-Degree Deletion.

Weighted Bounded-Degree Deletion (WBDD)

Input:	An undirected graph $G = (V, E)$, a budget $k \in \mathbb{N}$, a degree upper bound $d \in \mathbb{N}$ and a weight $w_v \in \mathbb{N}$ for each $v \in V$.
Question:	Does there exist a subset $V' \subseteq V$ with $|V'| \leq k$ such that each vertex $v \in V \setminus V'$ has degree at most $d - w_v$ in $G - V'$?

There is a straightforward parameterized reduction from Bounded-Degree Deletion to WBDD, where we set the weight of every vertex to zero.

Observation 3.6. *There is a parameterized reduction from* Bounded-Degree Deletion *parameterized by the feedback edge set number to* WBDD *parameterized by the same parameter.*

Proof. Let $\mathcal{I} = (G = (V, E), k, d)$ be an instance of Bounded-Degree Deletion. Then construct an equivalent instance $\mathcal{I}' = (G', k', d', w)$ of WBDD as follows. Set $G' := G$, $k' := k$ and $d' := d$. For each vertex $v \in V$ set $w_v := 0$. This reduction can be computed in $\mathcal{O}(|\mathcal{I}|)$ time, the instances are obviously equivalent, and $\text{FEN}(G') \leq \text{FEN}(G)$ holds, as $G' = G$. Hence this is a parameterized reduction. □

We will now present the first data reduction rules for WBDD that lead towards our linear-size problem kernel. Together they delete all vertices of degree at most one and even some vertices of higher degree. The data reduction rules are based on the observation that all vertices exceeding the degree bound even if all their neighbors are deleted must be in every solution set, and that picking the neighbor of a degree-one vertex in the solution set is at least as good as picking the degree-one vertex itself, if it exceeds the degree bound by at most one.

Reduction Rule 3.2.2. *Let $\mathcal{I} = (G = (V, E), k, d, w)$ be an instance of* WBDD *and let $v \in V$ be some vertex of G. If $w_v > d$, then delete v and decrease k by one.*

Observation 3.7. *Reduction Rule 3.2.2 is correct.*

Proof. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq d - w_u$ holds for each vertex $u \in V \setminus S$. If the vertex $v \notin S$, then $\deg_{G-S}(v) \leq d - w_v < 0$ must hold, as $w_v > d$, but this is a contradiction. Hence $v \in S$. Then $S' = S \setminus \{v\}$ is a solution for the modified instance $\mathcal{I}' = (G - \{v\}, k - 1, d, w)$ of size $|S'| = |S| - 1 \leq k - 1$. Therefore $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V \setminus \{v\}$ of size $|S'| \leq k - 1$ such that $\deg_{G-\{v\}-S'}(u) \leq d - w_u$ holds for each vertex $u \in (V \setminus \{v\}) \setminus S'$. Then $S' \cup \{v\}$ is a solution set for $\mathcal{I}$ of size $|S'| + 1 \leq k$. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent. □

Reduction Rule 3.2.3. *Let $\mathcal{I} = (G = (V, E), k, d, w)$ be an instance of* WBDD *and let $v \in V$ be some vertex of G with $\deg(v) = 0$. If $w_v \leq d$, then delete v.*

Observation 3.8. *Reduction Rule 3.2.3 is correct.*

Proof. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq d - w_u$ holds for each vertex $u \in V \setminus S$. Then the set $S' = S \setminus \{v\}$

is a solution for the modified instance $\mathcal{I}' = (G - \{v\}, k, d, w)$ of size at most $|S| \leq k$. Therefore $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V \setminus \{v\}$ of size $|S'| \leq k$ such that $\deg_{G-\{v\}-S'}(u) \leq d - w_u$ holds for each vertex $u \in (V \setminus \{v\}) \setminus S'$. Then $S = S'$ is a solution set for $\mathcal{I}$ of size $|S| = |S'| \leq k$, as $\deg_{G-S}(v) = 0 \leq d - w_v$. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent. □

Reduction Rule 3.2.4. *Let $\mathcal{I} = (G = (V, E), k, d, w)$ be an instance of* WBDD *and let $v \in V$ be some vertex of G with $N(v) = \{u\}$. If $w_v = d$, then delete both v and u, and decrease k by one.*

Observation 3.9. *Reduction Rule 3.2.4 is correct.*

Proof. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(x) \leq d - w_x$ holds for each vertex $x \in V \setminus S$. As $\deg_G(v) = 1 > d - w_v = 0$ holds, v or u must be in S. Then $S' = S \setminus \{u, v\}$ is a solution for the modified instance $\mathcal{I}' = (G - \{u, v\}, k - 1, d, w)$ of size $|S'| \leq |S| - 1 \leq k - 1$. Therefore $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V \setminus \{u, v\}$ of size $|S'| \leq k - 1$ such that $\deg_{G-\{u,v\}-S'}(x) \leq d - w_x$ holds for each vertex $x \in (V \setminus \{u, v\}) \setminus S'$. Then $S = S' \cup \{u\}$ is a solution set for $\mathcal{I}$ of size $|S| = |S'| + 1 \leq k$, as $\deg_{G-S}(v) = 0 \leq d - w_v = 0$. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent. □

Reduction Rule 3.2.5. *Let $\mathcal{I} = (G = (V, E), k, d, w)$ be an instance of* WBDD *and let $v \in V$ be some vertex of G with $N(v) = \{u\}$. If $w_v < d$, then delete v and increase w_u by one.*

Observation 3.10. *Reduction Rule 3.2.5 is correct.*

Proof. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(x) \leq d - w_x$ holds for each vertex $x \in V \setminus S$. If $v \in S$, then the set $S' = (S \cup \{u\}) \setminus \{v\}$ is a solution for the modified instance $\mathcal{I}' = (G - \{v\}, k, d, w')$ with $w'_x = w_x$ for each vertex $x \in V \setminus \{u, v\}$ and $w'_u = w_u + 1$ of size $|S'| \leq |S| \leq k$. If $v \notin S$, then the set $S' = S$ is a solution for $\mathcal{I}'$ of size $|S'| = |S| \leq k$, as either $u \in S'$ or $\deg_{G-\{v\}-S'}(u) + w'_u = \deg_{G-S}(u) + w_u \leq d$. Therefore $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V \setminus \{v\}$ of size $|S'| \leq k$ such that $\deg_{G-\{v\}-S'}(x) \leq d - w'_x$ holds for each vertex $x \in (V \setminus \{v\}) \setminus S'$. Then $S = S'$ is a solution set for $\mathcal{I}$ of size $|S| = |S'| \leq k$, as $\deg_G(v) = 1 \leq d - w_v$ and either $u \in S$ or $\deg_{G-S}(u) + w_u = \deg_{G-\{v\}-S'}(u) + w'_u \leq d$. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent. □

We will now prove that we can apply Reduction Rules 3.2.2 to 3.2.5 exhaustively in $\mathcal{O}(n + m)$ time to remove all vertices with degree at most one or weight greater than d.

Lemma 3.11. *Reduction Rules 3.2.2 to 3.2.5 can all be applied exhaustively in $\mathcal{O}(n+m)$ time.*

Proof. To apply the reduction rules exhaustively, we have to test for each vertex if one of the reduction rules can be applied and repeat this test for all vertices where at least one neighbor got deleted by applying one of the rules. This can be done in $\mathcal{O}(n+m)$ time by checking the weight and the degree of each vertex on a list of vertices that still need to be tested. If a vertex gets deleted, then all neighbors that are not on the list already need to be added again. Actually applying the rules wherever possible requires $\mathcal{O}(n+m)$ time, since deleting all vertices while still maintaining a correct graph representation is possible in this time. Combining the steps leads to $\mathcal{O}(n+m)$ time in total. □

Observation 3.12. *After applying Reduction Rules 3.2.2 to 3.2.5 exhaustively the graph does not contain any vertices with degree at most one and all vertices have weight at most d.*

Proof. Let $v \in V$ be a vertex of G. If $w_v > d$, then Reduction Rule 3.2.2 can be applied for v resulting in its deletion. Therefore all vertices remaining in G after applying the reduction rule have weight at most d. Now consider the case $\deg(v) = 1$. If $w_v = d$, then v can be deleted using Reduction Rule 3.2.4. On the other hand, if $w_v < d$, then applying Reduction Rule 3.2.5 on v results in its deletion. Hence all degree-one vertices are deleted after applying the reduction rules exhaustively. Finally, suppose $\deg(v) = 0$. If $w_v \leq d$, then v can be deleted using Reduction Rule 3.2.3. Therefore also all degree-zero vertices are deleted after applying the reduction rules exhaustively. □

While the presented data reduction rules bring us closer towards our goal by successfully removing vertices of degree at most one, it is still necessary for our approach that we upper-bound the number of degree-two vertices linearly in $|F|$. So we will now continue by showing how to obtain an equivalent instance where every connected component in $G[V_2(G)]$ contains at most six degree-two vertices. By Lemma 3.4 the graph then contains at most $18\,|F| - 6$ degree-two vertices and we obtain our linear-size problem kernel.

We will start by resolving subgraphs that are cycles consisting only of degree-two vertices. They are not adjacent to any vertices outside the cycle and therefore can be solved independently from the rest of the input graph. We can choose some vertex of the cycle, guess if it is in the solution set and alter the instance accordingly. Then the remaining vertices form a path, which we can resolve rather quickly by applying the already presented Reduction Rules 3.2.2 to 3.2.5 repeatedly until the whole cycle is resolved. Either there is a minimum solution set for resolving the cycle that contains the chosen vertex or there is one not containing the vertex. Hence rather than guessing we can try both options, pick the smaller solution set and thereby obtain a minimum solution set for resolving the cycle.

Reduction Rule 3.2.6. *Let $\mathcal{I} = (G = (V, E), k, d, w)$ be an instance of* WBDD. *Let $C = v_1 v_2 \dots v_c v_1$ be a cycle of degree-two vertices in G. Then remove $V(C)$ from G and decrease k by the size of any minimum solution set for resolving C.*

Lemma 3.13. *Reduction Rule 3.2.6 is correct.*

Proof. Let s be the size of any minimum solution set for resolving C. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq$

$d - w_u$ holds for each vertex $u \in V \setminus S$. Then $|S \cap V(C)| \geq s$ holds, as s is the size of any minimum solution set for resolving C. Then $S' = S \setminus V(C)$ is a solution for the modified instance $\mathcal{I}' = (G - V(C), k - s, d, w)$ of size $|S'| \leq k - |S \cap V(C)| \leq k - s$. Hence $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V \setminus V(C)$ of size $|S'| \leq k - s$ such that $\deg_{G-V(C)-S'}(u) \leq d - w_u$ holds for each vertex $u \in (V \setminus V(C)) \setminus S'$. Then there exists a subset $C' \subseteq V(C)$ of size $|C'| = s$ such that $S' \cup C'$ is a solution set for $\mathcal{I}$ of size $|S' \cup C'| \leq k - s + s = k$, because any minimum solution set for resolving C is of size s and the vertices in $V(C)$ are not adjacent to any vertices outside of C in G. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent. □

Not all degree-two vertices are part of a cycle that consists only of other degree-two vertices. In fact all subgraphs consisting only of degree-two vertices now remaining are paths. The paths themselves could be resolved quickly using the same approach with the Reduction Rules 3.2.2 to 3.2.5 again, but this is not immediately possible, because the paths are adjacent to vertices from the rest of the graph.

Every path consisting only of degree-two vertices is adjacent to either one vertex, in fact forming a cycle in G, or two distinct vertices. We will now present a data reduction rule for the first case and afterwards rules for the latter. The general idea of this data reduction rule is that we want to use as few vertex deletions as possible to resolve the path, but use this budget as good as possible to shrink down the remaining graph, i.e. the vertex adjacent to the degree-two vertices of the path. This essentially means that we will always pick a minimum solution set for resolving the path that contains the adjacent vertex or, if there is no such minimum solution set, contains as many neighbors of the adjacent vertex as possible. Note that the data reduction rule works regardless of the degree of the adjacent vertex, but if it has degree two, then Reduction Rule 3.2.6 could be used instead.

Reduction Rule 3.2.7. *Let $\mathcal{I} = (G = (V, E), k, d, w)$ be an instance of* WBDD. *Let $C = v_1 v_2 \ldots v_c v_1$ be a cycle in G with $\deg(v) = 2$ for all $v \in V(C) \setminus \{v_1\}$. Now distinguish the following four cases:*

1. *If there exists a minimum solution set for resolving C with weight w_{v_1} set to zero that contains v_1, then remove $V(C)$ from G and decrease k by the size of the solution set.*

2. *Otherwise, if there exists a minimum solution set for resolving C with weight w_{v_1} set to zero that contains both v_2 and v_c, then remove $V(C) \setminus \{v_1\}$ from G and decrease k by the size of the solution set.*

3. *Otherwise, if there exists a minimum solution set for resolving C with weight w_{v_1} set to zero that contains either v_2 or v_c, then remove $V(C) \setminus \{v_1\}$ from G, increase the weight w_{v_1} by one and decrease k by the size of the solution set.*

4. *Otherwise, remove $V(C) \setminus \{v_1\}$ from G, increase the weight w_{v_1} by two and decrease k by the size of the solution set.*

Lemma 3.14. *Reduction Rule 3.2.7 is correct.*

Proof. Let s be the size of a minimum solution set for resolving C with weight w_{v_1} set to zero. We will now prove the correctness in four cases.

We first cover Case 1. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq d - w_u$ holds for each vertex $u \in V \setminus S$. Then the set $S \setminus V(C)$ is a solution for the modified instance $\mathcal{I}' = (G - V(C), k - s, d, w)$ of size $|S \setminus V(C)| \leq k - |S \cap V(C)| \leq k - s$, as $|S \cap V(C)| \geq s$ holds. Hence $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V \setminus V(C)$ of size $|S'| \leq k - s$ such that $\deg_{G-V(C)-S'}(u) \leq d - w_u$ holds for each vertex $u \in (V \setminus V(C)) \setminus S'$. Then there exists a subset $C' \subseteq V(C)$ of size $|C'| = s$ containing v_1 such that $S' \cup C'$ is a solution set for $\mathcal{I}$ of size $|S' \cup C'| \leq k - s + s = k$, because the minimum solution set for resolving C containing v_1 is of size s and v_1 separates $V \setminus V(C)$ from $V(C) \setminus \{v_1\}$ in G. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent.

Now consider Case 2. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq d - w_u$ holds for each vertex $u \in V \setminus S$. Then the set $S \setminus (V(C) \setminus \{v_1\})$ is a solution for the modified instance $\mathcal{I}' = (G - (V(C) \setminus \{v_1\}), k - s, d, w)$ of size $|S \setminus (V(C) \setminus \{v_1\})| \leq k - |S \cap (V(C) \setminus \{v_1\})| \leq k - s$, as $|S \cap (V(C) \setminus \{v_1\})| \geq s$ holds. Hence $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V' := V \setminus (V(C) \setminus \{v_1\})$ of size $|S'| \leq k - s$ such that $\deg_{G[V']-S'}(u) \leq d - w_u$ holds for each vertex $u \in V' \setminus S'$. Then there exists a subset $C' \subseteq (V(C) \setminus \{v_1\})$ of size $|C'| = s$ containing both v_2 and v_c such that $S' \cup C'$ is a solution set for $\mathcal{I}$ of size $|S' \cup C'| \leq k - s + s = k$, because the minimum solution set for resolving C containing both v_2 and v_c is of size s, and $\{v_2, v_c\}$ separates V' from $V(C) \setminus \{v_1, v_2, v_c\}$ in G. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent.

Now consider Case 3. Let $\mathcal{I}' = (G' = G - (V(C) \setminus \{v_1\}), k - s, d, w')$ with $w'_{v_1} = w_{v_1} + 1$ and $w'_v = w_v$ for each $v \in V \setminus V(C)$ be the modified instance. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq d - w_u$ holds for each vertex $u \in V \setminus S$. If $v_1 \in S$ or $\{v_2, v_c\} \subseteq S$, then $|S \cap V(C)| \geq s + 1$ holds, because there does not exist a solution set for resolving C of size at most s that contains v_1 or both v_2 and v_c. Therefore the set $(S \setminus V(C)) \cup \{v_1\}$ is a solution for $\mathcal{I}'$ of size $|(S \setminus V(C)) \cup \{v_1\}| \leq k - |S \cap V(C)| + 1 \leq k - s$. Otherwise, that is, $v_2 \notin S$ or $v_c \notin S$, and $v_1 \notin S$, the set $S' = S \setminus V(C)$ is a solution for $\mathcal{I}'$ of size $|S \setminus V(C)| \leq k - |S \cap V(C)| \leq k - s$, because $|S \cap V(C)| \geq s$ and $\deg_{G'-S'}(v_1) + w'_{v_1} \leq \deg_{G-S}(v_1) + w_{v_1} \leq d$ holds, as $|N_{G-S}(v_1) \cap V(C)| \geq 1$, but $|N_{G'-S'}(v_1) \cap V(C)| = 0$. Hence $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V' := V \setminus (V(C) \setminus \{v_1\})$ of size $|S'| \leq k - s$ such that $\deg_{G'-S'}(u) \leq d - w'_u$ holds for each vertex $u \in V' \setminus S'$. Then there exists a subset $C' \subseteq (V(C) \setminus \{v_1\})$ of size $|C'| = s$ containing v_2 or v_c such that $S = S' \cup C'$ is a solution set for $\mathcal{I}$ of size $|S| = |S' \cup C'| \leq k - s + s = k$, because there exists a minimum solution set for resolving C containing v_2 or v_c of size s, v_1 separates $V \setminus V(C)$ from $V(C) \setminus \{v_1\}$ in G, and if $v_1 \notin S$, then $\deg_{G-S}(v_1) + w_{v_1} \leq \deg_{G'-S'}(v_1) + w'_{v_1} \leq d$, as $|S \cap \{v_2, v_c\}| \geq 1$. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent.

Finally, consider Case 4. Let $\mathcal{I}' = (G' = G - (V(C) \setminus \{v_1\}), k - s, d, w')$ with $w'_{v_1} = w_{v_1} + 2$ and $w'_v = w_v$ for each $v \in V \setminus V(C)$ be the modified instance. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq d - w_u$ holds for each vertex $u \in V \setminus S$. If v_1, v_2 or v_c are contained in S, then $|S \cap V(C)| \geq s+1$ holds, because there does not exist a solution set for resolving C of size at most s that contains v_1, v_2 or v_c. Therefore the set $(S \setminus V(C)) \cup \{v_1\}$ is a solution for $\mathcal{I}'$ of size $|(S \setminus V(C)) \cup \{v_1\}| \leq k - |S \cap V(C)| + 1 \leq k - s$. Otherwise, that is, $v_1, v_2, v_c \notin S$, the set $S' = S \setminus V(C)$ is a solution for $\mathcal{I}'$ of size $|S \setminus V(C)| \leq k - |S \cap V(C)| \leq k - s$, because $|S \cap V(C)| \geq s$ and $\deg_{G'-S'}(v_1) + w'_{v_1} = \deg_{G-S}(v_1) + w_{v_1} \leq d$ holds, as $v_2, v_c \in N_{G-S}(v_1)$, but $v_2, v_c \notin N_{G'-S'}(v_1)$. Hence $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V' := V \setminus (V(C) \setminus \{v_1\})$ of size $|S'| \leq k - s$ such that $\deg_{G'-S'}(u) \leq d - w'_u$ holds for each vertex $u \in V' \setminus S'$. Then there exists a subset $C' \subseteq (V(C) \setminus \{v_1\})$ of size $|C'| = s$ such that $S = S' \cup C'$ is a solution set for $\mathcal{I}$ of size $|S' \cup C'| \leq k - s + s = k$, because there exists a minimum solution set for resolving C containing neither v_1, v_2 nor v_c of size s, v_1 separates $V \setminus V(C)$ from $V(C) \setminus \{v_1\}$ in G, and if $v_1 \notin S$, then $\deg_{G-S}(v_1) + w_{v_1} \leq \deg_{G'-S'}(v_1) + w'_{v_1} \leq d$, as $\deg_{G-S}(v_1) \leq \deg_{G'-S'}(v_1) + 2$. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent.

As we have shown, $\mathcal{I}$ and $\mathcal{I}'$ are equivalent in all four cases. Thus the reduction rule is correct. □

After exhaustively applying the data reduction rules presented so far, all degree-two vertices in the remaining graph are part of a path of degree-two vertices that is adjacent to two distinct vertices of degree at least three.

We will now continue by presenting a data reduction rule that yields a graph where every path of consecutive degree-two vertices contains a constant number of vertices and thereby achieve our linear-size problem kernel. Again, the overall idea is that we want to use as few vertex deletions as possible to resolve the path, and use this budget as good as possible to shrink down the remaining graph, i.e. the two vertices adjacent to the endpoints of the path, but there is an important difference compared to the previous data reduction rule. It is not always clear which of the two endpoints should be shrunk more, if it is not possible to shrink both completely, or if actually a non-minimum solution set resolving the path might be optimal for the whole graph, as the solution set contains both endpoints. Hence we do not try to decide this seemingly difficult problem, but instead we replace every path with a path of bounded length whose minimum solution sets contain the same vertices near the endpoints and therefore behaves equally towards the rest of the graph. As there are many different cases and it is not immediately clear how a replacement path for each case looks, we will now provide a non-constructive proof for this data reduction rule in a first step.

Lemma 3.15. *There exists a reduction rule exhaustively applicable in $\mathcal{O}(n)$ time that yields a graph where every path of consecutive degree-two vertices contains at most ℓ vertices where $\ell \geq 1$ is some constant.*

Proof. We start by describing the reduction rule. Let $\mathcal{I} = (G = (V, E), k, d, w)$ be an instance of WBDD. Let $P = v_1 v_2 \ldots v_p$ be a path in G with $p > \ell + 2$ vertices and inner vertices of degree two. Our goal is to replace P with a path P' containing at most ℓ degree-two vertices and to adjust the budget to obtain an equivalent instance.

Let us introduce some notation. For a given path $P' = v'_1 v'_2 \ldots v'_{p'}$ with $p' \geq 3$ vertices and vertex weights w' let $s(P')$ be the size of a minimum solution set for resolving P'

with weights $w'_{v'_1}$ and $w'_{v'_{p'}}$ set to zero. Furthermore, for each $a \in \{1,2,3\}$ and for each $b \in \{-2,-1,0\}$ let $s_{a,b}(P')$ be the size of a minimum solution set $S \subseteq V(P')$ for resolving P' with weights $w'_{v'_1}$ and $w'_{v'_{p'}}$ set to zero, and with $v'_1 \in S$, if $a = 1$, or if $a = 2$, then $v'_2 \in S$ and $v'_1 \notin S$, or otherwise $S \cap \{v'_1, v'_2\} = \emptyset$, and with $v'_{p'} \in S$, if $b = 0$, or if $b = -1$, then $v'_{p'-1} \in S$ and $v'_p \notin S$, or otherwise $S \cap \{v'_{p'-1}, v'_{p'}\} = \emptyset$.

Remove $P - \{v_1, v_p\}$ from G, add the inner vertices of a path $P' = v'_1 v'_2 \dots v'_{p'}$ with $3 \leq p' \leq \ell + 2$ vertices, vertex weights w' and $s_{a,b}(P') - s(P') = s_{a,b}(P) - s(P)$ for each $a \in \{1,2,3\}$ and $b \in \{-2,-1,0\}$, connect v_1 with v'_2 and v_p with $v'_{p'-1}$, and decrease k by $s(P) - s(P')$. Such a path P' must exist, because otherwise set $\ell = p - 2$ and for any longer path P'' with $s_{a,b}(P'') - s(P'') = s_{a,b}(P) - s(P)$ for each $a \in \{1,2,3\}$ and $b \in \{-2,-1,0\}$ that may exist, P can be used as a replacement.

We now prove that the reduction rule is correct. Let $\mathcal{I}' = (G' = (V', E'), k - s(P) + s(P'), d, w'')$ with vertex set $V' = V \setminus (V(P) \setminus \{v_1, v_p\}) \cup (V(P') \setminus \{v'_1, v'_p\})$, edge set $E' = E(G - (V(P) \setminus \{v_1, v_p\})) \cup \{\{v_1, v'_2\}, \{v'_{p'-1}, v_p\}\} \cup \{\{v'_a, v'_{a+1}\} \mid 2 \leq a \leq p' - 2\}$, and weights $w''_a = w_a$ for each $a \in V \setminus (V(P) \setminus \{v_1, v_p\})$ and $w''_b = w'_b$ for each $b \in V(P') \setminus \{v'_1, v'_{p'}\}$ be the modified instance.

If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(x) \leq d - w_x$ holds for each vertex $x \in V \setminus S$. Let

$$a = \begin{cases} 1 & \text{if } v_1 \in S, \\ 2 & \text{if } v_2 \in S, v_1 \notin S, \\ 3 & \text{if } v_1, v_2 \notin S, \end{cases} \quad \text{and} \quad b = \begin{cases} 0 & \text{if } v_p \in S, \\ -1 & \text{if } v_{p-1} \in S, v_p \notin S, \\ -2 & \text{if } v_p, v_{p-1} \notin S. \end{cases}$$

Then $|S \cap V(P)| \geq s_{a,b}(P)$ holds by definition. Further there exists a vertex subset $S'' \subseteq V(P')$ with $v'_1 \in S''$, if $v_1 \in S$, and $v'_{p'} \in S''$, if $v_p \in S$, of size at most $s_{a,b}(P') = s_{a,b}(P) - s(P) + s(P')$ such that $S' = (S \setminus (V(P) \setminus \{v_1, v_p\})) \cup (S'' \setminus \{v'_1, v'_{p'}\})$ is a solution set for the modified instance $\mathcal{I}'$ of size $|S'| \leq |S| - |S \cap V(P)| + s_{a,b}(P) - s(P) + s(P') \leq k - s(P) + s(P')$, because S'' can be chosen such that v_1 and also v_p are either in both S and S' or their degree in both the original and the modified graph are equal. Hence $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V'$ of size $|S'| \leq k - s(P) + s(P')$ such that $\deg_{G'-S'}(u) \leq d - w''_u$ holds for each vertex $u \in V' \setminus S'$. Let

$$a = \begin{cases} 1 & \text{if } v'_1 \in S', \\ 2 & \text{if } v'_2 \in S', v'_1 \notin S', \\ 3 & \text{if } v'_1, v'_2 \notin S', \end{cases} \quad \text{and} \quad b = \begin{cases} 0 & \text{if } v'_{p'} \in S', \\ -1 & \text{if } v'_{p'-1} \in S', v'_p \notin S', \\ -2 & \text{if } v'_{p'}, v'_{p'-1} \notin S'. \end{cases}$$

Then $|S' \cap V(P')| \geq s_{a,b}(P')$ holds by definition. Further there exists a vertex subset $S'' \subseteq V(P)$ of size at most $s_{a,b}(P) = s_{a,b}(P') - s(P') + s(P)$ such that $S = (S' \setminus V(P')) \cup S''$ is a solution set for $\mathcal{I}$ of size $|S'| - |S' \cap V(P')| + s_{a,b}(P') - s(P') + s(P) \leq k - |S' \cap V(P')| + s_{a,b}(P') \leq k$, because S'' can be chosen such that v_1 and also v_p are either in both S and S' or their degree in both the original and the modified graph are equal. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent.

To apply the reduction rule exhaustively find all paths with more than ℓ vertices consisting only of degree-two vertices in $\mathcal{O}(n)$ time by traversing along degree-two paths

$$\begin{tikzcd}
\boxed{v_1, v_p \in S} & \boxed{v_2, v_p \in S,\, v_1 \notin S} \arrow[l] & \boxed{v_p \in S,\, v_1, v_2 \notin S} \arrow[l] \\
\boxed{v_1, v_{p-1} \in S,\, v_p \notin S} \arrow[u] & \boxed{\begin{array}{c} v_2, v_{p-1} \in S, \\ v_1, v_p \notin S \end{array}} \arrow[l] \arrow[u] & \boxed{\begin{array}{c} v_{p-1} \in S, \\ v_1, v_2, v_p \notin S \end{array}} \arrow[l] \arrow[u] \\
\boxed{v_1 \in S,\, v_{p-1}, v_p \notin S} \arrow[u] & \boxed{\begin{array}{c} v_2 \in S, \\ v_1, v_{p-1}, v_p \notin S \end{array}} \arrow[l] \arrow[u] & \boxed{v_1, v_2, v_{p-1}, v_p \notin S} \arrow[l] \arrow[u]
\end{tikzcd}$$

Figure 3.1: Relationship between different types of solutions for resolving the path $v_1 v_2 \ldots v_p$ with weights w_{v_1} and w_{v_p} set to zero. The arrows point to solutions that are called better. Two solutions are incomparable, if there is no directed path between them.

and marking the visited vertices along the way. For each detected path P compute the value $s_{a,b}(P)$ for each $a \in \{1, 2, 3\}$ and $b \in \{-2, -1, 0\}$ in $\mathcal{O}(n)$ time using Reduction Rules 3.2.2 to 3.2.5, as the paths always contain some vertex of degree at most one, once the containment of v_1 and v_p is fixed. Then replace the paths with matching paths of constant length in $\mathcal{O}(n)$ time. Combining all steps leads to $\mathcal{O}(n)$ time in total. □

We claim that the constant ℓ from Lemma 3.15 is six. To support this claim we will now state such a data reduction rule explicitly split in 21 different cases that together resolve every path containing more than six consecutive degree-two vertices. We will provide a proof of correctness for some exemplary cases and omit it for the remaining ones, as they can be proven analogously and the overall correctness is already given by Lemma 3.15.

To simplify notation we fix the following for the upcoming data reduction rules. Let $\mathcal{I} = (G = (V, E), k, d, w)$ be an instance of WBDD. Let $P = v_1 v_2 \ldots v_p$ be a path in G with $p \geq 9$ vertices and inner vertices of degree two. Now consider different solution sets for resolving P with weights w_{v_1} and w_{v_p} set to zero. Given two solution sets $S_1, S_2 \subseteq V(P)$ we say S_1 *is better than* S_2, if $v_1 \in S_1$ and $v_1 \notin S_2$, or $v_2 \in S_1$ and $v_1, v_2 \notin S_2$, or $v_p \in S_1$ and $v_p \notin S_2$, or $v_{p-1} \in S_1$ and $v_p, v_{p-1} \notin S_2$. See Figure 3.1 for an overview of the different relevant solution types and their relationship. An *optimal* solution set is of minimum size and no other solution set of minimum size is better. We also imply here that an optimal solution set is a solution set for resolving P with weights w_{v_1} and w_{v_p} set to zero.

We start with the case that there exists a solution set of minimum size containing both v_1 and v_p. This solution set is clearly optimal.

Reduction Rule 3.2.8. *If all optimal solution sets contain v_1 and v_p, then delete $V(P)$ and decrease k by the size of the solution set.*

Observation 3.16. *Reduction Rule 3.2.8 is correct.*

Proof. Let s be the size of an optimal solution set. If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq d - w_u$ holds for each vertex $u \in V \setminus S$. Then the set $S \setminus V(P)$ is a solution for the modified instance $\mathcal{I}' = (G - V(P), k - s, d, w)$ of size $|S \setminus V(P)| \le k - |S \cap V(P)| \le k - s$, as $|S \cap V(P)| \ge s$ holds. Therefore $\mathcal{I}'$ is also a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V' := V \setminus V(P)$ of size $|S'| \le k - s$ such that $\deg_{G-V(P)-S'}(u) \le d - w_u$ holds for each vertex $u \in V' \setminus S'$. Then there exists a subset $P' \subseteq V(P)$ of size $|P'| = s$ containing both v_1 and v_p such that $S' \cup P'$ is a solution set for $\mathcal{I}$ of size $|S' \cup P'| \le k - s + s = k$, because the minimum solution set for resolving P containing both v_1 and v_p is of size s and $\{v_1, v_p\}$ separates $V \setminus V(P)$ and $V(P) \setminus \{v_1, v_p\}$ in G. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent. $\square$

We will continue with the case that there does not exist an optimal solution set containing both v_1 and v_p, rather every minimum solution set containing both v_1 and v_p is one vertex larger than any optimal solution set.

Reduction Rule 3.2.9. *If every minimum solution set that contains both v_1 and v_p is one vertex larger than any optimal solution set, and*

1. *all optimal solution sets contain neither v_1, v_2, v_{p-1} nor v_p, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2a and decrease k by the size of the solution set minus two.*

2. *all optimal solution sets contain v_2, but not v_p or v_{p-1}, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2b and decrease k by the size of the solution set minus two.*

3. *all optimal solution sets contain either v_2, but not v_p or v_{p-1}, or v_{p-1}, but not v_1 or v_2, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2c and decrease k by the size of the solution set minus one.*

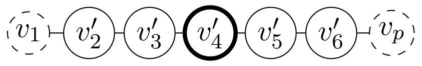
4. *all optimal solution sets contain v_1, but not v_p or v_{p-1}, then delete $V(P) \setminus \{v_p\}$, increase the weight w_p by one, and decrease k by the size of the solution set.*

5. *all optimal solution sets contain either v_1, but not v_p or v_{p-1}, or v_{p-1}, but not v_1 or v_2, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2d and decrease k by the size of the solution set minus two.*

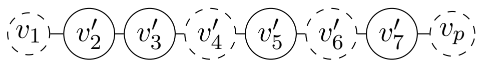
6. *all optimal solution sets contain either v_1, but not v_p or v_{p-1}, or v_p, but not v_1 or v_2, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2e and decrease k by the size of the solution set minus three.*

7. *all optimal solution sets contain either v_1, but not v_p or v_{p-1}, or v_p, but not v_1 or v_2, or both v_2 and v_{p-1}, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2f and decrease k by the size of the solution set minus two.*

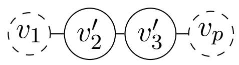
8. *all optimal solution sets contain both v_2 and v_{p-1}, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2g and decrease k by the size of the solution set minus two.*



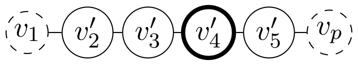
(a) Replacement path with minimum solution size two for Case 1 of Reduction Rule 3.2.9.



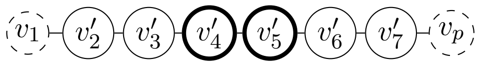
(b) Replacement path with minimum solution size two for Case 2 of Reduction Rule 3.2.9.



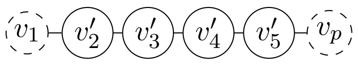
(c) Replacement path with minimum solution size one for Case 3 of Reduction Rule 3.2.9.



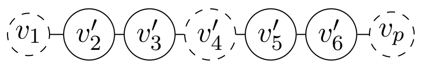
(d) Replacement path with minimum solution size two for Case 5 of Reduction Rule 3.2.9.



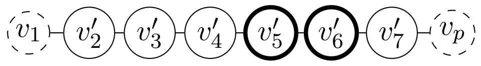
(e) Replacement path with minimum solution size three for Case 6 of Reduction Rule 3.2.9.



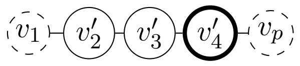
(f) Replacement path with minimum solution size two for Case 7 of Reduction Rule 3.2.9.



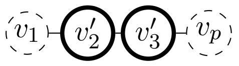
(g) Replacement path with minimum solution size two for Case 8 of Reduction Rule 3.2.9.



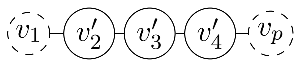
(h) Replacement path with minimum solution size three for Case 9 of Reduction Rule 3.2.9.



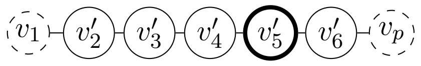
(i) Replacement path with minimum solution size two for Case 11 of Reduction Rule 3.2.9.



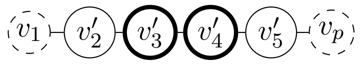
(j) Replacement path with minimum solution size two for Case 12 of Reduction Rule 3.2.9.



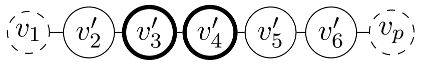
(k) Replacement path with minimum solution size one for Case 1 of Reduction Rule 3.2.10.



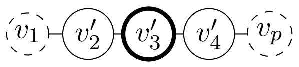
(l) Replacement path with minimum solution size two for Case 2 of Reduction Rule 3.2.10.



(m) Replacement path with minimum solution size two for Case 3 of Reduction Rule 3.2.10.



(n) Replacement path with minimum solution size two for Case 1 of Reduction Rule 3.2.11.



(o) Replacement path with minimum solution size one for Reduction Rule 3.2.12.

Figure 3.2: Paths that are used in the data reduction rules as short replacements for long paths. Vertices with weight d, $d-1$, and at most $d-2$ are marked with a bold, normal, and dashed border, respectively.

9. *all optimal solution sets contain either v_1, but not v_p or v_{p-1}, or both v_2 and v_{p-1}, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2h and decrease k by the size of the solution set minus three.*

10. *all optimal solution sets contain both v_1 and v_{p-1}, then delete $V(P) \setminus \{v_p\}$ and decrease k by the size of the solution set.*

11. *all optimal solution sets contain either both v_1 and v_{p-1}, or v_p, but not v_1 or v_2, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2i and decrease k by the size of the solution set minus two.*

12. *all optimal solution sets contain either both v_1 and v_{p-1}, or both v_p and v_2, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2j and decrease k by the size of the solution set minus two.*

Observation 3.17. *Case 1 of Reduction Rule 3.2.9 is correct.*

Proof. Let s be the size of an optimal solution set. Let $\mathcal{I}' = (G' = (V', E'), k-s+2, d, w')$ with vertex set $V' = V \setminus (V(P) \setminus \{v_1, v_p\}) \cup \{v'_2, \ldots, v'_6\}$, edge set $E' = E(G - (V(P) \setminus \{v_1, v_p\})) \cup \{\{v_1, v'_2\}, \{v'_6, v_p\}\} \cup \{\{v'_a, v'_{a+1}\} \mid 2 \leq a \leq 5\}$, and weights $w'_{v'_2} = w'_{v'_3} = w'_{v'_5} = w'_{v'_6} = d-1$ and $w'_{v'_4} = d$ and $w'_a = w_a$ for each $a \in V \setminus (V(P) \setminus \{v_1, v_p\})$ be the modified instance.

If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq d - w_u$ holds for each vertex $u \in V \setminus S$. Also $|S \cap V(P)| \geq s$ holds, as the minimum solution set for resolving P is of size s. If $v_1, v_2, v_{p-1}, v_p \notin S$, then the set $S' = (S \setminus V(P)) \cup \{v'_3, v'_5\}$ is a solution set for $\mathcal{I}'$ of size $|S'| = |S| - |S \cap V(P)| + 2 \leq k - s + 2$, because S' resolves the replacement path $v'_2 v'_3 \ldots v'_6$ and both $\deg_{G'-S'}(v_1) = \deg_{G-S}(v_1)$ and $\deg_{G'-S'}(v_p) = \deg_{G-S}(v_p)$, as $v_2, v_{p-1} \notin S$ and $v'_2, v'_6 \notin S'$.

If $\{v_1, v_2, v_{p-1}, v_p\} \cap S \neq \emptyset$, then $|S \cap V(P)| \geq s+1$ holds, as all minimum solution sets are of size s and contain neither v_1, v_2, v_{p-1} nor v_p. Construct a solution set S' for $\mathcal{I}'$ as follows. Initially $S' = (S \setminus (V(P) \setminus \{v_1, v_p\})) \cup \{v'_4\}$. If $v_1 \notin S$, then add v'_2 to S'. If $v_p \notin S$, then add v'_6 to S'. Now S' is a solution set for $\mathcal{I}'$, since S' resolves the replacement path $v'_2 v'_3 \ldots v'_6$, it holds that $\{v_1, v_p\} \cap S = \{v_1, v_p\} \cap S'$, and if v_1 and v_p are not in S, then $\deg_{G'-S'}(v_1) \leq \deg_{G-S}(v_1)$ and $\deg_{G'-S'}(v_p) \leq \deg_{G-S}(v_p)$, respectively. It is of size $|S'| = |S| - |S \cap V(P)| + 3 \leq k - (s+1) + 3 = k - s + 2$, because either v_1 or $v'_2 \in S'$ and either v_p or $v'_6 \in S'$. Hence $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V'$ of size $|S'| \leq k - s + 2$ such that $\deg_{G'-S'}(u) \leq d - w'_u$ holds for each vertex $u \in V' \setminus S'$. Also $|S' \cap \{v'_2, \ldots, v'_6\}| \geq 2$ holds, as the minimum solution set for resolving the path $v'_2 v'_3 \ldots v'_6$ is of size two. If $v_1, v'_2, v'_6, v_p \notin S'$, then there exists a subset $P' \subseteq V(P)$ of size $|P'| = s$ such that $S = S' \setminus \{v'_2, \ldots, v'_6\} \cup P'$ is a solution set for $\mathcal{I}$ of size $|S| = |S'| - |S' \cap \{v'_2, \ldots, v'_6\}| + s \leq k$, because the minimum solution set for resolving P is of size s, and if v_1 and v_p are not in P', then $\deg_{G-S}(v_1) \leq \deg_{G'-S'}(v_1)$ and $\deg_{G-S}(v_p) \leq \deg_{G'-S'}(v_p)$, respectively, as $v'_2, v'_6 \notin S'$.

If $\{v_1, v'_2, v'_6, v_p\} \cap S' \neq \emptyset$, then $|S' \cap \{v_1, v'_2, \ldots, v'_6, v_p\}| \geq 3$ holds, as all minimum solution sets for resolving the path $v_1 v'_2 \ldots v'_6 v_p$, even with weights w_{v_1} and w_{v_p} set to zero, are of size two, but do not contain v_1, v'_2, v'_6 or v_p. Therefore there exists a subset $P' \subseteq V(P)$ of size $|P'| = s+1$ containing both v_1 and v_p such that $S = S' \setminus \{v_1, v_2', \dots, v_6', v_p\} \cup P'$ is a solution set for $\mathcal{I}$ of size $|S| \leq |S'| - |S' \cap \{v_1, v_2', \dots, v_6', v_p\}| + s + 1 \leq k + 3 - |S' \cap \{v_1, v_2', \dots, v_6', v_p\}| \leq k$, because the minimum solution set containing v_1 and v_p for resolving P is of size $s+1$. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent. □

Observation 3.18. *Case 4 of Reduction Rule 3.2.9 is correct.*

Proof. Let s be the size of an optimal solution set. Let $\mathcal{I}' = (G - (V(P) \setminus \{v_p\}), k - s, d, w')$ with weights $w'_{v_p} = w_{v_p} + 1$ and $w'_a = w_a$ for each $a \in V \setminus V(P)$ be the modified instance.

If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \leq k$ such that $\deg_{G-S}(u) \leq d - w_u$ holds for each vertex $u \in V \setminus S$. Also $|S \cap V(P)| \geq s$ holds, as the minimum solution set for resolving P is of size s. If $v_p, v_{p-1} \notin S$, then the set $S' = S \setminus V(P)$ is a solution set for $\mathcal{I}'$ of size $|S'| = |S| - |S \cap V(P)| \leq k - s$, because $\deg_{G-(V(P)\setminus\{v_p\})-S'}(v_p) + w'_{v_p} = \deg_{G-S}(v_p) + w_{v_p} \leq d$, as $v_{p-1} \in N_{G-S}(v_p)$, but $v_{p-1} \notin N_{G-(V(P)\setminus\{v_p\})-S'}(v_p)$. If $v_p \in S$ or both $v_p \notin S$ and $v_{p-1} \in S$, then $|S \cap V(P)| \geq s+1$ holds, as all minimum solution set for resolving P are of size s, but do not contain v_p or v_{p-1}. Then the set $S' = (S \setminus V(P)) \cup \{v_p\}$ is a solution set for $\mathcal{I}'$ of size $|S'| = |S| - |S \cap V(P)| + 1 \leq k - s$. Hence $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V' := V \setminus (V(P) \setminus \{v_p\})$ of size $|S'| \leq k - s$ such that $\deg_{G-(V(P)\setminus\{v_p\})-S'}(u) \leq d - w'_u$ holds for each vertex $v \in V' \setminus S'$. Then there exists a subset $P' \subseteq V(P)$ of size $|P'| = s$ containing v_1 such that $S = S' \cup P'$ is a solution set for $\mathcal{I}$ of size $|S' \cup P'| \leq k - s + s = k$, because the minimum solution set for resolving P containing v_1 is of size s, it holds that $\deg_{G-S}(v_p) + w_{v_p} \leq \deg_{G-(V(P)\setminus\{v_p\})-S'}(v_p) + w'_{v_p} \leq d$, as $|N_{G-S}(v_p) \cap V(P)| \leq 1$, and $\{v_1, v_p\}$ separates $V \setminus V(P)$ from $V(P) \setminus \{v_1, v_p\}$ in G. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent. □

This concludes the rule for the case that every minimum solution set containing both v_1 and v_p is one vertex larger than any optimal solution set. We will continue with the case that every minimum solution set containing both v_1 and v_p is two vertices larger than any optimal solution set and every minimum solution set containing either both v_1 and v_{p-1} or both v_2 and v_p is one vertex larger than any optimal solution set.

Reduction Rule 3.2.10. *If every minimum solution set that contains both v_1 and v_p is two vertices larger than any optimal solution set, and every minimum solution set that contains either both v_1 and v_{p-1} or both v_2 and v_p is one vertex larger than any optimal solution set, and*

1. *all optimal solution sets contain neither v_1, v_2, v_{p-1} nor v_p, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2k and decrease k by the size of the solution set minus one.*

2. *all optimal solution sets contain v_2, but not v_p or v_{p-1}, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2l and decrease k by the size of the solution set minus two.*

3. *all optimal solution sets contain either v_2, but not v_p or v_{p-1}, or v_{p-1}, but not v_1 or v_2, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2m and decrease k by the size of the solution set minus two.*

4. *all optimal solution sets contain both v_2 and v_{p-1}, then delete $V(P) \setminus \{v_1, v_p\}$, and decrease k by the size of the solution set.*

Observation 3.19. *Case 1 of Reduction Rule 3.2.10 is correct.*

Proof. Let s be the size of an optimal solution set. Let $\mathcal{I}' = (G' = (V', E'), k-s+1, d, w')$ with vertex set $V' = V \setminus (V(P) \setminus \{v_1, v_p\}) \cup \{v_2', v_3', v_4'\}$, edge set $E' = E(G - (V(P) \setminus \{v_1, v_p\})) \cup \{\{v_1, v_2'\}, \{v_4', v_p\}\} \cup \{\{v_a', v_{a+1}'\} \mid 2 \le a \le 3\}$, and weights $w'_{v_2'} = w'_{v_3'} = w'_{v_4'} = d - 1$ and $w'_a = w_a$ for each $a \in V \setminus (V(P) \setminus \{v_1, v_p\})$ be the modified instance.

If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq V$ of size $|S| \le k$ such that $\deg_{G-S}(u) \le d - w_u$ holds for each vertex $u \in V \setminus S$. Also $|S \cap V(P)| \ge s$ holds, as the minimum solution set for resolving P is of size s. If $v_1, v_2, v_{p-1}, v_p \notin S$, then the set $S' = (S \setminus V(P)) \cup \{v_3'\}$ is a solution set for $\mathcal{I}'$ of size $|S'| = |S| - |S \cap V(P)| + 1 \le k - s + 1$, because S' resolves the replacement path $v_2'v_3'v_4'$ and both $\deg_{G-S}(v_1) = \deg_{G'-S'}(v_1)$ and $\deg_{G-S}(v_p) = \deg_{G'-S'}(v_p)$, as $v_2, v_{p-1} \notin S$ and $v_2', v_4' \notin S'$.

If $\{v_1, v_2, v_{p-1}, v_p\} \cap S \neq \emptyset$, but $v_1 \notin S$ or $v_p \notin S$, then $|S \cap V(P)| \ge s + 1$ holds, as all minimum solution sets are of size s and contain neither v_1, v_2, v_{p-1} nor v_p. Construct a solution set S' for $\mathcal{I}'$ as follows. Initially $S' = S \setminus (V(P) \setminus \{v_1, v_p\})$. If $v_1 \notin S$, then add v_2' to S'. If $v_p \notin S$, then add v_4' to S'. Now S' is a solution set for $\mathcal{I}'$, since S' resolves the replacement path $v_2'v_3'v_4'$, it holds that $\{v_1, v_p\} \cap S = \{v_1, v_p\} \cap S'$, and if v_1 and v_p are not in S, then $\deg_{G'-S'}(v_1) \le \deg_{G-S}(v_1)$ and $\deg_{G'-S'}(v_p) \le \deg_{G-S}(v_p)$, respectively. It is of size $|S'| = |S| - |S \cap V(P)| + 2 \le |S| - s + 1 \le k - s + 1$, because either v_1 or $v_2' \in S'$, and either v_p or $v_4' \in S'$.

If $v_1, v_p \in S$, then $|S \cap V(P)| \ge s + 2$ holds, as every minimum solution set that contains both v_1 and v_p is of size $s+2$. Therefore the set $S' = (S \setminus (V(P) \setminus \{v_1, v_p\})) \cup \{v_3'\}$ is a solution set for $\mathcal{I}'$ of size $|S'| = |S| - |S \cap V(P)| + 3 \le |S| - s + 1 = k - s + 1$, as S' resolves the replacement path $v_2'v_3'v_4'$. Hence $\mathcal{I}'$ is a yes-instance.

If $\mathcal{I}'$ is a yes-instance, then there exists a solution set $S' \subseteq V'$ of size $|S'| \le k - s + 1$ such that $\deg_{G'-S'}(u) \le d - w_u'$ holds for each vertex $u \in V' \setminus S'$. Also $|S' \cap \{v_2', v_3', v_4'\}| \ge 1$ holds, as the minimum solution set for resolving the path $v_2'v_3'v_4'$ is of size one. If $v_1, v_2', v_4', v_p \notin S'$, then there exists a subset $P' \subseteq V(P)$ of size $|P'| = s$ such that $S = S' \setminus \{v_2', v_3', v_4'\} \cup P'$ is a solution set for $\mathcal{I}$ of size $|S| = |S'| - |S' \cap \{v_2', v_3', v_4'\}| + s \le k$, because the minimum solution set for resolving P is of size s, and if v_1 and v_p are not in P', then $\deg_{G-S}(v_1) \le \deg_{G'-S'}(v_1)$ and $\deg_{G-S}(v_p) \le \deg_{G'-S'}(v_p)$, respectively, as $v_2', v_4' \notin S'$.

If $\{v_1, v_2', v_4', v_p\} \cap S' \neq \emptyset$, but $v_1 \notin S'$ or $v_p \notin S'$, then $|S' \cap \{v_1, v_2', v_3', v_4', v_p\}| \ge 2$ holds, as all minimum solution sets for resolving the path $v_1v_2'v_3'v_4'v_p$, even with weights w_{v_1} and w_{v_p} set to zero, are of size one, but do not contain v_1, v_2', v_4' or v_p. If $v_1 \in S'$, then there exists a subset $P' \subseteq V(P)$ of size $|P'| = s + 1$ containing both v_1 and v_{p-1} such that $S = S' \setminus \{v_1, v_2', v_3', v_4', v_p\} \cup P'$ is a solution set for $\mathcal{I}$ of size $|S| \le |S'| - |S' \cap \{v_1, v_2', v_3', v_4', v_p\}| + s + 1 \le k$, because the minimum solution set containing v_1 and v_{p-1} for resolving P is of size $s + 1$. If $v_p \in S'$, then there exists a subset $P' \subseteq V(P)$

of size $|P'| = s+1$ containing both v_2 and v_p such that $S = S' \setminus \{v_1, v_2', v_3', v_4', v_p\} \cup P'$ is a solution set for $\mathcal{I}$ of size $|S| \leq |S'| - |S' \cap \{v_1, v_2', v_3', v_4', v_p\}| + s + 1 \leq k$, because the minimum solution set containing v_2 and v_p for resolving P is of size $s+1$. If $v_1, v_p \notin S'$, then there exists a subset $P' \subseteq V(P)$ of size $|P'| = s+1$ containing both v_2 and v_{p-1} such that $S = S' \setminus \{v_1, v_2', v_3', v_4', v_p\} \cup P'$ is a solution set for $\mathcal{I}$ of size $|S| \leq |S'| - |S' \cap \{v_1, v_2', v_3', v_4', v_p\}| + s + 1 \leq k$, because the minimum solution set containing v_2 and v_{p-1} for resolving P is of size $s+1$. Furthermore, in all three sub cases it holds that if v_1 and v_p are not in S, then $\deg_{G-S}(v_1) = \deg_{G'-S'}(v_1)$ and $\deg_{G-S}(v_p) = \deg_{G'-S'}(v_p)$, respectively.

If $v_1, v_p \in S'$, then $|S' \cap \{v_1, v_2', v_3', v_4', v_p\}| \geq 3$ holds, as the minimum solution set containing both v_1 and v_p for resolving the path $v_1 v_2' v_3' v_4' v_p$, even with weights w_{v_1} and w_{v_p} set to zero, is of size three. Therefore there exists a subset $P' \subseteq V(P)$ of size $|P'| = s+2$ containing both v_1 and v_p such that $S = S' \setminus \{v_2', v_3', v_4'\} \cup P'$ is a solution set for $\mathcal{I}$ of size $|S| \leq |S'| - |S' \cap \{v_2', v_3', v_4'\}| + s + 2 \leq k$, because the minimum solution set containing both v_1 and v_p for resolving P is of size $s+2$. Hence $\mathcal{I}$ is a yes-instance and both instances are equivalent. □

This concludes the rule for the case that every minimum solution set containing both v_1 and v_p is two vertices larger than any optimal solution set and every minimum solution set containing either both v_1 and v_{p-1} or both v_2 and v_p is one vertex larger than any optimal solution set. We will continue with the case that every minimum solution set containing both v_1 and v_{p-1} is two vertices larger than any optimal solution set and every minimum solution set containing either v_1, but not v_p or v_{p-1}, or both v_2 and v_p is one vertex larger than any optimal solution set.

Reduction Rule 3.2.11. *If every minimum solution set that contains both v_1 and v_{p-1} is two vertices larger than any optimal solution set, and every minimum solution set that contains either v_1, but not v_p or v_{p-1}, or both v_2 and v_p is one vertex larger than any optimal solution set, and*

1. *all optimal solution sets contain neither v_1, v_2, v_{p-1} nor v_p, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2n and decrease k by the size of the solution set minus two.*

2. *all optimal solution sets contain v_2, but not v_p or v_{p-1}, then delete $V(P) \setminus \{v_1, v_p\}$, increase w_{v_p} by one, and decrease k by the size of the solution set.*

We will continue with the case that every minimum solution set containing both v_2 and v_p, or both v_1 and v_{p-1} is two vertices larger than any optimal solution set and every minimum solution set containing either v_1, but not v_p or v_{p-1}, or both v_2 and v_{p-1}, or v_p, but not v_1 or v_2 is one vertex larger than any optimal solution set.

Reduction Rule 3.2.12. *If every minimum solution set that contains both v_2 and v_p or both v_1 and v_{p-1} is two vertices larger than any optimal solution set, and every minimum solution set that contains either v_1, but not v_p or v_{p-1}, or both v_2 and v_{p-1}, or v_p, but not v_1 or v_2 is one vertex larger than any optimal solution set, then replace $V(P) \setminus \{v_1, v_p\}$ by the path shown in Figure 3.2o and decrease k by the size of the solution set minus one.*

The next rule will cover the final case that every minimum solution set containing both v_2 and v_{p-1} is two vertices larger than any optimal solution set. Note that this implies that every minimum solution set containing v_1, v_2, v_{p-1} or v_p is at least one vertex larger than any optimal solution set.

Reduction Rule 3.2.13. *If every minimum solution set that contains both v_2 and v_{p-1} is two vertices larger than any optimal solution set, then delete $V(P) \setminus \{v_1, v_p\}$, increase both w_{v_1} and w_{v_p} by one, and decrease k by the size of the solution set.*

This concludes all data reduction rules necessary to resolve all connected components in $G[V_2(G)]$ that contain more than six degree-two vertices. We will now prove that this is indeed the case and that we can apply them exhaustively in $\mathcal{O}(n + m)$ time.

Lemma 3.20. *After applying Reduction Rules 3.2.6 to 3.2.13 exhaustively, each connected component in $G[V_2(G)]$ contains at most six vertices.*

Proof. Every cycle and every path adjacent to only one vertex, respectively, is deleted from G by applying Reduction Rules 3.2.6 and 3.2.7 exhaustively, if they consist only of degree-two vertices. Then all remaining degree-two vertices are part of paths that consist only of degree-two vertices and are adjacent to two distinct neighbors. We will now show that such a path, if it contains more than six vertices, is resolved by applying the Reduction Rules 3.2.8 to 3.2.13 exhaustively. Then every connected component in $G[V_2(G)]$ contains at most six vertices and we are done. See again Figure 3.1 for an overview over different types of solutions.

Let $P = v_1 v_2 \ldots v_p$ be some path in G with $p \geq 9$ vertices and inner vertices of degree two. If there exists some minimum solution set containing both v_1 and v_p, then Reduction Rule 3.2.8 is applicable and results in the deletion of P. Otherwise every solution set for resolving P that contains both v_1 and v_p is larger than any minimum solution set by one or two vertices, because a solution set containing both v_1 and v_p can always be created by adding v_1 and v_p to any minimum solution set.

We now cover the cases that every minimum solution set containing both v_1 and v_p is one vertex larger than any optimal solution set. There can occur eight different types of optimal solutions. The case that all optimal solution sets have the same type, that is, for every pair of optimal solutions $S_1, S_2 \subseteq V(P)$ it holds that $S_1 \cap \{v_1, v_2, v_{p-1}, v_p\} = S_2 \cap \{v_1, v_2, v_{p-1}, v_p\}$, is covered by Cases 1, 2, 4, 8 and 10 of Reduction Rule 3.2.9. Note that Cases 2, 4 and 10 each cover two cases due to symmetry and therefore all eight cases are covered. The case that different types of optimal solution sets that are incomparable to each other appear simultaneously is covered by Cases 3, 5 to 7, 9, 11 and 12 of Reduction Rule 3.2.9. Due to symmetry Cases 5, 9 and 11 each cover two cases. As a result all ten cases are covered.

We now cover the cases that every minimum solution set that contains both v_1 and v_p is two vertices larger than any minimum solution set and every minimum solution set that contains either both v_1 and v_{p-1} or both v_2 and v_p is one vertex larger than any minimum solution set. Then no optimal solution set contains v_1 or v_p, otherwise there exists a solution set containing both v_1 and v_p that is only one vertex larger than this optimal solution set. Therefore only four different types of optimal solutions can occur. The four cases that all optimal solutions are of the same type are covered by Cases 1, 2 and 4 of Reduction Rule 3.2.10 with Case 2 covering two cases due to symmetry. There are two types of optimal solutions that are incomparable and the case that both appear simultaneously is covered by Case 3 of Reduction Rule 3.2.10.

We now cover the cases that every minimum solution set that contains both v_1 and v_{p-1} is two vertices larger than any minimum solution set and every minimum solution set that contains either v_1, but not v_{p-1} or v_p, or both v_2 and v_p is one vertex larger than any minimum solution set. Note that due to symmetry this also covers the case that every minimum solution set that contains both v_2 and v_p is two vertices larger than any minimum solution set and every minimum solution set that contains either v_p, but not v_1 or v_2, or both v_1 and v_{p-1} is one vertex larger than any minimum solution set. No optimal solution set can contain v_1, v_{p-1} or v_p, as otherwise there exists a solution set containing both v_1 and v_{p-1} that is only one vertex larger than this optimal solution set. Therefore only two different types of optimal solutions can occur. The two cases that all optimal solutions are of the same type are covered by Cases 1 and 2 of Reduction Rule 3.2.11. Since the two types of optimal solutions that can occur are comparable to each other, there cannot occur optimal solutions of different types simultaneously.

We now cover the case that every minimum solution set that contains both v_2 and v_p, or both v_1 and v_{p-1} is two vertices larger than any minimum solution set, and every minimum solution set that contains either v_1, but not v_p or v_{p-1}, or both v_2 and v_{p-1}, or v_p, but not v_1 or v_2 is one vertex larger than any minimum solution set. All optimal solution sets must contain neither v_1, v_2, v_{p-1} nor v_p, as otherwise there exists a solution set containing both v_2 and v_p or both v_1 and v_{p-1} that is only one vertex larger than this optimal solution set. Hence Reduction Rule 3.2.12 covers this case completely.

Finally, we cover the case that every minimum solution set that contains both v_2 and v_{p-1} is two vertices larger than any minimum solution set. All optimal solution sets must contain neither v_1, v_2, v_{p-1} nor v_p, as otherwise there exists a solution set containing both v_2 and v_{p-1} that is only one vertex larger than this optimal solution set. Hence Reduction Rule 3.2.13 covers this case completely.

Because every minimum solution set that contains neither v_1 nor v_2 or neither v_p nor v_{p-1} is at most one vertex larger than any minimum solution set, the discussed cases are exhaustive and we are done. □

Lemma 3.21. *Reduction Rules 3.2.2 to 3.2.13 can all be applied exhaustively in $\mathcal{O}(n+m)$ time.*

Proof. Check the degree and weight of each vertex in $\mathcal{O}(n)$ time to find all vertices where one of the Reduction Rules 3.2.2 to 3.2.5 can be applied. Furthermore, test for each degree-two vertex if it is part of a subgraph where one of the other reduction rules can be applied, that is, the vertex is part of a cycle consisting only of degree-two vertices, or a path consisting only of degree-two vertices that is either adjacent to only one vertex or contains at least seven vertices. This can also be done in $\mathcal{O}(n)$ time by traversing along degree-two paths and marking the visited vertices along the way. By applying the reduction rules the graph may be modified in a way such that a rule becomes applicable where it was not applicable before. Hence we have to repeat the test for vertices where we increase the weight above d, or decrease its degree to at most two. Since this only happens if at least one neighbor gets deleted, this is possible in $\mathcal{O}(m)$ time by adding all neighbors of the deleted vertices to a list of vertices that still need to be tested.

It is necessary to compute the size of two, five and nine different minimum solution sets for resolving certain subgraphs to apply Reduction Rules 3.2.6 and 3.2.7 and Reduction Rules 3.2.8 to 3.2.13, respectively. This can be done in $\mathcal{O}(n)$ time using Reduction Rules 3.2.2 to 3.2.5, as the subgraphs are paths that always contain some vertex of degree at most one. Modifying the graph according to the reduction rules requires $\mathcal{O}(n+m)$ time, because deletion of every vertex while maintaining a correct graph representation is possible in this time and apart from deleting vertices the rules only add shorter, constant-length replacement paths. Combining every step leads to a total running time of $\mathcal{O}(n+m)$. □

We can now use the presented data reduction rules and lemmata to obtain a linear-size problem kernel.

Theorem 3.22. Weighted Bounded-Degree Deletion *admits a linear-size problem kernel computable in $\mathcal{O}(n+m)$ time when parameterized by the feedback edge set number.*

Proof. Let (G, k, d, w) be the instance of WBDD. Let $F \subseteq E$ be a minimum feedback edge set for G computable in $\mathcal{O}(n+m)$ time using depth-first search. Apply Reduction Rules 3.2.2 to 3.2.13 exhaustively in $\mathcal{O}(n+m)$ time (see Lemma 3.21) to remove all vertices with degree at most one (see Observation 3.12) and to upper-bound the size of each connected component in $G[V_2(G)]$ by six vertices (see Lemma 3.20). Then the remaining graph contains at most $2|F|$ vertices of degree at least three and at most $18|F| - 6$ vertices of degree two, using Lemmas 3.3 and 3.4 respectively. If the budget $k \geq n$ or $d \geq n$, then output a trivial yes-instance, so assume $k, d < n$. Now it holds that $k, d < n \leq 20|F| - 6$ and $m \leq 21|F| - 7$. Hence the size of the remaining instance is bounded linearly in $|F|$. □

3.3 Parameterization by Distance to Disjoint Paths

In this section, we show that Biclique-Free Vertex Deletion is FPT when parameterized by the distance to disjoint paths. This parameter is at most the vertex cover number for which we already established that Biclique-Free Vertex Deletion is FPT, but at least the feedback vertex set number for which we know that the special case BDD is W[1]-hard. By providing an FPT algorithm when parameterized by the distance to disjoint paths we fill the "gap" between the known complexities with respect to the vertex cover number and the feedback vertex set number.

The distance to disjoint paths of a graph $G = (V, E)$ is the size of a minimum set $D \subseteq V$ such that every connected component in $G - D$ is a path. We assume that such a set D is given along with the problem input, but one can compute it in FPT time with respect to the distance to disjoint paths of the graph [KS21, Lemma 1].

We again split our algorithm in different cases. We start by presenting an algorithm for the case $i \geq 2$ or $j \geq 3$ and continue with the case $i = 1, j = 2$ after that. The remaining case of $i = j = 1$ known as Vertex Cover can be solved in FPT time with respect to the distance to disjoint paths using a well-known algorithm for Vertex

Cover parameterized by treewidth. See for example the version in the textbook of Cygan et al. [Cyg+15, Corollary 7.6] running in $\mathcal{O}(2^{|D|} \cdot |D|^2 \cdot n)$ time. Hence we can use this algorithm to solve the last case $i = j = 1$.

The following algorithm uses a depth-bounded search tree to resolve all bicliques based on the observation that each biclique containing at least one edge with both endpoints not in D contains at most three vertices that are not in D.

Theorem 3.23. *Let $D \subseteq V$ be a set such that every connected component in $G - D$ is a path. Then* Biclique-Free Vertex Deletion *with $i \geq 2$ or $j \geq 3$ can be solved in $\mathcal{O}(2^{|D|+|D|^2} \cdot |D|^2 \cdot n^2)$ time.*

Proof. If the budget $k \geq |D|$, then output *yes*, because deleting all vertices in D would be covered by the budget and the remaining paths cannot contain any bicliques with $i \geq 2$ or $j \geq 3$. So assume $k < |D|$ for the rest of this algorithm. Use the extension of the algorithm presented in Theorem 2.2 as discussed in Section 2.3 to obtain a set of parts of all bicliques in G in $\mathcal{O}(2^{\omega} \cdot \omega^2 \cdot n^2)$ time where ω denotes the treewidth of G. This also includes the time required for computing the necessary tree decomposition of G (see Section 2.1). Based on this set delete all vertices and edges that are not part of at least one $K_{i,j}$ (see Reduction Rule 2.3.1).

Let $P := V \setminus D$ be the set of vertices that form the disjoint paths that remain if one would delete D from G. Two vertices in P with a neighborhood that is a subset of D are of the same *type*, if they have the same neighborhood.

Now pick an arbitrary biclique that contains at least one edge from $E(G[P])$ and resolve it by deleting one of its vertices contained in D or P, or, if such a biclique does not exist, pick any remaining biclique and resolve it by deleting one of its vertices that is either contained in D or of some neighborhood type. Delete all edges and vertices that are no longer part of some biclique and update the list of parts of bicliques. Decrease k by one. Then repeat this procedure with any remaining bicliques, if possible one containing at least one edge from $E(G[P])$, until the budget is used up or the graph is biclique-free. Output *yes*, if the graph becomes biclique-free with a nonnegative budget remaining, otherwise output *no*. This concludes the description of the algorithm.

We now show two properties central for achieving the stated running time and for the correctness. If there does not exist any biclique that contains at least one edge from $E(G[P])$, then $|E(G[P])| = \emptyset$ holds, because all edges that are not part of at least one biclique are deleted. Therefore all vertices in P have a neighborhood that is a subset of D.

A biclique containing at least one edge from $E(G[P])$ contains at least one vertex from P on either side. Then the biclique contains at most three vertices from P in total, because the edges from the biclique between at least four of its vertices from P would not form a path.

This algorithm yields a depth-bounded search tree with at most $|D| + 2^{|D|} + 3$ options and at most $k < |D|$ choices. It runs in $\mathcal{O}(2^{|D|+|D|^2} \cdot |D|^2 \cdot n^2)$ time (using $|D| + 1 \geq \omega$), including the time required for applying Reduction Rule 2.3.1 exhaustively and maintaining necessary data structures, e.g. a list of parts of the remaining bicliques.

We do not need to distinguish between vertices of the same type, because if an instance of Biclique-Free Vertex Deletion has a solution containing a specific subset of vertices all of the same type, then it also has a solution that instead contains any other equally sized subset of vertices all of this type. Then this algorithm is correct, as the search tree will always find a solution of size at most k, if such a solution exists. $\square$

We will now solve the remaining case of $i = 1$, $j = 2$ using dynamic programming. Its running time is actually a bit shorter than the previous algorithm.

Theorem 3.24. *Let $D \subseteq V$ be a set such that every connected component in $G - D$ is a path. Then* Biclique-Free Vertex Deletion *with $i = 1$ and $j = 2$ can be solved in $\mathcal{O}(2^{2|D|} \cdot |D| \cdot (n + m))$ time.*

Proof. In order to decide which vertices from D should be in the solution set, guess $D' \subseteq D$, delete D' and decrease k by $|D'|$. Now it remains to be determined which vertices from $V \setminus D$ are in the solution set. To do so let us introduce some notation. Let $D^* := D \setminus D' = \{d_1, d_2, \ldots, d_{|D^*|}\}$ be the remaining vertices from D. Furthermore, let $P := \{v_1, v_2, \ldots, v_{|P|}\} = V \setminus D$ be the vertices in the disjoint paths such that for each $a, b \in [|P|]$ it holds that if $\{v_a, v_b\} \in E$, then $|a - b| = 1$. Also, let $P_a := \{v_p \in P \mid p \leq a\}$ be the subset of the first $a \in [|P|]$ vertices in the disjoint paths.

We define for each $p \in [|P|]$, for each $m_{d_c} \in \{0, 1\}$ with $d_c \in D^*$, for each $x \in \{\top, \bot\}$, and for each $\ell \in \{0, 1\}$ the table entry $t_p[m_{d_1}, m_{d_2}, \ldots, m_{d_{|D^*|}}, x, \ell]$ as the minimum possible size of a subset $S \subseteq P_p$ such that every vertex $d_c \in D^*$ has $\deg_{G[D^* \cup P_p] - S}(d_c) = m_{d_c}$, the vertex $v_p \in S$, if and only if $x = \top$, and if $x = \bot$, then the vertex v_p is part of a path of length ℓ in $G[P_p \setminus S]$. Note that if $x = \top$, then ℓ does not bear any meaning and can be ignored. If there is no such subset, then we define the table entry as infinity.

We compute the table entries using recursion starting with the base case $p = 1$. For each $m_{d_c} \in \{0, 1\}$ with $d_c \in D^*$, for each $x \in \{\top, \bot\}$, and for each $\ell \in \{0, 1\}$

$$t_1[m_{d_1}, \ldots, m_{d_{|D^*|}}, x, \ell] = \begin{cases} 0 & \text{if } x = \bot,\ \ell = 0 \text{ and } \deg_{G[D^* \cup \{v_1\}]}(d_c) = m_{d_c}\ \forall d_c \in D, \\ 1 & \text{if } x = \top \text{ and } \deg_{G[D^*]}(d_c) = m_{d_c}\ \forall d_c \in D, \\ \infty & \text{else.} \end{cases}$$

For the correctness of this formula observe that $P_1 = \{v_1\}$ holds. If $x = \bot$, then $v_1 \notin S = \emptyset$. Therefore the vertex v_1 must be part of a path of length zero in $G[P_1 - S]$. Hence the table entry is infinity or $\ell = 0$ holds. Furthermore, if $x = \bot$, then $D^* \cup P_1 \setminus S = D^* \cup \{v_1\}$. On the other hand, if $x = \top$, then $v_1 \in S = P_1$ and $D^* \cup P_1 \setminus S = D^*$.

For every $p \geq 2$ we compute the table entries as follows, once the table entries for $p-1$ are known. For each $m_{d_c} \in \{0, 1\}$ with $d_c \in D^*$ let $m'_{d_c} = m_{d_c} - 1$, if $\{d_c, v_p\} \in E$, otherwise $m'_{d_c} = m_{d_c}$, and

$$t_p[m_{d_1}, \ldots, m_{d_{|D^*|}}, \top] = \min_{\substack{x \in \{\top, \bot\} \\ \ell \in \{0,1\}}} \left\{ t_{p-1}[m_{d_1}, \ldots, m_{d_{|D^*|}}, x, \ell] \right\} + 1;$$

$$t_p[m_{d_1}, \ldots, m_{d_{|D^*|}}, \bot, 0] =$$

$$\begin{cases} \infty & \text{if } \exists d_c \in D^* \text{ s.t. } m'_{d_c} = -1, \\ \min\limits_{\substack{x \in \{\top, \bot\} \\ \ell \in \{0,1\}}} \left\{ t_{p-1}[m'_{d_1}, \ldots, m'_{d_{|D^*|}}, x, \ell] \right\} & \text{else if } \{v_p, v_{p-1}\} \notin E, \\ t_{p-1}[m'_{d_1}, \ldots, m'_{d_{|D^*|}}, \top] & \text{else;} \end{cases}$$

$$t_p[m_{d_1},\dots,m_{d_{|D^*|}},\bot,1] = \begin{cases} \infty & \text{if } \exists d_c \in D^* \text{ s.t. } m'_{d_c} = -1,\\ t_{p-1}[m'_{d_1},\dots,m'_{d_{|D^*|}},\bot,0] & \text{else if } \{v_p,v_{p-1}\} \in E,\\ \infty & \text{else.} \end{cases}$$

If $x = \top$, then $P_p \setminus S = P_{p-1} \setminus S$, so for each vertex $d_c \in D^*$ with $\deg_{G[D^* \cup P_p \setminus S]}(d_c) = m_{d_c}$ also $\deg_{G[D^* \cup P_{p-1} \setminus S]}(d_c) = m_{d_c}$ holds. Therefore the minimum subset is exactly one vertex larger than any minimum subset for $p-1$ with $m_{d_1},\dots,m_{d_{|D^*|}}$.

If $x = \bot$, then $v_p \notin S$ and $P_p \setminus S = (P_{p-1} \cup \{v_p\}) \setminus S$. Thus if $\{d_c, v_p\} \in E$, then $\deg_{G[D^* \cup P_p \setminus S]}(d_c) = m_{d_c} = m'_{d_c} + 1 = \deg_{G[D^* \cup P_{p-1} \setminus S]}(d_c) + 1$, and if $\{d_c, v_p\} \notin E$, then $\deg_{G[D^* \cup P_p \setminus S]}(d_c) = m_{d_c} = m'_{d_c} = \deg_{G[D^* \cup P_{p-1} \setminus S]}(d_c)$. Therefore the table entry must be infinity, when $m'_{d_c} = -1$ for some vertex $d_c \in D^*$. If $\ell = 0$, then the vertex v_p is part of a length-zero path in $G[P_p \setminus S]$. Therefore the minimum subset is of same size as any minimum subset for $p-1$ with $\{v_p, v_{p-1}\} \notin E$ or $v_{p-1} \in S$. If $\ell = 1$, then v_p is part of a length-one path in $G[P_p \setminus S]$. Thus both $\{v_p, v_{p-1}\} \in E$ and $v_{p-1} \notin S$ must hold and the minimum subset is of same size as this minimum subset for $p-1$. Hence we can conclude that the formulas are correct.

The size of a minimum solution set resolving every $K_{1,2}$ in G is the minimum of all table entries for the vertex $v_{|P|}$. Output *yes*, if it is at most k, otherwise output *no*.

The algorithm computes $\mathcal{O}(2^{|D^*|} \cdot n \cdot 3)$ table entries. To compute the entries for the base case, check the containment in D^* of all vertices in $\mathcal{O}(|D^*| \cdot n)$ time, and both count the number of neighbors in D^* for every vertex $d \in D^*$ and check if the edge $\{d, p_1\}$ exists in $\mathcal{O}(|D^*| \cdot m)$. Use this information to compute all $3 \cdot 2^{|D^*|}$ entries in $\mathcal{O}(2^{|D^*|} \cdot |D^*|)$ time. Thus the entries for the base case can be computed in $\mathcal{O}(2^{|D^*|} \cdot |D^*| + |D^*| \cdot (n+m))$ time in total. The entries for the other cases can be computed in $\mathcal{O}(2^{|D^*|} \cdot |D^*| \cdot n + |D^*| \cdot m)$ time by checking if the edge $\{v_p, v_{p-1}\}$ exists and testing for edges between D^* and $v_p \in P$ in $\mathcal{O}(|D^*| \cdot m)$ time. Finding the minimum table entry with $p = |P|$ and comparing it with k requires $\mathcal{O}(2^{|D^*|})$ time. As the guessing of $D' \subseteq D$ requires $2^{|D|}$ runs of the rest of the algorithm, the algorithm runs in total in $\mathcal{O}(2^{2|D|} \cdot |D| \cdot (n+m))$ time. □

By combining the running time of all three cases, we arrive at the following result for the general case.

Corollary 3.25. *Let $D \subseteq V$ be a set such that every connected component in $G - D$ is a path. Then* Biclique-Free Vertex Deletion *can be solved in $\mathcal{O}(2^{|D|+|D|^2} \cdot |D|^2 \cdot n^2)$ time.*

Chapter 4

Biclique-Free Edge Deletion

In this chapter, we will analyze the parameterized complexity of the BICLIQUE-FREE EDGE DELETION problem. It is known to be NP-complete in the special case $i = j = 2$, because in this case the forbidden subgraph is the $K_{2,2} = C_4$ and the problem of deleting edges such that the resulting graph does not contain cycles of specified length ℓ, for any $\ell \geq 3$, is NP-complete [Yan78]. The special case $k = 0$ is W[1]-hard with respect to the natural parameter j [Lin18].

We study BICLIQUE-FREE EDGE DELETION with respect to the same parameters for which we studied the parameterized complexity of BICLIQUE-FREE VERTEX DELETION in Chapter 3. This enables us to find similarities and differences in the complexity between the two variants of BICLIQUE-FREE DELETION. It turns out that the results are in fact similar. We show in Section 4.1 that BICLIQUE-FREE EDGE DELETION is FPT when parameterized by the vertex cover number and in Section 4.2 that it admits a linear-size problem kernel when parameterized by the feedback edge set number. We prove in Section 4.3 that it is unlikely to find FPT algorithms for two parameters upper-bounded by the vertex cover number or the feedback edge set number, that is, we show that BICLIQUE-FREE EDGE DELETION is W[1]-hard when parameterized by the feedback vertex set number or the treedepth. Again, this results resembles what is known for BICLIQUE-FREE VERTEX DELETION, but the parameterized complexity of BICLIQUE-FREE EDGE DELETION with parameter distance to disjoint paths remains open.

Let us consider the special case $i = 1$. BICLIQUE-FREE VERTEX DELETION is NP-hard even in this case (this is BOUNDED-DEGREE DELETION), but in contrast BICLIQUE-FREE EDGE DELETION becomes polynomial-time solvable. We show this using a reduction to the polynomial-time solvable DEGREE-CONSTRAINT SUBGRAPH problem [Gab83], also known as $(g, f)-$FACTOR [KFN21].

DEGREE-CONSTRAINT SUBGRAPH
Input: An undirected graph $G = (V, E)$, degree bounds $\ell_v, u_v \in \mathbb{N}$ for each $v \in V$, and a size lower bound $m' \in \mathbb{N}$ of the subgraph.
Question: Does G contain a subgraph $G' = (V, E')$ such that $|E'| \geq m'$ and $\ell_v \leq \deg_{G'}(v) \leq u_v$ for all $v \in V$?

Lemma 4.1. BICLIQUE-FREE EDGE DELETION *with $i = 1$ can be solved in $\mathcal{O}(m \cdot n \cdot j)$ time.*

Proof. Let $\mathcal{I} = (G = (V, E), 1, j, k)$ be an instance of BICLIQUE-FREE EDGE DELETION with $i = 1$. Then an equivalent instance $\mathcal{I}' = (G', \ell, u, m')$ of DEGREE-CONSTRAINT SUBGRAPH can be computed in $\mathcal{O}(n + m)$ time as follows: Set $G' := G$, $\ell_v := 0$ and $u_v := j - 1$ for all $v \in V$ and $m' := m - k$.

If $\mathcal{I}$ is a yes-instance, then there exists a solution set $S \subseteq E$ of size $|S| \leq k$ such that $G - S$ is $K_{1,j}$-free. The subgraph $G - S$ is a solution for $\mathcal{I}'$, because it contains $m - |S| \geq m - k$ edges and has maximum degree $j - 1$, as a $K_{1,j}$-free graph has maximum degree $j - 1$. Hence $\mathcal{I}'$ is a yes-instance.

Conversely, if $\mathcal{I}'$ is a yes-instance, then $G = (V, E)$ contains a subgraph $G' = (V, E') \subseteq G$ with maximum degree at most $j - 1$ such that $|E'| \geq m - k$. Then $E \setminus E'$ is of size $m - |E'| \leq k$ and $G - (E \setminus E')$ is $K_{1,j}$-free. Hence $\mathcal{I}$ is a yes-instance.

The DEGREE-CONSTRAINT SUBGRAPH problem can be solved in $\mathcal{O}(m \cdot \sum_{v \in V} u_v)$ time [Gab83]. Since $\sum_{v \in V} u_v = n \cdot j - n$, the total running time for first computing and then solving $\mathcal{I}'$ is $\mathcal{O}(m \cdot n \cdot j)$. □

4.1 Parameterization by Vertex Cover Number

In this section, we show that BICLIQUE-FREE EDGE DELETION is FPT when parameterized by the vertex cover number by providing an algorithm using integer linear programming (ILP). We assume that a minimum vertex cover S is given along with the problem input. Note that one can compute a minimum vertex cover for any graph in FPT time with respect to the size of the minimum vertex cover. Let $I := V \setminus S$ be the independent set corresponding to the minimum vertex cover S.

Designing an algorithm that guesses which edges to delete analogous to the algorithm for BICLIQUE-FREE VERTEX DELETION parameterized by the vertex cover number (see Section 3.1) turned out to be difficult, as it would have been necessary to upper-bound the budget k by some function of $|S|$. In our algorithm for BICLIQUE-FREE VERTEX DELETION we argue that deleting all vertices in S leaves a biclique-free independent set and therefore we can assume $k \leq |S|$, but deleting all edges with both endpoints in S does not remove any bicliques that are contained between S and I. The number of edges between S and I is at most $|S| \cdot (n - |S|)$ and therefore not bounded by any function of $|S|$ alone. Without an upper bound for k we cannot upper-bound the size of our guessed edge subset to delete from the graph.

We resort to the following ILP-based approach. We first guess a subset $E' \subseteq E(G[S])$ of edges with both endpoints in S, delete it and decrease the budget k by $|E'|$. After this step it remains to be determined if at most k edges with one endpoint in S and the other endpoint in I can be deleted to make G biclique-free. We solve this restricted variant of BICLIQUE-FREE EDGE DELETION using an equivalent ILP formulation with the number of variables bounded only by a function of $|S|$.

Our goal is to create an ILP formulation such that every variable assignment corresponds to a subset $E^* \subseteq \{\{u, v\} \in E \mid u \in S, v \in I\}$ of edges to delete from G to obtain some $G' := G - E^*$. We then add additional constraints to ensure that E^* contains at most k edges and that G' is biclique-free.

We introduce a variable $x_{T,T'} \in \mathbb{N}$ for each neighborhood type $T \subseteq S$ and for each $T' \subseteq T$. Herein, $x_{T,T'}$ is intended to denote the number of vertices from I whose neighborhood is T in G and will become $T \setminus T'$ in G' due to edge deletions. Note that we do not distinguish between individual vertices in I with the same neighborhood type in G (or G').

Since the number of vertices from I with some neighborhood T in G is given as input and cannot be changed, we add the following constraint to ensure that each solution for the ILP instance correctly represents this initial state of G.

$$\sum_{T' \subseteq T} x_{T,T'} = |\{u \in I \mid N(u) = T\}| \qquad \forall T \subseteq S \tag{4.1}$$

For each neighborhood $T \subseteq S$ the number of vertices in I with neighborhood T is equal to $\sum_{T' \subseteq T} x_{T,T'}$.

If a vertex $v \in I$ with neighborhood T in G is supposed to have a neighborhood $T \setminus T'$ in G', we have to delete T' edges. In fact we have to delete exactly the edges in $\{\{u, v\} \in E(G) \mid u \in T'\}$. In total we want to delete at most k edges to obtain G' from G, which we express in the following constraint.

$$\sum_{T \subseteq S} \sum_{T' \subseteq T} (|T'| \cdot x_{T,T'}) \leq k \tag{4.2}$$

This constraint allows us to omit an objective function (e.g. minimize the total number of edges deleted) and therefore the ILP formulation leads to an instance of the ILP Feasibility problem instead of the more extensive ILP Optimization problem.

Now we add further constraints to guarantee that G' is biclique-free. We utilize the fact that at least one side of every biclique contained in G is a subset of S. If the size of the common neighborhood of every subset $S' \subseteq S$ with size i, respectively j, is less than j, respectively i, then G' is biclique-free. We formalize any required decrement in the size of common neighborhoods using the following two constraints.

$$\sum_{S' \subseteq T \subseteq S} \sum_{\substack{T' \subseteq T \\ \text{with } T' \cap S' \neq \emptyset}} x_{T,T'} \geq \left| \bigcap_{v \in S'} N(v) \right| - j + 1 \qquad \forall S' \subseteq S \text{ with } |S'| = i \tag{4.3}$$

$$\sum_{S' \subseteq T \subseteq S} \sum_{\substack{T' \subseteq T \\ \text{with } T' \cap S' \neq \emptyset}} x_{T,T'} \geq \left| \bigcap_{v \in S'} N(v) \right| - i + 1 \qquad \forall S' \subseteq S \text{ with } |S'| = j \tag{4.4}$$

This concludes our ILP formulation for solving the restricted variant of Biclique-Free Edge Deletion. We now prove that the ILP formulation and the restricted variant of Biclique-Free Edge Deletion are equivalent.

Lemma 4.2. *Let S be a vertex cover for $G = (V, E)$ and let $I = V \setminus S$ be the corresponding independent set. Let $\mathcal{I} = (G = (V, E), i, j, k)$ be an instance of* Biclique-Free Edge Deletion *with the restriction that only edges from the set $\{\{u, v\} \in E \mid u \in S, v \in I\}$ may be deleted. Let $\mathcal{I}'$ be an* ILP Feasibility *instance consisting of Constraints (4.1) to (4.4) and the matching variables. Then $\mathcal{I}$ is a yes-instance, if and only if $\mathcal{I}'$ is a yes-instance.*

Proof. We first prove that if $\mathcal{I}$ is a yes-instance, then $\mathcal{I}'$ is a yes-instance. So let $E'' \subseteq \{\{u,v\} \in E \mid u \in S, v \in I\}$ be a solution set for $\mathcal{I}$ of size $|E''| \leq k$. Let $G' := G - E''$ be the graph that is obtained by deleting E'' from G. For each $T \subseteq S$ and for each $T' \subseteq T$ set the variable $x_{T,T'} = 0$ initially. Then, for each vertex $v \in I$, increase $x_{N_G(v),N_G(v)\setminus N_{G'}(v)}$ by one. This satisfies Constraint (4.1), as $N_G(v) \setminus N_{G'}(v) \subseteq N_G(v) \subseteq S$ holds. Also Constraint (4.2) is fulfilled, because G' is obtained from G by deleting $|E''| \leq k$ edges and E'' contains exactly $|N_G(v) \setminus N_{G'}(v)|$ edges for each vertex $v \in I$.

For each $S' \subseteq S$ with $|S'| = i$ it holds that $\left|\bigcap_{v\in S'} N_{G'}(v)\right| < j$, as G' is $K_{i,j}$-free. If $\left|\bigcap_{v\in S'} N_G(v)\right| \geq j$, then at least $\left|\bigcap_{v\in S'} N_G(v)\right| - j + 1$ edges between vertices in S' and the common neighborhood of S' must be deleted to make G' $K_{i,j}$-free and are therefore contained in E''. Hence there exist at least $\left|\bigcap_{v\in S'} N_G(v)\right| - j + 1$ vertices $v \in I$ with a neighborhood $N_G(v) \supseteq S'$ in G, but $|N_G(v) \cap S'| > |N_{G'}(v) \cap S'|$. Consequently, Constraint (4.3) and analogously also Constraint (4.4) must hold. Hence $\mathcal{I}'$ is a yes-instance.

Now we prove that if $\mathcal{I}'$ is a yes-instance, then $\mathcal{I}$ is a yes-instance. So let the variables from $\mathcal{I}'$ be assigned to values, such that all constraints are satisfied. Construct a solution set $E'' \subseteq \{\{u,v\} \in E \mid u \in S, v \in I\}$ for $\mathcal{I}$ as follows to obtain the graph $G' := G - E''$. For each $T \subseteq S$ and for each $T' \subseteq T$ choose a subset of $x_{T,T'}$ many vertices $X \subseteq \{u \in I \mid N(u) = T\}$ that were not chosen before and add all edges between the vertices in T' and X to E''. The edges exist, because $T' \subseteq T$ and $N(u) = T$ for all $u \in X$. There are sufficiently many vertices in $\{u \in I \mid N(u) = T\}$ that were not chosen before, because Constraint (4.1) ensures that $\sum_{T'\subseteq T} x_{T,T'} = |\{u \in I \mid N(u) = T\}|$ holds. We add $|T'| \cdot x_{T,T'}$ many edges to E'' for each $T \subseteq S$ and for each $T' \subseteq T$. Since Constraint (4.2) is fulfilled, this leads to at most k edges in total and $|E''| \leq k$ holds.

We continue by showing that G' is $K_{i,j}$-free. At least one side of every $K_{i,j}$ contained in G is a subset of S since vertices in an independent set are by definition not neighbors. If the size of the common neighborhood of every subset $S' \subseteq S$ with size i, respectively j, is less than j, respectively i, then G' is $K_{i,j}$-free. Hence we have to prove that $\left|\bigcap_{v\in S'} N_{G'}(v)\right| < j$ for all $S' \subseteq S$ with $|S'| = i$ and $\left|\bigcap_{v\in S'} N_{G'}(v)\right| < i$ for all $S' \subseteq S$ with $|S'| = j$. Assume towards contradiction that there exists a subset $S' \subseteq S$ with $|S'| = i$ such that $\left|\bigcap_{v\in S'} N_{G'}(v)\right| \geq j$ or a subset $S'' \subseteq S$ with $|S''| = j$ such that $\left|\bigcap_{v\in S''} N_{G'}(v)\right| \geq i$. We continue by covering only the first case, as the other case is analogous. Since Constraint (4.3) is fulfilled, at least $\left|\bigcap_{v\in S'} N_G(v)\right| - j + 1 \geq 1$ vertices contained in I and the common neighborhood of S' in G, that is, they have the neighborhood T in G for some $S' \subseteq T \subseteq S$, are not in the common neighborhood of S' in G', that is, they have the neighborhood $T \setminus T'$ in G' for some $T' \subseteq T$ with $T' \cap S' \neq \emptyset$. This yields the contradiction $j \leq \left|\bigcap_{v\in S'} N_{G'}(v)\right| \leq \left|\bigcap_{v\in S'} N_G(v)\right| - \left|\bigcap_{v\in S'} N_G(v)\right| + j - 1 = j - 1$. Hence $\mathcal{I}$ is a yes-instance. □

We now provide an FPT algorithm for Biclique-Free Edge Deletion parameterized by vertex cover number as the main result of this section. The algorithm uses the ILP Feasibility formulation to solve the restricted variant of Biclique-Free Edge Deletion parameterized by the vertex cover number in FPT time.

Theorem 4.3. *Let S be a vertex cover of G. Then* Biclique-Free Edge Deletion *can be solved in $\mathcal{O}^*(2^{2^{\mathcal{O}(|S|)}})$ time.*

Proof. Guess a subset $E' \subseteq \{\{u, v\} \in E \mid u, v \in S\}$ of edges with both endpoints in S, delete it from G and decrease the budget k by $|E'|$. Then construct an instance of ILP Feasibility as described above consisting of Constraints (4.1) to (4.4) and the matching variables. Solve it to obtain an answer for the remaining instance of Biclique-Free Edge Deletion now restricted to the deletion of edges with one endpoint in S and the other endpoint in I.

The algorithm is correct, because the ILP Feasibility instance is equivalent to the instance of the restricted variant of Biclique-Free Edge Deletion we obtained after we deleted the subset E' (see Lemma 4.2).

The algorithm tries all possible subsets of edges with both endpoints in S resulting in $2^{\mathcal{O}(|S|^2)}$ different ILP instances to construct and solve. Adding Constraints (4.3) and (4.4) to an instance requires $\mathcal{O}(2^{3|S|} \cdot |S| \cdot n)$ time. The remaining constraints can be added quicker. Hence in total each ILP instance can be constructed in $\mathcal{O}(2^{3|S|} \cdot |S| \cdot n)$ time. It uses $2^{\mathcal{O}(|S|)}$ many variables and can therefore be solved in $\mathcal{O}^*(2^{2^{\mathcal{O}(|S|)}})$ time [FT87; Kan87; Len83]. Combining every step sums up in total to $\mathcal{O}^*(2^{2^{\mathcal{O}(|S|)}})$ time. □

4.2 Parameterization by Feedback Edge Set Number

We will now show that Biclique-Free Edge Deletion admits a linear-size problem kernel when parameterized by the feedback edge set number. This is another parameter for which we also studied Biclique-Free Vertex Deletion. In fact we use the same idea we used to achieve the linear-size problem kernel for the case $i \geq 2$ of Biclique-Free Vertex Deletion (see Section 3.2.1) with only minor adjustments, but since Biclique-Free Edge Deletion with $i = 1$ is solvable in polynomial time this approach is not limited to $i \geq 2$. Recall that $V_2(G) := \{v \in V(G) \mid \deg(v) = 2\}$ for a graph G.

Theorem 4.4. Biclique-Free Edge Deletion *admits a linear-size problem kernel computable in $\mathcal{O}(n \cdot m \cdot j)$ time when parameterized by the feedback edge set number.*

Proof. If $i = 1$, then solve the instance in $\mathcal{O}(n \cdot m \cdot j)$ time (see Lemma 4.1) and output a trivial yes- or no-instance accordingly. For the remaining case $i \geq 2$ we can reduce Biclique-Free Edge Deletion to the following linear-size problem kernel that is very similar to the problem kernel we provided for Biclique-Free Vertex Deletion (see Theorem 3.5).

Assume $i, j \geq 2$ and let $F \subseteq E$ be a feedback edge set for the input graph G computable in $\mathcal{O}(n + m)$ time using depth-first search. If the budget $k \geq |F|$, then we can output a trivial yes-instance and are done. This is correct, because deleting all feedback edges is covered by the budget and results in a forest, which does not contain any $K_{i,j}$ with $i, j \geq 2$ as a subgraph. We can therefore assume that $k < |F|$.

Apply Reduction Rule 3.2.1 exhaustively in $\mathcal{O}(n + m)$ time (see Lemma 3.2) to remove all vertices with degree at most one. By Lemma 3.3 the remaining graph G

now contains at most $2|F|$ vertices of degree at least three. Also delete all vertices from G that are contained in a connected component of $G[V_2(G)]$ with more than four vertices, because they cannot be contained in any biclique of sufficient size. This is also possible in $\mathcal{O}(n+m)$ time using depth-first search. Then every connected component in $G[V_2(G)]$ contains at most four vertices. Using Lemma 3.4 we can conclude that there exist at most $12|F|-4$ vertices of degree two in G. By combining both upper bounds we achieve $n \leq 14|F|-4$ in total. Using $m \leq n+|F|-1$ yields $m \leq 15|F|-5$. If $j \geq n$, then output a trivial yes-instance, as the graph cannot contain any $K_{i,j}$ in this case. Now it holds that $k < |F|$, $i \leq j < n \leq 14|F|-4$, and $m \leq 15|F|-5$. Hence the size of the remaining instance is bounded linearly in $|F|$. □

4.3 Parameterization by Feedback Vertex Set Number

In this section, we show that Biclique-Free Edge Deletion parameterized by the feedback vertex set number is W[1]-hard. As a corollary, we also show that it is W[1]-hard for the parameter treedepth. Thereby we provide a result analogous to the known W[1]-hardness of Bounded-Degree Deletion [GKO21], and give evidence that the parameter feedback edge set number, for which we achieved fixed-parameter tractability earlier, is most likely the smallest established "tree-like" parameter for which we can hope to prove fixed-parameter tractability.

We provide a parameterized reduction from the same problem and based on the same underlying idea that Ganian, Klute, and Ordyniak [GKO21, Theorem 2] used for their parameterized reduction to show that Bounded-Degree Deletion parameterized by the feedback vertex set number or the treedepth is W[1]-hard. Nonetheless, the actual construction of an equivalent instance differs fundamentally, especially since Biclique-Free Edge Deletion works with edge deletions, but Bounded-Degree Deletion works with vertex deletions.

Let us introduce some notation. For a vector $b \in \mathbb{N}^k$ with $k \in \mathbb{N}$ rows we denote the value in the x'th row of b by $b[x]$. We write $a \geq b$ for two vectors $a, b \in \mathbb{N}^k$ with $k \in \mathbb{N}$ rows, if $a[x] \geq b[x]$ for each row $x \in [k]$. Furthermore, for a given vector $s \in \mathbb{N}^k$ with $k \in \mathbb{N}$ rows let $\max(s) := \max\{s[x] \mid x \in [k]\}$ be the largest value in s.

We show the W[1]-hardness using a parameterized reduction from the following problem.

Multidimensional Relaxed Subset Sum (MRSS) [GKO21]

Input: The number of rows $k \in \mathbb{N}$, a set $S = \{s_1, \dots, s_n\}$ of vectors with $s_i \in \mathbb{N}^k$ for each $i \in [n]$, a target vector $t \in \mathbb{N}^k$ and the budget $k' \in \mathbb{N}$.

Question: Is there a subset $S' \subseteq S$ with $|S'| \leq k'$ such that $\sum_{s \in S'} s \geq t$?

It is known that MRSS parameterized by $(k+k')$ is W[1]-hard.

Lemma 4.5 ([GKO21, Lemma 4]). Multidimensional Relaxed Subset Sum *parameterized by* $(k+k')$ *is* W[1]*-hard even if all numbers in the input are given in unary.*

Given an instance of MRSS we will now present the construction of an equivalent instance of Biclique-Free Edge Deletion, then we will prove that we can assume that solutions for the constructed instances have some property and using this observation prove the correctness of the overall parameterized reduction.

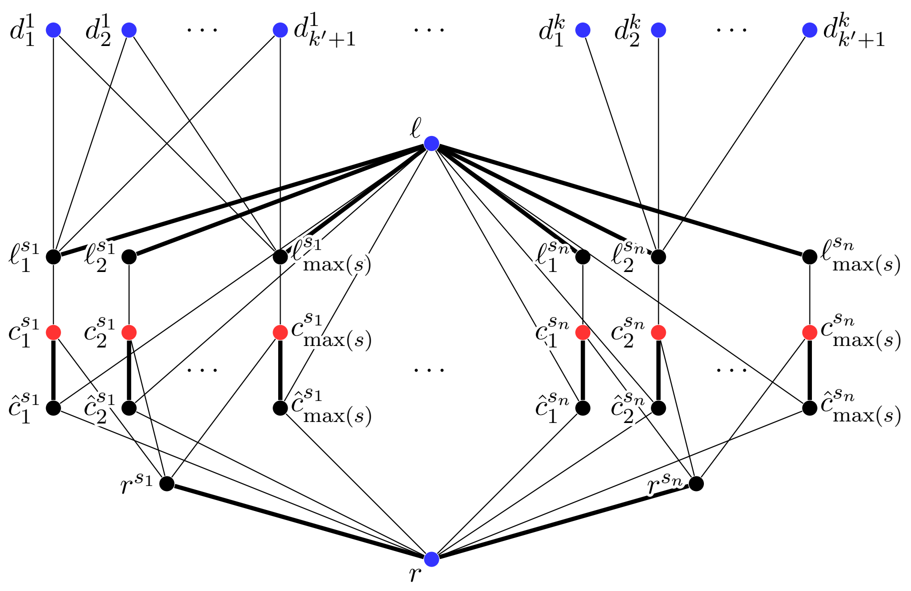


Figure 4.1: Example of a graph constructed by Construction 4.6 in the parameterized reduction from MRSS to Biclique-Free Edge Deletion parameterized by the feedback vertex set number. The blue vertices are feedback vertices. Every red vertex with either the vertex r or the vertex ℓ forms one side of a $K_{2,j}$. Also for each $x \in [k], y \in [k'+1]$ the vertex d_y^x with the vertex ℓ forms one side of a $K_{2,j}$. If there is a solution for the constructed instance, then there is a solution only containing a subset of the bold edges.

Construction 4.6. Let $\mathcal{I} = (k, S, t, k')$ be an instance of MRSS. We construct an equivalent instance $\mathcal{I}' = (G = (V, E), 2, j, k'')$ of Biclique-Free Edge Deletion as follows. See Figure 4.1 for an example of a constructed graph.

Add two feedback vertices r, ℓ to V. For each $s \in S$ add the gadget G^s to G consisting of

- the vertices $\ell_1^s, \ell_2^s, \dots, \ell_{\max(s)}^s$, each with an edge to ℓ,
- the vertex r^s with an edge to r,
- the vertices $c_y^s, \hat{c}_y^s$ and the edge $\{c_y^s, \hat{c}_y^s\}$ for each $y \in [\max(s)]$,
- the edges $\{c_1^s, \ell_1^s\}, \{c_2^s, \ell_2^s\}, \dots, \{c_{\max(s)}^s, \ell_{\max(s)}^s\}$,
- the edges $\{c_1^s, r^s\}, \{c_2^s, r^s\}, \dots, \{c_{\max(s)}^s, r^s\}$,
- the edges $\{\hat{c}_1^s, \ell\}, \{\hat{c}_2^s, \ell\}, \dots, \{\hat{c}_{\max(s)}^s, \ell\}$ and
- the edges $\{\hat{c}_1^s, r\}, \{\hat{c}_2^s, r\}, \dots, \{\hat{c}_{\max(s)}^s, r\}$.

Set the budget $k'' := k' + \sum_{s \in S} \max(s)$. For each row $x \in [k]$ add $(k'+1)$ feedback vertices $d_1^x, d_2^x, \dots, d_{k'+1}^x$ to V. Now add edges such that for each row $x \in [k]$, for each $y \in [k'+1]$ and for each $s \in S$ the vertex d_y^x has exactly $s[x]$ neighbors among the vertices $\ell_1^s, \ell_2^s, \dots, \ell_{\max(s)}^s$. Set $j := 2 + \max\{\sum_{s \in S} s[x] \mid x \in [k]\}$ such that G currently contains no $K_{2,j'}$ as a subgraph for all $j' > j$. For each $s \in S$ and for each $y \in [\max(s)]$ add new additional vertices to the common neighborhood of c^s_y and ℓ such that the size of the common neighborhood becomes j. More precisely, add a new vertex to G with edges to c^s_y and ℓ repeatedly until $\left|N(c^s_y) \cap N(\ell)\right| = j$ holds. Analogously, add new additional vertices to the common neighborhood of c^s_y and r to increase the size of the common neighborhood to j. For each row $x \in [k]$ and for each $y \in [k'+1]$ add new additional vertices to the common neighborhood of d^x_y and ℓ such that the size of the common neighborhood becomes $j + t[x] - 1$.

This finishes the construction of I'. Note that for each $y \in [\max(s)]$ the vertices c^s_y and ℓ have a common neighborhood of size j, which includes the vertices $\hat{c}^s_y$ and ℓ^s_y. Also the vertices c^s_y and r have a common neighborhood of size j. It includes the vertices $\hat{c}^s_y$ and r^s. Hence G^s contains $(2 \cdot \max(s))$ different bicliques $K_{i,j}$ that need to be resolved. There are many different possibilities to do so, but the following two options are the most relevant. In fact we will later show that we can assume that a solution only uses these two options to resolve all bicliques in the whole graph. The bicliques in G^s can be resolved by deleting the edges $\{\{c^s_1, \hat{c}^s_1\}, \{c^s_2, \hat{c}^s_2\}, \ldots, \{c^s_{\max(s)}, \hat{c}^s_{\max(s)}\}\} \subseteq E(G^s)$ or alternatively by deleting both the edge $\{r^s, r\} \in E(G^s)$ and the edges between the vertices $\ell^s_1, \ell^s_2, \ldots, \ell^s_{\max(s)}$ and ℓ. The first option requires $\max(s)$ edge deletions and is the optimum for each individual gadget. Choosing the second option requires one more edge deletion, but may help to resolve other bicliques intersecting G^s that will be discussed shortly and corresponds to picking the vector s in the solution set S' of $\mathcal{I}$. As the budget is $k'' = k' + \sum_{s \in S} \max(s)$, we can pick the second option up to k' times, therefore limiting the size of the corresponding solution set of $\mathcal{I}$.

Note that for each row $x \in [k]$ and for each $y \in [k'+1]$ the vertices d^x_y and ℓ have a common neighborhood of size $j + t[x] - 1$. Therefore d^x_y and ℓ are part of a biclique $K_{2,j+t[x]-1}$ contained in G as a subgraph that needs to be resolved by deleting at least $t[x]$ edges. Picking the second of the discussed two options to resolve the bicliques contained in the gadget G^s for all $s \in S' \subseteq S$ deletes $\sum_{s \in S'} s[x]$ of those edges. Hence the resulting graph is biclique-free, if and only if $\sum_{s \in S'} s[x] \geq t[x]$ holds for each row $x \in [k]$.

We will now prove that we can indeed assume that a solution for I' contains exactly one of two fixed subsets of edges from each gadget.

Observation 4.7. *If $\mathcal{I}'$ is a yes-instance, then there is a solution $E'' \subseteq E$ such that for each $s \in S$ we have either*

$$E'' \cap E(G^s) = \{\{c^s_1, \hat{c}^s_1\}, \{c^s_2, \hat{c}^s_2\}, \ldots, \{c^s_{\max(s)}, \hat{c}^s_{\max(s)}\}\} \text{ or} \tag{4.5}$$

$$E'' \cap E(G^s) = \{\{\ell^s_1, \ell\}, \{\ell^s_2, \ell\}, \ldots, \{\ell^s_{\max(s)}, \ell\}, \{r^s, r\}\}. \tag{4.6}$$

Proof. Let $E' \subseteq E$ be a solution for $\mathcal{I}'$ and let $s \in S$. If $|E' \cap E(G^s)| \leq \max(s)$, then Equation (4.5) must hold, because this is the only optimal solution set for resolving all bicliques in G^s and all other solutions are larger than $\max(s)$. If $|E' \cap E(G^s)| > \max(s)$, then $E'' = (E' \setminus E(G^s)) \cup \{\{r^s, r\}\} \cup \{\{\ell^s_y, \ell\} \mid y \in [\max(s)]\}$ is also a solution for $\mathcal{I}'$, because $|E''| = |E'| - |E' \cap E(G^s)| + \max(s) + 1 \leq |E'|$ and E'' resolves all bicliques in G^s and decreases the size of the common neighborhood of d^x_y and ℓ for each row $x \in [k]$ and for each $y \in [k'+1]$ at least equally. $\square$

We will now show that Construction 4.6 produces a parameterized reduction and allows us to conclude that Biclique-Free Edge Deletion parameterized by the feedback vertex set number is W[1]-hard.

Theorem 4.8. Biclique-Free Edge Deletion *parameterized by the feedback vertex set number is* W[1]*-hard.*

Proof. Given an instance $\mathcal{I}$ of MRSS parameterized by $(k+k')$ construct an equivalent instance $\mathcal{I}'$ of Biclique-Free Edge Deletion parameterized by the feedback vertex set number according to Construction 4.6.

The budget k'' and the size j can be computed in $\mathcal{O}(n \cdot k)$ time, as there are n vectors with k rows in S. Adding the gadgets including the enlarged common neighborhoods requires $\mathcal{O}(n \cdot \|S\|_{\max} \cdot \|S\|_\infty)$ many vertices and edges and therefore also time, where $\|\cdot\|_{\max}$ is the maximum norm and $\|\cdot\|_\infty$ is the maximum row sum norm. Adding the remaining feedback vertices with their incident edges is possible in $\mathcal{O}(k \cdot k' \cdot (\|S\|_\infty + \max(t)))$ time, as there are $\mathcal{O}(k \cdot k')$ of those vertices each with $\mathcal{O}(\|S\|_\infty + \max(t))$ edges. These running times add up to $\mathcal{O}(n \cdot k + n \cdot \|S\|_{\max} \cdot \|S\|_\infty + k \cdot k' \cdot (\|S\|_\infty + \max(t)))$ time for constructing the equivalent instance, which is a polynomial considering that we can assume that the numbers in the input are given in unary (see Lemma 4.5).

Deleting the vertices r, ℓ and $\{d_y^x \mid x \in [k], y \in [k'+1]\}$ from G results in a forest of height three. Hence the parameter $\mathrm{FVN}(G) \leq k \cdot (k'+1) + 2$ is bounded by the old parameter $(k+k')$.

We now prove that $\mathcal{I}$ and $\mathcal{I}'$ are equivalent and thereby finish the parameterized reduction. We start by showing that, given the MRSS instance $\mathcal{I}$ is a yes-instance, the modified instance $\mathcal{I}'$ is a yes-instance. So let $S' \subseteq S$ be a solution for $\mathcal{I}$. Then we construct a solution E' for $\mathcal{I}'$ as follows. For each $s \in S'$ add the edges between the vertices $\ell_1^s, \ell_2^s, \ldots, \ell_{\max(s)}^s \in V$ and $\ell \in V$, and the edge $\{r^s, r\} \in E$ to E'. For each $s \in S \setminus S'$ add the edges $\{\{c_1^s, \hat{c}_1^s\}, \{c_2^s, \hat{c}_2^s\}, \ldots, \{c_{\max(s)}^s, \hat{c}_{\max(s)}^s\}\} \subseteq E$ to E'. This resolves all bicliques $K_{2,j}$ in the gadget G^s for all $s \in S$ and decreases the size of the common neighborhood of d_y^x and ℓ by $\sum_{s \in S'} s[x]$ for each row $x \in [k]$ and for each $y \in [k'+1]$. This and the fact that $\sum_{s \in S'} s \geq t$ holds yields that the size of the common neighborhood of d_y^x and ℓ becomes $j + t[x] - 1 - \sum_{s \in S'} s[x] \leq j - 1 < j$, meaning that all bicliques $K_{2,j}$ are resolved in $G - E'$. Since $|S'| \leq k'$, there are at most $k' + \sum_{s \in S} \max(s)$ edges in E'. Hence $\mathcal{I}'$ is a yes-instance.

We now show that the MRSS instance $\mathcal{I}$ is a yes-instance, if $\mathcal{I}'$ is a yes-instance. So let $E' \subseteq E$ be a solution for $\mathcal{I}'$. By Observation 4.7 we may assume that, for each $s \in S$, either Equation (4.5) or Equation (4.6) holds. Then S' constructed as follows is a solution to $\mathcal{I}$. For each $s \in S$, if $\{\{\ell_1^s, \ell\}, \{\ell_2^s, \ell\}, \ldots, \{\ell_{\max(s)}^s, \ell\}, \{r^s, r\}\} \subseteq E'$, then add s to S'. It holds that $|S'| \leq k'$, because every $s \in S$ such that Equation (4.6) holds adds $1 + \max(s)$ edges to E', but $|E'| \leq k' + \sum_{s \in S} \max(s)$. Furthermore, it holds that $\left|E' \setminus \bigcup_{s \in S} E(G^s)\right| \leq k'$, because, for each $s \in S$, the gadget G^s requires at least $\max(s)$ edge deletions to make it biclique-free. Therefore for every row $x \in [k]$ there exists a $y \in [k'+1]$ such that $E' \cap \{\{d_y^x, v\} \mid v \in V\} = \emptyset$. The size of the common neighborhood of d_y^x and ℓ is per construction $j + t[x] - 1$, so E' must contain at least $t[x]$ edges from $\bigcup_{s \in S} E(G^s)$ with one endpoint being ℓ and the other endpoint being a neighbor of d_y^x. Only the gadgets in $\{G^s \mid s \in S'\}$ contain such edges, each gadget G^s exactly $s[x]$ many, so $\sum_{s \in S'} s[x] \geq t[x]$ must hold for every row and therefore also $\sum_{s \in S'} s \geq t$ holds. Hence $\mathcal{I}$ is a yes-instance and the two instances are equivalent.

The problem BICLIQUE-FREE EDGE DELETION parameterized by the feedback vertex set number is W[1]-hard, since there is a parameterized reduction from the W[1]-hard problem MRSS parameterized by $(k + k')$ to it. $\square$

Because deleting the feedback vertices results in a forest of constant height, i.e. three, the treedepth of G is also bounded by $(k + k')$. Hence Theorem 4.8 also proves that BICLIQUE-FREE EDGE DELETION is W[1]-hard when parameterized by treedepth.

Corollary 4.9. BICLIQUE-FREE EDGE DELETION *parameterized by the treedepth is* W[1]*-hard.*

Chapter 5

Conclusion

We studied the parameterized complexity of Biclique-Free Vertex Deletion and Biclique-Free Edge Deletion with respect to the structural graph parameters vertex cover number, distance to disjoint paths, feedback edge set number and feedback vertex set number.

We conclude that the parameterized complexity of Biclique-Free Vertex Deletion and Biclique-Free Edge Deletion is similar in many aspects, e.g. both are FPT for the parameter vertex cover number and admit a linear-size problem kernel with respect to the feedback edge set number. The special case $i = 1$ of Biclique-Free Vertex Deletion, i.e BDD, is W[1]-hard when parameterized by the feedback vertex set number or the treedepth [GKO21], while Biclique-Free Edge Deletion is W[1]-hard for the same parameters in general. However, Biclique-Free Edge Deletion appears to be simpler in some cases, e.g. Biclique-Free Edge Deletion with $i = 1$ can be solved in polynomial time and the linear-size problem kernel for the parameter feedback edge set number does not use a huge case distinction, but also Biclique-Free Vertex Deletion is simpler in some sense, e.g. the FPT algorithm for the parameter vertex cover number does not use ILP Feasibility and we also proved that Biclique-Free Vertex Deletion is FPT when parameterized by the distance to disjoint paths while the complexity for the edge deletion variant remains open for this parameter. Answering the following open question could help comparing the complexity of the two variants of Biclique-Free Deletion: Is Biclique-Free Edge Deletion with respect to the distance to disjoint paths FPT?

Taking a closer look it seems that Biclique-Free Vertex Deletion with $i = 1$, i.e. BDD, is actually significantly more involved than with $i \geq 2$. Our linear-size problem kernel for $i = 1$ and the parameter feedback edge set number requires a huge case distinction, while our linear-size problem kernel for $i \geq 2$ is simpler, although not faster computable. Our FPT algorithm for the parameter distance to disjoint paths handles the cases $i = j = 1$ and $i = 1, j = 2$ separately. Interestingly these cases actually have a slightly better running time bound compared to the case $i \geq 2$ or $j \geq 3$, and BDD is W[1]-hard when parameterized by the feedback vertex set number or the treedepth [GKO21], but the complexity of Biclique-Free Vertex Deletion with $i \geq 2$ regarding these two parameters remains open. Furthermore, BDD is in XP for the parameter treewidth [DJL93], but we could not find out if Biclique-Free

Vertex Deletion with $i \geq 2$ is also in XP for treewidth. So while BDD appears to be significantly more involved than Biclique-Free Vertex Deletion with $i \geq 2$ for some cases, the relationship between BDD and Biclique-Free Vertex Deletion with $i \geq 2$ is mostly open. Future work could analyze this relationship in more depth and answer questions left open: Is Biclique-Free Vertex Deletion with $i \geq 2$ W[1]-hard when parameterized by the feedback vertex set number? Is Biclique-Free Vertex Deletion with $i \geq 2$ in XP when parameterized by the treewidth?

We found two "FPT-borders" regarding structural graph parameters for Bounded-Degree Deletion and Biclique-Free Edge Deletion. Bounded-Degree Deletion is W[1]-hard for the parameter feedback vertex set number [GKO21], that is, the "distance to disjoint trees", but FPT for the parameter distance to disjoint paths, which is, based on the well-known hierarchy of established graph parameters, one of the next larger parameters. Apparently, Bounded-Degree Deletion is not easy to solve when the graph has a small feedback vertex set number, but it can be solved efficiently, if the graph has a small distance to disjoint paths. Biclique-Free Edge Deletion is W[1]-hard when parameterized by the feedback vertex set number, but FPT for the parameter feedback edge set number, which is also one of the next larger established parameters. Also Biclique-Free Edge Deletion appears to be difficult to solve when the graph has a small feedback vertex set number, but it can be solved efficiently, if the graph has a small feedback edge set number.

We would like to point out the following further results, which we believe are particularly interesting. We provide a non-constructive proof for a data reduction rule in our linear-size problem kernel for BDD when parameterized by the feedback edge set number (see Lemma 3.15). By arguing that there must exist a certain equivalent instance of constant size for each case that can occur, we avoid listing all cases and all equivalent instances each with a proof of correctness explicitly. We also showed an FPT algorithm solving the (i, j)-Biclique problem parameterized by treewidth, and showed how this algorithm can be used as a basis for a data reduction rule for Biclique-free Deletion, which deletes all vertices and edges that are not part of at least one biclique.

In addition to the questions raised so far, the following questions regarding the parameterized complexity of Biclique-free Deletion remain open for future research. Are the algorithms and kernels presented in this work optimal from a theoretical point of view and are they effective in practice? Finding even faster algorithms or proving that the presented ones are optimal may lead to new insight regarding the parameterized complexity of Biclique-free Deletion. Restricting a problem to a certain class of graphs can help finding efficient algorithm. What is the parameterized complexity of Biclique-free Deletion when restricted to a certain class of graphs, e.g. bipartite or line graphs?

Literature

[AH19] T. Alzahrani and K. Horadam. *Finding Maximal Bicliques in Bipartite Networks Using Node Similarity*. In: *Applied Network Science* 4.1 (2019), 21:1–21:25 (cit. on p. 10).

[Ale+04] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. L. Hammer, and B. Simeone. *Consensus Algorithms for the Generation of all Maximal Bicliques*. In: *Discrete Applied Mathematics* 145.1 (2004), pp. 11–21 (cit. on p. 10).

[BB+20] F. Bonomo-Braberman, J. R. Nascimento, F. S. Oliveira, U. S. Souza, and J. L. Szwarcfiter. *Linear-Time Algorithms for Eliminating Claws in Graphs*. In: *Proceedings of the 26th International Computing and Combinatorics Conference (COCOON 2020)*. LNCS 12273. Springer, 2020, pp. 14–26 (cit. on p. 11).

[BD11] S. Böcker and P. Damaschke. *Even Faster Parameterized Cluster Deletion and Cluster Editing*. In: *Information Processing Letters* 111.14 (2011), pp. 717–721 (cit. on p. 11).

[Bet+12] N. Betzler, R. Bredereck, R. Niedermeier, and J. Uhlmann. *On Bounded-Degree Vertex Deletion Parameterized by Treewidth*. In: *Discrete Applied Mathematics* 160.1 (2012), pp. 53–60 (cit. on pp. 11, 20, 22).

[Cai96] L. Cai. *Fixed-Parameter Tractability of Graph Modification Problems for Hereditary Properties*. In: *Information Processing Letters* 58.4 (1996), pp. 171–176 (cit. on pp. 10, 11).

[Cou90] B. Courcelle. *The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs*. In: *Information and Computation* 85.1 (1990), pp. 12–75 (cit. on p. 11).

[Cre+20] C. Crespelle, P. G. Drange, F. V. Fomin, and P. A. Golovach. *A Survey of Parameterized Algorithms and the Complexity of Edge Modification*. 2020. arXiv: `2001.06867v2 [cs.DS]` (cit. on p. 10).

[Cyg+14] M. Cygan, D. Marx, M. Pilipczuk, and M. Pilipczuk. *Hitting Forbidden Subgraphs in Graphs of Bounded Treewidth*. In: *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS 2014)*. LNCS 8635. Springer, 2014, pp. 189–200 (cit. on p. 11).

[Cyg+15] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015 (cit. on pp. 15, 16, 40).

[Cyg+17] M. Cygan, M. Pilipczuk, M. Pilipczuk, E. J. van Leeuwen, and M. Wrochna. *Polynomial Kernelization for Removing Induced Claws and Diamonds*. In: *Theory of Computing Systems* 60.4 (2017), pp. 615–636 (cit. on p. 11).

[Die17] R. Diestel. *Graph Theory*. 5th ed. Graduate Texts in Mathematics 173. Springer, 2017 (cit. on p. 13).

[DJL93] A. Dessmark, K. Jansen, and A. Lingas. *The Maximum k-Dependent and f-Dependent Set Problem*. In: *Proceedings of the 4th International Symposium on Algorithms and Computation (ISAAC 1993)*. LNCS 762. Springer, 1993, pp. 88–97 (cit. on pp. 11, 53).

[ELW15] L. Epstein, A. Levin, and G. J. Woeginger. *The (Weighted) Metric Dimension of Graphs: Hard and Easy Cases*. In: *Algorithmica* 72.4 (2015), pp. 1130–1171 (cit. on p. 21).

[Fel+11] M. R. Fellows, J. Guo, H. Moser, and R. Niedermeier. *A Generalization of Nemhauser and Trotter's Local Optimization Theorem*. In: *Journal of Computer and System Sciences* 77.6 (2011), pp. 1141–1158 (cit. on pp. 11, 19).

[FT87] A. Frank and E. Tardos. *An Application of Simultaneous Diophantine Approximation in Combinatorial Optimization*. In: *Combinatorica* 7.1 (1987), pp. 49–65 (cit. on pp. 16, 47).

[Gab83] H. N. Gabow. *An Efficient Reduction Technique for Degree-Constrained Subgraph and Bidirected Network Flow Problems*. In: *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC 1983)*. Association for Computing Machinery, 1983, pp. 448–456 (cit. on pp. 43, 44).

[GKO21] R. Ganian, F. Klute, and S. Ordyniak. *On Structural Parameterizations of the Bounded-Degree Vertex Deletion Problem*. In: *Algorithmica* 83.1 (2021), pp. 297–336 (cit. on pp. 11, 12, 19, 48, 53, 54).

[Kan87] R. Kannan. *Minkowski's Convex Body Theorem and Integer Programming*. In: *Mathematics of Operations Research* 12.3 (1987), pp. 415–440 (cit. on pp. 16, 47).

[KFN21] T. Koana, V. Froese, and R. Niedermeier. *Binary Matrix Completion Under Diameter Constraints*. In: *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*. Leibniz International Proceedings in Informatics (LIPIcs) 187. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021, 47:1–47:14 (cit. on p. 43).

[KK20] L. Kellerhals and T. Koana. *Parameterized Complexity of Geodetic Set*. In: *Proceedings of the 15th International Symposium on Parameterized and Exact Computation (IPEC 2020)*. Leibniz International Proceedings in Informatics (LIPIcs) 180. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 20:1–20:14 (cit. on p. 21).

[Kor21] T. Korhonen. *A Single-Exponential Time 2-Approximation Algorithm for Treewidth*. Accepted at FOCS 2021. 2021. arXiv: `2104.07463 [cs.DS]` (cit. on pp. 16, 17).

[KS21] M. Kučera and O. Suchý. *Minimum Eccentricity Shortest Path Problem with Respect to Structural Parameters*. In: *Proceedings of the 32nd International Workshop on Combinatorial Algorithms (IWOCA 2021)*. LNCS 12757. Springer, 2021, pp. 442–455 (cit. on p. 39).

[Len83] H. W. Lenstra. *Integer Programming with a Fixed Number of Variables*. In: *Mathematics of Operations Research* 8.4 (1983), pp. 538–548 (cit. on pp. 16, 47).

[Lin18] B. Lin. *The Parameterized Complexity of the k-Biclique Problem*. In: *Journal of the ACM* 65.5 (2018), 34:1–34:21 (cit. on pp. 10, 19, 43).

[Lok08] D. Lokshtanov. *Wheel-Free Deletion Is* W[2]*-Hard*. In: *Proceedings of the 3rd International Workshop on Parameterized and Exact Computation (IWPEC 2008)*. LNCS 5018. Springer, 2008, pp. 141–147 (cit. on p. 10).

[LY80] J. M. Lewis and M. Yannakakis. *The Node-Deletion Problem for Hereditary Properties is NP-Complete*. In: *Journal of Computer and System Sciences* 20.2 (1980), pp. 219–230 (cit. on pp. 9–11, 19).

[Mar10] D. Marx. *Chordal Deletion is Fixed-Parameter Tractable*. In: *Algorithmica* 57.4 (2010), pp. 747–768 (cit. on p. 10).

[NM06] J. Nešetřil and P. Ossona de Mendez. *Tree-Depth, Subgraph Coloring and Homomorphism Bounds*. In: *European Journal of Combinatorics* 27.6 (2006), pp. 1022–1041 (cit. on p. 14).

[Pee03] R. Peeters. *The Maximum Edge Biclique Problem is NP-Complete*. In: *Discrete Applied Mathematics* 131.3 (2003), pp. 651–654 (cit. on p. 10).

[PRS12] G. Philip, V. Raman, and S. Sikdar. *Polynomial Kernels for Dominating Set in Graphs of Bounded Degeneracy and Beyond*. In: *ACM Transactions on Algorithms* 9.1 (2012), 11:1–11:23 (cit. on p. 9).

[Sie19] S. Siebertz. *Greedy Domination on Biclique-Free Graphs*. In: *Information Processing Letters* 145.1 (2019), pp. 64–67 (cit. on p. 9).

[SS20] I. Sau and U. S. Souza. *Hitting Forbidden Induced Subgraphs on Bounded Treewidth Graphs*. In: *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*. Leibniz International Proceedings in Informatics (LIPIcs) 170. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 82:1–82:15 (cit. on p. 11).

[TV19] J. A. Telle and Y. Villanger. *FPT Algorithms for Domination in Sparse Graphs and Beyond*. In: *Theoretical Computer Science* 770.1 (2019), pp. 62–68 (cit. on p. 9).

[Yan78] M. Yannakakis. *Node-and Edge-Deletion NP-Complete Problems*. In: *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC 1978)*. Association for Computing Machinery, 1978, pp. 253–264 (cit. on pp. 9, 10, 43).