Vertex Deletion Problems: A Parameterized Point of View

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Alexander Göke

aus

Paderborn

Bonn, November 2020

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Erstgutachter: Prof. Dr. Matthias Mnich
Zweitgutachter: Prof. Dr. Heiko Röglin

Tag der Promotion: 12.03.2021
Erscheinungsjahr: 2022

Acknowledgment

First of all, I would like to thank my supervisor Matthias Mnich for his guidance and support throughout the past years. His enthusiasm and our many interesting discussions made for a great work environment.

I would also like to thank Heiko Röglin for his support. Many thanks especially for hosting me at the institute even at the time when I was employed at TU Hamburg.

I was fortunate to have Kristóf Bérczi, Dániel Marx, Lydia Mirabel Mendoza Cadena, and Matthias Mnich as my coauthors. Their ideas, questions and our many discussions have been extremely valuable.

Finally, I thank my colleagues in Bonn and in Hamburg. I very much enjoyed our discussions, lunch breaks and board game evenings.

Abstract

In this thesis we study various vertex deletion problems. A vertex deletion problem for a graph class $\mathcal{C}$ can be described as follows. Given a graph G and an integer k, delete at most k vertices from G such that the resulting graph belongs to the graph class $\mathcal{C}$.

One of the most prominent vertex deletion problems is Directed Feedback Vertex Set. Here the graph class $\mathcal{C}$ is the class of directed acyclic graphs. We study several generalizations of Directed Feedback Vertex Set and for each of them we either present a fixed-parameter algorithm or a hardness result. A fixed-parameter algorithm with parameter p is an algorithm whose run-time can be expressed as $f(p) \cdot \text{poly}(n)$, where $f(p)$ is some computable function that only depends on the parameter p and $\text{poly}(n)$ is a polynomial in the length n of the input.

Our first result is a fixed-parameter algorithm for Directed Long Cycle Hitting Set. This problem is the $\mathcal{C}_\ell$-Vertex Deletion problem where $\mathcal{C}_\ell$ is the class of graphs which do not contain any cycle of length greater than ℓ. We give a fixed-parameter algorithm for the parameter $k + \ell$. To achieve this we present a new generalization of important separators, as well as a new result on k-representative sets of paths.

Next we consider the problems Bounded Size Strongly Connected Component Vertex Deletion and 1-Out-Regular Vertex Deletion. For these problems, the graph class $\mathcal{C}$ is defined by the structure of the strongly connected components of the graphs in $\mathcal{C}$. In the first problem, every such component has to consist of a bounded number of vertices. In the second problem, every component has to be a simple cycle or a single vertex. We devise fixed-parameter algorithms for both problems.

Eventually, we consider the Negative Directed Feedback Arc Set problem. Here we are given a directed graph with integral arc weights. The task is to delete at most k arcs such that the resulting graph contains no negative cycles. This arc deletion problem generalizes the corresponding vertex deletion problem called Negative Directed Feedback Vertex Set. Moreover, it is related to the Minimum Feasibility Blocker problem from the area of linear programming. We give hardness results and fixed-parameter algorithms for various choices of parameters.

Contents

Chapter 1

Introduction

In this thesis we study various vertex deletion problems. Informally, a vertex deletion problem for a graph class $\mathcal{C}$ can be described as follows. Given a graph G and an integer k, delete at most k vertices from G such that the resulting graph belongs to $\mathcal{C}$.

$\mathcal{C}$-Vertex Deletion	
Instance:	A graph G and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq V(G)$ of size at most k such that $G - S \in \mathcal{C}$ or decide that no such set exists.

One of the most prominent vertex deletion problems is the Feedback Vertex Set problem, where the graph class $\mathcal{C}$ is the class of acyclic graphs. This problem is one of the 21 NP-hard problems on Karp's famous list [Kar72]. It has been studied extensively from the perspective of both exact and approximation algorithms [BBF99, RSS06, CLL+08, FGPR08, CCL15].

For the deletion size k being constant, $\mathcal{C}$-Vertex Deletion reduces to checking whether a given graph belongs to the graph class $\mathcal{C}$. To this end we enumerate all of the polynomially many vertex sets $X \subseteq V(G)$ of size at most k and check for each of them whether $G - X$ belongs to $\mathcal{C}$. Hence, if we can check membership in $\mathcal{C}$ in time $\text{poly}(n)$ for any graph G with $n = |V(G)|$, then we can solve $\mathcal{C}$-Vertex Deletion in time $n^k \cdot \text{poly}(n)$.

For undirected graphs, Robertson and Seymour [RS95, RS04] proved in their graph minor series that the $\mathcal{C}$ membership problem, and hence $\mathcal{C}$-Vertex Deletion, is solvable in polynomial time for a rich set of graph classes $\mathcal{C}$ and constant k. Their result applies to every graph class $\mathcal{C}$ that is closed under taking minors. In fact, Robertson and Seymour proved a stronger statement. The run-time of their algorithm for $\mathcal{C}$-Vertex Deletion on minor-closed graph classes takes the form $f(\mathcal{C}, k) \cdot \text{poly}(n)$, where $f(\mathcal{C}, k)$ is some computable function depending only on the graph class $\mathcal{C}$ and the deletion size k and $\text{poly}(n)$ is a polynomial depending only on $n = |V(G)|$. An algorithm with such a run-time is called a fixed-parameter algorithm with parameters $\mathcal{C}$ and k.

While the notion of fixed-parameter algorithms was not established by the time Robertson and Seymour proved their result, the area of fixed-parameter tractability has now grown into a vibrant field of research featuring a rich set of techniques and

results. The field of parameterized algorithms extends on the field of polynomial-time algorithms. Whereas the classical notion of a polynomial-time algorithm measures the run-time only in terms of the input length, in parameterized complexity one additionally considers certain *parameters*. Such parameters are numerical values associated with the instance, e.g., the deletion size k in $\mathcal{C}$-Vertex Deletion. We call an algorithm a *fixed-parameter algorithm* if its run-time can be expressed as $f(p_1, \ldots, p_t) \cdot \text{poly}(n)$, where $f(p_1, \ldots, p_t)$ is some computable function that only depends on the parameters $p_1, \ldots, p_t$ and $\text{poly}(n)$ is a polynomial in the length n of the input. If a problem admits a fixed-parameter algorithm, we call it *fixed-parameter tractable*. The complexity class of all fixed-parameter tractable problems is called FPT.

In the example of $\mathcal{C}$-Vertex Deletion the most natural parameter is k, i.e., the maximum size of a deletion set. Other popular choices for parameters are structural properties of the graph G like its treewidth, pathwidth or treedepth, or properties of the graph class $\mathcal{C}$.

Similar to the hardness distinction P vs. NP for polynomial-time algorithms, there is the hardness distinction FPT vs. W[1] for fixed-parameter algorithms. Indeed, there is a whole hierarchy $\mathsf{W}[t], t \in \mathbb{Z}_{>0}$ of complexity classes with $\mathsf{W}[i] \subseteq \mathsf{W}[i+1]$ and $\mathsf{FPT} \subseteq \mathsf{W}[1]$. Like $\mathsf{P} \neq \mathsf{NP}$, most researchers expect that $\mathsf{FPT} \neq \mathsf{W}[1]$ and thus W[1]-hard problems are not fixed-parameter tractable. We refer to [DF12, CFK+15] for a comprehensive introduction on the complexity class FPT, the W[1]-hierarchy, and hardness reductions for parameterized problems.

Two important results for vertex deletion problems in the area of fixed-parameter tractability are the existence of fixed-parameter algorithms for Feedback Vertex Set in undirected and directed graphs. The Feedback Vertex Set problem, is the $\mathcal{C}$-Vertex Deletion problem, where $\mathcal{C}$ is the class of all acyclic graphs. That is, one is tasked to delete k vertices from an undirected or directed graph such that no cycles remain in the graph. While the fixed-parameter algorithm for Undirected Feedback Vertex Set follows from the classical result by Robertson and Seymour on minor-closed graphs [RS95, RS04], the first fixed-parameter algorithm for Directed Feedback Vertex Set by Chen, Liu, Lu, O'Sullivan and Razgon [CLL+08] was a major breakthrough. Many results of this thesis build upon techniques introduced in this work.

We will study several generalizations of Directed Feedback Vertex Set. All graph classes we consider will be hereditary, that is, every induced subgraph of a graph in $\mathcal{C}$ also belongs to $\mathcal{C}$. For non-hereditary graph classes, deleting more vertices can be harmful, i.e., for sets $A \subsetneq B$ it can happen that $G - A$ is contained in $\mathcal{C}$ but $G - B$ is not. Our goal is to find fixed-parameter algorithms or hardness results for the vertex deletion problems we consider.

1.1 Results and Outline of this Thesis

In the following we briefly summarize our main results.

Directed Long Cycle Hitting Set

Our first result is a fixed-parameter algorithm for Directed Long Cycle Hitting Set. This problem is the $\mathcal{C}_\ell$-Vertex Deletion problem where $\mathcal{C}_\ell$ is the class of graphs

which do not contain any cycle of length greater than ℓ. We give a fixed-parameter algorithm for the parameters k and ℓ. Unless $\mathsf{P} = \mathsf{NP}$, none of the parameters can be omitted, because for $\ell = 1$, Directed Long Cycle Hitting Set is equivalent to Directed Feedback Vertex Set, and for $k = 0$ and $\ell = n - 1$, Directed Long Cycle Hitting Set is equivalent to Hamiltonian Cycle.

To design our algorithm, we generalize the standard tool of important separators. Important separators were used to tackle many vertex deletion problems, including the Skew Separator problem, which plays a major role in solving Directed Feedback Vertex Set. Our generalization allows to apply important separators to a richer set of $\mathcal{C}$-Vertex Deletion problems.

Another key part of our algorithm is a new result on k-representative sets of paths. Representative sets play an important role in the design of fixed-parameter algorithms [Mon85, Mar09, FLS14, FLPS14, SZ16]. We show how to obtain a small k-representative set of paths for strongly connected graphs in our graph class $\mathcal{C}_\ell$. We refer to Chapter 3 for more details on these results.

Bounded Size Strongly Connected Component Vertex Deletion

The next vertex deletion problem we study is Bounded Size Strongly Connected Component Vertex Deletion. Here the graph class $\mathcal{C}_s$ consists of all graphs whose strongly connected components contain at most s vertices. We give a fixed-parameter algorithm for this problem when parameterized in k and s. The Directed Feedback Vertex Set problem is the special case where $s = 1$. See Chapter 4 for details.

1-Out-Regular Vertex Deletion

A graph is r-out-regular if every vertex has exactly r outgoing arcs. In the r-Out-Regular Vertex Deletion problem our graph class $\mathcal{C}_r$ consists of all graphs for which each strongly connected component C is r_C-out-regular for some $r_C \leq r$. For $r = 0$ this is the Directed Feedback Vertex Set problem. For $r \geq 2$ the graph class is not hereditary anymore. In Chapter 5, we prove that 1-out-regular Vertex Deletion is fixed-parameter tractable when parameterized in k. To obtain this result, we construct a non-standard torso operation fine-tuned to our problem. Torso operations have been introduced by Chitnis, Hajiaghayi, and Marx to prove fixed-parameter tractability of the Directed Subset Feedback Vertex Set problem [CHM13].

Negative Cycle Deletion

Eventually, we consider the Negative Directed Feedback Arc Set problem. Here we are given a directed graph with integral arc weights. The task is to delete at most k arcs of this graph such that the resulting graph contains no negative cycles, or to decide that no such arc set exists. This arc deletion version generalizes the vertex deletion version called Negative Directed Feedback Vertex Set. Negative Directed Feedback Vertex Set in turn generalizes the Directed Feedback Vertex Set problem which is the special case where all arc weights are -1.

Negative Directed Feedback Arc Set is also related to the area of linear programming. A system of linear inequalities $(a_i \cdot x \leq b_i)_{i \in \{1,\dots,m\}}$ with $a_i \in \mathbb{Z}^n$, $b_i \in \mathbb{Z}$ is infeasible if there is no $x \in \mathbb{Z}^n$ that fulfills $a_i \cdot x \leq b_i$ for all $i \in \{1, \dots, m\}$.

A natural question to ask is whether we can make the system feasible by removing at most k inequalities. That is, does there exist an index set $I \subseteq \{1, \ldots, m\}$ of size at most k such that $(a_i \cdot x \leq b_i)_{i \in \{1,\ldots,m\} \setminus I}$ is feasible? We call this problem Minimum Feasibility Blocker. The special case of Minimum Feasibility Blocker where the linear inequality system only consists of difference constraints, i.e., inequalities of the form $x_i - x_j \leq b_{i,j}$, is equivalent to Negative Directed Feedback Arc Set.

We study the Negative Directed Feedback Arc Set problem with different sets of parameters and either prove hardness or fixed-parameter tractability. Negative Directed Feedback Arc Set parameterized only in k is W[1]-hard as we will show in Chapter 6. Therefore we also consider the parameters w_+ and w_- which denote the number of arcs with positive and negative weight respectively. The case $w_+ = 0$ with parameter k is equivalent to Directed Subset Feedback Vertex Set, which is known to be fixed-parameter tractable [CHM13]. Moreover, we consider the parameters treewidth, pathwidth, and treedepth that have recently been studied in the context of linear programming [FLS+18, EHK+19, CCK+20]. For a detailed overview of our results on Negative Directed Feedback Arc Set with different sets of parameters, see Table 6.1 in Section 6.4.

Parts of this thesis are based on joint work with Kristóf Bérczi, Dániel Marx, Lydia Mirabel Mendoza Cadena, and Matthias Mnich. Moreover, some parts of this thesis are based on the following publications.

- Alexander Göke, Dániel Marx, and Matthias Mnich. Parameterized algorithms for generalizations of directed feedback vertex set. In International Conference on Algorithms and Complexity (CIAC), pages 249–261, 2019.
- Alexander Göke, Dániel Marx, and Matthias Mnich. Hitting long directed cycles is fixed-parameter tractable. In International Colloquium on Automata, Languages and Programming (ICALP), 2020.
- Alexander Göke, Dániel Marx, and Matthias Mnich. Hitting long directed cycles is fixed-parameter tractable. arXiv:2003.05267, 2020.
- Alexander Göke, Dániel Marx, and Matthias Mnich. Parameterized algorithms for generalizations of directed feedback vertex set. arXiv:2003.02483, 2020.
- Alexander Göke, Lydia Mirabel Mendoza Cadena, and Matthias Mnich. Resolving infeasibility of linear systems: A parameterized approach. In International Symposium on Parameterized and Exact Computation (IPEC), 2019.

Chapter 2

General techniques

In this chapter we gather common algorithmic techniques for solving vertex deletion problems. Often, they allow a reduction from a general vertex deletion problem to a more restricted variant. These restricted variants have more structure to it and are thus easier to solve. The techniques presented here are used in many algorithms throughout this thesis.

2.1 Iterative Compression

Iterative compression is the most fundamental reduction technique for vertex deletion problems, as many other techniques rely on it. It was introduced by Reed, Smith and Vetta [RSV04] to solve the Odd Cycle Transversal problem. Since then it was applied to many vertex deletion problems. The technique allows us to solve a $\mathcal{C}$-Vertex Deletion problem instance $(G, \mathcal{F})$ by solving at most $n = |V(G)|$ instances of the so-called compression variant of this problem, as long as $\mathcal{C}$ is hereditary.

$\mathcal{C}$-Vertex Deletion Compression	
Instance:	A graph G, a vertex set $T \subseteq V(G)$ and an integer $k \in \mathbb{Z}_{\geq 0}$ such that $G - T \in \mathcal{C}$.
Task:	Find a set $S \subseteq V(G)$ of size at most k such that $G - S \in \mathcal{C}$ or decide that no such set exists.

Let us briefly describe how an algorithm the compression variant can be used to obtain an algorithm for hereditary vertex deletion problems.

Lemma 2.1 (Iterative Compression)**.** *Let $\mathcal{C}$ be a hereditary and non-empty graph class. Then an instance (G, k) of $\mathcal{C}$-Vertex Deletion can be solved in time*

$$\mathcal{O}(n \cdot A_{\mathsf{compression}}(n, k+1, k)),$$

where $A_{\mathsf{compression}}(n, t, k)$ is the run-time of an algorithm for $\mathcal{C}$-Vertex Deletion Compression on instances (G', T', k') with $|V(G')| \leq n$, $|T| \leq t$ and $k' \leq k$.

Proof. Fix an arbitrary numbering $v_1, \ldots, v_n$ of the vertices in $V(G)$. For $i \in \{0, \ldots, n\}$, we define $G_i = G[\{v_1, \ldots, v_i\}]$ to be the graph induced by the first i vertices. Assume that $(G = G_n, k)$ has a solution S. Then $S \cap V(G_i)$ is a solution to (G_i, k) as

- $|S \cap V(G_i)| \leq |S| \leq k$, and
- $G_i - (S \cap V(G_i)) \in \mathcal{C}$ because $G_i - (S \cap V(G_i))$ is a subgraph of $G - S \in \mathcal{C}$ and $\mathcal{C}$ is hereditary.

That means that if an instance (G_i, k) has no solution, neither has (G, k).

We are now going to iteratively construct solutions to $(G_0, k), (G_1, k), \ldots, (G_n, k)$ or conclude that no such solutions exist. We start with G_0 being the empty graph and thus $G_0 \in \mathcal{C}$ as $\mathcal{C}$ is hereditary and non-empty. Therefore, $S_0 = \emptyset$ is a solution for the instance (G_0, k).

Now we move on to arbitrary $i \in \{1, \ldots, n\}$. Assume we are given a solution S_{i-1} to (G_{i-1}, k). Then we have $G_i - (S_{i-1} \cup \{v_i\}) = G_{i-1} - S_{i-1} \in \mathcal{C}$. So $T_i = S_{i-1} \cup \{v_i\}$ is a potential solution to (G_i, k) of size $|T_i| = |S_{i-1}| + 1 \leq k + 1$. If $|T_i| \leq k$, our set T_i is already a solution to (G_i, k). Thus, we can set $S_i = T_i$ and continue with the next i. Otherwise (G_i, T_i, k) is an instance of $\mathcal{C}$-Vertex Deletion Compression such that the solutions are exactly the solutions of (G_i, k). We call an algorithm for $\mathcal{C}$-Vertex Deletion Compression on (G_i, T_i, k). If it concludes that there is no solution, than neither has (G, k) and we stop. Otherwise, we get a solution S_i and can move to the next i in our iteration.

In the final iteration we get a solution S_n to $(G_n, k) = (G, k)$ or the information that no solution exists. Returning this information solves our $\mathcal{C}$-Vertex Deletion instance.

For the run-time, notice that we do at most n calls to $\mathcal{C}$-Vertex Deletion Compression instances (G_i, T_i, k). Additionally, these instances fulfill $|V(G_i)| \leq n$ and $|T_i| \leq k + 1$, which proves the claimed run-time. □

Iterative compression can be strengthened in two further points, both which we deal with in a second. First, we can assume that our initial solution T and our sought after solution S are disjoint. Second, instead of the whole solution S we can search for a set $\mathcal{S}$ of bounded size that intersects our solution in at least one vertex. Both modifications add to the run-time of the resulting algorithm.

Disjoint $\mathcal{C}$-Vertex Deletion Compression

Instance: A graph G, a vertex set $T \subseteq V(G)$ and an integer $k \in \mathbb{Z}_{\geq 0}$ such that $G - T \in \mathcal{C}$.

Task: Find a set $S \subseteq V(G) \setminus T$ of size at most k such that $G - S \in \mathcal{C}$ or decide that no such set exists.

Lemma 2.2 (Disjoint Compression)**.** *An instance (G, T, k) of $\mathcal{C}$-Vertex Deletion Compression can be solved in time*

$$\mathcal{O}(2^{|T|} \cdot A_{\textit{disjoint compression}}(n, |T|, k)),$$

where $A_{\textit{disjoint compression}}(n, t, k)$ is the run-time of an algorithm for Disjoint $\mathcal{C}$-Vertex Deletion Compression on instances (G', T', k') with $|V(G')| \leq n$, $|T| \leq t$ and $k' \leq k$.

Proof. Denote by 2^T the set of all subsets of T including T and the empty set. For each $T' \in 2^T$ with $|T'| \leq k$ our algorithm proceeds as follows. We call the algorithm for Disjoint $\mathcal{C}$-Vertex Deletion Compression on the instance $(G-T', T \setminus T', k-|T'|)$. If the algorithm returns a solution S', we report the solution $S = S' \cup T'$ as solution to our $\mathcal{C}$-Vertex Deletion Compression (G, T, k). This is indeed a solution, as $|S| \leq |S'| + |T'| \leq k - |T'| + |T'| = k$ and $G - S = G - (S' \cup T') = (G - T') - S' \in \mathcal{C}$.

If for every $T' \in 2^T$ with $|T'| \leq k$ our algorithm call returns that there is no solution, we output that there is no solution to (G, T, k). We claim that this is correct since for any solution S of (G, T, k), the algorithm call for $T' = T \cap S$ should have returned a solution. Indeed, $T' = T \cap S$ fulfills $|T'| \leq |T| \leq k$ and we did an algorithm call to the instance $(G - (T \cap S), T \setminus S, k - |T \cap S|)$. Now the set $S \setminus T$ is a solution to this instance as $G - (T \cap S) - (S \setminus T) = G - S \in \mathcal{C}$, $|S \setminus T| = |S| - |S \cap T| \leq k - |T \cap S|$ and $(T \setminus S) \cap (S \setminus T) = \emptyset$. Hence, our algorithm is correct.

For the run-time note that our algorithm does at most $2^{|T|}$ calls (one for every element of 2^T) to instances (G', T', k') with G' being a subgraph of G, $T' \subseteq T$ and $k' \leq k$ and thus the run-time follows. □

Corollary 2.3. *Let $\mathcal{C}$ be a hereditary and non-empty graph class. Then an instance (G, k) of $\mathcal{C}$-Vertex Deletion can be solved in time*

$$\mathcal{O}(2^{k+1} n \cdot A_{\textsf{disjoint compression}}(n, k+1, k)),$$

where $A_{\textsf{disjoint compression}}(n, t, k)$ is the run-time of an algorithm for Disjoint $\mathcal{C}$-Vertex Deletion Compression on instances (G', T', k') with $|V(G')| \leq n$, $|T| \leq t$ and $k' \leq k$.

By spending an additional run-time of $\mathcal{O}(kn \cdot A_{\textsf{membership}}(n))$, we can assume that for all instances (G, T, k) passed to the Disjoint $\mathcal{C}$-Vertex Deletion Compression, we have that T is an inclusion-wise minimal set with $G - T \in \mathcal{C}$. Here, $A_{\textsf{membership}}(n)$ is the run-time needed to check for a graph with n vertices whether it belongs to $\mathcal{C}$.

Proof. First apply Lemma 2.1 to reduce the problem to n instances of $\mathcal{C}$-Vertex Deletion Compression. For every of these n instances (G', T', k') we can check in time $(k+1)A_{\textsf{membership}}(n)$, whether $G - (T - t) \in \mathcal{C}$ for any $t \in T'$. If it is, we can directly return $T' - t$ as solution. Otherwise we know by $\mathcal{C}$ being hereditary that T' is inclusion-wise minimal with $G' - T' \in \mathcal{C}$. We then apply Lemma 2.2 to the remaining instances. □

As last refinement of our compression strategy we want to introduce an "imprecise" version of our algorithm. This algorithm is allowed to compute – instead of a "precise" solution – a set of bounded size that intersects some solution.

Disjoint $\mathcal{C}$-Vertex Deletion Compression Intersection	
Instance:	A graph G, a vertex set $T \subseteq V(G)$ and an integer $k \in \mathbb{Z}_{\geq 0}$ such that $G - T \in \mathcal{C}$.
Task:	Find a vertex set $\mathcal{S}_{\text{intersect}} \subseteq V(G)$ such that if there is a set $S \subseteq V(G) \setminus T$ of size at most k such that $G - S \in \mathcal{C}$, then there is such a set S with $S \cap \mathcal{S}_{\textsf{intersect}} \neq \emptyset$.

Lemma 2.4 (Disjoint Compression Intersection)**.** *An instance (G, T, k) of* Disjoint $\mathcal{C}$-Vertex Deletion Compression *can be solved in time*

$$\mathcal{O}(k(f_{\mathsf{intersect}}(n, |T|, k))^k \cdot (A_{\mathsf{disjoint\ compression\ intersection}}(n, |T|, k) + A_{\mathsf{membership}}(n))),$$

where

- $A_{\mathsf{disjoint\ compression\ intersection}}(n, t, k)$ *is the run-time of an algorithm for* Disjoint $\mathcal{C}$-Vertex Deletion Compression Intersection *on instances (G', T', k') that fulfill $|V(G')| \leq n$, $|T| \leq t$ and $k' \leq k$,*
- $f_{\mathsf{intersect}}(n, t, k)$ *is a size bound on the set $\mathcal{S}_{\mathsf{intersect}}$ output by the algorithm for* Disjoint $\mathcal{C}$-Vertex Deletion Compression Intersection *on these instances, and*
- $A_{\mathsf{membership}}(n)$ *is the run-time needed to test whether $G' \in \mathcal{C}$ for $|V(G')| \leq n$.*

Proof. We start our algorithm with $S = \emptyset$ and do a branching procedure that works as follows. Given some partial candidate solution $S' \subseteq V(G) \setminus T$, we first check by oracle call whether $G - S' \in \mathcal{C}$. If it is, we return $S = S'$ as solution to our instance. If it is not and $|S'| = k$, we return that there is no solution $S \supseteq S'$. In the remaining case, we have that S' is not a solution and $|S'| < k$ so there still might be a solution $S \supseteq S'$. To check for such a solution, we call our Disjoint $\mathcal{C}$-Vertex Deletion Compression Intersection algorithm on the instance $I^\star = (G - S', T, k - |S'|)$. Note that for any solution $S \supseteq S'$ to our original instance, we have that $S \setminus S'$ is a solution to $I^\star$ as Disjoint $\mathcal{C}$-Vertex Deletion Compression instance by $|S \setminus S'| = |S| - |S'| \leq k - |S'|$ and $(G - S') - (S \setminus S') = G - S \in \mathcal{C}$. Vice versa, for any solution $S^\star$ to $I^\star$ as Disjoint $\mathcal{C}$-Vertex Deletion Compression instance, we have that $S^\star \cup S'$ is a solution to our original instance, as $|S^\star \cup S'| = |S^\star| + |S'| \leq k - |S'| + |S'| = k$ and $G - (S^\star \cup S') = (G - S') - S^\star \in \mathcal{C}$. So if our original instance has a solution $S \supseteq S'$ our algorithm call returns a set $\mathcal{S}_{\text{intersect}} \subseteq V(G) \setminus S'$ that intersects a solution $S^\star$ to $I^\star$ as Disjoint $\mathcal{C}$-Vertex Deletion Compression instance. For every vertex $v \in \mathcal{S}_{\text{intersect}} \setminus T$, we then do a recursive call of our procedure with the partial candidate solution $S' \cup \{v\}$. For the right choice of v (i.e. $v \in S^\star$) our recursive call will (by induction) output that there is a solution S to our original instance with $S \supseteq S' \cup \{v\}$. If there is no solution $S \supseteq S'$ all our recursive calls will return that there is no solution $S \supseteq S' \cup \{v\}$ and we return the same to our parent algorithm call or in the case of $S' = \emptyset$, we return that there is no solution.

We argued for correctness while describing our algorithm. It only remains to prove the run-time. In every branch of our algorithm we do at most one oracle call to each, the membership and the Disjoint $\mathcal{C}$-Vertex Deletion Compression Intersection oracle. So it suffices to bound the number of branches by $k(f_{\mathsf{intersect}}(n, |T|, k))^k$. Denote by $B(i)$ the number of branches where $i = |S'|$. We start with $S' = \emptyset$ and only add items to it, so $B(0) = 1$. For each of the branches with $|S'| < k$ we do branch in $|\mathcal{S}_{\text{intersect}}| \leq f_{\mathsf{intersect}}(|G - S'|, |T|, k - |S'|) \leq f_{\mathsf{intersect}}(n, |T|, k)$ many branches, so $B(i) \leq f_{\mathsf{intersect}}(n, |T|, k) \cdot B(i-1)$ for all $i \in \{1, \ldots, n\}$. Thus, the number of overall branches is

$$\sum_{i=0}^{k} B(i) \leq \sum_{i=0}^{k} (f_{\mathsf{intersect}}(n, |T|, k))^i B(0) = k(f_{\mathsf{intersect}}(n, |T|, k))^k + 1.$$

□

2.2 Important Separators

An important tool for designing fixed-parameter algorithms for vertex deletion problems are the so-called important separators. These are $X \to Y$-separators that inclusion-wise maximize the number of reachable vertices from X among all separators of at most their size. The notion of important separators was introduced by Marx [Mar06] and has since been applied implicitly or explicitly to many $\mathcal{C}$-Deletion problems.

Definition 2.5. *Let G be a graph and let $X, Y \subseteq V(G)$ be two vertex sets. An $X \to Y$-separator C is said to be* dominated *by another $X \to Y$-separator C' if*

$$R_{G-C}(X) \subsetneq R_{G-C'}(X) \text{ and } |C'| \leq |C| \qquad \text{for undirected graphs, or}$$
$$R^+_{G-C}(X) \subsetneq R^+_{G-C'}(X) \text{ and } |C'| \leq |C| \qquad \text{for directed graphs.}$$

An $X \to Y$-separator is said to be important *if there is no $X \to Y$-separator dominating it.*

The main reason these important separators are useful is that their number can be bounded in terms of their maximum size. This can be done for directed and undirected graphs, as well as for vertex separators and edge/arc cuts. The earliest proofs of such statements can be contributed to Marx [Mar06] (undirected vertex version, 4^{k^2} size bound) and Chen et al. [CLL+08] (undirected vertex version, 4^k size bound).

Theorem 2.6 ([CFK+15, Theorem 8.11])**.** *Let G be a graph, let $X, Y \subseteq V(G)$ be two vertex sets and let $k \in \mathbb{Z}_{\geq 0}$ be an integer. Then there are at most 4^k important $X \to Y$-separators of size at most k. Moreover, the set of all important $X \to Y$-separators of size at most k can be enumerated in time $\mathcal{O}(4^k k \cdot (n+m))$.*

We give a short sketch how to prove such a statement here, for details refer to the textbook by Cygan et al. [CFK+15, Theorem 8.11]. The prove is done by induction on $2k - \lambda_G(X,Y)$, where $\lambda_G(X,Y)$ is the minimum size of an $X \to Y$-separator in G. The statement holds for $k < \lambda_G(X,Y)$, as then no (important) $X \to Y$-separator of size at most k exists. So we focus on the cases where $k \geq \lambda_G(X,Y)$. Note that there is a unique important separator C_{min} of size $\lambda_G(X,Y)$ (by "uncrossing" on the neighborhood of the reachable vertices). Consider now an arbitrary vertex $v \in C_{\mathsf{min}}$. Then we have for any important $X \to Y$-separator C and that either $v \in C$ or $v \in R^+_{G-C}(X)$. In the former case, we can recurse on $G - v$ with k being decreased by one. In the latter case, including v in X increases the size of a minimum separator C_{min} by at least one. Thus, in both cases we can prove the statement by applying the induction hypothesis.

We will generalize this argument in Section 3.1.4 to separators where we restrict the set $R^+_{G-C}(X)$ to "yes"-instances of a hereditary $\mathcal{C}$-Vertex Deletion problem.

2.3 Shadow Covering

In this section we take a look at the shadow covering framework for vertex deletion problems. It was developed by Marx and Razgon [MR14] for the Undirected Multicut problem and further improved by Chitnis et al. [CCHM15] for the Directed Subset Feedback Vertex Set problem.

The technique is aimed at vertex deletion problems in directed graphs exhibiting a certain connectivity structure. For these problems the technique can be used as follows: After applying the iterative compression technique and the disjoint compression technique (see above), we have to solve the disjoint compression variant. That is, we are already given a solution $T \subseteq V(G)$ and want to find a solution $S \subseteq V(G) \setminus T$ of size at most k. The shadow covering technique now allows us to identify a superset of those vertices of G that can either not reach T in $G - S$ or are not reachable from T in $G - S$. Special treatment of these vertices often allows to recover somewhat of the structure the analogue vertex deletion problem on undirected graphs has.

We will now discuss the result more formally. Most of the notation closely follows the notation of Chitnis et al. [CCHM15]. The set of vertices we want to identify, i.e. the vertices that are in some direction separated from T by a solution S are called the shadow of S with respect to T.

Definition 2.7 (shadows)**.** *Let G be a directed graph and let $S, T \subseteq V(G)$ be two vertex sets. The* forward shadow *of S with respect to T is the set of vertices $v \in V(G) \setminus (S \cup T)$ such that S is a $T \to \{v\}$-separator. The* reverse shadow *of S with respect to T is the set of vertices $v \in V(G) \setminus (S \cup T)$ such that S is a $\{v\} \to T$-separator. The* shadow *of S with respect to T is the union of the forward and the reverse shadow of S with respect to T.*

We now go into more detail what connectivity structure our problem must exhibit. The central concepts for this are T-connectedness and $\mathcal{F}$-transversals.

Definition 2.8 (T-connected and $\mathcal{F}$-transversal)**.** *Let G be a directed graph and $\mathcal{F}$ a family of subgraphs of G. For a vertex set $T \subseteq V(G)$ the family $\mathcal{F}$ is called* T-connected*, if for every subgraph $F \in \mathcal{F}$ and every $v \in V(F)$ there is a $\{v\} \to (T \cap V(F))$-walk in F and a $(T \cap V(F)) \to \{v\}$-walk in F. A set $X \subseteq V(G)$ is called an* $\mathcal{F}$-transversal *if for every $F \in \mathcal{F}$ we have that $X \cap V(F) \neq \emptyset$.*

With these definitions in place, we can now state the main theorem of the shadow covering technique. We restrict ourselves to the deterministic variant. For a randomized variant, as well as for the proofs, see Chitnis et al. [CCHM15].

Theorem 2.9 (Deterministic Covering of Shadows, [CCHM15, Theorem 3.6])**.** *Let G be a directed graph and let $T \subseteq V(G)$ be a vertex set. Then we can construct a set $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_t\}$ with $t = 2^{\mathcal{O}(k^2)} \log^2 n$ in time $2^{\mathcal{O}(k^2)} \cdot \mathrm{poly}(n)$ such that for any set $\mathcal{F}$ of T-connected subgraphs, if there exists an $\mathcal{F}$-transversal of size at most k, then there is an $\mathcal{F}$-transversal S of size at most k such that for at least one $Z_i \in \mathcal{Z}$ we have*

1. *$S \cap Z_i = \emptyset$, and*
2. *Z_i contains the shadow of S with respect to T.*

2.4 Bounding Powers of Logarithms

For our vertex deletion algorithms we aim to have run-times of the form $f(k)\,\mathrm{poly}(n)$. In practice this is often achieved by finding some set X of bounded size that intersects some solution of size at most k. This guessing adds a factor of $|X|^k$ to the run-time (see Lemma 2.4). If $|X|$ is bounded by a function of k only, it is clear that $|X|^k$ is also bounded by some $g(k)$. More complicated is the case where $|X| \in \Theta(\log n)$. However, in this case we can use the following lemma.

Lemma 2.10. *For $n > 2$ and $f(k) > 0$ we have*

$$(\log n)^{f(k)} \leq (2f(k)^2)^{f(k)} + n \quad \in 2^{\mathcal{O}(f(k)\log f(k))} + n.$$

Proof. We distinguish two cases, and add the upper bounds for $(\log n)^{f(k)}$ from both cases. Our case distinction depends on the relation of $f(k)$ to $\frac{\log n}{\log\log n}$. Note that by $n > 2$ this fraction is well-defined.

If $f(k) \leq \frac{\log n}{\log\log n}$ then we have $n \geq 2^{f(k)(\log\log n)} = (\log n)^{f(k)}$.

Otherwise, we have $f(k) > \frac{\log n}{\log\log n}$. This implies

$$\frac{f(k)^2}{\log n} > \frac{\log n}{(\log\log n)^2}.$$

The expression $\frac{\log n}{(\log\log n)^2}$ obtains its global minimum in the domain $(2, \infty)$ for $n = 2^{e^2}$ with a value of $0.25e^2 \ln^2(2) > 0.5$. Thus $\frac{f(k)^2}{\log n} \geq 0.5$ holds, which is equivalent to $2f(k)^2 \geq \log n$. This in turn implies $(\log n)^{f(k)} \leq (2f(k)^2)^{f(k)}$.

Adding the bounds on $(\log n)^{f(k)}$ from both cases we get

$$(\log n)^{f(k)} \leq (2f(k)^2)^{f(k)} + n$$

which using $f(k) = 2^{\log f(k)}$ lies in $2^{\mathcal{O}(f(k)\log f(k))} + n$. □

2.5 Parameterized Reductions

In the field of polynomial-time algorithms, polynomial reductions take a special place. They allow to solve problems by reducing them to other already solved problems, while keeping the algorithms running in polynomial time. Conversely, they can also show NP-hardness for a problem by reducing it to another problem, which is already known to be NP-hard. It is desirable to have another class of reductions that take this place for FPT and W[1]-hardness. These are the "parameterized reductions". See [CFK+15, Section 13.1] for a thorough introduction.

Definition 2.11 ([CFK+15, Definition 13.1])**.** *Let A, B be two parameterized problems.* A parameterized reduction *from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that*

1. *(x, k) is a "yes"-instance of A if and only if (x', k') is a "yes"-instance of B,*
2. *$k' \leq g(k)$ for some computable function g, and*
3. *the run-time is $f(k) \cdot \text{poly}(|x|)$ for some computable function f.*

These definitions of reductions are those needed to transfer membership in FPT and W[1]-hardness between parameterized problems. Sometimes, we want even stronger reductions than those, in the sense that they should only take polynomial run-time and that the new parameter is bounded by a polynomial in the old. These are known as "polynomial parameter transformations".

Definition 2.12 ([CFK+15, Definition 15.14])**.** *Let A, B be two parameterized problems.* A polynomial parameter transformation *from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that*

1. *(x, k) is a "yes"-instance of A if and only if (x', k') is a "yes"-instance of B,*
2. *$k' \leq p(k)$ for some polynomial p, and*
3. *the run-time is of the algorithm is* $\text{poly}(|x|)$.

These are indeed a subset of the parameterized reductions and thus are able to transfer membership in FPT and W[1]-hardness. However, these are also useful in other matters of parameterized hardness. See [CFK+15, Section 15.2.2] for more details on this.

Chapter 3

Directed Long Cycle Hitting Set

In this chapter we discuss the Directed Long Cycle Hitting Set problem. Here we want to delete at most k vertices from a directed graph such that no directed cycle exceeds a prescribed length ℓ.

Directed Long Cycle Hitting Set	
Instance:	A graph G and two integers $k, \ell \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq V(G)$ of size at most k such that every directed cycle of $G - S$ has length at most ℓ or decide that no such set exists.

The length of a longest directed cycle of a graph is also known as its *circumference*. If the graph is acyclic the *circumference* is defined as 0 or ∞ depending on the application. For our context it will be useful to define it as 0.

Definition 3.1. *Let G be a directed graph. The* circumference $\mathrm{cf}(G)$ *is defined as the length of the longest directed cycle in G. If G is acyclic, define* $\mathrm{cf}(G) = 0$.

With this we can write the requirement on our set S as $\mathrm{cf}(G - S) \leq \ell$. Our main result will be that Directed Long Cycle Hitting Set is indeed fixed-parameter tractable in $k + \ell$.

Theorem 3.2. *There is an algorithm that solves instances (G, k, ℓ) of Directed Long Cycle Hitting Set in time $2^{\mathcal{O}(\ell^6 + k^3\ell + k^4 \log k)} \operatorname{poly}(n)$, where $n = |V(G)|$.*

We get this result by a series of reductions working on evermore sophisticated problems. By applying the iterative compression technique (see Chapter 2) to our problem, we are able to work on graphs of bounded circumference most of the time. Thus, an important section of this chapter is dedicated to structural properties of these graphs. While many of these are well known results, we add a new tool which we call k-representative sets of paths.

Definition 3.3. *Let G be a directed graph, $x, y \in V(G)$ and $k \in \mathbb{Z}_{\geq 0}$. A set $\mathcal{P}$ of $x \to y$-paths is a* k-representative set of $x \to y$-paths*, if for every $S \subseteq V(G)$ of size at most k holds:*

If there is an $x \to y$-path in $G - S$, there is an $x \to y$-path $P \in \mathcal{P}$ in $G - S$.

A small k-representative sets of paths allows for an efficient enumeration of structures that are disjoint from an unknown solution of size k. On the other hand, if a set S of size k is an $x \to y$-cut, we have that it has to intersect all the paths in a k-representative set of $x \to y$-paths. This helps to find vertices that lie in S. For graphs of bounded circumference we are able to obtain such k-representative sets of paths.

Theorem 3.4. *Let G be a strongly connected, directed graph, $x, y \in V(G)$ and $k \in \mathbb{Z}_{\geq 0}$. Then we can find a k-representative set of $x \to y$-paths having size $\mathrm{cf}(G)^{\mathcal{O}(k^2 \log k)} \cdot \log n$ in time $\mathrm{cf}(G)^{\mathcal{O}(k^2 \log k)} \cdot \mathrm{poly}(n)$.*

This result can be of independent interest for other problems on graphs of bounded circumference. Combined with Theorem 3.2, it may also be possible to apply it to graphs which are a small vertex deletion set (a so-called circumference modulator) away from having bounded circumference.

The Directed Long Cycle Hitting Set problem generalizes a series of well-known vertex deletion problems for different values of ℓ. For $\ell = 0$ it corresponds to Directed Feedback Vertex Set. For $\ell = 2$ it generalizes (undirected) Feedback Vertex Set as well as Feedback Vertex set in Mixed Graphs (graphs with both directed arcs and undirected edges). As Feedback Vertex Set and Directed Feedback Vertex Set are both NP-hard as shown by Karp [Kar72], there is no $f(\ell)\,\mathrm{poly}(n)$ algorithm for Directed Long Cycle Hitting Set, unless $\mathsf{P} = \mathsf{NP}$.

Moreover, the special case of $k = 0$ and $\ell = n - 1$ is equivalent to checking whether a graph has a Directed Hamiltonian Cycle. As also Directed Hamiltonian Cycle was shown to be NP-hard by Karp [Kar72], there is no $f(k)\,\mathrm{poly}(n)$ algorithm for Directed Long Cycle Hitting Set, unless $\mathsf{P} = \mathsf{NP}$. Even checking whether a set S is a solution to Directed Long Cycle Hitting Set is NP-hard by above argument. In this sense, an $f(k, \ell)\,\mathrm{poly}(n)$ algorithm is optimal.

For the following special cases of Directed Long Cycle Hitting Set fixed-parameter algorithms were already known:

- Feedback Vertex Set is fixed-parameter tractable in the solution size by the graph minor algorithm due to Robertson and Seymour [RS95].
- Directed Feedback Vertex Set can be solved in time $4^k k!\,\mathrm{poly}(n)$ due to Chen et al. [CLL+08].
- Feedback Vertex Set in Mixed Graphs can be solved in time $2^{\mathcal{O}(k)} k!\,\mathrm{poly}(n)$ due to Bonsma and Lokshtanov [BL11].
- ℓ-long Cycle Detection, the task of finding a cycle of length at least ℓ, was shown to be solvable in time $2^{\mathcal{O}(\ell)}\,\mathrm{poly}(n)$ by Zehavi [Zeh16].

Our algorithm for Directed Long Cycle Hitting Set generalizes all of these algorithms. A reduction of the above mentioned problems to Directed Long Cycle Hitting Set can be found in the last section of this chapter. There we will also show that the arc and vertex deletion variant of Directed Long Cycle Hitting Set can be reduced to each other in a parameter preserving way. Thus, our algorithm also solves the arc deletion variant of Directed Long Cycle Hitting Set.

The results of this chapter are joint work with Dániel Marx and Matthias Mnich. An extended abstract of this work has previously appeared at ICALP 2020 [GMM20a]. A preliminary version of the full results has been published on arXiv [GMM20b].

3.1 Technical Tools

In this section we gather structural observations that are used in our main result but are independent of it. This includes size bounds on sets defining a separator, properties of graphs with bounded circumference and, most interestingly, our result about k-representative sets of paths. Moreover, we generalize the concept of important separators by restricting them to separators that also allow for a $\mathcal{C}$-Deletion problem to be solved on one side of the separator. We derive similar bounds on the number of these in comparison to the original important separators.

3.1.1 Size Bounds on Sets Defining a Separator

A well known result about directed separators is that there are at most 4^k many important $X \to Y$-separators of size at most k (cf. Theorem 2.6). Here we are interested in bounds on X and Y instead, i.e. are there $X' \subseteq X$ and $Y' \subseteq Y$ of bounded size such that any $X' \to Y'$-separator of size at most k is also an $X \to Y$-separator. We will derive such bounds in the following. A key ingredient to this is the following lemma.

Lemma 3.5. *Let G be a directed graph and let $x, y_1, \ldots, y_r$ be vertices of G. Let $S_1, \ldots, S_r$ be sets of vertices of size at most k each, such that the following holds for each $i = 1, \ldots, r$:*

- *y_i is reachable from x in $G - S_i$, but*
- *for each $j \in \{1, \ldots, r\} \setminus \{i\}$, there is no $x \to y_j$-path in $G - S_i$.*

Then $r \leq (k+1)4^{k+1}$.

Proof. Create a graph G' from G by adding a new vertex $y^\star$ together with the arcs $(y_i, y^\star)$ for each $i = 1, \ldots, r$. Observe that each vertex y_i is part of an $x \to y^\star$-separator $S_i' = S_i \cup \{y_i\}$ of size $k+1$ and moreover $R^+_{G' \setminus S_i'}(x)$ contains some vertex v_i such that (v_i, y_i) is an arc of G. Therefore, there exists an important $x \to y^\star$-separator S_i'' such that $R^+_{G' \setminus S_i'}(x) \subseteq R^+_{G' \setminus S_i''}(x)$, which implies that $v_i \in R^+_{G' \setminus S_i''}(s)$ and $y_i \in S_i''$. Consequently, each vertex y_i belongs to some important $x \to y^\star$-separator of size at most $k+1$. By Theorem 2.6 there are at most 4^{k+1} important separators of size at most $k+1$ and thus at most $(k+1)4^{k+1}$ such vertices, i.e. $r \leq (k+1)4^{k+1}$. □

In other words: at most $(k+1)4^{k+1}$ terminals define the structure of a separator of size at most k. We will now show how to construct for Y a small "witness" set Y' of size at most $(k+1)4^{k+1}$ such that all $x \to Y'$-separators of size at most k are also $x \to Y$-separators.

Lemma 3.6. *Let G be a directed graph, $x \in V(G)$, $Y \subseteq V(G)$, and $k \in \mathbb{Z}_{\geq 0}$. Then in time $2^{\mathcal{O}(k)} \cdot \mathrm{poly}(n)$ we can identify a set $Y' \subseteq Y$ of size at most $(k+1)4^{k+1}$ such that:*

if for $S \subseteq V(G)$ with $|S| \leq k$ there is an $x \to Y$-path in $G - S$,
then there is also an $x \to Y'$-path in $G - S$. (†)

Proof. Initially, we start with $Y' = Y$, which certainly satisfies property (†). For every $v \in Y'$ we check whether $Y' \setminus \{v\}$ also satisfies property (†). For this purpose, we need to check whether there is a set S of at most k vertices such that some vertex of Y is reachable from x in $G - S$, but no vertex of $Y' \setminus \{v\}$ is reachable. As Y' satisfies the assumptions of the lemma, if Y is reachable, then some vertex of Y' is reachable. Therefore, what we need is a set S such that v is reachable from x in $G - S$, but no vertex of $Y' \setminus \{v\}$ is reachable.

Let us introduce a new vertex $y^\star$ into G and add an arc from every vertex of $Y' \setminus \{v\}$ to $y^\star$. Observe that S is an $x \to y^\star$-separator (clearly, we have $x \notin S$). We claim that if there is an $x \to y^\star$-separator S of size at most k such that v is reachable from x in $G - S$, there is such an important separator S'. Indeed, if S' is an important separator with $|S'| \leq |S|$ and $R^+_{G-S}(x) \subseteq R^+_{G-S'}(x)$, then v is reachable from x also in $G - S'$. Therefore, we can test existence of the required separator S by testing every important $s \to y^\star$-separator of size at most k. If none of them satisfies the requirements, then we can conclude that $Y' \setminus \{v\}$ also satisfies property (†) and we can continue the process with the smaller set $Y' \setminus \{v\}$.

Suppose now that for every $v \in Y'$, we have found a set S_v of at most k vertices such that v is reachable from x in $G - S_v$, but $Y' - \{v\}$ is not. Then Lemma 3.5 implies that $|Y'| \leq (k+1)4^{k+1}$. □

Next, we prove a "set extension" of the previous lemma, in which the vertex x is enlarged to a set X. Then we apply the result to multiple sets X_i.

Lemma 3.7. *Let G be a directed graph, let $X, Y \subseteq V(G)$ be sets of vertices, and let $k \in \mathbb{Z}_{\geq 0}$. Then we can identify sets $X' \subseteq X, Y' \subseteq Y$ each of size at most $(k+1)4^{k+1}$ such that: If for $S \subseteq V(G)$ with $|S| \leq k$ there is an $X \to Y$-path in $G - S$, then there is also an $X' \to Y'$-path in $G - S$.*

These sets can be computed in time $2^{\mathcal{O}(k)} \cdot \mathrm{poly}(n)$.

Proof. Let us introduce a new vertex x into G and add an arc from x to every vertex of X. Let us use the algorithm of Lemma 3.6 to find a set $Y' \subseteq Y$ of size at most $(k+1)4^{k+1}$. Let $\overleftarrow{G}$ be the directed graph obtained from G by reversing the orientation of all arcs. Add a vertex $\overleftarrow{x}$ to $\overleftarrow{G}$ and add an arc $(\overleftarrow{x}, v)$ for every vertex $v \in Y'$. Apply the algorithm of Lemma 3.6 on $\overleftarrow{G}$ with $\overleftarrow{x}$ playing the role of x and X playing the role of Y; let X' be the set returned by the algorithm.

We claim that X' and Y' satisfy the requirements of the lemma. Suppose that there is an $X \to Y$-path P in $G - S$. By the way we obtained Y', we may assume that P ends in Y'. Then the reverse of P is a $Y' \to X$-path in $\overleftarrow{G} - S$. Therefore, by the way we obtained X' there is a path Q in $\overleftarrow{G} - S$ from Y' to X'. Now the reverse of Q is an $X' \to Y'$-path in $G - S$, as required. □

Lemma 3.8. *Let G be a directed graph, let $X_1, \ldots, X_t \subseteq V(G)$ be sets of vertices, and $k \in \mathbb{Z}_{\geq 0}$. Then we can identify sets $X'_i \subseteq X_i$ of size at most $2(t-1)(k+1)4^{k+1}$ for every $i \in \{1, \ldots, t\}$, such that: If for $S \subseteq V(G)$ with $|S| \leq k$ there is an $X_i \to X_j$-path in $G - S$ for some $i \neq j$, then there is also an $X'_i \to X'_j$-path in $G - S$.*

These sets can be computed in time $t^2 2^{\mathcal{O}(k)} \cdot \mathrm{poly}(n)$.

Proof. For every ordered pair (i, j) apply Lemma 3.7 to X_i and X_j to obtain sets $X_i^{(i,j)}$ and $X_j^{(i,j)}$. Let

$$X_i' = \bigcup_{\substack{j=1 \\ j \neq i}}^{t} \left(X_i^{(i,j)} \cup X_i^{(j,i)} \right) .$$

These have the desired properties, as for a $X_i \to X_j$-path in $G - S$ for $i \neq j$ there is by construction a $X_i^{(i,j)} \to X_j^{(i,j)}$-path in $G - S$. The size bound follows directly. □

3.1.2 Properties of Directed Graphs with Bounded Circumference

Here we gather results about directed graphs with bounded circumference. Many of them are well known and show that distances in those graphs cannot be two asymmetric.

Lemma 3.9. *Let G be a directed graph and let $x, y \in V(G)$.*

If P_1 is an $x \to y$-path and P_2 is a $y \to x$-path, then $|P_1| \leq (\mathrm{cf}(G) - 1)|P_2|$. Consequently, we have $\mathrm{dist}_G(x, y) \leq (\mathrm{cf}(G) - 1)\,\mathrm{dist}_G(y, x))$.

Proof. Note that the statement trivially holds if $x = y$: then every simple path between the two vertices has length 0. Let x', y' be any two distinct vertices of P_1 such that no internal vertex $P[x', y']$ is on P_2 (see Fig. 3.1). Going from y to x on P_2, let x'' be the first vertex of P_2 that is in $P_1[x, x']$ (possibly x'' is equal to x or x') and let y'' be the last vertex of P_2 before x'' that is on P_1. Note that y'' has to be between y' and y (possibly y'' is y' or y). As no internal vertex of $P_2[y'', x'']$ is on P_1, concatenating $P_2[y'', x'']$ and $P_1[x'', y'']$ gives a simple cycle; note that $P_1[x'', x']$ and $P_1'[y', y'']$ may contain vertices of P_2 outside $P_2[y'', x'']$. The length of this cycle is at most $\mathrm{cf}(G)$, hence $|P_1[x', y']| \leq |P_1[x'', y'']| \leq \mathrm{cf}(G) - 1$. It follows that for any $\mathrm{cf}(G) - 1$ consecutive vertices of P_1, at least one of these vertices is used by P_2. As the first and last vertices of P_1 (that is, x and y) are in P_2, it is easy to see that if n_i denotes the number of vertices of P_i, then $n_1 \leq (n_2 - 1)(\mathrm{cf}(G) - 1) + 1$. In other words, $|P_1| \leq |P_2|(\mathrm{cf}(G) - 1)$, what we had to show. □

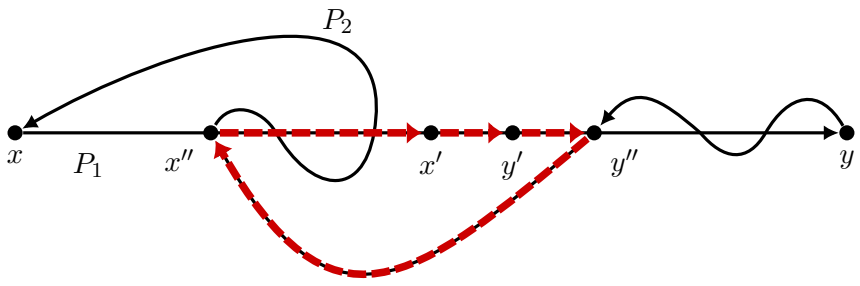


Figure 3.1: Proof of Lemma 3.9.

Note that the ratio $\mathrm{cf}(G) - 1$ in Lemma 3.9 is tight; see the solid blue and dashed green paths in Fig. 3.2 for an example with $\mathrm{cf}(G) = 4$. Examples for arbitrary $\mathrm{cf}(G) = \ell$ can be constructed similarly.

By using that there is always a backward path in strongly connected, directed graphs, applying above result twice yields:

Lemma 3.10. *Let G be a strongly connected, directed graph and $x, y \in V(G)$. Then $|P_1| \leq (\mathrm{cf}(G)-1)^2 \operatorname{dist}_G(x,y)$ for every $x \to y$-path P_1.*

Proof. If $x = y$, then every simple $x \to y$-path has length 0, and the statement holds trivially. Otherwise, let $P^\star$ be a shortest $x \to y$-path. As G is strongly connected, every arc of $P^\star$ is in a cycle of length at most $\mathrm{cf}(G)$. Thus, for every arc (u,v) of $P^\star$, there is a $v \to u$-path of length at most $\mathrm{cf}(G)-1$. Concatenating these paths for every arc of $P^\star$, we obtain a $y \to x$ walk of length at most $(\mathrm{cf}(G)-1)|P^\star| = (\mathrm{cf}(G)-1)\operatorname{dist}_G(x,y)$, and hence there is a $y \to x$-path P_2 of at most this length. By Lemma 3.9, we have $|P_1| \leq (\mathrm{cf}(G)-1)|P_2| \leq (\mathrm{cf}(G)-1)^2|P^\star| = (\mathrm{cf}(G)-1)^2 \operatorname{dist}_G(x,y)$. □

Again, the ratio $(\mathrm{cf}(G)-1)^2$ in Lemma 3.10 is tight, see the dotted red and dashed green paths in Fig. 3.2.

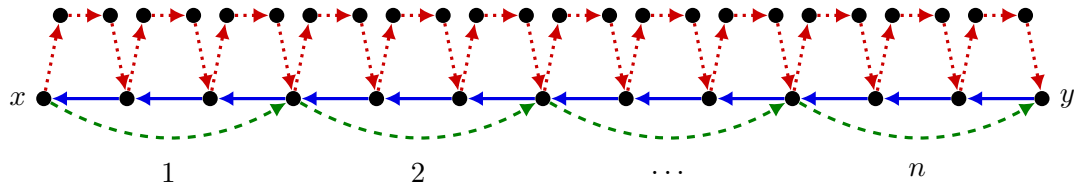


Figure 3.2: A strongly connected, directed graph G with circumference $\mathrm{cf}(G) = 4$. There is an $x \to y$-path of length n (dashed green), an $x \to y$-path of length $(\mathrm{cf}(G)-1)^2 n = 9n$ (dotted red), and a $y \to x$-path of length $(\mathrm{cf}(G)-1)n = 3n$ (solid blue).

If considering the structure of those forward and backwards paths more carefully, one gets that single vertices on paths may not lie to far away from paths running in parallel.

Lemma 3.11. *Let G be a strongly connected, directed graph, $x, y \in V(G)$ two vertices, and P_1, P_2 be two $x \to y$-paths. For every vertex v of P_1, we have $\operatorname{dist}_G(P_2, v) \leq 2(\mathrm{cf}(G)-2)$ and $\operatorname{dist}_G(v, P_2) \leq 2(\mathrm{cf}(G)-2)$.*

Proof. The claim is true if $x = y$, as then both P_1 and P_2 have length 0, and the statement holds trivially. Otherwise, as G is strongly connected, every arc of P_2 is in a cycle of length at most $\mathrm{cf}(G)$. Thus, for every arc (u,v) of P, there is a $v \to u$-path of length at most $\mathrm{cf}(G)-1$. Concatenating these paths for every arc of P_2, we obtain a walk from y to x where every vertex is at distance at most $\mathrm{cf}(G)-2$ from P_2. This implies that there is a $y \to x$-path P_3 where every vertex is at distance at most $\mathrm{cf}(G)-2$ from P_2.

Observe that if (u,v) is an arc of P_1, then $\operatorname{dist}_G(P_2, v) \leq \operatorname{dist}_G(P_2, u)+1$. Therefore, if P_1 has a vertex v with $\operatorname{dist}_G(P_2, v) > 2(\mathrm{cf}(G)-2)$, then there is a subpath $P_1[v', v'']$ with $\operatorname{dist}_G(P_2, v') = \mathrm{cf}(G)-2$, $\operatorname{dist}_G(P_2, v'') = 2\,\mathrm{cf}(G)-3$, and every internal vertex of $P_1[v', v'']$ is at distance more than $\mathrm{cf}(G)-2$ from P_2. This means that P_3 does not contain any internal vertex of $P_1[v', v'']$, since every vertex of P_3 is at distance at most $\mathrm{cf}(G)-2$ from P_2. Now $P_1[v'', y] \circ P_3 \circ P_1[x, v']$ is a $v'' \to v'$ walk that does not contain any internal vertex of $P_1[v', v'']$ and hence there is a simple cycle containing $P[v', v'']$.

Note that the length of any $v'' \to v$-path is at least 2: P_1 has no arc (v'', v') and such an arc cannot appear in P_3 either, as $\mathrm{dist}_G(P_2, v) > \mathrm{cf}(G) - 2$. Therefore, the length of this cycle is at least $|P[v', v'']| + 2(\mathrm{cf}(G) - 1) + 2 > \mathrm{cf}(G)$, a contradiction. This proves $\mathrm{dist}_G(P_2, v) \leq 2(\mathrm{cf}(G) - 2)$.

To prove the second bound $\mathrm{dist}_G(v, P_2) \leq 2(\mathrm{cf}(G) - 2)$, let us reverse the arcs of the graph and apply the first bound on the two $y \to x$-paths corresponding to P_1 and P_2. □

The bound $2(\mathrm{cf}(G) - 2)$ in Lemma 3.11 is tight: in Fig. 3.2, the dotted red $x \to y$-path has vertices at distance exactly $2(\mathrm{cf}(G) - 2) = 4$ from the dashed green path (the example can be generalized to larger $\mathrm{cf}(G)$).

3.1.3 k-Representative Sets of Paths

Let us briefly recall the definition of k-representative sets of paths.

Definition 3.3. *Let G be a directed graph, $x, y \in V(G)$ and $k \in \mathbb{Z}_{\geq 0}$. A set $\mathcal{P}$ of $x \to y$-paths is a* k-representative set of $x \to y$-paths*, if for every $S \subseteq V(G)$ of size at most k holds:*

If there is an $x \to y$-path in $G - S$, there is an $x \to y$-path $P \in \mathcal{P}$ in $G - S$.

Representative sets of paths (and also of other objects) are important tools in the design of parameterized algorithms [Mon85, FLS14, FLPS14, SZ16, Mar09].

The algorithm of Bonsma and Lokshtanov [BL11] for the case $\mathrm{cf}(G) \leq 2$ uses the following observation in an essential way. Let G be a strongly connected, directed graph and let $\langle G \rangle$ denote the underlying undirected graph of G. If $\mathrm{cf}(G) \leq 2$ then $\langle G \rangle$ is a tree (with bidirected arcs in G), and hence there is a unique $x \to y$-path P for any pair x, y of distinct vertices in G. This means that for any set $S \subseteq V(G)$, either P is an $x \to y$-path in $G - S$ or there is no $x \to y$-path in $G - S$ at all. In other words, the set $\{P\}$ is a k-representative family for every k.

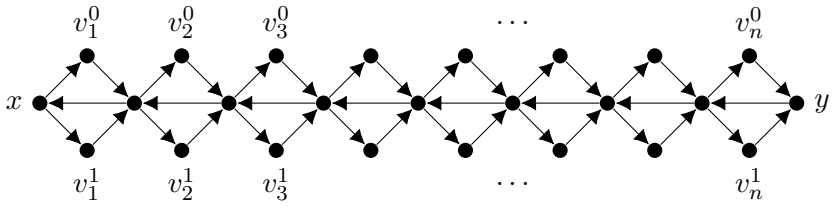


Figure 3.3: A directed graph G with $\mathrm{cf}(G) = 3$ where every k-representative set of $x \to y$-paths has size $2^{\Omega(k)} \log n$.

The situation is significantly different even for $\mathrm{cf}(G) = 3$. Consider the strongly connected, directed graph in Fig. 3.3. There are exactly 2^n different $x \to y$-paths in G; each such path corresponds to a 0-1 vector of length n by going through v_i^0 or v_i^1 depending on whether the vector has a 0 or 1 at the i-th coordinate. Thus, if we remove vertex v_i^0 (v_i^1), then only those paths survive that have 1 (0) at the i-th coordinate. Therefore, a collection of paths in this graph is k-representative only if no matter how we fix the values of k arbitrary coordinates, there is a vector in the collection satisfying these constraints. Kleitman and Spencer [KS73] proved that every collection of vectors

of length n satisfying this property has size $2^{\Omega(k)} \cdot \log n$ (more precisely, they gave a lower bound on the dual question of k-independent families, but it can be easily rephrased into this lower bound). The main result of this subsection is that in a directed graph of bounded circumference, we can construct a k-representative family of paths whose size is somewhat worse than this lower bound: assuming that the circumference is bounded by a constant, there is such a family of size $2^{\mathcal{O}(k^2 \log k)} \cdot \log n$ (Theorem 3.4).

If the paths we are considering have bounded length, then the following result of Monien [Mon85] gives a representative set of bounded size:

Theorem 3.12 ([Mon85])**.** *Let G be a directed graph, let $x, y \in V(G)$ and let $k \in \mathbb{Z}_{\geq 0}$. If every $x \to y$-path in G has length at most ℓ, then a k-representative set of $x \to y$-paths containing at most ℓ^k elements can be found in time $\ell^{\mathcal{O}(k)} \cdot \mathrm{poly}(n)$.*

Recently, Fomin et al. [FLS14] improved the computation of representative sets of paths, both in terms of the size of the set and the run-time, but Theorem 3.12 will be sufficient for our purposes.

We will show that in strongly connected, directed graphs of bounded circumference, a k-representative set of bounded size can be found even if there is no bound on the length of the $x \to y$-paths. The proof uses so called *k-perfect families of hash functions*.

Definition 3.13. *Let U be a finite set and $k \in \mathbb{Z}_{>0}$. We say that a family $\mathcal{F}$ of functions $f : U \to \{1, \ldots, k\}$ is a* k-perfect family of hash functions *if for every $X \subseteq U$, there is an $f \in \mathcal{F}$ such that $f|_X$ is injective, i.e. $f(x) \neq f(x')$ for all distinct $x, x' \in X$.*

We use the following construction of Alon et al. [AYZ95] that gives us a k-perfect family of hash functions of bounded size.

Theorem 3.14 ([AYZ95])**.** *Let U be a finite set and $k \in \mathbb{Z}_{>0}$. Then there is a k-perfect family of hash functions $\mathcal{F}$ with $|\mathcal{F}| \in 2^{\mathcal{O}(k)} \log |U|$. Moreover, $\mathcal{F}$ can be constructed in time $2^{\mathcal{O}(k)} \mathrm{poly}(|U|)$.*

Before presenting the construction of representative sets for strongly connected, directed graphs of bounded circumference, let us explain how k-perfect families of hash functions can be used for the construction in the case of the graph of Fig. 3.3. Let $\mathcal{F}$ be a k-perfect family of hash functions over the universe $U = \{1, \ldots, n\}$. For every $f \in \mathcal{F}$ and every function $h : \{1, \ldots, k\} \to \{0, 1\}$, we add to the set $\mathcal{P}$ the path P that uses vertex $v_i^{h(f(i))}$ for every $i \in \{1, \ldots, n\}$. Now we can write any set $S \subseteq V(G)$ of size k as $\{v_i^{g(i)} \mid i \in X\}$ for some $X \subseteq U$ of size k and function $g : X \to \{0, 1\}$. As $\mathcal{F}$ is a k-perfect family, there is an $f \in \mathcal{F}$ that is injective on X. For every $i \in X$, let us define $h(f(i)) = 1 - g(i)$; as f is injective on X, this is well-defined and gives a function $h : \{1, \ldots, k\} \to \{0, 1\}$. We claim that the path P introduced into $\mathcal{P}$ for this choice of f and h is disjoint from S. For $i \notin X$, it does not matter if P uses v_i^0 or v_i^1. For $i \in X$, set S contains $v_i^{f(i)}$. By our definition of h, we have $h(f(i)) = 1 - g(i)$, hence P uses $v_i^{1-g(i)}$, avoiding S. Thus, P is indeed disjoint from S.

The following proof generalizes this construction to arbitrary strongly connected, directed graphs of bounded circumference: we construct the path by concatenating a series of fairly independent "short jumps." The short jumps are taken from a representative set of short paths constructed by Theorem 3.12. The choice of which short path to select is determined by a k-perfect family of hash function, similarly to the argument in the previous paragraph.

Theorem 3.4. *Let G be a strongly connected, directed graph, $x, y \in V(G)$ and $k \in \mathbb{Z}_{\geq 0}$. Then we can find a k-representative set of $x \to y$-paths having size $\operatorname{cf}(G)^{\mathcal{O}(k^2 \log k)} \cdot \log n$ in time $\operatorname{cf}(G)^{\mathcal{O}(k^2 \log k)} \cdot \operatorname{poly}(n)$.*

Proof. Let us fix an arbitrary $x \to y$-path R (which exists as G is strongly connected) to guide our construction. Denote by r the length of R and by $v_0 = x, v_1, \ldots, v_{r-1}, v_r = y$ its vertices. We only consider a subset of vertices z_i at distance $d = 2\operatorname{cf}(G)^4$ from each other or more formally $z_i = v_{i \cdot d}$. These z_i will be the anchor vertices for our short jumps. We divide the z_i further into $k+1$ subsets Z^o by taking every $(k+1)$st vertex starting at offset o. Formally we define $z_i^o = z_{i(k+1)+o}$ and $Z^o = \{z_i^o\}$. These subsets have the advantage that one of these is far away from a deletion set S of size at most k. For this we fix a set S of size at most k such that an $x \to y$-path in $G - S$ exists.

Claim 1. *There is some $o_S \in \{0, \ldots, k\}$ such that*
- $\operatorname{dist}_G(Z^{o_S}, S) > 2(\operatorname{cf}(G) - 2)$ *and*
- $\operatorname{dist}_G(S, Z^{o_S}) > 2(\operatorname{cf}(G) - 2)$.

Proof of Claim 1. We claim that for every $s \in S$ there is at most one $o \in \{0, \ldots, k\}$ such that $\operatorname{dist}_G(Z^o, s) \leq 2\operatorname{cf}(G)^2$. Suppose that $\operatorname{dist}_G(w_1, s), \operatorname{dist}_G(w_2, s) \leq 2\operatorname{cf}(G)^2$ for some $w_1 \in Z^{o_1}$ and $w_2 \in Z^{o_2}$ with $o_1 \neq o_2$. Assume, without loss of generality, that w_1 appears before w_2 on R; then $R[w_1, w_2]$ has length at least d (as different z_i have distance at least d). By Lemma 3.9, we have

$$\operatorname{dist}_G(s, w_1) \leq (\operatorname{cf}(G) - 1) \operatorname{dist}_G(w_1, s) \leq (\operatorname{cf}(G) - 1) \cdot 2\operatorname{cf}(G)^2,$$

and thus $\operatorname{dist}_G(w_2, w_1) \leq \operatorname{dist}_G(w_2, s) + \operatorname{dist}_G(s, w_1) \leq 2\operatorname{cf}(G)^3$. Again by Lemma 3.9, we have $d \leq |R[w_1, w_2]| \leq (\operatorname{cf}(G) - 1) \operatorname{dist}_G(w_2, w_1) < 2\operatorname{cf}(G)^4$, a contradiction. Hence, for each of the k vertices $s \in S$ there is at most one value $o \in \{0, \ldots, k\}$ such that s is at distance at most $2\operatorname{cf}(G)^2$ from Z^o. Therefore, by the pigeon-hole principle there is an $o_S \in \{0, \ldots, k\}$ such that $\operatorname{dist}_G(Z^{o_S}, S) > 2\operatorname{cf}(G)^2$. By Lemma 3.9 this also implies $\operatorname{dist}_G(S, Z^{o_S}) > 2\operatorname{cf}(G)^2/(\operatorname{cf}(G) - 1) > 2(\operatorname{cf}(G) - 2)$. This completes the proof of Claim 1. ■

Thus, we know that a small surrounding of one of the Z^o's will be disjoint from S. Furthermore, Lemma 3.10 gives a bound on the length of a path P between two consecutive vertices z_i^o and z_{i+1}^o of Z^o, by $|P| \leq (\operatorname{cf}(G) - 1)^2 |R[z_i^o, z_{i+1}^o]| = \mathcal{O}(\operatorname{cf}(G)^6 k)$. This allows us to introduce sets $\mathcal{P}_i^o$ of k-representative $z_i^o \to z_{i+1}^o$-paths using the algorithm of Theorem 3.12 and have their size bounded by some $B = \mathcal{O}(\operatorname{cf}(G)^6 k)^k = \operatorname{cf}(G)^{\mathcal{O}(k \log k)}$ (using $k = 2^{\log k}$ and $\operatorname{cf}(G) \geq 2$). By duplicating paths as necessary we can assume that every P_i^o has size exactly B.

To make sure that our path collections with offset are connected to x and y we construct additional sets $\mathcal{P}_x^o$ and $\mathcal{P}_y^o$ as follows: Let z_x^o be the first vertex in Z^o *after* x and z_y^o the last vertex *before* y. Then compute, using the algorithm of Theorem 3.12, $\mathcal{P}_x^o$ as a k-representative set of $x \to z_x^o$-paths and $\mathcal{P}_y^o$ as a k-representative set of $z_y^o \to y$-paths. As the distances between these pairs of vertices are bounded by the distance of neighboring vertices in Z^o we can analogously get a size bound of B for $\mathcal{P}_x^o$ and $\mathcal{P}_y^o$. Note that for some offsets o either $\mathcal{P}_x^o$ or $\mathcal{P}_y^o$ may align with some $\mathcal{P}_i^o$; then we leave out this $\mathcal{P}_i^o$ as we do not need it anymore. We define the set of these relevant sets as $\mathcal{P}^o := \{\mathcal{P}_x^o, \mathcal{P}_y^o\} \cup \{\mathcal{P}_i^o\}_i$ for each o.

Claim 2. *Every $\mathcal{P}^{o_S}_T \in \mathcal{P}^{o_S}$ contains a path disjoint from S.*

Proof of Claim 2. Consider a set $\mathcal{P}^{o_S}_T$ with $T \in \{x, y, i\}$ such that the paths in $\mathcal{P}^{o_S}_T$ are $x_T \to y_T$-paths. As above sets are k-representative sets of paths, we must only show that there is any $x_T \to y_T$-path in $G - S$.

By assumption there is an $x \to y$ path Q in $G - S$. By Lemma 3.11 we can find a $q_x \in V(Q)$ such that $\mathrm{dist}_G(x_T, q_x) \leq 2(\mathrm{cf}(G) - 2)$ and a $x_T \to q_x$-path Q_x in G achieving this distance. By Claim 1 we know that Q_x is disjoint from S and therefore, $Q_x \circ Q[q_x, y]$ is an $x_T \to y$ walk disjoint from S. Let $\hat{Q}_x$ be an $x_T \to y$-path contained in this walk. Another application of Lemma 3.11 yields a vertex $q_y \in V(\hat{Q})$ with $\mathrm{dist}_G(q_y, y_T) \leq 2(\mathrm{cf}(G) - 2)$ and a $q_y \to y_T$-path Q_y in G achieving this distance. Again, by Claim 1, Q_y is disjoint from S. Then $\hat{Q}_x[x_T, q_y] \circ Q_y$ contains a $x_T \to y_T$-path as proposed. This completes the proof of Claim 2. ■

Of course, enumerating all possible tuples of paths would construct too many candidates, as the size of $\mathcal{P}^{o_S}$ can be $\Omega(m)$. Therefore, we want to use a $f(k)$-perfect family of hash functions. This is possible if we can bound the number of intersections of S with different sets in $\mathcal{P}^{o_S}$ by some function $f(k)$.

Claim 3. *The set S intersects at most $2k$ sets of $\mathcal{P}^{o_S}$.*

Proof of Claim 3. We show that a single $s \in S$ can intersect at most two sets of $\mathcal{P}^{o_S}$ and those have to share an endpoint, thus achieving the claimed size bound. Suppose, for sake of contradiction, that s intersects two paths Q_1 and Q_2 out of sets in $\mathcal{P}^{o_S}$ that do not share an endpoint. Let each Q_i be an $x_i \to y_i$-path. Assume, without loss of generality, that the order in which the endpoints appear on R is x_1, y_1, x_2, y_2, and that $|R[y_1, x_2]| \geq 2\,\mathrm{cf}(G)^5$ (by the distance of the z_i). At the same time, $R[x_i, y_i]$ and Q_i connect the same endpoints, hence Lemma 3.11 implies that there is a $t_1 \in V(R[x_1, y_1])$ with $\mathrm{dist}_G(t_1, s) \leq 2(\mathrm{cf}(G) - 2)$ and a $t_2 \in V(R[x_2, y_2])$ with $\mathrm{dist}_G(s, t_2) \leq 2(\mathrm{cf}(G) - 2)$ as $s \in Q_1 \cap Q_2$. This implies that $\mathrm{dist}_G(t_1, t_2) \leq \mathrm{dist}_G(t_1, s) + \mathrm{dist}_G(s, t_2) \leq 4(\mathrm{cf}(G) - 2)$. If we now consider $R[t_1, t_2]$, we get

$$|R[t_1, t_2]| \geq |R[y_1, x_2]| \geq 2\,\mathrm{cf}(G)^4 > (\mathrm{cf}(G) - 1)^2 \cdot \mathrm{dist}_G(t_1, t_2),$$

in contradiction to Lemma 3.10. This completes the proof of Claim 3. ■

We can now construct a $2k$-perfect family Ψ^o of hash functions over the universe $\mathcal{P}^o$ for each o. For o_S this family contains an element ψ which assigns all sets of $\mathcal{P}^{o_S}$ that are intersected by S a different number in $\{1, \ldots, 2k\}$ (by Claim 3). Further, there is a map π_{free} that maps the numbers of $\{1, \ldots, 2k\}$ to a number of $\{1, \ldots, B\}$, such that for every $\mathcal{P} \in \mathcal{P}^{o_S}$ which has a path intersected by S, we have that the $\psi \circ \pi_{\mathsf{free}}(\mathcal{P})$th path of $\mathcal{P}$ is not intersected by S. There is such a path by Claim 2. Denote by $Q_{\psi, \pi_{\mathsf{free}}}(\mathcal{P})$ this path. As we cannot know π_{free} in advance we create a set Π of all possible functions from $\{1, \ldots, 2k\}$ to $\{1, \ldots, B\}$.

We know that for the specific choices of o_S, ψ and π_{free} we get a that the union of paths in $\{Q_{\psi, \pi_{\mathsf{free}}}(\mathcal{P}) | \mathcal{P} \in \mathcal{P}^{o_S}\}$ forms an $x \to y$ walk W in $G - S$. Every $x \to y$-path within W is also disjoint from S. Therefore, the set $\mathcal{P}_{x,y,k}$ created as follows contains a path disjoint from S: For every $o \in \{1, \ldots, k + 1\}$, every $\psi \in \Psi$ and every $\pi \in \Pi$ consider the $x \to y$-walk $\bigcup_{\mathcal{P} \in \mathcal{P}^o} Q_{\psi, \pi}(\mathcal{P})$ and introduce an arbitrary $x \to y$-path in it into $\mathcal{P}_{x,y,k}$.

The size bound on $\mathcal{P}_{x,y,k}$ is proven by multiplying the possibilities for each choice:

$$\underbrace{(k+1)}_{\text{choice of } o} \cdot \underbrace{2^{\mathcal{O}(k)} \log m}_{|\Psi|} \cdot \underbrace{B^{2k}}_{|\Pi|} = \mathrm{cf}(G)^{\mathcal{O}(k^2 \log k)} \log n.$$

The run-time follows similarly. □

The previous lemma is very useful if we have a strongly connected, directed graph of bounded circumference. However, if we have a graph G and a subset of vertices T such that $\mathrm{cf}(G-T) \leq \ell$ it is not clear how to get a k-representative set of paths. Instead, we give a weaker result which suffices for our algorithm. We restrict our deletion sets from arbitrary sets S of size at most k to sets which additionally fulfill $\mathrm{cf}(G-S) \leq \ell$. Additionally, instead of paths we consider closed walks connecting at least two vertices of T after the deletion of S. The following lemma helps us to structure the surroundings of T.

Lemma 3.15. *Let G be a directed graph, let $s,t \in V(G)$, and let $k,d \in \mathbb{Z}_{\geq 0}$. In time $2^{\mathcal{O}(kd)}\,\mathrm{poly}(n)$ we can construct collections $\mathcal{R}_{\leq 2d}$ and $\mathcal{R}_{>2d}$, each of size $2^{\mathcal{O}(kd)}$, with*

- *$\mathcal{R}_{\leq 2d}$ contains only paths of length at most $2d$,*
- *$\mathcal{R}_{>2d}$ contains pairs (P_s, P_t), where each P_s is a path of length d starting at s, and each P_t is a path of length d ending at t.*

Then, for every set S of size at most k, if there is an $s \to t$-path P disjoint from S then there is an $s \to t$-path P' disjoint from S with

- *$P' \in \mathcal{R}_{\leq 2d}$ if $|P'| \leq 2d$ or*
- *$(P'_s, P'_t) \in \mathcal{R}_{>2d}$, where P'_s and P'_t are the disjoint subpaths of P' containing the first and the last d arcs of P' respectively, if $|P'| > 2d$.*

Proof. The proof is by induction on d. For $d = 0$, the construction is easy: If $s = t$ introduce the zero length path $\{s\}$ into $\mathcal{R}_{\leq 2d}$. Otherwise set $\mathcal{R}_{\leq 2d} = \emptyset$. In any case let $\mathcal{R}_{>2d}$ contain a single pair (P_s, P_t) with P_s and P_t being zero length paths containing only vertices s and t respectively. This construction is correct since every $s \to t$-path starts in s and ends in t and a zero length path does only exist (and is unique) if $s = t$.

Suppose that $d > 0$, and that the statement of the lemma holds for $d-1$. We start with $\mathcal{R}_{\leq 2d} = \mathcal{R}_{>2d} = \emptyset$. If $s = t$ or s and t are adjacent, then we introduce to $\mathcal{R}_{\leq 2d}$ the path of length 0 or 1, respectively. Afterwards, let us invoke the algorithm of Lemma 3.6 on $X = N^+(s)$ and $Y = N^-(t)$. For every pair (s', t') with $s' \in X'$ and $t' \in Y'$, let us use the induction hypothesis and invoke our algorithm on the directed graph $G - \{s,t\}$, vertices s', t', and integers k and $d-1$ to enumerate the collections $\mathcal{R}'_{\leq 2(d-1)}$ and $\mathcal{R}'_{>2(d-1)}$. We add to $\mathcal{R}_{\leq 2d}$ all paths obtained by extending a path $P \in \mathcal{R}'_{\leq 2(d-1)}$ into sPt. Moreover, we add to $\mathcal{R}_{>2d}$ all pairs obtained by extending a pair $(P'_s, P'_t) \in \mathcal{R}'_{>2(d-1)}$ to $(sP'_s, P'_t t)$.

To prove that the resulting collections $\mathcal{R}_{\leq 2d}$ and $\mathcal{R}_{>2d}$ satisfy the requirements, consider a path P disjoint from an arbitrary set $S \subseteq V(G)$ of size at most k. If P has length 0 or 1, we introduced this path into $\mathcal{R}_{\leq 2d}$ and are done. Otherwise, let s' and t' be the neighbors of s and t on P, respectively. As $|P| \geq 2$, $P[s', t']$ is a subpath of P and therefore disjoint from S. Moreover, it is disjoint from s and t by definition. By choice of X' and Y' (see Lemma 3.6), there is an $x \to y$-path Q in $G - (S \cup \{s,t\})$ with $x \in X'$ and $y \in Y'$. Therefore, by the induction hypothesis, there is a path Q' in $G - (S \cup \{s,t\})$ such that either $Q' \in \mathcal{R}'_{\leq 2(d-1)}$ or $(Q'_s, Q'_t) \in \mathcal{R}'_{>2(d-1)}$, where Q'_s and Q'_t contain the first and last $d-1$ arcs of Q', respectively.

In the first case, $sQ't$ is an $s \to t$-path in $G - S$ which we introduced to $\mathcal{R}_{\leq 2d}$. In the second case, $(sQ'_s, Q'_t t)$ will appear in $\mathcal{R}_{>2d}$ and satisfy the requirements.

It can be proven inductively that $\mathcal{R}_1$ and $\mathcal{R}_2$ have size $2^{\mathcal{O}(kd)}$. The overall run-time for their construction is $2^{\mathcal{O}(kd)} \operatorname{poly}(n)$. □

Lemma 3.16. *Let G be a directed graph with two vertices s, t and let k, ℓ be integers such that $\operatorname{cf}(G - \{s,t\}) \leq \ell$. Then in time $2^{\mathcal{O}(k\ell + k^2 \log k)} \cdot \operatorname{poly}(n)$, we can compute a collection $\mathcal{Q}$ of $2^{\mathcal{O}(k\ell + k^2 \log k)} \log^2 n$ closed walks in G, each containing both s and t, such that the following holds:*
if $S \subseteq V(G)$ is a set of at most k vertices in G such that $\operatorname{cf}(G - S) \leq \ell$ and $G - S$ has a closed walk containing both s and t, then there is a closed walk in $\mathcal{Q}$ disjoint from S.

Proof. We first compute a collection $\mathcal{P}_{s,t}$ of $s \to t$-paths. Let us use the algorithm of Lemma 3.15 with directed graph G and $d = \ell$ to compute the collection $\mathcal{R}_{\leq 2\ell}$ and $\mathcal{R}_{>2\ell}$. Let us introduce every path in $\mathcal{R}_{\leq 2\ell}$ into $\mathcal{P}_{s,t}$. We will introduce further paths into $\mathcal{P}_{s,t}$ based on $\mathcal{R}_{>2\ell}$ the following way. For every $(P_s, P_t) \in \mathcal{R}_{>2\ell}$ with $x \in V(P_s)$ and $y \in V(P_t)$, if x and y are in the same strongly connected component C of $G - \{s,t\}$, invoke the algorithm of Theorem 3.4 to obtain a collection $\mathcal{P}_{x,y,k}$. For each $Z \in \mathcal{P}_{x,y,k}$, we have that $P_s[s,x] \circ Z \circ P_t[y,t]$ forms an $s \to t$-walk Z^*. Thus, we can for every $Z \in \mathcal{P}_{x,y,k}$ introduce an $s \to t$-path using only the vertex set of Z^* into $\mathcal{P}_{s,t}$. Observe that the size of $\mathcal{P}_{s,t}$ obtained this way can be bounded by $2^{\mathcal{O}(k\ell + k^2 \log k)} \log n$.

We repeat a similar construction step with the roles of s and t reversed, to obtain a collection $\mathcal{P}_{t,s}$ of $t \to s$-paths. Then for every choice of $P_{s,t} \in \mathcal{P}_{s,t}$ and $P_{t,s} \in \mathcal{P}_{t,s}$, we introduce the concatenation of $P_{s,t}$ and $P_{t,s}$ into $\mathcal{Q}$. Clearly, every member of $\mathcal{Q}$ is a closed walk containing both s and t, and the size of $\mathcal{Q}$ is $2^{\mathcal{O}(k\ell + k^2 \log k)} \log^2 n$.

To prove the correctness of the construction, suppose that S is a set of at most k vertices such that $\operatorname{cf}(G - S) \leq \ell$ and $G - S$ has a closed walk containing both s and t. This means that there is an $s \to t$-path $P_{s,t}$ and a $t \to s$-path $P_{t,s}$, both disjoint from S. We claim that both $\mathcal{P}_{s,t}$ and $\mathcal{P}_{t,s}$ contain paths disjoint from S. If this is true, then it follows by construction that $\mathcal{Q}$ contains a closed walk disjoint from S.

Let us prove that $\mathcal{P}_{s,t}$ contains a path disjoint from S (the statement for $\mathcal{P}_{t,s}$ follows symmetrically). As there is an $s \to t$-path $P_{s,t}$ disjoint form S by Lemma 3.15, there is an $s \to t$-path Q such that either Q in $\mathcal{R}_{\leq 2\ell} \subseteq \mathcal{P}_{s,t}$ or $(Q_s, Q_t) \in \mathcal{R}_{>2\ell}$ with Q_s and Q_t being the subpaths of its first and last ℓ arcs respectively. In the first case, we have that $Q \in \mathcal{P}_{s,t}$ and are done. Otherwise, consider the last vertex x of Q_s and the first vertex y of Q_t. Then $Q[x,y]$ is a certificate that there is an $x \to y$-path in $(G - \{s,t\}) - S$.

We want to argue that x and y are in the same strongly connected component of $G - \{s,t\}$. Consider the path $P_{t,s}$. As both Q and $P_{t,s}$ exist in $G - S$ the closed walk W they form must contain no cycle of length greater than ℓ. Therefore, the path Q_s must be intersected by $P_{t,s}$ outside of s, as otherwise the cycle in W containing the segment Q_s has length greater than ℓ. Let x' be the last vertex of $P_{t,s} - s$ that intersects Q_s. By the same argument, Q_t must be intersected by $P_{t,s} - t$. Let y' the first vertex of $P_{t,s} - t$ that intersects Q_t. Then $Q[x', y'] \circ P_{t,s}[y', x']$ is a closed walk in $G - \{s,t\}$ containing x and y. Therefore, x and y are in the same strongly connected component of $G - \{s,t\}$.

Then, by choice of $\mathcal{P}_{x,y,k}$, there is a path $Z \in \mathcal{P}_{x,y,k}$ that is disjoint from S, and we have extended Z to Z^* by adding P_s and P_t to it and then introduced it into $\mathcal{P}_{s,t}$. As Z, P_s, and P_t are all disjoint from S, it follows that $\mathcal{P}_{s,t}$ contains a path disjoint from S. □

Lemma 3.17. *Let G be a strongly connected, directed graph, $T \subseteq V(G)$ and $k, \ell \in \mathbb{Z}_{\geq 0}$ with $\mathrm{cf}(G - T) \leq \ell$. Then in time $2^{\mathcal{O}(k\ell + k^2 \log k)} \mathrm{poly}(n)$, we can find a set $\mathcal{Q}$ of $|T|^2 2^{\mathcal{O}(k\ell + k^2 \log k)} \log^2 n$ closed walks with the following property:*

If there is a set $S \subseteq V(G)$ of size at most k with $\mathrm{cf}(G - S) \leq \ell$ and there are two vertices of T in the same strongly connected component of $G - S$ then there is a closed walk in $Q \in \mathcal{Q}$ containing two vertices of T that is disjoint from S.

Proof. We construct $\mathcal{Q}$ the following way. For every pair $s, t \in T$, we invoke the algorithm on Lemma 3.16 in $G - (T \setminus \{s, t\})$ and vertices s, t to obtain a collection $\mathcal{Q}_{s,t}$. Moreover, we invoke for every ordered pair $s, t \in T$ Theorem 3.12 to obtain a k-representative set of $s \to t$-paths $\mathcal{R}^{\leq \ell}_{s,t}$ of length at most ℓ. For this it is enough to invoke Theorem 3.12 on the subgraph where every vertex is at distance at most ℓ from s. The collection $\mathcal{Q}$ will be the union of the $\binom{|T|}{2}$ collections $\mathcal{Q}_{s,t}$ and all $|T|^2 \ell^{2k}$ closed walks of the form $R_{s,t} \circ R_{t,s}$ with $R_{p,q} \in \mathcal{R}^{\leq \ell}_{p,q}$.

To prove the correctness, suppose that $G - S$ has a strongly connected component C containing at least two vertices of T. This means that there is a closed walk R containing at least two vertices of T; let us choose R such that $|T \cap V(R)|$ is at least two, and subject to that, R is of minimum length. If R contains exactly two vertices s, t of T, then Lemma 3.16 guarantees that a member of $\mathcal{Q}$ is disjoint from S. Suppose that R contains a set T_0 of at least three vertices of T. If R is a simple cycle, then it has length at most ℓ as $\mathrm{cf}(G - S) \leq \ell$. This means that there is an $s \to t$-path $P_{s,t}$ and a $t \to s$-path $P_{t,s}$ of length at most ℓ in $G - S$. By choice of the $\mathcal{R}^{\leq \ell}_{p,q}$, we have we introduced a closed walk $R_{s,t} \circ R_{t,s}$ containing s and t into $\mathcal{Q}$ that is disjoint from S.

Now assume that R is not a simple cycle. Then, there is a vertex $x \in V(R)$ that is visited at least twice during the walk, meaning that the walk can be split into two closed walks R_1 and R_2, meeting at x (this is true even if R visits x more than twice). As $|T \cap V(R)| \geq 3$, we can assume without loss of generality that R_1 visits at least two vertices of T. This means that R_1 visits at least two vertices of T, but is a strict subset of R, contradicting the minimal choice of R. □

3.1.4 Important Ranged $\mathcal{C}$-Deletion Separator

In this section we consider the W-Ranged $\mathcal{C}$-Deletion Separator problem. Similar to to the $\mathcal{C}$-Deletion problem, we assume a fixed graph class $\mathcal{C}$ that is given, but not part of the input to our problem. The input to our problem then consists of a directed graph G, three vertex sets $W, X, Y \subseteq V(G)$ with $W \subseteq X$, and an integer $k \in \mathbb{Z}_{\geq 0}$. Our goal is to compute a pair of vertex sets (D, S) such that S is an inclusion-wise minimal $X \to Y$-separator and the graph $G[R^+_{G-(D \cup S)}(W)]$ lies in $\mathcal{C}$. Meanwhile k is a size bound on the total size of the two sets. We call such pairs W-ranged $\mathcal{C}$-deletion $X \to Y$-separators or short ranged deletion separators.

Definition 3.18. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$ and let $\mathcal{C}$ be a graph class. A W-*ranged $\mathcal{C}$-deletion $X \to Y$-separator *is a pair (D, S) with $D, S \subseteq V(G) \setminus (W \cup X \cup Y)$ such that S is an inclusion-wise minimal $X \to Y$-separator and $G[R^+_{G-(D \cup S)}(W)] \in \mathcal{C}$. The size of (D, S), denoted by $|(D, S)|$, is defined as $|D| + |S|$.*

Note that $\mathcal{C}$-deletion problems can be turned into W-Ranged $\mathcal{C}$-Deletion Separator problems by adding a new vertex w with forward arcs to all original vertices and apply the same modification to every graph in $\mathcal{C}$. Choosing $W = X = \{w\}$ and $Y = \emptyset$ then yields an equivalent instance. As we do not know how to solve $\mathcal{C}$-Deletion in general, we assume we are given an oracle for W-Ranged $\mathcal{C}$-Deletion, i.e. for an instance (G, k, W) finding a set $D \subseteq V(G) \setminus W$ with $|D| \leq k$ such that $G[R^+_{G-D}(W)] \in \mathcal{C}$. This is exactly the remaining problem after the set S has been fixed. Unfortunately, for general graph classes $\mathcal{C}$, it is not clear how the choice of S interacts with D. We therefore restrict ourselves to hereditary graph classes $\mathcal{C}$, i.e. classes for which if $G \in \mathcal{C}$ any subgraph of G also lies in $\mathcal{C}$. For these graph classes finding some W-ranged $\mathcal{C}$-deletion $X \to Y$-separator is relatively straight forward (given the above mentioned oracle). The main observation is the following.

Lemma 3.19. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$ and let $\mathcal{C}$ be a hereditary graph class. Then for any W-ranged $\mathcal{C}$-deletion $X \to Y$-separator (D, S) and any inclusion-wise minimal $X \to Y$-separator S' with $R^+_{G-S'}(X) \subseteq R^+_{G-S}(X)$, we have that for $D' = D \cap R^+_{G-S'}(W)$ the set (D', S') is also an W-ranged $\mathcal{C}$-deletion $X \to Y$-separator.*

Proof. We know that S' is an inclusion-wise minimal $X \to Y$-separator. Thus, for (D', S') to be a W-ranged $\mathcal{C}$-deletion $X \to Y$-separator, we only have to check that $G[R^+_{G-(D' \cup S')}(W)] \in \mathcal{C}$. For this note that by definition of D' we have

$$G[R^+_{G-(D' \cup S')}(W)] = G[R^+_{G-(D \cup S')}(W)].$$

Moreover, by definition of $R^+_{G-S'}(W)$ we have for every $v \in R^+_{G-S'}(W)$ that there is an $W \to v$-path in $R^+_{G-S'}(W) \subseteq R^+_{G-S'}(X) \subseteq R^+_{G-S}(X)$.

This implies $R^+_{G-S'}(W) \subseteq R^+_{G-S}(W)$. Thus, we have

$$G[R^+_{G-(D' \cup S')}(W)] = G[R^+_{G-(D \cup S')}(W)] = G[R^+_{G-D}(W) \cap R^+_{G-S'}(W)]$$

which is a subgraph of $G[R^+_{G-D}(W) \cap R^+_{G-S}(W)] = G[R^+_{G-(D \cup S)}(r)] \in \mathcal{C}$. As $\mathcal{C}$ is hereditary, this implies $G[R^+_{G-(D' \cup S')}(W)] \in \mathcal{C}$. □

Lemma 3.19 leads to the following approach for solving W-Ranged $\mathcal{C}$-Deletion Separator: enumerate all $X \to Y$-separators S of size at most k, which inclusion-wise minimize $R^+_{G-S}(X)$ among all $X \to Y$-separators of smaller or equal size. I.e. enumerate all $X \to Y$-separators S of size at most k for which there is no $X \to Y$-separators S' with $R^+_{G-S'}(X) \subsetneq R^+_{G-S}(X)$ and $|S'| \leq |S|$, and there is no subset of S that is an $X \to Y$-separator. Note that these are exactly the important $Y \to X$-separators in the graph $\overleftarrow{G}$, where every arc is reversed. Then call an oracle for W-Ranged $\mathcal{C}$-Deletion on $(G[R^+_{G-S}(W)], k - |S|, r)$. If it returns a set D, we know that (D, S) is a ranged deletion separator as

$$G[R^+_{G-(D \cup S)}(W)] = (G[R^+_{G-S}(W)])[R^+_{G[R^+_{G-S}(W)]-D}(X)].$$

On the other hand, if there is a solution we have that by Lemma 3.19 there is also a solution whose separator inclusion-wise minimizes $R^+_{G-S}(X)$. Thus if none of the calls returns a set D, we return truthfully that the instance has no solution.

We now want to tackle the opposite problem. Namely, inclusion-wise maximizing $R^+_{G-S}(X)$ among all ranged deletion separators (D, S). This leads to a definition similar to important separators.

Definition 3.20. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$ and let $\mathcal{C}$ be a graph class. An W-ranged $\mathcal{C}$-deletion $X \to Y$-separator (D, S) is said to* dominate *another W-ranged $\mathcal{C}$-deletion $X \to Y$-separator (D', S') if $R^+_{G-S}(X) \subseteq R^+_{G-S'}(X)$, $|(D, S)| \leq |(D', S')|$ and the inclusion or inequality is strict.*

We call an W-ranged $\mathcal{C}$-deletion $X \to Y$-separator important *if it is not dominated by any other W-ranged $\mathcal{C}$-deletion $X \to Y$-separator.*

We call two W-ranged $\mathcal{C}$-deletion $X \to Y$-separators (D, S) and (D', S') range equivalent *if $R^+_{G-S}(X) = R^+_{G-S'}(X)$. Let $\mathcal{DS}$ be some subset of the ranged $\mathcal{C}$-deletion $X \to Y$-separators of G. The* range equivalence classes *of $\mathcal{DS}$ are the equivalence classes of $\mathcal{DS}$ formed by the "range equivalence" equivalence relation.*

Note that for W being the empty set and $\mathcal{C}$ including the empty graph, this notion generalizes important $X \to Y$-separators. Also note that there may exist range equivalent important ranged deletion separators if there are several solutions to the W-Ranged $\mathcal{C}$-Deletion problem in $G[R^+_{G-S}(r)]$ of the same (minimum) size. We now want to find at least one important ranged deletion separators out of every range equivalent class. In this case we cannot simply push S towards Y (making it an important $X \to Y$-separator) as this may turn the W-Ranged $\mathcal{C}$-Deletion problem on $G[R^+_{G-S}(W)]$ unsolvable (and thus there is no ranged deletion separator containing this important separator). Our approach is to focus our attention on minimum $X \to Y$-separators S_{min} and depending on whether there is a ranged deletion separator (D, S_{min}) refine how potential important deletion separators could look like. The key tool to refine our search is the following lemma.

Lemma 3.21. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$ and let $\mathcal{C}$ be an hereditary graph class. Let $(D^\star, S^\star)$ be an important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator. If $S^\star$ is an $X \cup \{x\} \to Y \cup \{y\}$-separator for some $x \in R^+_{G-S^\star}(X)$ and $y \in V(G)$, then $(D^\star, S^\star)$ is also an important W-ranged $\mathcal{C}$-deletion $X \cup \{x\} \to Y \cup \{y\}$-separator.*

Proof. First note that we still have $W \subseteq X \cup \{x\}$. Suppose that the statement does not hold. Then either

- $S^\star$ is not an inclusion-wise minimal $X \cup \{x\} \to Y \cup \{y\}$-separator,
- $G[R^+_{G-(D^\star \cup S^\star)}(W)] \notin \mathcal{C}$, or
- $(D^\star, S^\star)$ is dominated by a W-ranged $\mathcal{C}$-deletion $X \cup \{x\} \to Y \cup \{y\}$-separator.

We can rule out $G[R^+_{G-(D^\star \cup S^\star)}(W)] \notin \mathcal{C}$ as $(D^\star, S^\star)$ is an important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator. Moreover, any $X \cup \{x\} \to Y \cup \{y\}$-separator is an $X \to Y$-separator. By inclusion-wise minimality of $S^\star$ as $X \to Y$-separator, this implies that it is inclusion-wise minimal as $X \cup \{x\} \to Y \cup \{y\}$-separator. Thus, $(D^\star, S^\star)$ is dominated by another W-ranged $\mathcal{C}$-deletion $X \cup \{x\} \to Y \cup \{y\}$-separator (D', S').

As S' is an inclusion-wise minimal $X \cup \{x\} \to Y \cup \{y\}$-separator, it includes an inclusion-wise minimal $X \to Y$-separator $S'' \subseteq S'$. Then for $D'' = D' \cup (S' \setminus S'')$, we have that $D' \cup S' = D'' \cup S''$, implying $G[R^+_{G-(D'' \cup S'')}(W)] = G[R^+_{G-(D' \cup S')}(W)] \in \mathcal{C}$.

Thus, (D'', S'') is a W-ranged $\mathcal{C}$-deletion $X \to Y$-separator. By choice of D'' we have that $|(D'', S'')| = |(D', S')| \leq |(D^\star, S^\star)|$. Suppose for a moment that $R^+_{G-S'}(X \cup \{x\}) = R^+_{G-S'}(X)$ holds. Then, we have by $x \in R^+_{G-S^\star}(X)$ that

$$R^+_{G-S^\star}(X) = R^+_{G-S^\star}(X \cup \{x\}) \subseteq R^+_{G-S'}(X \cup \{x\}) = R^+_{G-S'}(X) \subseteq R^+_{G-S''}(X).$$

Moreover, either this range inclusion or the size inequality is strict by (D', S') dominating $(D^\star, S^\star)$ as W-ranged $\mathcal{C}$-deletion $X \cup \{x\} \to Y \cup \{y\}$-separator. This then implies that (D'', S'') dominates $(D^\star, S^\star)$ as W-ranged $\mathcal{C}$-deletion $X \to Y$-separator. This would give us a contradiction to the importance of $(D^\star, S^\star)$ as W-ranged $\mathcal{C}$-deletion $X \to Y$-separator.

So it remains to show $R^+_{G-S'}(X \cup \{x\}) = R^+_{G-S'}(X)$. As $x \in R^+_{G-S^\star}(X)$ we have that there is an $X \to x$-path in $G - S^\star$ completely contained in $R^+_{G-S^\star}(X)$. By $R^+_{G-S^\star}(X) = R^+_{G-S^\star}(X \cup \{x\}) \subseteq R^+_{G-S'}(X \cup \{x\})$, this path also exists in $G - S'$ and thus $x \in R^+_{G-S'}(X)$. This then shows $R^+_{G-S'}(X \cup \{x\}) = R^+_{G-S'}(X)$. □

We first analyze the case where there is a ranged deletion separator (D, S_{min}).

Lemma 3.22. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$, let $\mathcal{C}$ be an hereditary graph class and let $k \in \mathbb{Z}_{\geq 0}$ be a non-negative integer. Let S_{min} be a minimum $X \to Y$-separator for which an W-ranged $\mathcal{C}$-deletion $X \to Y$-separator (D, S_{min}) of size at most k exists. Then for any important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator $(D^\star, S^\star)$ of size k, we have that either $S^\star = S_{min}$ or $(D^\star, S^\star)$ is an important W-ranged $\mathcal{C}$-deletion $X \cup \{v\} \to Y$-separator for some $v \in S_{min} \cap R^+_{G-S^\star}(X)$.*

Proof. Suppose for sake of contradiction that the statement does not hold. First note that $R^+_{G-S^\star}(X)$ cannot be a strict subset of $R^+_{G-S_{\mathsf{min}}}(X)$ as otherwise (D, S_{min}) would dominate $(D^\star, S^\star)$ (note that $|(D, S_{\mathsf{min}})| \leq k = |(D^\star, S^\star)|$), a contradiction to the importance of the latter. By inclusion-wise minimality of the $X \to Y$-separator in a W-ranged $\mathcal{C}$-deletion $X \to Y$-separator, we have that $S^\star = N^+(R^+_{G-S^\star}(X))$ and $S_{\mathsf{min}} = N^+(R^+_{G-S_{\mathsf{min}}}(X))$. Thus, the ranges cannot be equal, as then $S_{\mathsf{min}} = N^+(R^+_{G-S_{\mathsf{min}}}(X)) = N^+(R^+_{G-S^\star}(X)) = S^\star$, in contradiction to our assumption. This implies that we have some vertex $z \in R^+_{G-S^\star}(X) \setminus R^+_{G-S_{\mathsf{min}}}(X)$. For this vertex there is an $x \to z$-path P in $G - S^\star$ for some $x \in X$. Let v be the first vertex in $R^+_{G-S^\star}(X) \setminus R^+_{G-S_{\mathsf{min}}}(X)$ of P, which exists as z is such a vertex. Then $P[x, v]$ shows that $v \in N^+(R^+_{G-S_{\mathsf{min}}}(X)) = S_{\mathsf{min}}$.

If $Y = \emptyset$, we have that all minimum $X' \to Y$-separators are empty. Then all W-ranged $\mathcal{C}$-deletion $X' \to Y$-separator have the form $(D, \emptyset)$, implying $S_{\mathsf{min}} = S^\star$. Otherwise, apply Lemma 3.21 to $x = v$ and an arbitrary $y \in Y$. As $S^\star$ is an $X \cup \{v\} \to Y$-separator by $v \in R^+_{G-S^\star}(X)$, we get that $(D^\star, S^\star)$ is an important W-ranged $\mathcal{C}$-deletion $X \cup \{v\} \to Y$-separator. □

Lemma 3.23. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$, let $\mathcal{C}$ be an hereditary graph class and let $k \in \mathbb{Z}_{\geq 0}$ be a non-negative integer. Let S_{min} a minimum $X \to Y$-separator for which there is no W-ranged $\mathcal{C}$-deletion $X \to Y$-separator (D, S_{min}) of size at most k. Then for any important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator $(D^\star, S^\star)$ of size k, we have that $(D^\star, S^\star)$ is an important W-ranged $\mathcal{C}$-deletion $X \to Y \cup \{y\}$-separator for some $y \in S_{min}$.*

Proof. For the statement to hold, we have to show that $S^\star$ is an inclusion-wise minimal $X \to Y \cup \{y\}$-separator and that no W-ranged $\mathcal{C}$-deletion $X \to Y \cup \{y\}$-separator dominates $(D^\star, S^\star)$. We start by showing, that $S^\star$ is indeed an $X \to Y \cup \{y\}$-separator for some $y \in S_{\mathsf{min}}$. If there is a $y \in S_{\mathsf{min}} \setminus (S^\star \cup R^+_{G-S^\star}(X))$, we are done. Otherwise, we have that $S_{\mathsf{min}} \subseteq S^\star \cup R^+_{G-S^\star}(X)$, which implies $S_{\mathsf{min}} \cup R^+_{G-S_{\mathsf{min}}}(X) \subseteq S^\star \cup R^+_{G-S^\star}(X)$. Note that by inclusion-wise minimality of $S^\star$ and S_{min} as $X \to Y$-separators, we have $S^\star = N^+(R^+_{G-S^\star}(X))$ and $S_{\mathsf{min}} = N^+(R^+_{G-S_{\mathsf{min}}}(X))$. So we can subtract the out-neighborhood on both sides of the inclusion and get $R^+_{G-S_{\mathsf{min}}}(X) \subseteq R^+_{G-S^\star}(X)$. In this case Lemma 3.19 certifies us that there is a W-ranged $\mathcal{C}$-deletion $X \to Y$-separator (D, S_{min}) of size $\leq k$, a contradiction. So we have that $S^\star$ is indeed an $X \to Y \cup \{y\}$-separator for some $y \in S_{\mathsf{min}}$.

If $X = \emptyset$, we have that all minimum $X \to Y \cup \{y\}$-separators are empty. Then all W-ranged $\mathcal{C}$-deletion $X \to Y \cup \{y\}$-separator have the form $(D, \emptyset)$, implying $S_{\mathsf{min}} = S^\star$. But then there is W-ranged $\mathcal{C}$-deletion $X \to Y$-separator $(D^\star, S_{\mathsf{min}})$ of size $|D^\star| \leq k$, a contradiction. Otherwise, apply Lemma 3.21 to y and an arbitrary $x \in X$. As $S^\star$ is an $X \to Y \cup \{y\}$-separator, and by $x \in X \subseteq R^+_{G-S^\star}(X)$, we get that $(D^\star, S^\star)$ is an important W-ranged $\mathcal{C}$-deletion $X \to Y \cup \{y\}$-separator. □

With these lemmas in place we can formulate our main result.

Theorem 3.24. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$, let $\mathcal{C}$ be an hereditary graph class and let $k \in \mathbb{Z}_{\geq 0}$ be a non-negative integer. Then there are at most $4^{k \log k}$ range equivalence classes among the important W-ranged $\mathcal{C}$-deletion $X \to Y$-separators of size exactly k.*

Moreover, for an instance (G, W, X, Y, k) we can find a representative of every range equivalence class of important W-ranged $\mathcal{C}$-deletion $X \to Y$-separators of size exactly k in time

$$4^{k \log k} \operatorname{poly}(n) + 4^{k \log k} n \cdot A_{\mathcal{C}\text{-Deletion}},$$

where $A_{\mathcal{C}\text{-Deletion}}$ is the maximum run-time of an oracle for W-RANGED $\mathcal{C}$-DELETION over all subgraphs $G[R^+_{G-S}(W)]$ of G, where S is some $X \to Y$-separator.

Note that the maximum pf the oracle run-times is well defined as we are maximising over a finite number of choices for S. The idea for the proof is to take an inclusion-wise maximum chain of minimum $X \to Y$-separators and use the position where ranged separators stop existing to refine the search by adding vertices to X and Y by the help of Lemma 3.21.

Proof of Theorem 3.24. We are going to prove the following stronger statement instead. The number of range equivalence classes of the important W-ranged $\mathcal{C}$-deletion $X \to Y$-separators of size exactly k in a graph G where the minimum $X \to Y$-cut has size $\lambda_G(X, Y)$ is at most

$$\begin{cases} 0 & \text{if } k < \lambda_G(X,Y), \\ \left((\lambda_G(X,Y))^2 + 1\right)^{k - \lambda_G(X,Y)} & \text{otherwise.} \end{cases}$$

This implies the original statement as for $k < \lambda_G(X, Y)$, we have that $0 \leq 4^{k \log k}$, for $k = \lambda_G(X, Y)$, we have $\left((\lambda_G(X,Y))^2 + 1\right)^{k - \lambda_G(X,Y)} = 1 \leq 4^{k \log k}$ and for $k > \lambda_G(X, Y)$, we have $\left((\lambda_G(X,Y))^2 + 1\right)^{k - \lambda_G(X,Y)} \leq (k^2)^k = 4^{k \log k}$.

We are going to prove the statement by induction over $k - \lambda_G(X, Y)$. However, before we begin the induction we take care of the case $\lambda_G(X, Y) = 0$. As for any W-ranged $\mathcal{C}$-deletion $X \to Y$-separator (D, S) we must choose S as inclusion-wise minimal $X \to Y$-separator and the empty set is an $X \to Y$-separator, we have that all ranged deletion separators have the form $(D, \emptyset)$. Thus, for any possible range deletion separator we have that the range with respect to S is $R^+_{G-\emptyset}(X)$ and thus they are all range equivalent. We can check whether such a deletion separator exists by calling $A_{\mathcal{C}\text{-Deletion}}$ on $(G[R^+_G(W)], k, W)$ to find a solution (and on $(G[R^+_G(W)], k-1, W)$ to rule out smaller solutions).

We now move on to the induction over $k - \lambda_G(X, Y)$. If $k - \lambda_G(X, Y) < 0$, we have that $k < \lambda_G(X, Y)$ and thus there are no W-ranged $\mathcal{C}$-deletion $X \to Y$-separators of size at most k. This shows the base case.

Now assume that the statement holds for all (X', Y', k') with $k' - \lambda_G(X', Y') < k - \lambda_G(X, Y)$. As we already covered the case $\lambda_G(X, Y) = 0$, we can assume $\lambda_G(X, Y) > 0$. Now take an inclusion-wise maximal chain of minimum $X \to Y$-separators, i.e. inclusion-wise maximal family of vertex sets $X \subseteq U_1 \subseteq \ldots \subseteq U_p \subseteq V(G) \setminus Y$, such that $S_i = N^+(U_i)$ is an $X \to Y$-separator of size $|S_i| = \lambda_G(X, Y)$.

We now call for each S_i our algorithm $A_{\mathcal{C}\text{-Deletion}}$ on $(G[R^+_{G-S_i}(W)], k - |S_i|, W)$. If it finds a solution D_i for some S_i, we know that (D_i, S_i) is a W-ranged $\mathcal{C}$-deletion $X \to Y$-separator. There is either a solution (D_p, S_p), no solution of the form (D_1, S_1), or an index $1 \leq i < p$ such that there is a solution (D_i, S_i), but no solution for $i + 1$.

Case 1: There is a solution (D_p, S_p).

Consider the minimum $X \to Y$-separator S_p. By Lemma 3.22 we have that for any important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator $(D^\star, S^\star)$ of size k either $S^\star = S_p$ holds or $(D^\star, S^\star)$ is an important W-ranged $\mathcal{C}$-deletion $X \cup \{x\} \to Y$-separator for some $x \in S_p$. If $S^\star = S_p$ we have that (D_p, S_p) is a range equivalent solution and we recover it. Otherwise, we branch on the vertices $x \in S_p$. Note that for every $x \in S_p$, we have that a minimum $X \cup \{x\} \to Y$-separator $S_{\mathsf{min}} = N^+(U_{\mathsf{min}})$ is also an $X \to Y$-separator. By uncrossing we get that

$$|S_{\mathsf{min}}| + |S_p| \geq |N^+(U_{\mathsf{min}} \cup U_p)| + |N^+(U_{\mathsf{min}} \cap U_p)|.$$

As $N^+(U_{\mathsf{min}} \cap U_p)$ is an $X \to Y$-separator and S_p is a minimum $X \to Y$-separator, we can subtract them from their sides and get

$$|S_{\mathsf{min}}| \geq |N^+(U_{\mathsf{min}} \cup U_p)|.$$

Now $N^+(U_{\mathsf{min}} \cup U_p)$ is an $X \to Y$-separator with $U_{\mathsf{min}} \cup U_p \supseteq U_p \cup \{x\} \supsetneq U_p$.

By inclusion-wise maximality of our minimum $X \to Y$-separator chain, we have that $N^+(U_{\mathsf{min}} \cup U_p)$ is not a minimum separator and thus has size at least $\lambda_G(X, Y) + 1$. This implies $\lambda_G(X \cup \{x\}, Y) = |S_{\mathsf{min}}| \geq \lambda_G(X, Y) + 1$. Thus, we can find an important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator range equivalent to $(D^\star, S^\star)$ by finding one of every range equivalent class of important W-ranged $\mathcal{C}$-deletion $X \cup \{x\} \to Y$-separators. For these we have $k - \lambda_G(X \cup \{x\}, Y) < k - \lambda_G(X, Y)$ and we can apply induction.

Case 2: There is no solution of the form (D_1, S_1).

Consider the minimum $X \to Y$-separator S_1. By Lemma 3.23 for any important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator $(D^\star, S^\star)$ we have that $(D^\star, S^\star)$ is an important

W-ranged $\mathcal{C}$-deletion $X \to Y \cup \{y\}$-separator for some $y \in S_1$. We branch on the vertices $y \in S_1$. Note that for every $y \in S_1$, we have that a minimum $X \to Y \cup \{y\}$-separator $S_{\mathsf{min}} = \delta(U_{\mathsf{min}})$ is also an $X \to Y$-separator. By uncrossing we get that

$$|S_{\mathsf{min}}| + |S_1| \geq |N^+(U_{\mathsf{min}} \cup U_1)| + |N^+(U_{\mathsf{min}} \cap U_1)|.$$

As $N^+(U_{\mathsf{min}} \cup U_1)$ is an $X \to Y$-separator and S_1 is a minimum $X \to Y$-separator, we can subtract them from their sides and get

$$|S_{\mathsf{min}}| \geq |N^+(U_{\mathsf{min}} \cap U_1)|.$$

Now $N^+(U_{\mathsf{min}} \cap U_1)$ is an $X \to Y$-separator with $U_{\mathsf{min}} \cap U_1 \subseteq U_1 \setminus \{y\} \subsetneq U_1$. By inclusion-wise maximality of our minimum $X \to Y$-separator chain, we have that $N^+(U_{\mathsf{min}} \cap U_1)$ is not a minimum separator and thus has size at least $\lambda_G(X, Y) + 1$. This implies $\lambda_G(X, Y \cup \{y\}) = |S_{\mathsf{min}}| \geq \lambda_G(X, Y) + 1$. Thus, we can find an important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator range equivalent to $(D^\star, S^\star)$ by finding one of every range equivalent class of important W-ranged $\mathcal{C}$-deletion $X \to Y \cup \{y\}$-separators. For these we have $k - \lambda_G(X, Y \cup \{y\}) < k - \lambda_G(X, Y)$ and we can apply induction.

Case 3: For some $i \in \{1, \ldots, p-1\}$ there is a solution (D_i, S_i) but no solution of the form (D_{i+1}, S_{i+1}).

We want to recover a fixed important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator $(D^\star, S^\star)$ or an important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator range equivalent to it.

Consider the minimum $X \to Y$-separator S_i. By Lemma 3.22 we either have $S^\star = S_i$ or $(D^\star, S^\star)$ is an important W-ranged $\mathcal{C}$-deletion $X \cup \{x\} \to Y$-separator for some $x \in S_i \cap R^+_{G-S^\star}(X)$. If $S^\star = S_i$ we have that (D_i, S_i) is a range equivalent solution and we recover it. Otherwise, we branch on the vertices $x \in S_i$. For the right choice of x we have $x \in S_i \cap R^+_{G-S^\star}(X)$ implying $R^+_{G-S^\star}(X \cup \{x\}) = R^+_{G-S^\star}(X)$.

Consider the minimum $X \to Y$-separator S_{i+1}. By Lemma 3.23 we have that $(D^\star, S^\star)$ is an important W-ranged $\mathcal{C}$-deletion $X \to Y \cup \{y\}$-separator for some $y \in S_{i+1}$. We branch on the vertices $y \in S_{i+1}$, for $|S_i| \cdot |S_{i+1}| = (\lambda_G(X, Y))^2$ branches total.

Now $S^\star$ is an $X \cup \{x\} \to Y \cup \{y\}$-separator. Then Lemma 3.21 implies that $(D^\star, S^\star)$ is an important W-ranged $\mathcal{C}$-deletion $X \cup \{x\} \to Y \cup \{y\}$-separator.

Consider a minimum $X \cup \{x\} \to Y \cup \{y\}$-separator $S_{\mathsf{min}} = \delta(U_{\mathsf{min}})$. Note that any $X \cup \{x\} \to Y \cup \{y\}$-separator is an $X \to Y$-separator. By uncrossing twice we get that

$$\begin{aligned}
&|S_{\mathsf{min}}| + |S_i| + |S_{i+1}| \\
&\geq |N^+(U_{\mathsf{min}} \cup U_i)| + |N^+(U_{\mathsf{min}} \cap U_i)| + |S_{i+1}| \\
&\geq |N^+((U_{\mathsf{min}} \cup U_i) \cap U_{i+1})| + |N^+(U_{\mathsf{min}} \cup U_i \cup U_{i+1})| + |N^+(U_{\mathsf{min}} \cap U_i)|.
\end{aligned}$$

As $N^+(U_{\mathsf{min}} \cap U_i)$ and $N^+(U_{\mathsf{min}} \cup U_i \cup U_{i+1})$ are $X \to Y$-separators and S_i and S_{i+1} are minimum $X \to Y$-separators, we can subtract them from their sides and get

$$|S_{\mathsf{min}}| \geq |N^+((U_{\mathsf{min}} \cup U_i) \cap U_{i+1})|.$$

Now $N^+((U_{\mathsf{min}} \cup U_i) \cap U_{i+1})$ is an $X \to Y$-separator with

$$U_i \subsetneq U_i \cup \{x\} \subseteq (U_{\mathsf{min}} \cup U_i) \cap U_{i+1}) \subseteq U_{i+1} \setminus \{y\} \subsetneq U_{i+1}.$$

By inclusion-wise maximality of our minimum $X \to Y$-separator chain, we have that $N^+((U_{\mathsf{min}} \cup U_i) \cap U_{i+1})$ is not a minimum $X \to Y$-separator and thus has size at least $\lambda_G(X,Y)+1$. This implies $\lambda_G(X \cup \{x\}, Y \cup \{y\}) = |S_{\mathsf{min}}| \geq \lambda_G(X,Y)+1$. Therefore, we can find an important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator which is range equivalent to $(D^\star, S^\star)$ by finding one of every range equivalent class of important W-ranged $\mathcal{C}$-deletion $X \cup \{x\} \to Y \cup \{y\}$-separators. Here we can apply induction by

$$k - \lambda_G(X \cup \{x\}, Y \cup \{y\}) < k - \lambda_G(X,Y).$$

Note that in every of these cases we recovered at most one ranged deletion separator and created at most $(\lambda_G(X,Y))^2$ branches with decreased $k - \lambda_G(X',Y')$. By above argument we have recovered an important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator out of every range equivalence class. So it remains to bound number of recovered ranged deletion separators and number of oracle calls. Note that the number of oracle calls is bounded by the number of recovered ranged deletion separators times n. Thus, we try to bound the number of recovered ranged deletion separators.

If $k = \lambda_G(X,Y)$ each of the branches had $k - \lambda_G(X',Y') < 0$ and thus returned no ranged deletion separator. In this case our upper bound on recovered range deletion separators is $1 = \left((\lambda_G(X,Y))^2+1\right)^0 = \left((\lambda_G(X,Y))^2+1\right)^{k-\lambda_G(X,Y)}$, so our induction hypothesis holds. Otherwise, by induction, the number recovered range deletion separators is at most

$$1 + (\lambda_G(X,Y))^2 \cdot \Big((\lambda_G(X,Y)-1)^2+1\Big)^{k-\lambda_G(X,Y)-1}.$$

By $(\lambda_G(X,Y)-1)^2+1 \geq 1$ and $k - \lambda_G(X,Y) - 1 \geq 0$, we have that

$$\Big((\lambda_G(X,Y)-1)^2+1\Big)^{k-\lambda_G(X,Y)-1} \geq 1.$$

This implies

$$\begin{aligned}
&1 + (\lambda_G(X,Y))^2 \cdot \Big((\lambda_G(X,Y)-1)^2+1\Big)^{k-\lambda_G(X,Y)-1} \\
&\leq \Big((\lambda_G(X,Y))^2+1\Big) \cdot \Big((\lambda_G(X,Y)-1)^2+1\Big)^{k-\lambda_G(X,Y)-1} \\
&\leq \Big((\lambda_G(X,Y))^2+1\Big)^{k-\lambda_G(X,Y)}.
\end{aligned}$$

Finally, note that we are only passing subgraphs $G[R^+_{G-S}(W)]$ of G to our oracle, where S is some $X \to Y$-separator. □

3.2 The Algorithm

In this section we will present our fixed-parameter algorithm for hitting all long cycles of the input directed graph. Recall the formal problem definition:

Directed Long Cycle Hitting Set	
Instance:	A graph G and two integers $k, \ell \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq V(G)$ of size at most k such that every directed cycle of $G - S$ has length at most ℓ or decide that no such set exists.

3.2.1 Outline of the Algorithm

Our algorithm performs a sequence of reductions to special cases of the original Directed Long Cycle Hitting Set problems. Each reduction is presented in its own section. All these sections are modular and just need the problem formulation at the end of the previous section to understand. Furthermore, every section ends with a theorem summarizing the findings of the section.

The overall algorithm can be described as follows: The first section starts with the standard arguments of disjoint compression. This allows us to solve the easier problem where we are already given a solution T and search for a smaller solution S. Moreover, we discuss how to verify solution such that it suffices to find a bounded size set intersecting some solution. In the second section, Section 3.2.3, we then lay the crucial foundation of our algorithm. By applying a graph structural contraction argument we reduce to instances where our given solution T has at most one vertex in every strongly connected component of $G - S$.

The resulting problem bears similarities to the Skew Separator problem which appears naturally in the DFVS algorithm by Chen et al. [CLL+08]. There it was solved by a pushing argument involving important separators. For us important separators will not do the trick, leading to the definition of hitting separators. We use the technique of W-ranged $\mathcal{C}$-deletion separators to reduce to the special case of the Directed Long Cycle Hitting Set compression variant, where T has size exactly one.

Afterwards, in Section 3.2.5, we use this together with the shadow covering technique to reduce to strongly connected, directed graphs where every arc lies on a short cycle (and still $|T| = 1$). Eventually, in Section 3.2.6, we consider the structure of strongly connected components of $G - T$ to reduce to the Directed Multiway Cut which is known to be solvable in $2^{\mathcal{O}(k^2)} \operatorname{poly}(n)$.

3.2.2 Compression Intersection

In this subsection we apply the general techniques of disjoint compression and compression intersection. This way we get instances where we are given an existing solution T for which we have to find a disjoint solution S. Moreover, we only have to find some set of bounded size, which intersects such solutions.

In order to apply disjoint compression we have to check whether Directed Long Cycle Hitting Set is a $\mathcal{C}$-Vertex Deletion problem (see Chapter 1 for the def-

inition) with a non-empty and hereditary graph class $\mathcal{C}$. The obvious graph class $\mathcal{C}_\ell$ for Directed Long Cycle Hitting Set is the class of all graphs G with $\mathrm{cf}(G) \leq \ell$. The empty graph has no cycles and thus by definition circumference 0. That means $\mathcal{C}_\ell$ contains the empty graph for all $\ell \in \mathbb{Z}_{\geq 0}$ and thus is non-empty. To see that $\mathcal{C}_\ell$ is hereditary, note that for every subgraph H of a graph $G \in \mathcal{C}_\ell$, we have that all cycles of H exist in G and thus $\mathrm{cf}(H) \leq \mathrm{cf}(G) \leq \ell$. By $\mathcal{C}_\ell$ being non-empty and hereditary, we can apply Corollary 2.3 (see Chapter 2) to reduce to the following problem.

Disjoint Directed Long Cycle Hitting Set Compression	
Instance:	A graph G, a vertex set $T \subseteq V(G)$ and two integers $k, \ell \in \mathbb{Z}_{\geq 0}$ s.t. $\mathrm{cf}(G - T) \leq \ell$.
Task:	Find a set $S \subseteq V(G) \setminus T$ with $\lvert S\rvert \leq k$ and $\mathrm{cf}(G - S) \leq \ell$ or decide that no such set exists.

The reduction procedure is summarized in the following lemma.

Lemma 3.25. *An instance (G, k, ℓ) of Directed Long Cycle Hitting Set can be solved in time $\mathcal{O}(n \cdot 2^{k+1} \cdot A_{\textsf{disjoint compression}}(n, k+1, k))$, where $A_{\textsf{disjoint compression}}(n, t, k)$ is the run-time of an algorithm for Disjoint Directed Long Cycle Hitting Set Compression on instances (G', T', k') with $\lvert V(G')\rvert \leq n$, $\lvert T\rvert \leq t$ and $k' \leq k$.*

Proof. Apply Corollary 2.3 to $\mathcal{C}_\ell$ which is hereditary and non-empty. □

In the next step we want to apply the compression intersection technique (cf. Chapter 2, Lemma 2.4), where we do not need to find a solution but only need to find a bounded size set which intersects some solution (given there exists a solution). This allows us to use statements of the type "either S intersects X or our instance has property Y" in the following way. If we can solve the intersection variant on instances with property Y by returning some set $\mathcal{S}_{\textsf{intersect}}$, we can solve our original problem by returning $\mathcal{S}_{\textsf{intersect}} \cup X$. Let us formulate the intersection variant.

Disjoint Long Cycle Hitting Set Compression Intersection	
Instance:	A graph G, a vertex set $T \subseteq V(G)$ and two integer $k, \ell \in \mathbb{Z}_{\geq 0}$ s.t. $\mathrm{cf}(G - T) \leq \ell$.
Task:	Find a vertex set $\mathcal{S}_{\textsf{intersect}} \subseteq V(G)$ such that if there is a set $S \subseteq V(G) \setminus T$ with $\lvert S\rvert \leq k$ and $\mathrm{cf}(G - S) \leq \ell$, then there is such a set S with $S \cap \mathcal{S}_{\textsf{intersect}} \neq \emptyset$.

In order to apply Lemma 2.4, we need an algorithm to check whether a graph G fulfills $\mathrm{cf}(G) \leq \ell$. For this we use a fixed-parameter algorithm by Zehavi [Zeh16].

Theorem 3.26 ([Zeh16])**.** *There is an algorithm that decides in time $2^{\mathcal{O}(\ell)} \cdot \mathrm{poly}(n)$ whether an n-vertex directed graph G contains a cycle of length more than ℓ.*

Now the disjoint compression intersection technique results in the following lemma.

Lemma 3.27. *Directed Long Cycle Hitting Set can be solved in time*

$$\mathcal{O}(2^{\mathcal{O}(\ell)}\operatorname{poly}(n)+n\cdot k2^{k+1}(f_{\mathsf{intersect}}(n,|T|,k,\ell))^k\cdot A_{\mathsf{disjoint\ compression\ intersection}}(n,|T|,k,\ell)),$$

where $A_{\mathsf{disjoint\ compression\ intersection}}(n,t,k,\ell)$ is the run-time of an algorithm for Disjoint Directed Long Cycle Hitting Set Compression Intersection on instances (G',T',k') with $|V(G')|\leq n$, $|T|\leq t$ and $k'\leq k$ and $f_{\mathsf{intersect}}(n,t,k,\ell)$ is a size bound on the set $\mathcal{S}_{\mathsf{intersect}}$ output by the algorithm on these instances.

Proof. Apply Lemma 2.4 with the membership oracle of Theorem 3.26 to Lemma 3.25. □

3.2.3 Isolation by Contraction

Except for the intersection step, the reductions of the last section have been used by Chen et al. [CLL+08] in their algorithm for Directed Feedback Vertex Set. The next key observation in their algorithm for DFVS is that every vertex of T must lie in their own strongly connected component of $G-S$. The reason is that for every directed feedback vertex set S of G, each strongly connected component of $G-S$ is a single vertex. For Directed Long Cycle Hitting Set, the situation is way more complicated, as strongly connected components of $G-S$ contain cycles of length up to ℓ. Moreover, those cycles can concatenate to arbitrarily large strongly connected components. So it is possible that $G-S$ contains strongly connected components with more than one vertex of T.

We want to restrict our solution space to solutions S such that every strongly connected component of $G-S$ contains at most one vertex of T. We call such solutions isolating long cycle hitting sets.

Definition 3.28. *Let G be a directed graph, $\ell\in\mathbb{Z}_{\geq 0}$ and $T\subseteq V(G)$ be some subset of vertices. A vertex set $U\subseteq V(G)\setminus T$ is called* isolating long cycle hitting set *(with respect to T) if $\operatorname{cf}(G-U)\leq\ell$ and every strongly connected component contains at most one vertex of T.*

To reduce to this solution space, we want to contract (parts of) strongly connected components with more than one vertex of T to a single vertex. Eventually, after several contractions, every strongly connected component of $G-S$ contains at most one vertex of T. A structural result allowing for such contractions is the following lemma:

Lemma 3.29. *Let G be a directed graph and let $X\subseteq V(G)$ be such that $G[X]$ is strongly connected and $\operatorname{cf}(G[X])\leq\ell$. Suppose that the following two properties hold:*

1. *Every cycle of G has length at most ℓ or length at least ℓ^2.*
2. *For any $a,b\in X$ there cannot be both*
   (a) *an $a\to b$-path P_{ab} of length at least ℓ in $G[X]$*
   (b) *a $b\to a$-path P_{ba} of length at most ℓ in $G-(X\setminus\{a,b\})$*

Let G/X be the directed graph obtained by contracting X to a single vertex x.

- *If $\operatorname{cf}(G-S)\leq\ell$ for some $S\subseteq V(G)\setminus X$, then $\operatorname{cf}(G/X-S)\leq\ell$.*
- *If $\operatorname{cf}(G/X-S')\leq\ell$ for some $S'\subseteq V(G/X)\setminus\{x\}$, then $\operatorname{cf}(G-S')\leq\ell$.*

Proof. For Statement 1, suppose that $G/X - S$ has a cycle C of length more than ℓ. If C does not go through x, then C is a cycle of G disjoint from S. Otherwise, if C goes through x, then the arcs of C correspond to a walk of G going from some vertex $x_1 \in X$ to a vertex $x_2 \in X$ and having length more than ℓ. If $x_1 = x_2$ then this walk is a cycle of length more than ℓ in $G - S$, a contradiction. Suppose therefore that $x_1 \neq x_2$, in which case this walk is a simple path P. As $G[X]$ is strongly connected, there is an $x_2 \to x_1$-path Q in $G[X]$. The paths P and Q create a cycle in G that is disjoint from S and has length more than ℓ, a contradiction.

For Statement 2, suppose that $G - S'$ has a cycle C of length greater than ℓ. Let us choose C such that it has the minimum number of vertices outside X. By assumption, cycle C cannot be fully contained in X. If C is disjoint from X, then C is a cycle of G/X disjoint from S', a contradiction. If C contains exactly one vertex of X, then there is a corresponding cycle C' in the contracted directed graph with the same length and disjoint from S', a contradiction. Assume therefore that C contains more than one vertex of X; let P be an $x_1 \to x_2$-subpath of C with both endpoints in X and no internal vertex in X. If P has length more than ℓ, then there is a corresponding cycle C' in the contracted directed graph with the same length and disjoint from S', a contradiction. Let v be an arbitrary internal vertex of P and let $G^\star = G[X \cup V(C) \setminus \{v\})]$. By the minimality of the choice of C, we have $\mathrm{cf}(G^\star) \leq \ell$. As $G^\star[X] = G[X]$ is strongly connected, it contains an $x_2 \to x_1$-path P_1. Also, the subpath P_2 of C from x_1 to x_2 is in $G^\star$. By property (P1), the length of C is at least ℓ^2, hence $|P_2| = |C| - |P| \geq \ell^2 - \ell$. Thus, Lemma 3.9 implies that $|P_1| \geq |P_2|/(\ell - 1) \geq \ell$. But then P and P_1 contradict property (2). □

To algorithmically use this result we have to make sure its requirements are fulfilled. This is easy if G has a cycle C with $\ell < |C| < \ell^2$. We can detect these cycles in time $2^{\mathcal{O}(\ell^2)} \cdot \mathrm{poly}(n)$ by using color coding (see Alon et al. [AYZ95]).

Theorem 3.30 ([AYZ95])**.** *A simple directed cycle of length p in a directed graph G can be found in time $2^{\mathcal{O}(p)}\mathcal{O}(nm)$.*

Note that it is crucial for the reported cycles to have length bounded in ℓ. This is the reason we cannot make use of the algorithm by Zehavi (Theorem 3.26) as the cycle reported there may have length $\Omega(n)$. If there exists a cycle of length $> \ell$, any long cycle hitting set has to intersect it. Also, its length is bounded, so we can return it as set $\mathcal{S}_{\mathsf{intersect}}$. We need an even larger gap between short and long cycles later on. We call the cycles we want to delete now "medium-length cycles".

Definition 3.31. *For an integer $\ell \in \mathbb{Z}_{\geq 0}$ and a directed graph G, a cycle C in G is called* medium-length cycle *if the length of C fulfills $\ell < |\ell(C)| < 2\ell^6$.*

The other requirement to use Lemma 3.29 is to find a set X with the remaining properties. To find a suitable set X we build on a tool called "k-representative set of paths" which has already been introduced in the technical tools section. We will use the following variant specialized to our needs, which was already proven in the technical tools section.

Lemma 3.17. *Let G be a strongly connected, directed graph, $T \subseteq V(G)$ and $k, \ell \in \mathbb{Z}_{\geq 0}$ with $\mathrm{cf}(G - T) \leq \ell$. Then in time $2^{\mathcal{O}(k\ell + k^2 \log k)}\,\mathrm{poly}(n)$, we can find a set $\mathcal{Q}$ of $|T|^2 2^{\mathcal{O}(k\ell + k^2 \log k)} \log^2 n$ closed walks with the following property:*

If there is a set $S \subseteq V(G)$ of size at most k with $\mathrm{cf}(G - S) \leq \ell$ and there are two vertices of T in the same strongly connected component of $G - S$ then there is a closed walk in $Q \in \mathcal{Q}$ containing two vertices of T that is disjoint from S.

We now use Lemma 3.17 to find sets suitable for Lemma 3.29. Ideally, we would like to compute $\mathcal{Q}$ as in Lemma 3.17 for our directed graph G and vertex set T and use the walks inside $\mathcal{Q}$. Alas, we cannot use Lemma 3.29 directly, as the second condition may not be fulfilled. We handle this issue via the following lemma:

Lemma 3.32. *Let $I = (G, k, \ell, T)$ be an instance of Disjoint Directed Long Cycle Hitting Set Compression Intersection. There are two set $\mathcal{S}_{\mathsf{intersect}}$ and $\mathcal{X}$ with the following properties. If I has a solution then either*

- *I contains an isolating long cycle hitting set, or*
- *a solution of I intersects $\mathcal{S}_{\mathsf{intersect}}$, or*
- *there is an $X \in \mathcal{X}$ such that*
  - *$G[X]$ is strongly connected, $\mathrm{cf}(G[X]) \leq \ell$, and $|X \cap T| \geq 2$.*
  - *For any $a, b \in X$ there cannot be both*
    1. *an $a \to b$-path P_{ab} of length at least ℓ in $G[X]$,*
    2. *a $b \to a$-path P_{ba} of length at most ℓ in $G - (X \setminus \{a, b\})$*

Moreover, we have $|\mathcal{S}_{\mathsf{intersect}}|, |\mathcal{X}| \leq |T|^2 2^{\mathcal{O}(k\ell + k^2 \log k)} \log^2 n$ and the sets can be computed in $2^{\mathcal{O}(k\ell + k^2 \log k)} \cdot \mathrm{poly}(n)$ time.

Proof. To prove correctness of the algorithm we are about to construct, assume that the instance (G, k, ℓ, T) has a solution but does not have an isolating long cycle hitting set, i.e. for every solution S one of the components of $G - S$ contains two vertices of T. Otherwise the algorithm may return any sets $|\mathcal{S}_{\mathsf{intersect}}|, |\mathcal{X}|$ that fulfill the size bounds.

Our algorithm works in two steps. First we create our set $\mathcal{X}$ that consists of closed walks by taking the set $\mathcal{Q}$ produced by Lemma 3.17. In a second step we make sure that the walks in $\mathcal{X}$ fulfill the properties we need them to have, by adding vertices to $\mathcal{S}_{\mathsf{intersect}}$ if this is not the case.

Since we assume that $G - S$ contains a strongly connected component with two vertices of T, we know that there is a closed walk $X \in \mathcal{X}$ that contains at least two vertices of T and is disjoint from S. By definition, this walk lies inside one strongly connected component of $G - S$ and thus has to fulfill $\mathrm{cf}(G[X]) \leq \ell$.

Assume in the following we have picked the right walk X. Our algorithm cannot distinguish this walk and thus applies the procedure to every $X \in \mathcal{X}$.

First we remove a walk X if $\mathrm{cf}(G[X]) > \ell$. It remains to ensure that no paths P_{ab} and P_{ba} like in the theorem statement exist. Our algorithm then checks for every $a, b \in X$ whether an $a \to b$-path P_{ab} in $G[X]$ of length at least ℓ exists and whether a $b \to a$-path P_{ba} in $G - (X \setminus \{a, b\})$ of length at most ℓ exists. This again can be done by standard color-coding techniques (see Alon et al. [AYZ95]).

If for some $a, b \in X$ both paths exists, the closed walk $W = P_{ab} \circ P_{ba}$ is in fact a cycle, as the paths only intersect in a and b. As P_{ab} has length at least ℓ, the cycle W has to be intersected by our solution S. Moreover, $V(P_{ab}) \subseteq X \subseteq V(G) \setminus S$, thus we know that S has to intersect $V(P_{ba})$, which has size at most ℓ. Thus, we can add $V(P_{ba})$ to $\mathcal{S}_{\mathsf{intersect}}$ and continue with the next set $X \in \mathcal{X}$. This completes the description and correctness of the algorithm.

The size bounds can be seen as follows. A bound of $|T|^2 2^{\mathcal{O}(k\ell+k^2\log k)}\log^2 n$ on $|\mathcal{X}|$ follows directly from Lemma 3.17. The only things we added to $\mathcal{S}_{\mathsf{intersect}}$ were vertex sets of size $\leq \ell$. This we did once in the first step and once for every element of $\mathcal{X}$. Thus, the same size bound applies, as the factor of ℓ can be incorporated into the exponent.

The run-time can be divided in those of the color coding techniques (which take $2^{\mathcal{O}(\ell)}\operatorname{poly}(n)$ time) applied to every $X \in \mathcal{Q}$ and the run-time of Lemma 3.17. This gives a combined run-time of $2^{\mathcal{O}(k\ell+k^2\log k)} \cdot \operatorname{poly}(n)$. □

We may use the previous lemma to find sets suitable for contraction by Lemma 3.29. Note that these contractions reduce the size of T without affecting the size of S. Therefore, it might occur that T is smaller than the sought after S. But as we search for a disjoint solution we may not return T as solution.

Now we combine Lemma 3.32, Lemma 3.29, and the elimination of medium-length cycles into a single algorithm. Afterwards we will be left with the following problem:

Isolating Long Cycle Hitting Set Intersection

Instance: A graph G, a vertex set $T \subseteq V(G)$ and two integer $k, \ell \in \mathbb{Z}_{\geq 0}$ s.t.
- G has no medium-length cycles,
- $\operatorname{cf}(G-T) \leq \ell$.

Task: Find a vertex set $\mathcal{S}_{\text{intersect}} \subseteq V(G)$ such that
if there is an isolating long cycle hitting set $S \subseteq V(G) \setminus T$,
then there is such a set S with $S \cap \mathcal{S}_{\mathsf{intersect}} \neq \emptyset$.

The resulting algorithm using oracle calls to an algorithm for Isolating Long Cycle Hitting Set Intersection is summarized in the following lemma. Note that the size bound on the set returned by the oracle does only contribute to the size of the new solution and not to the run-time of the algorithm.

Lemma 3.33. *An instance (G,T,k,ℓ) of Disjoint Directed Long Cycle Hitting Set Compression Intersection can be solved in time*

$$\left(|T|^{2|T|}2^{\mathcal{O}(\ell^6+|T|k\ell+|T|k^2\log k)}\log^{2|T|} n\right)\left(A_{\mathsf{isolating}}(n,|T|,k,\ell)+\operatorname{poly}(n)\right),$$

where $A_{\mathsf{isolating}}(n,t,k,\ell)$ denotes the run-time of an algorithm for Isolating Long Cycle Hitting Set Intersection that is called on instances (G',T',k',ℓ) with $|V(G')| \leq n$, $|T| \leq t$ and $k' \leq k$.

If $f_{\mathsf{isolating}}(n,t,k,\ell)$ is a size bound on the set $\mathcal{S}_{\mathsf{isolating}}$ output by the Isolating Long Cycle Hitting Set Intersection algorithm on these instances, then the set $\mathcal{S}_{\mathsf{intersect}}$ returned has size at most

$$\left(|T|^{2|T|}2^{\mathcal{O}(|T|k\ell+|T|k^2\log k)}\log^{2|T|} n\right)\left(2\ell^6+f_{\mathsf{isolating}}(n,|T|,k,\ell)\right).$$

Proof. If (G,T,k,ℓ) has no long cycle hitting set, we may return any set that fulfills the size bounds. So assume while arguing for correctness that (G,T,k,ℓ) has a long cycle hitting set.

Starting with $\mathcal{S}_{\mathsf{intersect}} = \emptyset$, we call the following recursive procedure on (G, T, k, ℓ). Let $I' = (G', T', k, \ell)$ the instance in one of the recursive calls. To argue correctness, we want that if I' has a long cycle hitting set that the final set $\mathcal{S}_{\mathsf{intersect}}$ intersects it. As we do the initial call with (G, T, k, ℓ), this shows correctness.

We start by checking our instance for medium-length cycles via Theorem 3.30. For every $\ell < p \leq 2\ell^6$, we call the algorithm of Theorem 3.30. If it finds a cycle C, we add $V(C) \setminus T'$ to $\mathcal{S}_{\mathsf{intersect}}$ and return early. This is correct, as any long cycle hitting set of I' needs to contain a vertex of $V(C) \setminus T'$.

Otherwise, G' has no medium-length cycles. We call $A_{\mathsf{isolating}}$ on (G', T', k, ℓ) and add the result to $\mathcal{S}_{\mathsf{intersect}}$. So if the instance I' has an isolating long cycle hitting set, $\mathcal{S}_{\mathsf{intersect}}$ intersects it. Note, that if $|T'| \leq 1$, any long cycle hitting set is an isolating long cycle hitting set and will intersect $\mathcal{S}_{\mathsf{intersect}}$. Again, we can return early in this case.

Otherwise, we have $|T| \geq 1$. Then we call Lemma 3.32 on this instance. This yields us a set $\mathcal{S}'_{\mathsf{intersect}}$ which might intersect a solution and a set $\mathcal{X}$ as candidates for a possible contraction. We add $\mathcal{S}'_{\mathsf{intersect}}$ to $\mathcal{S}_{\mathsf{intersect}}$. So if (G', T', k, ℓ) has a long cycle hitting set that is not already covered by our set $\mathcal{S}_{\mathsf{intersect}}$, we have that there is an $X \in \mathcal{X}$, which fulfills the requirements of Lemma 3.29. This tells us that $I_X = (G'/X, (T' \setminus X) \cup \{x\}, k, \ell)$ is a valid instance of Disjoint Directed Long Cycle Hitting Set Compression Intersection and moreover that I' and I_X share the same long cycle hitting sets that are disjoint of $T \cup X$ and $(T' \setminus X) \cup \{x$ respectively. It is therefore sufficient to call our recursive procedure on all instances I_X with $X \in \mathcal{X}$.

It remains to show run-time and size bounds. For this we need to bound the number of instances on which our recursive procedure is called. Note that in each recursive call we have that $|(T' \setminus X) \cup \{x\}| = |T'| - |T' \cap X| + 1 \leq |T'| - 1$ as $|T' \cap X| \geq 2$. Thus, our recursive calls nest at most $|T|$ levels deep. Using the conservative size bound

$$|\mathcal{X}| \leq |T|^2 2^{\mathcal{O}(k\ell + k^2 \log k)} \log^2 n$$

for all depths, we get that there are at most

$$(|T|^2 2^{\mathcal{O}(|T|k\ell + |T|k^2 \log k)} \log^2 n)^{|T|}$$

calls to our subroutine. Each subroutine calls $A_{\mathsf{isolating}}$, Theorem 3.30 and Lemma 3.32 exactly once. This means the overall run-time is

$$\begin{aligned}
&\left(|T|^2 2^{\mathcal{O}(k\ell + k^2 \log k)} \log^2 n\right)^{|T|} \left(2^{\mathcal{O}(\ell^6 + k\ell + k^2 \log k)} \cdot \mathrm{poly}(n) + A_{\mathsf{isolating}}(n, |T|, k, \ell)\right) \\
= &\left(|T|^{2|T|} 2^{\mathcal{O}(\ell^6 + |T|k\ell + |T|k^2 \log k)} \log^{2|T|} n\right) \left(A_{\mathsf{isolating}}(n, |T|, k, \ell) + \mathrm{poly}(n)\right).
\end{aligned}$$

For the size bound note that in each subroutine call if we return early we add at most $\max\{2\ell^6, f_{\mathsf{isolating}}(n, |T|, k, \ell)\}$ vertices to $\mathcal{S}_{\mathsf{intersect}}$. Otherwise, we add an amount bounded in the number of new subroutine calls. This gives us a size bound of

$$\left(|T|^{2|T|} 2^{\mathcal{O}(|T|k\ell + |T|k^2 \log k)} \log^{2|T|} n\right) \left(2\ell^6 + f_{\mathsf{isolating}}(n, |T|, k, \ell)\right). \qquad \square$$

3.2.4 Pushing by Important Hitting Separators

In the previous sections we reduced the Directed Long Cycle Hitting Set problem to the Isolating Long Cycle Hitting Set Intersection problem, a variant where we are already given a solution T of size at most $k+1$ and search for a solution S disjoint from T of size at most k. Additionally, we know that T has at most one vertex in each strongly connected component of $G - S$. The situation is now similar, but not identical, to the situation in the DFVS algorithm by Chen et al. [CLL+08]. They guess a topological ordering on the vertices of T in $G - S$ and observe that there is no path from a vertex in T to earlier vertices in the topological ordering or to itself. Then they solve it by the Skew Separator problem, which is essentially guessing an important $t \to T$-separator (for $t \in T$ being the last vertex of the topological ordering) that dominates the range of some solution .

In our case we need to drop the constraint that a vertex of T may not have a path to itself, as there may exist short cycles containing a vertex of T. We want to apply essentially the same argument, i.e. pushing by an important separator. However, our solution additionally hits long cycle that go through t only. Therefore, we consider *hitting separators*, $t \to (T \setminus \{t\})$-separators that also intersect those cycles. We show that is a special case of the W-Ranged $\mathcal{C}$-Deletion $X \to Y$-Separator problem we introduced in the technical tools section.

Definition 3.34. *Let (G, k, ℓ, T) be an instance of Isolating Long Cycle Hitting Set Intersection and let S be an isolating long cycle hitting set in it. A vertex $t \in T$ is called* last vertex of T *(with respect to S) if there is a topological ordering of strongly connected components of $G - S$ such that t appears in the last component that contains some vertex of T.*

Note that a last vertex of some solution S may not be unique (as there exist different topological orderings). Yet, no last vertex of a solution may reach another vertex of T in $G - S$.

Lemma 3.35. *Let (G, k, ℓ, T) be an instance of Isolating Long Cycle Hitting Set Intersection and let S be an isolating long cycle hitting set in it. Let $t \in T$ be a last vertex of T with respect to S. Then S is a $t \to T \setminus \{t\}$-separator.*

Proof. Suppose, for sake of contradiction, that there exists a $t \to T \setminus \{t\}$-path P in $G - S$ and it ends in some t'. Then P is a certificate that t' must be in no earlier strongly connected component than t in every topological ordering of strongly connected components of $G - S$. As t was in the last component containing a vertex t in such an ordering, they have to be in the same component. This is a contradiction to S being an isolating long cycle hitting set. □

Definition 3.36. *Let (G, k, ℓ, T) be an instance of Isolating Long Cycle Hitting Set Intersection and $t \in T$. A* hitting $t \to T \setminus \{t\}$-separator *is a pair (D_{hit}, S_{sep}) such that*

- *S_{sep} is an inclusion-wise minimal $t \to T \setminus \{t\}$-separator, and*
- *$G[R^+_{G-(D_{hit} \cup S_{sep})}(t)]$ contains no cycle of length greater ℓ.*

The size of (D_{hit}, S_{sep}), denoted by $|(D_{hit}, S_{sep})|$, is defined as $|D_{hit}| + |S_{sep}|$.

Lemma 3.37. *Let (G, k, ℓ, T) be an instance of* Isolating Long Cycle Hitting Set Intersection *and let S be an isolating long cycle hitting set in it. Let $t \in T$ be a last vertex of T with respect to S. Then S contains a hitting $t \to T \setminus \{t\}$-separator (D_{hit}, S_{sep}) with $D_{hit} \subseteq R^+_{G-S_{sep}}(t)$.*

Proof. By Lemma 3.35 we know that S is a $t \to T \setminus \{t\}$-separator. Let $S_{\mathsf{sep}} \subseteq S$ be an inclusion-wise minimal $t \to T \setminus \{t\}$-separator. Define D_{hit} to be the set $S \cap R^+_{G-S_{\mathsf{sep}}}(t)$. Then $(D_{\mathsf{hit}}, S_{\mathsf{sep}})$ is contained in S with $D_{\mathsf{hit}} \subseteq R^+_{G-S_{\mathsf{sep}}}(t)$. It remains to show that it is a hitting $t \to T \setminus \{t\}$-separator by verifying that there are no long cycles in $G' = G[R^+_{G-(D_{\mathsf{hit}} \cup S_{\mathsf{sep}})}(t)]$. Assume for contradiction that there is a cycle O in G' of length greater ℓ. We know that O also exists in G and thus $V(O) \cap T \neq \emptyset$. As S_{sep} is a $t \to T \setminus \{t\}$-separator, we have $V(G') \cap T = \{t\}$ and thus $t \in V(O)$. Now, S is a long cycle hitting set and thus $S \cap V(O) \neq \emptyset$. Let $s \in S \cap V(O)$ be the first vertex of S that O visits after t. Then $O[t, s]$ is a witness that either $s \in S_{\mathsf{sep}}$ or $s \in S \cap R^+_{G-S_{\mathsf{sep}}}(t) = D_{\mathsf{hit}}$. This is a contradiction to O being a cycle in G' which is a subgraph of $G - (D_{\mathsf{hit}} \cup S_{\mathsf{sep}})$ (and thus $V(O) \cap (D_{\mathsf{hit}} \cup S_{\mathsf{sep}}) = \emptyset$). □

We now lay the foundation for our pushing argument. That is we show how to replace a hitting $t \to T \setminus \{t\}$-separator $(D_{\mathsf{hit}}, S_{\mathsf{sep}})$ by another hitting $t \to T \setminus \{t\}$-separator $(D'_{\mathsf{hit}}, S'_{\mathsf{sep}})$ with $|(D'_{\mathsf{hit}}, S'_{\mathsf{sep}})| \leq |(D_{\mathsf{hit}}, S_{\mathsf{sep}})|$ and $R^+_{G-S_{\mathsf{sep}}}(t) \subseteq R^+_{G-S'_{\mathsf{sep}}}(t)$.

Lemma 3.38. *Let (G, k, ℓ, T) be an* Isolating Long Cycle Hitting Set Intersection *instance and let S be an isolating long cycle hitting set of size at most k in it. Let $t \in T$ be a last vertex of T with respect to S and (D_{hit}, S_{sep}) a hitting $t \to T \setminus \{t\}$-separator in S with $D_{hit} \subseteq R^+_{G-S_{sep}}(t)$. For any hitting $t \to T \setminus \{t\}$-separator (D'_{hit}, S'_{sep}) with $|(D'_{hit}, S'_{sep})| \leq |(D_{hit}, S_{sep})|$ and $R^+_{G-S_{sep}}(t) \subseteq R^+_{G-S'_{sep}}(t)$, we have that $S' = (S \setminus (D_{hit} \cup S_{sep})) \cup (D'_{hit} \cup S'_{sep})$ is also an isolating long cycle hitting set of size at most k.*

Proof. First we check the size bound. We have that

$$|S'| \leq |S| - |D_{\mathsf{hit}} \cup S_{\mathsf{sep}}| + |D'_{\mathsf{hit}} \cup S'_{\mathsf{sep}}| = |S| + |(D'_{\mathsf{hit}}, S'_{\mathsf{sep}})| - |(D_{\mathsf{hit}}, S_{\mathsf{sep}})| \leq |S| \leq k.$$

Now assume for contradiction that S' is not an isolating long cycle hitting set. Then there is either a cycle O in $G - S'$ with $|O| \geq \ell$ or a closed walk W in $G - S'$ connecting two vertices of T. In any case O and W did not exist in $G - S$ thus there is a vertex $s \in S \setminus S' \subseteq (D_{\mathsf{hit}} \cup S_{\mathsf{sep}})$ contained in them. Now S_{sep} and S'_{sep} are inclusion-wise minimal $t \to T \setminus \{t\}$-separators and thus $S_{\mathsf{sep}} = N^+(R^+_{G-S_{\mathsf{sep}}}(t))$ and $S'_{\mathsf{sep}} = N^+(R^+_{G-S'_{\mathsf{sep}}}(t))$. This implies $S_{\mathsf{sep}} \cup R^+_{G-S_{\mathsf{sep}}}(t) \subseteq S'_{\mathsf{sep}} \cup R^+_{G-S'_{\mathsf{sep}}}(t)$.

By $D_{\mathsf{hit}} \subseteq R^+_{G-S'_{\mathsf{sep}}}(t)$, we have that $s \in S_{\mathsf{sep}} \cup R^+_{G-S_{\mathsf{sep}}}(t) \subseteq S'_{\mathsf{sep}} \cup R^+_{G-S'_{\mathsf{sep}}}(t)$. S'_{sep} is a $t \to (T \setminus \{t\})$-separator, so no vertex of $T \setminus \{t\}$ is reachable from s. By $s \in O$ and $s \in W$ this shows $V(O) \cap T \subseteq \{t\}$ and $V(W) \cap T \subseteq \{t\}$, respectively. This already rules out the case that there is a closed walk W in $G - S'$ connecting two vertices of T. Now consider the long cycle O, which has to contain a vertex of T. From above argument this must be t and thus $V(O) \subseteq R^+_{G-S'}(t) \subseteq R^+_{G-(D'_{\mathsf{hit}} \cup S'_{\mathsf{sep}})}(t)$. This is a contradiction to $G[R^+_{G-(D'_{\mathsf{hit}} \cup S'_{\mathsf{sep}})}(t)]$ containing no long cycles by definition of hitting $t \to T \setminus \{t\}$-separators. □

We now recall the definition of W-ranged $\mathcal{C}$-deletion $X \to Y$-separators from the technical tools section.

Definition 3.18. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$ and let $\mathcal{C}$ be a graph class. A* W-ranged $\mathcal{C}$-deletion $X \to Y$-separator *is a pair (D, S) with $D, S \subseteq V(G) \setminus (W \cup X \cup Y)$ such that S is an inclusion-wise minimal $X \to Y$-separator and $G[R^+_{G-(D \cup S)}(W)] \in \mathcal{C}$. The size of (D, S), denoted by $|(D, S)|$, is defined as $|D| + |S|$.*

Note that for $W = \{t\}$, $\mathcal{C}$ being the set of graphs with circumference at most ℓ, $X = \{t\}$ and $Y = T \setminus \{t\}$ the W-ranged $\mathcal{C}$-deletion $X \to Y$-separators are exactly our hitting $t \to T \setminus \{t\}$-separators. To apply the pushing argument we make use of the range equivalence classes of important W-ranged $\mathcal{C}$-deletion $X \to Y$-separators.

Definition 3.20. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$ and let $\mathcal{C}$ be a graph class. An W-ranged $\mathcal{C}$-deletion $X \to Y$-separator (D, S) is said to* dominate *another W-ranged $\mathcal{C}$-deletion $X \to Y$-separator (D', S') if $R^+_{G-S}(X) \subseteq R^+_{G-S'}(X)$, $|(D, S)| \leq |(D', S')|$ and the inclusion or inequality is strict.*

We call an W-ranged $\mathcal{C}$-deletion $X \to Y$-separator important *if it is not dominated by any other W-ranged $\mathcal{C}$-deletion $X \to Y$-separator.*

We call two W-ranged $\mathcal{C}$-deletion $X \to Y$-separators (D, S) and (D', S') range equivalent *if $R^+_{G-S}(X) = R^+_{G-S'}(X)$. Let $\mathcal{DS}$ be some subset of the ranged $\mathcal{C}$-deletion $X \to Y$-separators of G. The* range equivalence classes *of $\mathcal{DS}$ are the equivalence classes of $\mathcal{DS}$ formed by the "range equivalence" equivalence relation.*

Our main result for important W-ranged $\mathcal{C}$-deletion $X \to Y$-separators was that a representative of every range equivalent class can be computed efficiently if we are given an oracle for the W-Ranged $\mathcal{C}$-deletion problem.

Theorem 3.24. *Let G be a directed graph, let $W, X, Y \subseteq V(G)$ be three vertex sets with $W \subseteq X$, let $\mathcal{C}$ be an hereditary graph class and let $k \in \mathbb{Z}_{\geq 0}$ be a non-negative integer. Then there are at most $4^{k \log k}$ range equivalence classes among the important W-ranged $\mathcal{C}$-deletion $X \to Y$-separators of size exactly k.*

Moreover, for an instance (G, W, X, Y, k) we can find a representative of every range equivalence class of important W-ranged $\mathcal{C}$-deletion $X \to Y$-separators of size exactly k in time

$$4^{k \log k} \operatorname{poly}(n) + 4^{k \log k} n \cdot A_{\mathcal{C}\text{-Deletion}},$$

where $A_{\mathcal{C}\text{-Deletion}}$ is the maximum run-time of an oracle for W-ranged $\mathcal{C}$-Deletion over all subgraphs $G[R^+_{G-S}(W)]$ of G, where S is some $X \to Y$-separator.

We want to use this theorem to find range-maximal hitting $t \to T \setminus \{t\}$-separators. Our oracle then needs to solve the following problem.

Directed Long Cycle Hitting Set from Singular Solution

Instance: A graph G, a vertex $t \in V(G)$ and two integer $k, \ell \in \mathbb{Z}_{\geq 0}$ s.t.
- G has no medium-length cycles,
- $\operatorname{cf}(G - t) \leq \ell$.

Task: Find a set $S \subseteq V(G) \setminus \{t\}$ with $|S| \leq k$ and $\operatorname{cf}(G - S) \leq \ell$ or decide that no such set exists.

Lemma 3.39. *An instance (G, T, k, ℓ) of* Isolating Long Cycle Hitting Set Intersection *can be solved in time*

$$|T|k4^{k \log k} A_{\mathsf{singular}}(n, k, \ell) \operatorname{poly}(n),$$

where $A_{\mathsf{singular}}(n, k, \ell)$ is the run-time of an algorithm for Directed Long Cycle Hitting Set from Singular Solution *on instances (G', t, k', ℓ) with $|V(G')| \leq n$ and $k' \leq k$. The set $\mathcal{S}_{\mathsf{intersect}}$ returned has size at most $|T|k^2 4^{k \log k}$.*

Proof. If (G, T, k, ℓ) does not have a solution we are allowed to return any set, as long as we obey run-time and size bounds. So assume for now that (G, T, k, ℓ) has a isolating long cycle hitting set S of size at most k. We guess a vertex $t \in T$ as last vertex of T with respect to S. From Lemma 3.37 we know that S contains a hitting $t \to T \setminus \{t\}$-separator $(D_{\mathsf{hit}}, S_{\mathsf{sep}})$ with $D_{\mathsf{hit}} \subseteq R^+_{G-S_{\mathsf{sep}}}(t)$. By definition the $t \to T \setminus \{t\}$-separators are exactly the W-ranged $\mathcal{C}$-deletion $X \to Y$-separators for $W = \{t\}$, $\mathcal{C}$ being the set of graphs with circumference at most ℓ, $X = \{t\}$ and $Y = T \setminus \{t\}$. Lemma 3.38 tells us that for every hitting $t \to T \setminus \{t\}$-separator with $|(D'_{\mathsf{hit}}, S'_{\mathsf{sep}})| \leq |(D_{\mathsf{hit}}, S_{\mathsf{sep}})|$ and $R^+_{G-S_{\mathsf{sep}}}(t) \subseteq R^+_{G-S'_{\mathsf{sep}}}(t)$, there is an isolating long cycle hitting set S' of size at most k containing it. Now either $(D_{\mathsf{hit}}, S_{\mathsf{sep}})$ is important as W-ranged $\mathcal{C}$-deletion $X \to Y$-separator or there is an important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator $(D'_{\mathsf{hit}}, S'_{\mathsf{sep}})$ that fulfills Lemma 3.38. Moreover, we can take any important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator $(D'_{\mathsf{hit}}, S'_{\mathsf{sep}})$ of the same range equivalence class and still fulfill Lemma 3.38. Thus, it is enough to report a vertex set $\mathcal{S}_{\mathsf{intersect}}$ containing one important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator of every range equivalent class.

We guess the size $k^\star \leq k$ of such an important W-ranged $\mathcal{C}$-deletion $X \to Y$-separator. By Theorem 3.24, we get an algorithm finding one ranged deletion separator out of every range equivalent class if we can solve the W-Ranged $\mathcal{C}$-deletion problem on subgraphs $G[R^+_{G-S_{\mathsf{sep}}}(W)]$ of G, where S_{sep} is some $X \to Y$-separator. In our case this is finding a set $S \subseteq V(G) \setminus T$ such that $G[R^+_{G-S_{\mathsf{sep}}}(W)] - S$ contains no long cycles where S_{sep} is some $t \to (T \setminus \{t\})$-separator. We now make sure that (G', t, k', ℓ) with $G' = G[R^+_{G-S_{\mathsf{sep}}}(W)]$ and some $k' \leq k^\star \leq k$ is indeed a Directed Long Cycle Hitting Set from Singular Solution instance. First, as G' is a subgraph of G it contains no medium-length cycles. Moreover, as S_{sep} is some $t \to (T \setminus \{t\})$-separator, we know that G' is a subgraph of $G - (T \setminus \{t\})$ and thus $\operatorname{cf}(G' - t) \leq \operatorname{cf}(G - T) \leq \ell$.

We have $|T|k$ choices for $t \in T$ and $k^\star \leq k$. For every choice we call Theorem 3.24, which takes $4^{k^\star \log k^\star} A_{\mathsf{singular}} \operatorname{poly}(n)$ time and returns $4^{k^\star \log k^\star}$ sets of size $k^\star \leq k$. Thus, we get a run-time bound of $|T|k4^{k \log k} A_{\mathsf{singular}} \operatorname{poly}(n)$ and size bound of $|T|k^2 4^{k \log k}$. □

3.2.5 Reduction to Strongly Connected Graphs with Many Short Cycles

In the last section we managed to shrink our given solution to a single vertex t. The advantage of this is twofold. First, we can get rid of all parts of the graph that are not in the same strongly connected component as t, as they will never lie on a short cycle. Second, the shadow covering technique becomes more powerful as any vertex outside of shadow and solution must be connected to t. However, before we start, we apply the compression intersection technique again.

Corollary 3.40. *An instance (G, t, k, ℓ) of Directed Long Cycle Hitting Set from Singular Solution can be solved in time*

$$\mathcal{O}(k(f_{\mathsf{intersect}}(n,k,\ell))^k \cdot A_{\mathsf{singular\ intersection}}(n,k,\ell)) + 2^{\mathcal{O}(\ell)}\operatorname{poly}(n),$$

where $A_{\mathsf{singular\ intersection}}(n,k,\ell)$ is the run-time of an algorithm for the intersection variant of Directed Long Cycle Hitting Set from Singular Solution on instances (G', t, k', ℓ) with $|V(G')| \leq n$ and $k' \leq k$, and $f_{\mathsf{intersect}}(n,k,\ell)$ is a size bound on the set $\mathcal{S}_{\mathsf{intersect}}$ output by the algorithm on these instances.

Proof. Apply Lemma 2.4 with the membership oracle from Theorem 3.26. □

Lemma 3.41. *Let $I = (G, t, k, \ell)$ be an instance of Directed Long Cycle Hitting Set from Singular Solution. Let $C^\star$ be the strongly connected component of G containing t. Define a new instance $I^\star = (G^\star, t, k, \ell)$ with $G^\star = G[C^\star]$. Then any solution $S^\star$ to $I^\star$ is a solution to I. Moreover, for any solution S to I, $S \cap C^\star$ is a solution to $I^\star$.*

Proof. In both cases we have that the size of the claimed solution does not increase. It is therefore enough to check that no long cycles are introduced.

Consider any solution $S^\star$ to $I^\star$. Assume for contradiction that $G - S^\star$ contains a long cycle O. As $t \in V(O)$ this cycle lies entirely in $C^\star$ and thus in $G^\star - S^\star$, a contradiction to $S^\star$ being a solution to $I^\star$.

Now consider a solution S to I. Assume for contradiction that there is a long cycle O in $G - (S \cap C^\star)$. As O does not exist in $G - S$, we have that there is an $s \in V(O) \cap (S \setminus C^\star)$. We know that $t \in V(O)$ and thus $s \in V(O) \subseteq C^\star$, a contradiction to $s \notin C^\star$. Hence, we know that $G - (S \cap C^\star)$ contains no long cycle and thus neither does its subgraph $G^\star - (S \cap C^\star)$, which shows the claim. □

As a next step, we want to apply the "Shadow Covering" technique as described in Chapter 2. This helps us to get rid of arcs that lie only on long cycles. The main observation about the usefulness of sets covering the shadow is summarized by the following lemma.

Lemma 3.42. *Let G be a strongly connected, directed graph, $\ell \in \mathbb{Z}_{\geq 0}$ and $t \in V(G)$ with $\operatorname{cf}(G - t) \leq \ell$. Let $S \subseteq V(G) \setminus \{t\}$ be a vertex set with $\operatorname{cf}(G - S) \leq \ell$ and let $(v, w) \in A(G)$ be an arc with $\operatorname{dist}_G(w, v) \geq \ell$. For any $Z \subseteq V(G) \setminus \{t\}$ we can identify in time $\mathcal{O}(n+m)$ a vertex set $\mathcal{S}_{\mathsf{intersect}}$ with $|\mathcal{S}_{\mathsf{intersect}}| \leq 2$, such that if $Z \subseteq V(G) \setminus (S \cup \{t\})$ is a set covering the shadow of S with respect to t, then $S \cap \mathcal{S}_{\mathsf{intersect}} \neq \emptyset$.*

Proof. Find a $w \to t$-path P and a $t \to v$-path Q in G. Let x be the last vertex of Q not in Z and y be the first vertex of P not in Z. We claim that $S_{\mathsf{intersect}} = \{x, y\}$ fulfills the statement. The run-time and size bounds are clear.

Assume that $S \cap \mathcal{S}_{\mathsf{intersect}} = \emptyset$. Then x and y do not lie in the shadow of S with respect to t and not in S itself. By definition of shadow there is an $y \to t$-path P' and an $t \to x$-path Q' in $G - S$. By choice of x and y we have that $Q[x, v]$ and $P[w, y]$ are contained in $Z \cup S_{\mathsf{intersect}} \subseteq V(G) \setminus S$. Thus, the closed walk $(v, w) \circ P[w, y] \circ P' \circ Q' \circ Q[x, v]$ exists in $G - S$. Now every walk on a closed walk lies also on a cycle completely contained in that closed walk. Hence, (v, w) lies on a cycle O in $G - S$. However, since $\operatorname{dist}_G(w, v) \geq \ell$ this cycle has length at least $\ell + 1$, a contradiction to $\operatorname{cf}(G - S) \leq \ell$. □

Applying the deterministic covering of shadows we get the following result.

Lemma 3.43. *Let $I = (G, t, k, \ell)$ be an instance of* Directed Long Cycle Hitting Set from Singular Solution *such that G is strongly connected and there is an arc $(v, w) \in A(G)$ with $\mathrm{dist}_G(w, v) \geq \ell$. Then we can find in time $2^{\mathcal{O}(k^2)} \cdot \mathrm{poly}(n)$ a set $\mathcal{S}_{\mathsf{intersect}}$ with $|\mathcal{S}_{\mathsf{intersect}}| \leq 2^{\mathcal{O}(k^2)} \log^2 n$ such that any long cycle hitting set of size at most k intersects $\mathcal{S}_{\mathsf{intersect}}$.*

Proof. Let $\mathcal{F}$ be the set of all cycles in G of length more than ℓ. Then the graphs of $\mathcal{F}$ are t-connected and the long cycle hitting sets of I are exactly the $\mathcal{F}$-transversals. Fix an arbitrary long cycle hitting set S of size at most k. By Theorem 2.9 we can compute a set $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_p\}$ with $p = 2^{\mathcal{O}(k^2)} \log^2 n$ in time $2^{\mathcal{O}(k^2)} \cdot \mathrm{poly}(n)$ such that for some $i \in \{1, \ldots, p\}$ we have that $S \cap Z_i = \emptyset$ and Z_i contains the shadow of S with respect to t. Now apply Lemma 3.42 to every of the sets Z_i and let $\mathcal{S}_{\mathsf{intersect}}$ be the union of the returned sets. For the right choice of i, we have that the returned set and thus $\mathcal{S}_{\mathsf{intersect}}$ intersects S. The run-time and size bounds follow directly. □

Applying the above lemmas we can simplify our problem to the following.

Directed Long Cycle Hitting Set with many Short Cycles

Instance: A strongly connected, directed graph G, a vertex $t \in V(G)$ and two integer $k, \ell \in \mathbb{Z}_{\geq 0}$ such that
- G has no medium-length cycles,
- $\mathrm{cf}(G - t) \leq \ell$,
- every arc of G lies on a cycle of length at most ℓ.

Task: Find a vertex set $\mathcal{S}_{\mathrm{intersect}} \subseteq V(G)$ such that if there is a long cycle hitting set $S \subseteq V(G) \setminus \{t\}$ of size at most k, then there is such a set S with $S \cap \mathcal{S}_{\mathsf{intersect}} \neq \emptyset$.

Lemma 3.44. *An instance (G, t, k, ℓ) of* Directed Long Cycle Hitting Set from Singular Solution *can be solved in time*

$$\mathcal{O}(k(\max\{f_{\mathsf{intersect}}(n, k, \ell), 2^{\mathcal{O}(k^2)} \log^2 n\})^k \cdot A_{\mathsf{short\ cycles}}(n, k, \ell)) + 2^{\mathcal{O}(\ell)} \mathrm{poly}(n),$$

where $A_{\mathsf{short\ cycles}}(n, k, \ell)$ is the run-time of an algorithm for Directed Long Cycle Hitting Set with many Short Cycles *on instances (G', t, k', ℓ) with $|V(G')| \leq n$ and $k' \leq k$, and $f_{\mathsf{intersect}}(n, k, \ell)$ is a size bound on the set $\mathcal{S}_{\mathsf{intersect}}$ output by the algorithm on these instances.*

Proof. Apply Corollary 3.40 to reduce to the intersection variant of Directed Long Cycle Hitting Set from Singular Solution. If the resulting instance is not strongly connected, we apply Lemma 3.41 to shrink the instance to a strongly connected subgraph. Then for every $(v, w) \in A(G)$, we compute $\mathrm{dist}_G(w, v)$. If it is at least ℓ for some arc, we apply Lemma 3.43 and return the resulting set of size $2^{\mathcal{O}(k^2)} \log^2 n$. Otherwise, we have that all arcs lie on a cycle of length at most ℓ. Thus, the remaining instance is a Directed Long Cycle Hitting Set with many Short Cycles instance and we apply $A_{\mathsf{short\ cycles}}$. □

3.2.6 Portals and Clusters

Right now we reduced the problem to strongly connected, directed graphs where we are given a single vertex t intersecting all long cycles and we are searching for a solution disjoint from it. The next step simplify this problem further by consideration of the strongly connected components of $G - t$. The deletion of t reduces the long cycles in G to paths. We observe that every long path must be traversing a long distance in some strongly connected component of $G - t$. These long distances allow us to identify clusters of vertices that need to be separated from each other. In the end this problem becomes a Directed Multiway Cut problem.

Let us start with the structure of G after the deletion of t. Let $\mathcal{C}$ be the set of strongly connected components of $G - t$. For each $C \in \mathcal{C}$, we identify certain "portal" vertices that can be used to enter/leave the component.

Definition 3.45. *Let G be a graph and let $C \subseteq V(G)$. A vertex $v \in C$ is a* portal vertex *of C, if $\Delta_G(v) > \Delta_{G[C]}(v)$, where $\Delta_H(v)$ is the number of incident arcs (both in-coming and out-going) of v in a graph H. We denote by X_C the set of all portal vertices of C.*

Lemma 3.46. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance and $\mathcal{C}$ the strongly connected components of $G - t$. Then for any $C \in \mathcal{C}$ and any $v \in X_C$ there is a cycle of length at most ℓ in G going through v and t.*

Proof. As $\Delta_G(v) > \Delta_{G[C]}(v)$ there is an arc $a \in A(G)$ incident to v with its other endpoint w not contained in C. We know that a lies on a cycle of length at most ℓ in G. As $a \notin G[C]$ this cycle exists in G but not in $G-t$; thus, the cycle goes through t. □

For every $C \in \mathcal{C}$ and $v \in X_C$, fix an arbitrary cycle as in Lemma 3.46, and let O_v be the vertex set of that cycle.

Lemma 3.47. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance and $\mathcal{C}$ the strongly connected components of $G - t$. Then for any $C \in \mathcal{C}$ and any $v_1, v_2 \in X_C$, either $\mathrm{dist}_{G[C]}(v_1, v_2) \leq 2\ell^2$ or $\mathrm{dist}_{G[C]}(v_1, v_2) \geq 2\ell^6 - 2\ell$.*

Proof. Suppose, for sake of contradiction, that P_1 is a $v_1 \to v_2$-path of $G[C]$ with $2\ell^2 + 1 \leq |P_1| \leq 2\ell^6 - 2\ell - 1$. There is a $v_2 \to v_1$-path P_2 using only the vertices of O_{v_2} and O_{v_1} and hence has length at most 2ℓ.

Consider the directed graph G' induced by the vertices of the $v_1 \to v_2$-path P_1 and the $v_2 \to v_1$-path P_2. This graph has at most $|P_1| + |P_2| \leq |P_1| + 2\ell < 2\ell^6$ vertices. As G contains no medium-length cycles (i.e. no cycles with length in $(\ell, 2\ell^6]$), we get $\mathrm{cf}(G') \leq \ell$. Applying Lemma 3.9 on P_1 and P_2 in G', we get $|P_1| \leq (\mathrm{cf}(G') - 1)|P_2| \leq (\ell - 1) \cdot 2\ell < 2\ell^2 + 1$, a contradiction. □

For any $C \in \mathcal{C}$, we partition X_C into clusters the following way. Let $\ell_{\max} := 2\ell^2$. For every $v \in X_C$, denote by X_v the subset of X_C that is at distance at most $\ell_{\max}$ from v in $G[C]$. By definition, we have that $v \in X_v$.

Lemma 3.48. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance and $\mathcal{C}$ the strongly connected components of $G - t$. For every $C \in \mathcal{C}$ and $v_1, v_2 \in X_C$, the sets X_{v_1} and X_{v_2} are either disjoint or equal.*

Proof. Suppose that $x \in X_{v_1} \cap X_{v_2}$ and, without loss of generality, $y \in X_{v_1} \setminus X_{v_2}$. Now

$$\begin{aligned}
\operatorname{dist}_{G[C]}(v_2, y) &\leq \operatorname{dist}_{G[C]}(v_2, x) + \operatorname{dist}_{G[C]}(x, v_1) + \operatorname{dist}_{G[C]}(v_1, y) \\
&\leq \ell_{\max} + (\ell - 1) \operatorname{dist}_{G[C]}(v_1, x) + \ell_{\max} && \text{(Lemma 3.9)} \\
&\leq \ell_{\max} + (\ell - 1) \cdot \ell_{\max} + \ell_{\max} \\
&= (\ell + 1)\ell_{\max} \\
&\leq 2\ell^6 - 2\ell - 1,
\end{aligned}$$

and hence, by Lemma 3.47, we have $\operatorname{dist}_{G[C]}(v_2, y) \leq 2\ell^2 = \ell_{\max}$, implying $y \in X_{v_2}$. □

Therefore, the sets X_v for $v \in X_C$ define a partition of X_C. We call the classes of this partition the *clusters* of X_C.

An example for portals and clusters can be found in Fig. 3.4.

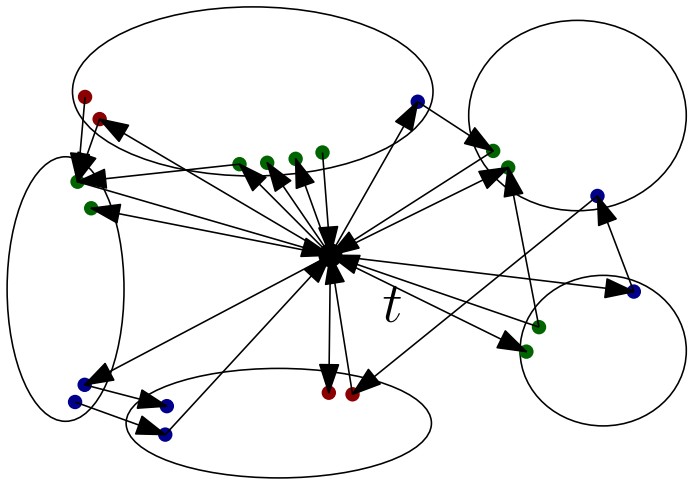


Figure 3.4: An example for the structure of $G - t$. The large circles form the strongly connected components $C \in \mathcal{C}$. The colored dots represent the portals with their color corresponding to their cluster.

The huge distance between the clusters and the absence of medium-length cycles allows for the following structural insight:

Lemma 3.49. *Let (G, t, k, ℓ) be a* Directed Long Cycle Hitting Set with many Short Cycles *instance and $\mathcal{C}$ the strongly connected components of $G - t$. Let R be a cycle of length more than ℓ in G. Then R contains a path between two different clusters of some $C \in \mathcal{C}$.*

Proof. As $\operatorname{cf}(G - t) \leq \ell$ we have that $t \in R$. Starting from t, let $x_0, \ldots, x_p$ be the vertices of R that are in $\bigcup_{C \in \mathcal{C}} X_C$. By definition, the vertex after t is in X_C for some $C \in \mathcal{C}$ and the vertex before t is in $X_{C'}$ for some $C' \in \mathcal{C}$. Thus, $R[x_0, x_p]$ contains every vertex of R except t, yielding $|R[x_0, x_p]| = |R| - 2$. If an arc (u, v) of $R[x_0, x_p]$ has u and v in different strongly connected components $C_1 \in \mathcal{C}$ and $C_2 \in \mathcal{C}$ respectively, then $u \in X_{C_1}$ and $v \in X_{C_2}$, hence both appear in the sequence $x_0, \ldots, x_p$. Therefore, for $i = 0, \ldots, p - 1$ the subpath $R[x_i, x_{i+1}]$ is either fully contained in a single component $C \in \mathcal{C}$ or consists of only one arc.

If x_i and x_{i+1} are in the same strongly connected component $C \in \mathcal{C}$, and they are in two different clusters of C, then we are done. Otherwise, if x_i and x_{i+1} are in the same cluster, then $\text{dist}_{G[C]}(x_i, x_{i+1}) \leq \ell_{\max}$ by the definition of the clusters. Thus, Lemma 3.10 implies $|R[x_i, x_{i+1}]| \leq (\text{cf}(G[C]) - 1)^2 \cdot \ell_{\max} < (\ell - 1)^2 \cdot \ell_{\max}$. Therefore, if $p < \ell^2$, we have $|R| = |R[x_0, x_p]| + 2 \leq p \cdot (\ell - 1)^2 \cdot \ell_{\max} + 2 < 2\ell^6$, contradicting that G has no medium-length cycles. Otherwise, consider the vertex x_{ℓ^2}; we have $\ell^2 \leq |R[x_0, x_{\ell^2}]| \leq \ell^2(\ell - 1)^2\ell_{\max}$. By Lemma 3.46, there is an $x_{\ell^2} \to t$-path of length at most $\ell - 1$. As x_0 is an out-neighbor of t, this means that there is an $x_{\ell^2} \to x_0$-path Q of length at most ℓ in G. Let G' be the directed graph induced by $R[x_0, x_{\ell^2}]$ and Q. As G' has at most $|R[x_0, x_{\ell^2}]| + |Q| \leq \ell^2(\ell - 1)^2\ell_{\max} + \ell < 2\ell^6$ vertices and G contains no medium-length cycles (i.e., no cycles with length in $(\ell, 2\ell^6]$), we have that $\text{cf}(G') \leq \ell$. Applying Lemma 3.9 on $R[x_0, x_{\ell^2}]$ and Q in G' we get $|R[x_0, x_{\ell^2}]| \leq (\ell - 1)|Q| < \ell^2 \leq |R[x_0, x_{\ell^2}]|$—a contradiction. □

We now focus again on finding the long cycle hitting sets in G.

Lemma 3.50. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance and $\mathcal{C}$ the strongly connected components of $G - t$. Let $x_1 \in L_1, x_2 \in L_2$ for distinct clusters L_1, L_2 of some strongly connected component $C \in \mathcal{C}$. For each $S \subseteq V(G)$ with $\text{cf}(G-S) \leq \ell$ and $S \cap (O_{x_1} \cup O_{x_2}) = \emptyset$, there is no $x_1 \to x_2$-path in $G - S$.*

Proof. Let S be disjoint from $O_{x_1} \cup O_{x_2}$ and suppose, for sake of contradiction, that there is an $x_1 \to x_2$-path P_1 in $G[C] - S$. As x_1 and x_2 are in distinct clusters, we have $|P_1| > \ell_{\max}$. There is a $t \to x_1$-path of G using only the vertices of O_{v_1} and hence has length at most ℓ. Similarly, there is an $x_2 \to t$-path in G using only vertices of O_{v_2} and having length at most ℓ. The concatenation of these two paths gives an $x_2 \to x_1$-walk using only the vertices $O_{v_1} \cup O_{v_2}$ and having length at most 2ℓ. This walk contains an $x_2 \to x_1$-path P_2 of length at most 2ℓ.

By the assumptions of the lemma, P_1 and P_2 are disjoint from S. Applying Lemma 3.9 on the $x_1 \to x_2$-path P_1 and the $x_2 \to x_1$-path P_2 in $G - S$, we get $|P_1| \leq (\text{cf}(G - S) - 1)|P_2| \leq (\ell - 1) \cdot 2\ell < \ell_{\max}$—a contradiction. □

Our next goal is to use Lemma 3.50 to argue that there cannot be too many clusters in a component $C \in \mathcal{C}$. This may in general not be the case, but if there are many clusters we can identify vertices lying in all long cycle hitting sets. For this we again apply the shadow covering technique (see Chapter 2).

Lemma 3.51. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance, $Z \subseteq V(G)$ and $C \in \mathcal{C}$ be a component with distinct clusters L_1, L_2. Let $x_1 \in L_1$, $x_2 \in L_2$ and $x_3 \in C \setminus Z$. Then any $S \subseteq V(G)$ with $\text{cf}(G - S) \leq \ell$, $S \cap (O_{x_1} \cup O_{x_2}) = \emptyset$ and that is shadowless with respect to Z cannot have both an $x_1 \to x_3$-path P_1 and an $x_2 \to x_3$-path P_2 in $G[C] - S$.*

Proof. Let S be as in the statement and suppose, for sake of contradiction, that both P_1 and P_2 exist. Let R_i be a $t \to x_i$-path in O_{x_i} for $i = 1, 2$. The concatenation of R_1 and P_1 shows that there is a $t \to x_3$-path in $G - S$. By $x_3 \in V(G) \setminus Z$ and S being shadowless with respect to Z, there is an $x_3 \to t$-path Q disjoint from S in $G - S$. Let (a, b) be the last arc of Q that is not entirely in C (as $t \notin C$, there is such an arc). As $Q[x_3, a]$ is a path disjoint from t which starts and ends in C, the path Q is

entirely contained in the strongly connected component C of $G - t$. Thus, a is in X_C, and hence in some cluster L. Now for $i = 1, 2$ we have that $P_i \circ Q[x_3, a]$ is a walk from L_i to L, fully contained in $G[C] - S$. However, L is different from at least one of L_1 and L_2. Without loss of generality, let $L \neq L_1$. Then $P_1 \circ Q[x_3, a]$ contains an $x_1 \to a$-path of length at least $\ell_{\max}$ by definition of clusters. Likewise, $P_1 \circ Q[x_3, t]$ contains an $x_1 \to t$-path $R^\star$ of length at least $\ell_{\max}$ (as $Q[a, t]$ is outside of C). Consider again the $t \to x_1$-path R_1 inside O_{x_1}. As it lies inside O_{x_1}, it is disjoint from S and has length at most ℓ. If we now compare the length of $R^\star$ and R_1 with help of Lemma 3.9, we get $|R^\star| \leq (\mathrm{cf}(G - S) - 1)|R_1| \leq \ell^2 < \ell_{\max} \leq |R^\star|$—a contradiction. □

The following lemma gives us an intersecting set when a vertex is reachable from many clusters on (mostly) disjoint paths.

Lemma 3.52. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance, $Z \subseteq V(G)$ and $\mathcal{C}$ the strongly connected components of $G - t$. Let $x_1, \ldots, x_{k+2}$ be vertices in distinct clusters of a component $C \in \mathcal{C}$ and $v \in C \setminus Z$ be a vertex. Furthermore, let $P_1, \ldots, P_{k+2}$ be paths in $G[C]$ such that P_i is an $x_i \to v$-path and these paths share vertices only in $Z \cup \{v\}$. Then any $S \subseteq V(G)$ with $\mathrm{cf}(G - S) \leq \ell$ and $|S| \leq k$ that is shadowless with respect to Z intersects $v \cup \bigcup_{i=1}^{k+2} O_{x_i}$.*

Proof. As $|S| \leq k$ and S is disjoint from Z, at least two of the P_i's have their internal vertices disjoint from S. Assume, without loss of generality, that S contains no internal vertex of P_1 and P_2. If S is disjoint from $O_{x_1} \cup O_{x_2} \cup \{v\}$, then Lemma 3.51 gives a contradiction. □

To check whether Lemma 3.52 is applicable, we reduce to a maximum flow problem.

Lemma 3.53. *It can be tested in polynomial time whether Lemma 3.52 is applicable.*

Proof. For every $C \in \mathcal{C}$ and $v \in C \setminus Z$, we solve the following maximum flow problem with vertex capacities: introduce a new source adjacent to each cluster of C, set v to be the only sink, vertices in $Z \cup \{v\}$ have infinite capacity, and every other vertex of C has unit capacity. An integral flow of value at least $k + 2$ corresponds directly to the vertices in the Lemma 3.52. As a maximum integral flow can be found in polynomial time and we have at most $|V(G)|$ choices for v and C (choosing v fixes C) we can check for this in polynomial time. □

If Lemma 3.52 is not applicable and a strongly connected component $C \in \mathcal{C}$ with many clusters exists, we can find a simple set intersecting every $S \subseteq V(G)$ with $\mathrm{cf}(G - S) \leq \ell$ and $|S| \leq k$.

Lemma 3.54. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance, $Z \subseteq V(G)$ and $\mathcal{C}$ the strongly connected components of $G - t$. Let $x_0, \ldots, x_{k(k+1)+1}$ be vertices from different clusters of some $C \in \mathcal{C}$. If Lemma 3.52 is not applicable then every $S \subseteq V(G)$ with $\mathrm{cf}(G - S) \leq \ell$ and $|S| \leq k$ that is shadowless with respect to Z intersects $\bigcup_{i=1}^{k(k+1)+1} O_{x_i}$.*

Proof. Suppose that S is disjoint from every O_{x_i}. For every $i = 1, \ldots, k(k+1)+1$ let us fix an $x_i \to x_0$-path P_i in C. By Lemma 3.50, S intersects the path P_i for every $i = 1, \ldots, k(k+1)+1$. Denote by y_i be the first vertex of S on P_i.

By pigeonhole principle, there has to be a vertex of S that appears as y_i for at least $k+2$ values of i. Assume without loss of generality, that $y_1 = \ldots = y_{k+2} =: y$. If for some $1 \leq j_1 < j_2 \leq k+2$, paths $P_{j_1}[x_{j_1}, y]$ and $P_{j_2}[x_{j_2}, y]$ share a vertex different from y, then by Lemma 3.51 this vertex has to be in Z. Therefore, the paths $P_1[x_1, y], \ldots, P_{k+2}[x_{k+2,y}]$ share vertices only in $Z \cup \{y\}$, and hence Lemma 3.52 would be applicable, a contradiction. □

Next, we find an intersection set if many components have more than two clusters.

Lemma 3.55. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance and $\mathcal{C}$ the strongly connected components of $G - t$. If there exist strongly connected components $C_1, \ldots, C_{k+1} \in \mathcal{C}$, each containing at least two clusters, then for arbitrary vertices x_i, y_i of different clusters of C_i, every $S \subseteq V(G)$ with $\mathrm{cf}(G - S) \leq \ell$ and $|S| \leq k$ intersects $\bigcup_{i=1}^{k+1}(O_{x_i} \cup O_{y_i})$.*

Proof. Suppose that S is disjoint from $\bigcup_{i=1}^{k+1}(O_{x_i} \cup O_{y_i})$. Then some C_i is disjoint from S, implying that there is an $x_i \to y_i$-path in $G[C_i] - S$, contradicting Lemma 3.50. □

Corollary 3.56. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance, $Z \subseteq V(G)$ and $\mathcal{C}$ the strongly connected components of $G - t$. If*

- *either there is a $C \in \mathcal{C}$ with more than $k(k+1)+1$ clusters,*
- *or there are more than k components in $\mathcal{C}$ with at least two clusters,*

then there is a set $\mathcal{S}_{\text{many clusters}} \subseteq V(G)$ of size at most $(k^2+k+1)(\ell-1)$ that intersects every $S \subseteq V(G)$ with $\mathrm{cf}(G - S) \leq \ell$ and $|S| \leq k$ which is shadowless with respect to Z. Moreover, the set $\mathcal{S}_{\text{many clusters}}$ can be found in polynomial time.

Proof. If there is a $C \in \mathcal{C}$ with more than $k(k+1)+1$ clusters either Lemma 3.52 or Lemma 3.54 is applicable. If there are more than k components in $\mathcal{C}$ with at least two clusters Lemma 3.55 is applicable. We can check whether these conditions are accurate (by possibly using Lemma 3.53) in polynomial time. We can also compute the sets $\mathcal{S}$ they produce in polynomial time and use one of them as set $\mathcal{S}_{\text{many clusters}}$. By taking the maximum over their size bounds and using that $|O_x - t| \leq (\ell - 1)$, we obtain the promised size bound. □

Now we need to handle the remaining case. Namely, that there are only k components in $\mathcal{C}$ with more than two clusters and these have at most $k(k+1)+1$ clusters.

Lemma 3.57. *Let (G, t, k, ℓ) be a Directed Long Cycle Hitting Set with many Short Cycles instance and let $\mathcal{C}$ be the set of strongly connected components in $G - t$. If Corollary 3.56 is not applicable, then there is a set $\mathcal{S}_{\text{few clusters}} \subseteq V(G)$ of size at most $2^{\mathcal{O}(k+\log \ell)}$ such that for every $S \subseteq V(G)$ with $\mathrm{cf}(G - S) \leq \ell$ and $|S| \leq k$,*

- *either S intersects $\mathcal{S}_{\text{few clusters}}$, or*
- *for every $C \in \mathcal{C}$ the set S intersects all paths between different clusters of C in $G[C]$.*

Furthermore, the set $\mathcal{S}_{\text{few clusters}}$ can be found in time $2^{\mathcal{O}(k)} \cdot \mathrm{poly}(n)$.

Proof. Construct $\mathcal{S}_{\mathsf{few\ clusters}}$ as follows:

Input : directed graph G, integers k, ℓ and a vertex $t \in V(G)$
Output: A vertex set $\mathcal{S}_{\mathsf{few\ clusters}} \subseteq V(G)$

```
1 Let S_few clusters = ∅.
2 foreach C ∈ 𝒞 with at least two clusters do
3     Let L_1, ..., L_r the clusters of C.
4     Apply Lemma 3.8 to L_1, ..., L_r to obtain L'_1, ..., L'_r.
5     foreach x ∈ L'_1 ∪ ... ∪ L'_r do
6         Add O_x to S_few clusters.
```

Note that as Corollary 3.56 is not applicable, we have that there are at most k components that have at least two clusters and all of them have $r \leq k^2 + k + 1$ clusters. Together with $|O_x| \leq \ell$, the bound of $|\mathcal{S}_{\mathsf{few\ clusters}}| \leq \ell \cdot 2(r-1)(k+1)4^{k+1} = 2^{\mathcal{O}(k+\log \ell)}$ follows.

For correctness, let $S \subseteq V(G)$ with $\mathrm{cf}(G - S) \leq \ell$ and $|S| \leq k$ such that S is disjoint from $\mathcal{S}_{\mathsf{few\ clusters}}$. Suppose, for sake of contradiction, that there is a path P in $G[C] - S$ between different clusters of C for some $C \in \mathcal{C}$. Without loss of generality, let P be a $L_1 \to L_2$-path. By Lemma 3.8 there is also an $x \to y$-path Q in $G[C] - S$ with $x \in L'_1, y \in L'_2$. But S is disjoint from $O_x \cup O_y \subseteq \mathcal{S}_{\mathsf{few\ clusters}}$ in contradiction to Lemma 3.50. □

A vertex set hitting all $X_i \to X_j$-paths between different vertex sets $X_1, \ldots, X_r$ is known as multiway cut.

Definition 3.58. *Let G be a directed graph and let $X_1, \ldots, X_r \subseteq V(G)$.*

A set $S \subset V(G) \setminus (X_1 \cup \ldots \cup X_r)$ is called $X_1, \ldots, X_r$-multiway cut *if $G - S$ contains no $X_i \to X_j$-path for any $i \neq j$.*

Directed Multiway Cut	
Instance:	A graph G, vertex sets $X_1, \ldots, X_r \subseteq V(G)$ and an integer $p \in \mathbb{Z}_{\geq 0}$.
Task:	Find a $X_1, \ldots, X_r$-multiway cut of size at most p or decide that none exits.

The Directed Multiway Cut was first shown to be fixed-parameter tractable by Chitnis, Hajiaghayi, and Marx [CHM13], who later, together with Cygan, improved significantly on the run-time of their main routine [CCHM15].

Theorem 3.59 ([CHM13][CCHM15])**.** *Let G be a directed graph, let $X_1, \ldots, X_r \subseteq V(G)$, and let $p \in \mathbb{Z}_{\geq 0}$. The Directed Multiway Cut problem for $(G, X_1, \ldots, X_r, p)$ can be solved in $2^{\mathcal{O}(p^2)} \cdot \mathrm{poly}(n)$ time. Further, an $X_1, \ldots, X_r$-multiway cut of size at most p can be found in the same time, if it exists.*

Lemma 3.60. *There is an algorithm that solves instances (G, t, k, ℓ) of Directed Long Cycle Hitting Set with many Short Cycles in time $2^{\mathcal{O}(k^2)} \cdot \mathrm{poly}(n)$ by a set of size $2^{\mathcal{O}(k^2 + \log \ell)} \log^2 n$.*

Proof. Let $\mathcal{C}$ be the strongly connected components of $G - t$. If either there is a $C \in \mathcal{C}$ with more than $k(k+1)+1$ clusters, or there are more than k components in $\mathcal{C}$ with at least two clusters, we apply shadow covering combined with Corollary 3.56.

Let $\mathcal{F}$ be the set of all cycles in G of length more than ℓ. Then the graphs of $\mathcal{F}$ are t-connected and the long cycle hitting sets of our instance are exactly the $\mathcal{F}$-transversals. Fix an arbitrary long cycle hitting set S of size at most k. By Theorem 2.9 we can compute a set $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_p\}$ with $p = 2^{\mathcal{O}(k^2)} \log^2 n$ in time $2^{\mathcal{O}(k^2)} \cdot \text{poly}(n)$ such that for some $i \in \{1, \ldots, p\}$ we have that $S \cap Z_i = \emptyset$ and Z_i contains the shadow of S with respect to t. Start with $\mathcal{S}_{\mathsf{intersect}} = \emptyset$. For every set Z_i we apply Corollary 3.56 and add the resulting set $\mathcal{S}_{\mathsf{many\ clusters}} \subseteq V(G)$ of size at most $(k^2+k+1)(\ell-1)$ to $\mathcal{S}_{\mathsf{intersect}}$. Afterwards we return the set $\mathcal{S}_{\mathsf{intersect}}$.

Otherwise, there are at most k components in $\mathcal{C}$ with at least two clusters and each of these has at most $k(k+1)+1$ clusters. Moreover, we can apply Lemma 3.57 to get a set $\mathcal{S}_{\mathsf{few\ clusters}}$ of size at most $2^{\mathcal{O}(k+\log \ell)}$ that either intersects all long cycle hitting sets of size $\leq k$ or for every $C \in \mathcal{C}$ the set S intersects all paths between different clusters of C in $G[C]$. Add this set $\mathcal{S}_{\mathsf{few\ clusters}}$ to $\mathcal{S}_{\mathsf{intersect}}$.

In a last step we compute a minimum size $L_1^C, \ldots L_r^C$-multiway cut M_C for every component $C \in \mathcal{C}$ that has clusters $L_1^C, \ldots L_r^C$ with $r > 1$. This we can do by n calls to Theorem 3.59. If the union $\bigcup_{C \in \mathcal{C}} M_C$ has size at most k, we add it to $\mathcal{S}_{\mathsf{intersect}}$.

The run-time and size bound follow directly from the involved lemmas. It remains to argue for correctness. If we could apply Corollary 3.56 we returned a correct solution as some Z_i did cover the shadow of S. Otherwise, either a long cycle hitting set of size at most k intersects $\mathcal{S}_{\mathsf{few\ clusters}}$ or for every $C \in \mathcal{C}$ the long cycle hitting set intersects all paths between different clusters of C in $G[C]$. In other words, in the latter case it is a $L_1^C, \ldots L_r^C$-multiway cut for every component $C \in \mathcal{C}$ that has clusters $L_1^C, \ldots L_r^C$ with $r > 1$. Thus, the union of our minimum size multiway cuts has size at most k. Consider now any such union $M = \bigcup_{C \in \mathcal{C}} M_C$. We claim that M is a long cycle hitting set.

Take any cycle R of length more than ℓ in G. By Lemma 3.49, our cycle R contains a path between two different clusters of some $C \in \mathcal{C}$. As M_C is an $L_1^C, \ldots L_r^C$-multiway cut, our cycle R is intersected by M. Thus, M is a long cycle hitting set of size at most k. In particular, there is a long cycle hitting set of size at most k intersecting $\mathcal{S}_{\mathsf{intersect}} \supseteq M$. □

3.2.7 Putting Everything Together

This section combines the previous sections to an overall algorithm solving Directed Long Cycle Hitting Set. We do this in two steps. First we combine Lemma 3.27, Lemma 3.33, and Lemma 3.39 to reduce Directed Long Cycle Hitting Set to Directed Long Cycle Hitting Set from Singular Solution. Then in a second step we solve Directed Long Cycle Hitting Set from Singular Solution by combining Lemma 3.44 and Lemma 3.60.

Corollary 3.61. *An instance (G, k, ℓ) of Directed Long Cycle Hitting Set can be solved in time*

$$2^{\mathcal{O}(\ell^6 + k^3\ell + k^4 \log k)} \log^{\mathcal{O}(k^2)} n \left(n^2 A_{\mathit{singular}}(n, k, \ell) + \text{poly}(n) \right),$$

where $A_{\mathit{singular}}(n, k, \ell)$ is the run-time of an algorithm for Directed Long Cycle Hitting Set from Singular Solution on instances (G', t, k', ℓ) with $|V(G')| \leq n$ and $k' \leq k$.

Proof. Note that throughout the algorithm, we always have $|T| \leq k+1$. Thus, we can simplify Lemma 3.39 to the statement that Isolating Long Cycle Hitting Set Intersection can be solved in time $2^{\mathcal{O}(k \log k)} n \cdot A_{\mathsf{singular}}(n,k,\ell)$ by a set of size $2^{\mathcal{O}(k \log k)}$. Plugging this into Lemma 3.33 gives us an algorithm for Disjoint Directed Long Cycle Hitting Set Compression Intersection with run-time

$$2^{\mathcal{O}(\ell^6+k^2\ell+k^3 \log k)} \log^{\mathcal{O}(k)} n \left(n A_{\mathsf{singular}}(n,k,\ell) + \mathrm{poly}(n)\right)$$

that returns a set of size

$$2^{\mathcal{O}(k^2\ell+k^3 \log k)} \log^{\mathcal{O}(k)} n.$$

Finally, using this algorithm in Lemma 3.27 gives the claimed run-time. □

Corollary 3.62. *There is an algorithm that solves instances (G,t,k,ℓ) of Directed Long Cycle Hitting Set from Singular Solution in time*

$$\left(2^{\mathcal{O}(k^3+\log \ell)} \log^{2k} n + 2^{\mathcal{O}(\ell)}\right) \mathrm{poly}(n).$$

Proof. Use Lemma 3.60 in Lemma 3.44. □

Finally, we can combine above statements to get our main result.

Theorem 3.63. *There is an algorithm solving Directed Long Cycle Hitting Set with run-time $2^{\mathcal{O}(\ell^6+k^3\ell+k^4 \log k)} \mathrm{poly}(n)$.*

Proof. By combining Corollary 3.61 and Corollary 3.62, we get that Directed Long Cycle Hitting Set can be solved in time

$$\begin{aligned} &2^{\mathcal{O}(\ell^6+k^3\ell+k^4 \log k)} \log^{\mathcal{O}(k^2)} n \left(n^2 \left(2^{\mathcal{O}(k^3+\log \ell)} \log^{2k} n + 2^{\mathcal{O}(\ell)}\right) \mathrm{poly}(n) + \mathrm{poly}(n)\right) \\ = \; &2^{\mathcal{O}(\ell^6+k^3\ell+k^4 \log k)} \log^{\mathcal{O}(k^2)} n \, \mathrm{poly}(n). \end{aligned}$$

Using Lemma 2.10 we get $\log^{\mathcal{O}(k^2)} n = 2^{\mathcal{O}(k^2 \log k)} \mathrm{poly}(n)$ and the theorem follows. □

3.3 Reductions for Directed Long Cycle Vertex Deletion

In this section we deal with reductions arising in the context of Directed Long Cycle Hitting Set. First, we will show that the arc deletion version can be reduced to an instance of the vertex deletion version of the same size. This is simply done by taking the directed line graph.

Theorem 3.64. *There exists a polynomial parameter transformation from instances (G,k,ℓ) of the arc-deletion variant of Directed Long Cycle Hitting Set to an instance (G',k,ℓ) of the vertex-deletion variant of Directed Long Cycle Hitting Set.*

Proof. Given an instance (G,k,ℓ) of the arc-deletion variant of Directed Long Cycle Hitting Set, create a directed graph G' from G by letting G' be the directed line graph of G. I.e. we set $V(G') = A(G)$ and

$$A(G) = \{((v_1,v_2),(v_3,v_4)) \mid (v_1,v_2),(v_3,v_4) \in A(G) \text{ with } v_2 = v_3\}.$$

We further set $k' = k$ and $\ell' = \ell$; then (G', k', ℓ') is an instance of the vertex-deletion variant of Directed Long Cycle Hitting Set.

In the forward direction, let S be a set of at most k arcs of G such that in $G - S$ every simple cycle has length at most ℓ. Then, since every cycle of G is mapped to a cycle of G' with the same length, the set S' of vertices to which the arcs in S get mapped to in G' is such that $G' - S'$ does not have any simple cycles of length strictly more than ℓ.

In the backward direction, let S' be a set of at most k vertices of G' such that every simple cycle of $G' - S'$ has length at most ℓ. Then, since every cycle of G' is mapped to a cycle of G with the same length, the set S of arcs to which the vertices in S' get mapped to in G is such that $G - S$ does not have any simple cycles of length strictly more than ℓ. □

There is also a reduction in the other direction by splitting each vertex v into v^- and v^+ and adding an arc from v^- to v^+. Arcs (u, v) are replaced by (u^+, v^-) and are made undeletable by taking enough copies.

Theorem 3.65. *There exists a polynomial parameter transformation from instances (G, k, ℓ) of the vertex-deletion variant of Directed Long Cycle Hitting Set to an instance $(G', k, 2\ell)$ of the arc-deletion variant of Directed Long Cycle Hitting Set.*

Proof. Given an instance (G, k, ℓ) of the vertex-deletion variant of Directed Long Cycle Hitting Set, create a directed graph G' from G by splitting each vertex $v \in V(G)$ into two vertices v^+, v^-, adding an arc from v^- to v^+, and connecting all in-neighbors u of v in G by $k+1$ parallel arcs from u^+ to v^- in G', and all out-neighbors u of v in G by an arc from v^+ to u^- in G'. In other words, $V(G') = \{v^+, v^- \mid v \in V(G)\}$ and $A(G') = \{(v^-, v^+) \mid v \in V(G)\} \cup \{(u^+, v^-)^k \mid u \in N_G^-(v), v \in V(G)\} \cup \{(v^+, u^-)^k \mid u \in N_G^+(v), v \in V(G)\}$. We further set $k' = k$ and $\ell' = 2\ell$; then (G', k', ℓ') is an instance of the arc-deletion variant of Directed Long Cycle Hitting Set.

In the forward direction, let S be a set of at most k vertices such that any simple cycle of $G - S$ has length at most ℓ. We let $S' = \{(v^-, v^+) \mid v \in S\}$, it follows that S is a set of at most k arcs such that any simple cycle of $G' - S'$ has length at most $\ell' = 2\ell$.

In the backward direction, let S' be a set of at most k' arcs such that $G' - S'$ does not have any simple cycles of length strictly more than $\ell' = 2\ell$. We may assume that S' only contains arcs of the form (v^-, v^+) for some vertex $v \in V(G)$, as S' contains at most k arcs and there are $k + 1$ parallel arcs between any two vertices of G' that correspond to distinct vertices of G. Therefore, the set $S = \{v \mid (v^-, v^+) \in S'\}$ is a set of at most k vertices in G such that $G - S$ does not have any simple cycles of length more than ℓ. □

It is clear that the Directed Long Cycle Hitting Set problem generalizes the Directed Feedback Vertex Set problem for parameter $\ell = 0$. We now show that this problem also generalizes the Feedback Vertex Set in Mixed Graphs problem, but this time for the parameter $\ell = 2$. A mixed graph $G = (V, A, E)$ is a graph on a vertex set V that has a set of directed arcs A, as well as a set of undirected edges E. In the Feedback Vertex Set in Mixed Graphs problem, we are given as input a mixed graph $G = (V, A, E)$, where each arc in A can be traversed only along

its direction and each edge in E can be traversed in both directions, together with an integer k, and we are seeking a set S of at most k vertices such that $G - S$ does not contain any cycles.

Theorem 3.66. *There is a polynomial parameter transformation from instances* $(G = (V, A, E), k)$ *of* Feedback Vertex Set in Mixed Graphs *to an instance* $(G', k, 2)$ *of* Directed Long Cycle Hitting Set.

Proof. Let $(G = (V, A, E), k)$ be an instance of the Feedback Vertex Set in Mixed Graphs problem. We can assume that G is loop-free, as vertices with loops need to be removed in any solution. Now, we will create a directed graph G' such that $(G', k, 2)$ as instance of Directed Long Cycle Hitting Set has a solution if and only if (G, k) has one. For this we replace every arc $a \in A$ by a path P_a of length two in the same direction. Afterwards we replace all edges depending on the existence of other arcs/edges between its endpoints: If for an edge $e = \{v, w\} \in E$ the only arc/edge between v and w in G is e (i.e. $G[\{v, w\}]$ contains a cycle) we replace e by arcs $\overrightarrow{e} = (v, w)$ and $\overleftarrow{e} = (w, v)$ in both directions. Otherwise, we replace $e = \{v, w\} \in E$ by two paths $\overrightarrow{P_e}$ and $\overleftarrow{P_e}$ of length two in both directions. The resulting graph is G'.

Let now S be a solution to (G, k). Then the only cycles in $G' - S$ must be those formed by replacing an edge with forward and backward paths/arcs. Only the edges replaced by paths can form cycles of length longer than two. But those edges had another arc/edge between their endpoints, thus forming a cycle in G. As S intersects this cycle, only cycles of length two survive in $G' - S$.

For the opposite direction, let S' be a solution to $(G', k, 2)$. We can assume that $S' \subseteq V(G)$ as all other vertices lie in the middle of paths (i.e. have degree two) and we could include an endpoint of the path instead. As cycles in G get replaced by longer cycles in G', $G - S'$ contains only cycles of length two which don't get longer when transforming to G'. These cycles can only contain two edges between the same vertices (as arcs get longer). But these get replaced by paths so the cycles would have length at least four in G' and would be deleted by S'. Thus, $G - S'$ contains no cycles. □

As the (undirected) Feedback Vertex Set problem is a special case of Feedback Vertex Set in Mixed Graphs, we have that Directed Long Cycle Hitting Set also generalizes Feedback Vertex Set for $\ell \geq 2$.

Chapter 4

Bounded Size Strongly Connected Component Vertex Deletion

In this chapter we want to consider the Bounded Size Strongly Connected Component Vertex Deletion problem. Given a directed graph G and two integers $k, \ell \in \mathbb{Z}_{\geq 0}$ it asks whether there is a set S of at most k vertices such that every strongly connected component of $G - S$ has at most s vertices.

Bounded Size Strongly Connected Component Vertex Deletion	
Instance:	A graph G and two integers $k, s \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq V(G)$ of size at most k such that for every strongly connected component C of $G - S$ holds $\lvert V(C)\rvert \leq s$ or decide that no such set exists.

The main result of this chapter is that Bounded Size Strongly Connected Component Vertex Deletion is fixed-parameter tractable in the deletion size k and the target component size s.

Theorem 4.1. *Let G be a directed graph with n vertices and $k, s \in \mathbb{Z}_{\geq 0}$. There is an algorithm that solves the Bounded Size Strongly Connected Component Vertex Deletion instance (G, k, s) in time $2^{2k+1}(ks + k + s)! \cdot \mathcal{O}(n^4)$.*

In particular, the run-time has the same asymptotic dependence on k as does the algorithm by Chen et al. [CLL+08] for the Directed Feedback Vertex Set problem, which corresponds to the special case $s = 1$.

Since its first publication this algorithm has been improved by the work of Neogi, Ramanujan, Saurabh and Sharma [NRSS20] who gave an $2^{\mathcal{O}(k(\log k + \log s))}\operatorname{poly}(n)$ algorithm instead of the above $2^{\mathcal{O}((k+s)(\log k + \log s))}\operatorname{poly}(n)$ algorithm.

Moreover, we will show that for the vertex deletion variant and the arc deletion variant (both with parameters k and s), there are polynomial parameter transformations in both directions. Therefore also Bounded Size Strongly Connected Component Arc Deletion is fixed-parameter tractable when parameterized in k and s.

This result is joint work with Dániel Marx and Matthias Mnich [GMM20c]. An extended abstract of this work has previously appeared at CIAC 2019 [GMM19].

4.1 The Fixed-Parameter Algorithm

4.1.1 Applying Disjoint Compression

Our algorithm first makes use of the techniques "Iterative Compression" and "Disjoint Compression" from Chapter 2. For this let for a fix $s \in \mathbb{Z}_{\geq 0}$ the graph family $\mathcal{C}_s$ be the family of all directed graphs, whose strongly connected components have at most s vertices. Note that this graph family is non-empty as it always contains the empty graph and hereditary as for any graph $G \in \mathcal{C}_s$ and any subgraph G' of G, the strongly connected components of G' are subgraphs of the strongly connected components of G. Thus, any strongly connected component of G' has at most s vertices and $G' \in \mathcal{C}_s$.

For this choice of $\mathcal{C}_s$, our Bounded Size Strongly Connected Component Vertex Deletion problem is exactly the $\mathcal{C}_s$-Vertex Deletion problem. By $\mathcal{C}_s$ being non-empty and hereditary, we can apply Corollary 2.3 to reduce to the following problem.

Disjoint Bounded Size Strongly Connected Component Deletion Compression	
Instance:	A graph G, a vertex set $T \subseteq V(G)$ and two integer $k, s \in \mathbb{Z}_{\geq 0}$ s.t. for every strongly connected component C of $G - T$ holds $\lvert V(C)\rvert \leq s$.
Task:	Find a set $S \subseteq V(G) \setminus T$ of size at most k such that for every strongly connected component C of $G - S$ holds $\lvert V(C)\rvert \leq s$ or decide that no such set exists.

The reduction procedure is summarized in the following lemma.

Lemma 4.2. *An instance (G, k, s) of Bounded Size Strongly Connected Component Vertex Deletion can be solved in time*

$$\mathcal{O}(n \cdot 2^{k+1} \cdot A_{\textit{disjoint compression}}(n, k+1, k)),$$

where $A_{\textit{disjoint compression}}(n, t, k)$ is the run-time of an algorithm for Disjoint Bounded Size Strongly Connected Component Deletion Compression on instances (G', T', k', s) with $\lvert V(G')\rvert \leq n$, $\lvert T\rvert \leq t$ and $k' \leq k$.

Proof. Apply Corollary 2.3 to $\mathcal{C}_s$ which is hereditary and non-empty. □

4.1.2 Reduction to Skew Separator Problem

In this section, we solve the Disjoint Bounded Size Strongly Connected Component Deletion Compression problem by reducing it to the problem of finding a small "skew separator" in one of a bounded number of reduced instances. Let us briefly introduce the Skew Separator problem.

Definition 4.3. *Let G be a directed graph, and let $\mathcal{X} = (X_1, \ldots, X_t), \mathcal{Y} = (Y_1, \ldots, Y_t)$ be two ordered collections of $t \geq 1$ subsets of $V(G)$. A* skew separator *S for $(G, \mathcal{X}, \mathcal{Y})$ is a vertex subset of $V(G) \setminus \bigcup_{i=1}^{t}(X_i \cup Y_i)$ such that for any index pair (i, j) with $1 \leq j \leq i \leq t$, there is no path from X_i to Y_j in the graph $G - S$.*

Skew Separator	
Instance:	A graph G, two ordered collections $\mathcal{X}, \mathcal{Y}$ of subsets of $V(G)$, and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find a skew separator $S \subseteq V(G) \setminus \bigcup_i (X_i \cup Y_i)$ of size at most k or decide that no such separator exists.

The Skew Separator problem was introduced by Chen et al. [CLL+08] and they showed that it is fixed-parameter tractable when parameterized in k.

Theorem 4.4 ([CLL+08, Thm. 3.5])**.** *There is an algorithm solving Skew Separator in time $4^k k \cdot \mathcal{O}(n^3)$ for n-vertex directed graphs G.*

Now we want to reduce from Disjoint Bounded Size Strongly Connected Component Deletion Compression to Skew Separator. Note that, by definition, we have that every strongly connected component of $G - T$ has size at most s. Moreover, we can assume that every strongly connected component of $G[T]$ has size at most s, as otherwise there is no solution S of (G, k, s) that is disjoint from T. Let $\{t_1, \ldots, t_{k+1}\}$ be an arbitrary labeling of the vertices in T.

Lemma 4.5. *Let (G, T, k, s) be an instance of Disjoint Bounded Size Strongly Connected Component Deletion Compression. Then there is an algorithm that in time $(ks+s-1)! \cdot \mathcal{O}(n)$ computes a collection $\mathcal{C}$ of at most $(ks+s-1)!$ vectors such that:*

- *every vector $C = (C_1, \ldots, C_{k+1})$ has length $k+1$, and*
- *for every $h = 1, \ldots, k+1$, we have $t_h \in C_h \subseteq V(G)$, and*
- *if there is a solution S to (G, T, k, s), then there is a vector $C \in \mathcal{C}$ such that the strongly connected component of $G - S$ containing t_h is exactly $G[C_h]$ for $h = 1, \ldots, k+1$.*

Proof. Fix a hypothetical solution S of (G, T, k, s) that is disjoint from T. The algorithm computes, for each vertex $t_h \in T$, a set $C_h \ni t_h$ of at most s vertices such that C_h induces a strongly connected component of $G - S$. These sets C_h must exist as S is required to be disjoint from T.

Notice that the algorithmic computation of C_h must only depend on t_h but not on S. Vertices in C_h (other than t_h) may or may not belong to T and in particular it can be that $t_{h'} \in C_h$ for some $h' \in \{1, \ldots, k+1\} \setminus \{h\}$. Thus, for distinct $t_h, t_{h'} \in T$, sets C_h and $C_{h'}$ possibly overlap.

The algorithm constructs the sets C_h iteratively by a simple branching algorithm along the following lines. It starts with an initial set $C_h^0 = \{t_h\}$ and a guessed set $S = \emptyset$. For $i \geq 0$, suppose that it has already constructed a set C_h^i that must be a subset of C_h, and we want to either extend C_h^i to a proper superset C_h^{i+1} or decide that $C_h = C_h^i$. If there is a path P in $G - S$ of length at least two and length at most $s - |C_h^i|$ whose end vertices are in C_h^i and whose internal vertices are all outside C_h^i, then it branches into two cases:

- either some internal vertex u of P belongs to the deletion set S (add u to S), or
- the entire path P belongs to the candidate set C_h^{i+1} (add $V(P)$ to C_h^i).

Thus, in each branching step, either the size of S strictly increases, or the size of C_h^i strictly increases. Note that the size of S is bounded by k, and the size of $C_h^i \subseteq C_h$ is bounded by s. Hence, in the first branch, adding u to S implies that the budget $k-|S|$ strictly decreases to $k-|S\cup\{u\}|$, whereas in the second branch, adding $V(P)$ to C_h^i strictly decreases the budget $s-|C_h^i|$ to $s-|C_h^i\cup V(P)|$. The latter is a decrease by at least one since P contains an internal vertex not in C_h^i. We repeat this branching until the size of S reaches the limit of k or the size of C_h^i reaches the limit of s, or if there are no paths left of length at most $s-|C_h^i|$ with both end vertices inside C_h^i and all internal vertices outside C_h^i. At this point, the set C_h^i will not be further extended, and $C_h := C_h^i$ is a candidate set for t_h. We then continue with t_{h+1} and C_{h+1}. This completes the algorithm description.

Next, we analyze the run-time of the algorithm. The run-time mainly depends on the number of branches. For a single branch node, the number of branches originating from this node can be bounded by a function $f(k',q)$ for the remaining value k' of k at this node and $q=\sum_{h=1}^{k+1}(s-|C_h^i|)$ is the sum of the remaining capacities of the C_h's. By the above branching algorithm, this function satisfies the recursion

$$f(k,q) \leq (s-|C_h^i|)f(k-1,q)+f(k,q-1),$$

as in the first branch there are at most $s-|C_h^i|$ choices for vertex u each of which reduces the budget of $k-|S|$ by 1, and one branch which reduces the budget of q by the number of internal vertices of P which is at least 1.

To obtain an upper bound on the growth of f, we first notice that $f(0,q)=1$ for all $q\in\mathbb{Z}_{\geq 0}$ and $f(k,0)=1$ for all $k\in\mathbb{Z}_{\geq 0}$. We then claim that $f(k,q)\leq(q+k)!$, since by induction for $k,q\in\mathbb{Z}_{\geq 0}$ it holds

$$\begin{aligned} f(k,q) &\leq (s-|C_i^h|)f(k-1,q)+f(k,q-1) \\ &\leq (s-|C_i^h|)(q+k-1)!+(q-1+k)! \\ &= (s-|C_i^h|+1)(q+k-1)! \\ &= \left(\frac{s-|C_h^i|+1}{q+k}\right)(q+k)! \\ &\leq (q+k)!, \end{aligned}$$

where in the last inequality we used that $q\geq s-|C_i^h|$ and $k\geq 1$. Hence, the search tree has at most $(q+k)!$ leaves and each leaf corresponding to some vector $C\in\mathcal{C}$. The initial capacity q satisfies $q=(k+1)(s-1)$, and thus $|\mathcal{C}|\leq(ks+s-1)!$. In each branching step we need to find a edge-wise shortest $C_h^i\to C_h^i$-path which is not a single edge, which can be achieved by a modified breath-first search in linear time. Also all set operations in a branching step can be done in linear time. Thus, since each branching step can be executed in linear time, the search tree (and hence the set $\mathcal{C}$) can be constructed in time $(q+k)!\cdot\mathcal{O}(n)$. Thus, the overall run-time is $(q+k)!\cdot\mathcal{O}(n)=(ks+s-1)!\cdot\mathcal{O}(n)$.

As the number of branches also bounds the number of produced vectors, we have $|\mathcal{C}|\leq(ks+s-1)!$. □

Armed with Lemma 4.5, we can restrict our search for a solution S of (G,T,k,s) to those that additionally are "compatible" with a vector C in $\mathcal{C}$. Formally, we call a solution S of (G,T,k,s) *compatible* with a vector $C=(C_1,\ldots,C_{k+1})\in\mathcal{C}$ if the strongly connected component of $G-S$ containing t_h is exactly C_h for $h=1,\ldots,k+1$.

To determine whether for a given vector $C = (C_1, \ldots, C_{k+1}) \in \mathcal{C}$ some solution S of (G, T, k, s) that is compatible with C exists, we create several instances of the Skew Separator problem. To this end, note that if two sets $C_h, C_{h'}$ for distinct $t_h, t'_h \in T$ overlap, then we must actually have $C_h = C_{h'}$ (and $t_h, t'_h \in C_h$) or no solution S can be compatible with them. So for each set C_h we choose exactly one (arbitrary) *representative T-vertex* among all vertices of T in C_h with consistent choice among overlapping (and thus equal) C_h's. Let $T' \subseteq T$ be the set of these representative vertices. Now we generate precisely one instance $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'}, k)$ of Skew Separator for each permutation σ' of T'. The graph G' is the same in all these instances, and is obtained from G by replacing each unique set C_h by two vertices t_h^+, t_h^- (where t_h is the representative of C_h), and connecting all in-neighbors of C_h in G by an in-arc to t_h^+ and all out-neighbors of C_h in G by an arc outgoing from t_h^-. This way also arcs of the type (t_j^-, t_h^+) are added but none of type (t_j^-, t_h^-), (t_j^+, t_h^-) or (t_j^+, t_h^+). Notice that this operation is well-defined and yields a simple directed graph G', even if $t_{h'} \in C_h$ for some distinct h, h'.

Next we iterate over all possible permutation σ' of $\{1, \ldots, |T'|\}$. For every such permutation σ' we define the ordered collections $\mathcal{X}_{\sigma'}$ and $\mathcal{Y}_{\sigma'}$ of "sources" and "sinks". For this let $\mathcal{X}_{\sigma'} = (t^-_{\sigma'(1)}, \ldots, t^-_{\sigma'(|T'|)})$ and let $\mathcal{Y}_{\sigma'} = (t^+_{\sigma'(1)}, \ldots, t^+_{\sigma'(|T'|)})$.

This way we generate for every $C \in \mathcal{C}$ at most $|T'|! \leq |T|! = (k+1)!$ instances $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'}, k)$ of the Skew Separator problem. We now prove that at least one of the created instances is equivalent to our original instance (G, T, k, s) when restricted to C-compatible solutions.

Lemma 4.6. *If an instance (G, T, k, s) admits a solution S compatible with C for which $(C_{\sigma'(1)}, \ldots, C_{\sigma'(|T'|)})$ can be extended to a topological order of the strongly connected components of $G' - S$, then S forms a skew separator of size at most k for $(G, \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'})$.*

Proof. Suppose, for the sake of contradiction, that the claim is false. Then one of the two following cases must hold:

1. For two vertices $t_h, t_{h'} \in T'$ with $\sigma'(h) < \sigma'(h')$, there would be a path P_1 from $t^-_{h'}$ to t^+_h in $G' - S$. This corresponds to a $C_{h'} \to C_h$ path in $G - S$. Either there is also a $C_h \to C_{h'}$ path in G meaning that $C_h = C_{h'}$ in contradiction to our choice of T' or the topological order σ' of strongly connected components was incorrect (as $C_{h'}$ must be before C_h).

2. The in-vertex t_h^- of the strongly connected component containing v_h would be reachable by a path P_2 from the out-vertex t_h^+ of this strongly connected component in the graph $G' - S$. Then the strongly connected component of t_h would contain the internal vertices of P_2, contradicting that the strongly connected component of $G' - S$ containing t_h is exactly C_h (by definition of S being compatible with C. □

Lemma 4.7. *Conversely, if S is a skew separator of $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'})$ with size at most k, then S is a solution of (G, T, k, s).*

Proof. By choice of $C, \mathcal{X}$ and $\mathcal{Y}$, we have that $S \subseteq V(G) \setminus \bigcup_{i=1}^{t'}(X_i \cup Y_i) \subseteq V(G) \setminus T$. That means that S is disjoint of T. Now suppose, for the sake of contradiction, that S is not a solution for (G, T, k, s). Then there is some strongly connected component Q in $G - S$ of size more than s. By abuse of notation let $C = \bigcup_{h=1}^{k+1} C_h$.

Since neither $G[C]$ (by choice of C) nor $V(G) - C$ (as subgraph of $G[V(G) \setminus T]$) contain strongly connected components of size greater than s, this component Q must contain vertices from both C and $V(G) \setminus C$. Let K be a closed walk of Q that intersects both $G[C]$ and $G[V(G) \setminus C]$. Such a closed walk K must exist by Q being strongly connected.

We consider two cases:

1. The closed walk K intersects a single unique component C_h. Then all other vertices of K are in $V(G) \setminus C_h$. Let t_h be the representative of C_h. As K intersects $G[V(G) \setminus C]$, K leaves and enters C_h at least once. This means that there is a walk P_1 in $G' - S$ that starts with the vertex t_h^- and ends with the vertex t_h^+, and all internal vertices of P_1 (of which there is at least one) are outside T'. But this contradicts the assumption that S is a skew separator for the tuple $(G', (t^-_{\sigma'(1)}, \ldots, t^-_{\sigma'(|T'|)}), (t^+_{\sigma'(1)}, \ldots, t^+_{\sigma'(|T'|)}))$ that should cut all walks from t_h^- to t_h^+.

2. The closed walk K intersects several different components C_h. In this case, denote by $(C_{h_1}, \ldots, C_{h_d}, C_{h_1})$ the components in the order we encounter them when traversing along the walk K for some $d > 1$. Let $(t_{h_1}, \ldots, t_{h_d}, t_{h_1})$ be the corresponding representative vertices. Then there must be an index j such that h_j occurs after $h_{j+1 \pmod{d+1}}$ in $(\sigma'(1), \sigma'(2), \ldots \sigma'(|T'|)$. Consider the subpath P_2 of K that starts from C_{h_j}, ends in $C_{h_{j+1}}$ and has its interior disjoint from both. Since all internal vertices on P_2 (by definition of P_2) are not in any C_h, all such internal vertices of P_2 must be from $G - S - \cup_{i=1}^{|T'|}(X_i \cup Y_i)$, and the path P_2 corresponds to a path P_2' in the graph $G' - S$ that starts from vertex $t^-_{h_j}$ and ends at vertex $t^+_{h_{j+1}}$. Again, this contradicts the assumption that S is a skew separator for the tuple $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'})$.

This proves that the skew separator S must be a solution for (G, T, k, s). □

In summary, we have reduced a single instance (G, T, k, s) of Disjoint Bounded Size Strongly Connected Component Deletion Compression to at most $|\mathcal{C}| \cdot |T'|!$ many instances $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'}, k)$ of the Skew Separator problem, where each such instance corresponds to a permutation σ' of T'. The reduction just described implies that:

Lemma 4.8. *An input (G, k, s, T) to the Disjoint Bounded Size Strongly Connected Component Deletion problem is a "yes"-instance if and only if at least one of the instances $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'}, k)$ is a "yes"-instance for the Skew Separator problem.*

So we invoke the algorithm of Theorem 4.4 for each of the instances $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'}, k)$. If at least one of them is a "yes"-instance then (G, T, k, s) is a "yes"-instance, otherwise (G, T, k, s) is a "no"-instance. Hence, we conclude that Disjoint Bounded Size Strongly Connected Component Deletion Compression is fixed-parameter tractable with respect to the joint parameter $k + s$, and so is Size Strongly Connected Component Vertex Deletion by Lemma 4.2.

For the overall run-time, note that Lemma 4.2 provides a factor of $2^{k+1}\mathcal{O}(n)$ to the run-time. The generation of $\mathcal{C}$ takes time $(ks+s-1)! \cdot \mathcal{O}(n)$ and produces a set $\mathcal{C}$ of size $(ks + s - 1)!$ by Lemma 4.5. The iteration over permutations of $|T'|$ provides another

factor of $|T'|!$ to the run-time. Finally, the algorithm for solving Skew Separator runs in time $4^k k\mathcal{O}(n^3)$ by Theorem 4.4. The overall run-time of the algorithm is thus bounded by

$$\begin{aligned} 2^{k+1}\mathcal{O}(n) \cdot |\mathcal{C}| \cdot |T'|! \cdot 4^k k\mathcal{O}(n^3) &= (ks+s-1)! \cdot (k+1)! \cdot 2^{2k+1} \cdot \mathcal{O}(n^4) \\ &= 2^{2k+1}(ks+k+s)! \cdot \mathcal{O}(n^4). \end{aligned}$$

This completes the proof of Theorem 4.1.

4.2 Reductions between Vertex and Arc Version

In this section we prove the existence of polynomial parameter transformations between Bounded Size Strongly Connected Component Arc Deletion and Bounded Size Strongly Connected Component Vertex Deletion in both directions. This implies that they are parameter equivalent and thus Bounded Size Strongly Connected Component Arc Deletion is also fixed-parameter tractable in $k+s$.

Polynomial Parameter Transformation from Arc to Vertex Version

Our first transformation reduces an instance of Bounded Size Strongly Connected Component Arc Deletion to an instance of Bounded Size Strongly Connected Component Vertex Deletion. While our transformation keeps the parameter k constant, the parameter s increases to $(k+1)s^3$. This is due to a replacement of all vertices by complete graphs of size roughly ks^2, therefore increasing the size of eligible components.

Lemma 4.9. *Given an instance (G,k,s) of Bounded Size Strongly Connected Component Arc Deletion we can compute in polynomial time an equivalent instance (G',k',s') of Bounded Size Strongly Connected Component Vertex Deletion with $k'=k$ and $s'=(k+1)s^3$.*

Proof. We first bound the number of parallel arcs in G. Note that if there are more than $k+1$ arcs between a pair of vertices running in the same direction, we can remove additional arcs as at least one of these arcs remains after the removal of k arcs. Thus we can restrict ourselves to instances with at most $k+1$ parallel arcs per ordered vertex pair. In such directed graphs any subgraph with at most s vertices has at most $s_a := (k+1)s(s-1)$ arcs. The idea is now to subdivide the arcs by a vertex and replace the original vertices by complete directed graphs of size s_a+k+1. Then the inclusion of an original vertex into a strongly connected component has more impact than any of the artificial vertices needed to subdivide the arcs. Formally, we define our new directed graph G' as follows:

$$\begin{aligned} V(G') &= \{v_i \mid v \in V(G), 1 \le i \le s_a+k+1\} \cup \{u_a \mid a \in A(G)\}, \\ A(G') &= \{(v_i,v_j) \mid v \in V(G), 1 \le i,j \le s_a+k+1, i \ne j\} \\ &\quad \cup \{(v_i,u_a) \mid a=(v,w) \in A(G), 1 \le i \le s_a+k+1\} \\ &\quad \cup \{(u_a,w_i) \mid a=(v,w) \in A(G), 1 \le i \le s_a+k+1\}. \end{aligned}$$

Finally, we set $s' = s(s_a+k+1)+s_a = (k+1)s^3$ and get the resulting instance (G', k, s') of Bounded Size Strongly Connected Component Vertex Deletion. It remains to show that the two instances are indeed equivalent.

For the forward direction, let S be a set of at most k arcs such that every strongly connected component of $G - S$ has at most s vertices. Let $S' = \{u_a \mid a \in S\}$. By construction, we have that $G' - S'$ is equivalent to applying above transformation to the graph $G - S$. As our transformation preserves connectedness we have a one to one correspondence between strongly connected components of $G' - S'$ and $G - S$. Let X' be a strongly connected component of $G' - S'$ and X it's corresponding set in $G - S$. We know that $A(G[X])$ has size at most $(k+1)|X|(|X|-1) \leq s_a$. Thus, X' contains at most s_a vertices of type u_a. Furthermore, there are at most $|X|(s_a+k+1) \leq s(s_a+k+1)$ vertices of type v_i. Hence, we have $|X'| \leq s_a + s(s_a+k+1) = s'$, and by $|S'| = |S| \leq k$ we know that S' is a valid solution to (G', k, s').

For the reverse direction, let S' be a set of at most k vertices such that every strongly connected component of $G' - S'$ has at most s' vertices. Then, we claim that the set $S = \{a \in A(G) \mid u_a \in S'\}$ is a solution to (G, k, s). Obviously, $|S| \leq |S'| \leq k$. We now want to show that the strongly connected components in $G - S$ do contain at most s vertices. As $s_a + k + 1 > k$ we know that for every $v \in V(G)$ at least one v_i remains in $G' - S'$. Because all v_i have the same neighbors, removing the vertices of type v_i from S' does not change connectivity of $G - S'$. Now again there is a one-to-one correspondence between the strongly connected components of $G - S$ and $G' - S'$. The strongly connected components of $G' - S'$ are missing at most k vertices of type v_i which are in S'. Let X be a strongly connected component in $G' - S'$. Let $W \subset V(G)$ be the set of all vertices $w \in V(G)$ in G such that X contains a vertex w_i. If $|W| > s$ then X contains at least $(s_a + k + 1)|W| - k \geq (s_a + k + 1)s + s_a + k + 1 - k = s' + 1$ vertices, a contradiction to the fact that S' was solution for (G', k, s'). Thus, $|W| \leq s$ and by the one to one correspondence of strongly connected components, we know that W is indeed a strongly connected component of $G - S$. As X was chosen arbitrary and all strongly connected components of $G - S$ have counterpart in $G' - S'$, this completes the proof. □

Polynomial Parameter Transformation from Vertex to Arc Version

Here we state a polynomial parameter transformation from Bounded Size Strongly Connected Component Vertex Deletion to Bounded Size Strongly Connected Component Arc Deletion. Note that, unlike the reduction in backwards direction, the parameter increase of s is only linear and does not depend on k.

Lemma 4.10. *Given an instance (G, k, s) of Bounded Size Strongly Connected Component Vertex Deletion we can compute in polynomial time an equivalent instance (G', k', s') of Bounded Size Strongly Connected Component Arc Deletion with $k' = k$ and $s' = 2s$.*

Proof. Given an instance (G, k, s) of Bounded Size Strongly Connected Component Vertex Deletion, create a directed graph G' from G by splitting each vertex $v \in V(G)$ into two vertices v^+, v^-, adding the arc from v^- to v^+, and connecting all in-neighbors u of v in G by $k+1$ parallel arcs from u^+ to v^- in G', and all out-neighbors u of v in G by $k+1$ parallel arcs from v^+ to u^- in G'.

In other words, we set

$$\begin{aligned} V(G') &= \{v^+, v^- \mid v \in V(G)\}, \\ A(G') &= \{(v^-, v^+) \mid v \in V(G)\} \\ &\quad \cup \{(u^+, v^-)^{k+1} \mid u \in N_G^-(v), v \in V(G)\} \\ &\quad \cup \{(v^+, u^-)^{k+1} \mid u \in N_G^+(v), v \in V(G)\} \ . \end{aligned}$$

We further set $k' = k$ and $s' = 2s$. Then (G', k', s') is an instance of Bounded Size Strongly Connected Component Arc Deletion. It remains to check equivalence of the instances.

In the forward direction, let S be a set of at most k vertices such that in $G - S$ every strongly connected component has at most s vertices. Let $S' = \{(v^-, v^+) \mid v \in S\}$ be the corresponding set of k arcs in G'. The number of vertices in each strongly connected component of $G' - S'$ is now exactly twice the number of vertices in its corresponding component in $G - S$. Therefore, every strongly connected component of $G' - S'$ consists of at most $s' = 2s$ vertices.

In the backward direction, let S' be a set of at most k arcs such that in $G' - S'$ every strongly connected component has at most $s' = 2s$ vertices. We first argue that we can change S' in such a way that it will only consist of arcs of the form (v^-, v^+) for some vertex $v \in V(G)$. As for all other arcs we inserted $k + 1$ parallel arcs, any deletion of k arcs, does not change the connectivity of the parallel arcs. Thus, we can remove those arcs from S'. This justifies the assumption that $S' = \{(v^-, v^+) \mid v \in V(G)\}$.) Now let $S = \{v \in V(G) \mid (v^-, v^+) \in S'\}$ be the set of at most k vertices in G corresponding to the arcs in S'. The number of vertices in each strongly connected component of $G - S$ is now exactly half the number of vertices in its corresponding component in $G' - S'$. Therefore, every strongly connected component of $G - S$ consists of at most $s = s'/2$ vertices. □

Chapter 5

1-Out-Regular Vertex Deletion

In this chapter we study the 1-Out-Regular Vertex Deletion problem. Here, for a given directed graph G and an integer k, we are to find a set $S \subseteq V(G)$ of size at most k such that every strongly connected component of $G - S$ is either a directed cycle or a single vertex. That is for every strongly connected component C, all vertices have the same out-degree in $G[C]$ and this out-degree is at most 1, hence the name.

Definition 5.1. *Let G be a directed graph and let $r \in \mathbb{Z}_{\geq 0}$ be an integer. A subgraph H of G is called* r-out-regular*, if for every $v \in V(H)$ we have $|\delta^+_H(v)| = r$.*

1-Out-Regular Vertex Deletion	
Instance:	A graph G and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq V(G)$ of size at most k such that every strongly connected component C of $G - S$ is r_C-out-regular with $r_C \leq 1$ or decide that no such set exists.

Our main result is that 1-Out-Regular Vertex Deletion is fixed-parameter tractable when parameterized in k.

Theorem 5.2. *Let G be an n-vertex directed graph G and $k \in \mathbb{Z}_{\geq 0}$. There is an algorithm solving the 1-Out-Regular Vertex Deletion instance (G, k) in time $2^{\mathcal{O}(k^3)} \operatorname{poly}(n)$.*

By giving a polynomial parameter transformation from 1-Out-Regular Arc Deletion to 1-Out-Regular Vertex Deletion, we extend this result to the arc deletion variant.

Since its first publication above algorithm has been improved by the work of Neogi, Ramanujan, Saurabh and Sharma [NRSS20] who gave a $2^{\mathcal{O}(k \log k)} \operatorname{poly}(n)$ algorithm.

The motivation for studying 1-Out-Regular Vertex Deletion comes from the Eulerian Strongly Connected Component Vertex Deletion problem. Here one wishes to delete k vertices from a directed graph G such that every strongly connected component of the remaining graph is Eulerian. As we will see, this problem is $W[1]$-hard when parameterized by k.

Theorem 5.3. *Eulerian Strongly Connected Component Vertex Deletion is* NP*-hard and* W[1]*-hard parameterized by solution size k, even for $(k+1)$-strongly connected, directed graphs.*

The main difficulty seems to lie in the fact that the family of graphs with Eulerian strongly connected components is not hereditary. Even for 2-out-regular graphs, the deletion of a vertex may lead to a graph that is not 2-out-regular anymore.

An example for this is the graph on five vertices $v_0, \ldots, v_4$ with the arcs (v_i, v_{i+1}) and (v_i, v_{i+2}) for all $i \in \{0, \ldots, 4\}$, where indices are taken modulo 5. Here the deletion of a single vertex v_i leaves the graph strongly connected (because of the arc (v_{i-1}, v_{i+1})), but the out-degree of v_{i-2} and v_{i-1} decreased by one, whereas the out-degree of v_{i+1} and v_{i+2} did not change.

However, the graph class of all graphs where every strongly connected component C is r_C-out-regular for some $r_C \leq 1$, is hereditary. Any deletion of vertices can only break the 1-out-regular strongly connected components, which are directed cycles, into strongly connected components that consist of isolated vertices, which are 0-out-regular. So 1-Out-Regular Vertex Deletion is the natural restriction of Eulerian Strongly Connected Component Vertex Deletion to hereditary graph classes.

This result is joint work with Dániel Marx and Matthias Mnich [GMM20c]. An extended abstract of this work has previously appeared at CIAC 2019 [GMM19].

5.1 The Fixed-Parameter Algorithm

5.1.1 Applying Disjoint Compression

As discussed in the introduction to this chapter, the graph class $\mathcal{C}_{\text{1-out-regular}}$, of graphs whose strongly connected components C are r_C-out-regular with $r_C \leq 1$, is hereditary. Moreover, it is non-empty, as it contains the empty graph.

For this choice of $\mathcal{C}_{\text{1-out-regular}}$, our 1-Out-Regular Vertex Deletion problem is exactly the $\mathcal{C}_{\text{1-out-regular}}$-Vertex Deletion problem. By $\mathcal{C}_{\text{1-out-regular}}$ being non-empty and hereditary, we can apply Corollary 2.3 to reduce to the following problem.

Disjoint 1-Out-Regular Vertex Deletion Compression	
Instance:	A graph G, a vertex set $T \subseteq V(G)$ and an integer $k \in \mathbb{Z}_{\geq 0}$ such that T is an inclusion-wise minimal set with the property that every strongly connected component C of $G - T$ is r_C-out-regular for some $r_C \leq 1$.
Task:	Find a set $S \subseteq V(G) \setminus T$ of size at most k such that every strongly connected component C of $G - S$ is r_C-out-regular with $r_C \leq 1$ or decide that no such set exists.

The reduction procedure is summarized in the following lemma.

Lemma 5.4. *An instance (G, k) of 1-Out-Regular Vertex Deletion can be solved in time $\mathcal{O}(n \cdot 2^{k+1} \cdot A_{\text{disjoint compression}}(n, k+1, k))$, where $A_{\text{disjoint compression}}(n, t, k)$ is the run-time of an algorithm for Disjoint 1-Out-Regular Vertex Deletion Compression on instances (G', T', k') with $|V(G')| \leq n$, $|T| \leq t$ and $k' \leq k$.*

Proof. Apply Corollary 2.3 to $\mathcal{C}_{\text{1-out-regular}}$ which is hereditary and non-empty. □

5.1.2 Covering of Shadows

As a next step, we want to apply the "Shadow Covering" technique as described in Chapter 2. Given an instance (G, T, k) of Disjoint 1-Out-Regular Vertex Deletion Compression, we let $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$ be the family of strongly connected subgraphs C of G that are not r_C-out-regular for some $r_C \leq 1$. This means that every graph of $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$ contains at least one vertex with out-degree greater one. Moreover, every graph of $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$ is strongly connected. Thus, for any graph $H \in \mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$ and any $v \in V(H)$ we have that there is a $v \to T$-path and a $T \to v$-path in H, i.e. all graph of $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$ are T-connected. Additionally, any solution S of (G, T, k) must intersect every graph in $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$ as otherwise the graph $G-S$ contains a strongly connected component of $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$, which is not r_C-out-regular for some $r_C \leq 1$. Thus, any solution S must be an $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$-transversal. Also, any $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$-transversal S of size at most k must be a solution, as any remaining strongly connected component of $G - S$ cannot be in $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$. Thus, we can apply Theorem 2.9 to obtain the following result.

Lemma 5.5. *Let (G, T, k) be an instance of 1-Out-Regular Vertex Deletion. Then we can construct in time $2^{\mathcal{O}(k^2)} \operatorname{poly}(n)$ a set $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_t\}$ with $t = 2^{\mathcal{O}(k^2)} \log^2 n$ such that if (G, T, k) has a solution, there is a solution S and a $Z_i \in \mathcal{Z}$ with*

1. *$S \cap Z_i = \emptyset$, and*
2. *Z_i contains the shadow of S with respect to T.*

Proof. Apply Theorem 2.9 to $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$, which is T connected. Moreover, the $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$-transversals are exactly our solutions. □

This allows us to solve the following easier problem instead:

Shadow-Covered 1-Out-Regular Vertex Deletion

Instance: A graph G, vertex sets $T, Z \subseteq V(G)$ and an integer $k \in \mathbb{Z}_{\geq 0}$ such that
- T is an inclusion-wise minimal set with the property that every strongly connected component C of $G - T$ is r_C-out-regular for some $r_C \leq 1$,
- $T \cap Z = \emptyset$,
- $G[Z \cup T]$ contains no subgraph in $\mathcal{F}_{\text{not } \{0,1\}\text{-out-regular}}$.

Task: Find a set $S \subseteq V(G) \setminus T$ of size at most k such that
- every strongly connected component C of $G - S$ is r_C-out-regular with $r_C \leq 1$,
- $S \cap Z = \emptyset$, and
- Z contains the shadow of S with respect to T

or decide that no such set exists.

Note that our problem contains the two additional properties that $T \cap Z = \emptyset$ and $G[Z \cup T]$ contains no subgraph in $\mathcal{F}_{\mathsf{not}\ \{0,1\}\text{-out-regular}}$. The first property we can assume as T is never in the shadow of some $S \subseteq V(G) \setminus T$. The second property follows from the fact that $Z \cup T$ is disjoint form our sought-after solution. The following lemma summarizes those findings.

Lemma 5.6. *Given an instance (G, T, k) of Disjoint 1-Out-Regular Vertex Deletion Compression, then we can compute in time $2^{\mathcal{O}(k^2)}\operatorname{poly}(n)$ a collection $Z_1, Z_2, \ldots, Z_t$ of $t = 2^{\mathcal{O}(k^2)}\log^2 n$ vertex sets $Z_i \subseteq V(G)$, such that for the $I_j = (G, T, Z_j, k)$ the following holds:*

- *the I_j are Shadow-Covered 1-Out-Regular Vertex Deletion instances,*
- *any solution to one of the I_j instances is a solution to (G, T, k), and*
- *if (G, T, k) admits a solution then one of the I_j has a solution.*

Proof. By Lemma 5.5, we can compute in $2^{\mathcal{O}(k^2)}\operatorname{poly}(n)$ time at most $2^{\mathcal{O}(k^2)}\log^2 n$ many sets Z_i such that if (G, T, k) has a solution, it has a solution S that is disjoint of Z_i and Z_i covers the shadow of S with respect to T for some i. We can assume that these Z_i are disjoint of T, as a vertex of T is never in any shadow with respect to T. That is we take $Z_i \setminus T$, if they are not. Moreover, for any such Z_i, we have that $G[Z_i \cup T]$ contains no subgraph of $\mathcal{F}_{\mathsf{not}\ \{0,1\}\text{-out-regular}}$, as S is disjoint of $Z_i \cup T$ and thus $G - S$ would contain a strongly connected component that is in $\mathcal{F}_{\mathsf{not}\ \{0,1\}\text{-out-regular}}$, a contradiction to S being a solution. We can check for this in linear time for every Z_i and remove these sets. Thus, we can restrict us to Z_i where (G, T, Z_i, k) is a Shadow-Covered 1-Out-Regular Vertex Deletion instance. On the other hand, any solution to a Shadow-Covered 1-Out-Regular Vertex Deletion instance (G, T, Z, k) with arbitrary Z is also a solution to the Disjoint 1-Out-Regular Vertex Deletion Compression instance (G, T, k). The run-time follows from the run-time and size bounds of Lemma 5.5. □

5.1.3 Reduction by Torso Operation

Normally after using the "shadow covering" technique, one uses the set Z to create a new instance of the same problem that has a shadowless solution (if there is any). In contrast to this we reduce slightly different problem here that retains a bit more information by marking arcs as good or bad.

Definition 5.7. *Let (G, T, Z, k) be an Shadow-Covered 1-Out-Regular Vertex Deletion instance. Then $\mathsf{torso}(G, Z)$ defines the directed graph with vertex set $V(G) \setminus Z$ and a labeling of the arcs as* good *and* bad*. An arc (u, v) for $u, v \notin Z$ is introduced whenever there is an $u \to v$-path P in G (of length at least 1) whose internal vertices are all in Z. We mark (u, v) as* good *arc if P is unique and there is no cycle O in $G[Z]$ with $O \cap P \neq \emptyset$. Otherwise, we mark it as* bad *arc.*

Note that every arc (x, y) for $x, y \in V(G) \setminus Z$ also forms a path as above. Therefore, $G[V(G) \setminus Z]$ is a subgraph of $\mathsf{torso}(G, Z)$. Also, $\mathsf{torso}(G, Z)$ may contain self loops at vertices v from cycles with only the vertex v outside of Z. In $\mathsf{torso}(G, Z)$, we call a cycle *good* if it consists of only good arcs. A non-good cycle in $\mathsf{torso}(G, Z)$ can contain both good arcs and bad arcs.

We call the resulting new problem Good 1-Out-Regular Vertex Deletion.

GOOD 1-OUT-REGULAR VERTEX DELETION

Instance: A graph G with arcs labeled *good* and *bad*, a vertex set $T \subseteq V(G)$ and an integer $k \in \mathbb{Z}_{\geq 0}$ such that T is an inclusion-wise minimal set with the property that every strongly connected component C of $G - T$

- is r_C-out-regular for some $r_C \leq 1$,
- containing only good arcs.

Task: Find a set $S \subseteq V(G) \setminus T$ of size at most k such that every strongly connected component C of $G - S$

- is r_C-out-regular with $r_C \leq 1$,
- contains only good arcs, and
- is shadowless with respect to T

or decide that no such set exists.

We will now show solution equivalence for the two problems in the following lemmas.

Lemma 5.8. *Let (G, T, Z, k) be an instance of* SHADOW-COVERED 1-OUT-REGULAR VERTEX DELETION *and let (G', T, k) be the corresponding* GOOD 1-OUT-REGULAR VERTEX DELETION *instance with $G' = \mathsf{torso}(G, Z)$.*

Then any solution $S \subseteq V(G) \setminus (Z \cup T)$ to (G, T, Z, k) is a solution to (G', T, k).

Proof. First note, that $S \subseteq V(G') \setminus T$. Assume for contradiction that $G' - S$ contains a strongly connected component C' that is either not $r_{C'}$-out-regular for some $r_{C'} \leq 1$ or contains a bad arc. Transform C' back to a subgraph of G as follows. All good arcs are replaced by their unique path as defined in the torso operation. For a bad arc (x, y) we insert all $x \to y$-paths whose internal vertices completely belong to Z. If there is only a single such path P then by definition there is a cycle O in $G[Z]$ that intersects P. We also insert all cycles O of this type. Call the resulting graph C.

This directed graph C is a subgraph of $G - S$ as $(V(C') \cup Z) \cap S = \emptyset$. Moreover, C is strongly connected as C' was strongly connected and all added vertices have a path from and to $V(C')$. Now, either C' was not a cycle, then C is also not a cycle or it contained a bad arc, and we have inserted at least two parallel paths or a cycle. In any case, we have that C is not r_C-out-regular for an $r_C \leq 1$. □

Lemma 5.9. *Let (G, T, Z, k) be an instance of* SHADOW-COVERED 1-OUT-REGULAR VERTEX DELETION *and let (G', T, k) be the corresponding* GOOD 1-OUT-REGULAR VERTEX DELETION *instance with $G' = \mathsf{torso}(G, Z)$.*

Then any solution $S' \subseteq V(G') \setminus T$ to (G', T, k) is a solution to (G, T, Z, k).

Proof. First note that $S' \subseteq V(G) \setminus (Z \cup T)$. Assume for contradiction that $G - S'$ has a strongly connected component C that is not r_C-out-regular for some $r_C \leq 1$. We will show that $C' = \mathsf{torso}(C, Z)$ is a strongly connected subgraph of $G' - S$ that is not $r_{C'}$-out-regular for some $r_{C'} \leq 1$ or contains a bad arc. Note that the torso operation preserves subgraph relations and connection. So C' exists in $G' - S$ and is strongly connected.

By the input property of Shadow-Covered 1-Out-Regular Vertex Deletion, that $G[Z \cup T]$ contains only $\{0,1\}$-out-regular subgraphs, we know hat there is a $v \in V(C) \setminus (Z \cup T)$. Furthermore, we know that there is also a $t \in V(C) \cap T$ as T is a solution to G. From $Z \cap T = \emptyset$ by definition we know that $v, t \notin Z$ and hence in $V(C')$. As C is strongly connected, there is a closed walk O through v and t in C.

Claim 1. *O is a cycle.*

Proof of Claim 1. Suppose, for sake of contradiction, that O is not a cycle. Let w be a vertex that is visited at least twice when traversing O. Let x_1, w, y_1 be the first traversal and x_2, w, y_2 be the second one. Without loss of generality, we can assume that $x_1, x_2, y_1, y_2 \notin Z$ by replacing them by the next vertex on O outside Z.

If $w \in Z$ then the arcs $(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)$ all exist in the strongly connected subgraph $\mathsf{torso}(O, Z)$. Thus, $\mathsf{torso}(O, Z)$ is not r_O-out-regular for $r_O \leq 1$, in contradiction to the fact that $\mathsf{torso}(O, Z)$ is a subgraph of $G' - S'$ and S' a solution to I'.

Otherwise, $w \notin Z$ and thus, the arcs $(x_1, w), (x_2, w), (w, y_1), (w, y_2)$ would exist in $\mathsf{torso}(O, Z)$, giving the same contradiction. This completes the proof of the claim. ■

Now C is strongly connected and not a cycle, and therefore has to contain a (possibly closed) $x \to y$ path R with the following properties:

- $x, y \in V(O)$,
- R contains no arc from O,
- all internal vertices of R are disjoint of $V(O)$.

As O contains at least two vertices v, t that are not in Z, there is an $x_1 \to x_2$-path O_x and a $y_1 \to y_2$-path O_y in O such that

- the endpoints of O_x and O_y are not in Z but all their interior vertices, and
- $x \in V(O_x)$, $y \in V(O_y)$.

For example, if $x \notin Z$, set $x_1 = x_2 = x$. Otherwise, take the subpath of O that starts in the first vertex not in Z before x and ends in the first vertex not in Z after x.

Now, if R contains some interior vertex $u \notin Z$, the path $O_x[x_1, x] \circ R[x, u]$ is in C and shrinks to a $x_1 \to u$-path in C'. As $u \notin V(O)$, we get that x_1 has at least two out-arcs $(x_1, x_2), (x_1, u)$ in C' and therefore $C' = \mathsf{torso}(C, Z)$ is not $r_{C'}$-out-regular for some $r_{C'} \leq 1$, a contradiction. Thus, the interior of R lies in Z.

Next, if $(x_1, x_2) \neq (y_1, y_2)$ then $O_x[x_1, x] \circ R \circ O_y[y, y_2]$, is a $x_1 \to y_2$ path in C. Note that $x_2 \neq y_2$ as O is a cycle, $(x_1, x_2) \neq (y_1, y_2)$ and all interior vertices of O_x and O_y are in Z. Therefore the path is shrunk by the torso operation to the arc (x_1, y_2). But then x_1 has two outgoing arcs in C' and as C' is still strongly connected, $C' = \mathsf{torso}(C, Z)$ is not $r_{C'}$-out-regular for some $r_{C'} \leq 1$, a contradiction.

Finally, we have $(x_1, x_2) = (y_1, y_2)$ and also $O_x = O_y$ as otherwise the arc would be bad (because there are two different $x_1 \to x_2$-paths). If x lies before y on O_x the path $P = O_x[x_1, x] \circ R \circ O_x[y, x_2]$ is a $x_1 \to x_2$-path in C. As the interior of O_x and R is in Z, this would give a second $x_1 \to x_2$-path, making (x_1, x_2) bad. This is a contradiction to C' being a strongly connected subgraph of $G' - S'$ and S' being a solution.

The last case is if y lies before x on O_x. Then $R \circ O_x[y, x]$ forms a cycle in Z which intersects O_x at least in the vertex x, again showing that (x_1, x_2) is bad, a contradiction. □

The above lemmas show that S is a solution of a Shadow-Covered 1-Out-Regular Vertex Deletion instance (G, T, Z, k) if and only if it is a solution of the Good 1-Out-Regular Vertex Deletion instance $(\mathsf{torso}(G, Z), T, k)$. As the torso graph contains no vertices in Z, we can reduce our search for $(\mathsf{torso}(G, Z), T, k)$ to shadowless solutions.

5.1.4 Finding a Shadowless Solution

Consider an instance (G, T, k) of Good 1-Out-Regular Vertex Deletion. Normally, after the torso operation a pushing argument is applied. We deviate from this by constructing an algorithm that recovers the last strongly connected component of $G - S$ (given that a solution S exists). A crucial observation for this is that the last strongly connected component of $G - S$ has to contain a vertex of T.

Lemma 5.10. *Let $I = (G, T, k)$ be an instance of Good 1-Out-Regular Vertex Deletion. If I has a solution S, then for any topological ordering $C_1, \ldots, C_\ell$ of the strongly connected components of $G - S$, we have $V(C_\ell) \cap T \neq \emptyset$.*

Proof. Note that every vertex of $G-S$ is in some strongly connected component (maybe containing only one vertex). Thus, for every $v \in V(C_\ell)$, the only reachable vertices from v lie in C_ℓ by definition of a topological ordering. Now, for S to be a solution, it must fulfill that the shadow of S with respect to T is empty. This implies that every $v \in V(C_\ell)$ has to reach some $t \in T$. By above argument we have that $t \in V(C_\ell) \cap T$. □

Lemma 5.11. *Let $I = (G, T, k)$ be an instance of Good 1-Out-Regular Vertex Deletion. If I has a solution S, then for any topological ordering $C_1, \ldots, C_\ell$ of the strongly connected components of $G - S$, we have $N^+(V(C_\ell)) \neq \emptyset$.*

Proof. Assume for sake of contradiction that $N^+(V(C_\ell)) = \emptyset$. Now, we consider the graph $G - (T \setminus V(C_\ell))$. Lemma 5.10 shows that $T \setminus V(C_\ell) \subsetneq T$. Thus, by inclusion-wise minimality of T, we know that $G - (T \setminus V(C_\ell))$ contains a strongly connected component C that is not r_C-out-regular for some $r_C \leq 1$. As C does not exist in $G - T$, we have that there is a $t \in V(C) \cap (V(C_\ell) \cap T)$. Moreover, C is not a subgraph of C_ℓ as C_ℓ is r_{C_ℓ}-out-regular for some $r_{C_\ell} \leq 1$ and this property is hereditary. Thus, there is also a $v \in V(C) \setminus V(C_\ell)$. By C being strongly connected, there is a $t \to v$-path in G. As $t \in V(C_\ell)$ and $v \notin V(C_\ell)$, this path has to leave $V(C_\ell)$ at least once and the first vertex outside $V(C_\ell)$ is thus an element of $N^+(V(C_\ell))$. This shows $N^+(V(C_\ell)) \neq \emptyset$. □

By the previous lemmas we have that in order to identify at least one vertex of some solution S, it suffices to recover the last strongly connected component of $G - S$. We will later reapply the procedure up to k times to identify all vertices.

Lemma 5.12. *There is an algorithm that given an instance $I = (G, T, k)$ of Good 1-Out-Regular Vertex Deletion, computes in time $2^{\mathcal{O}(k \log k)} \operatorname{poly}(n)$ a collection of subgraphs $C_1, \ldots, C_r$ with $r \leq (k+1)^{k+1}$ such that if I has a solution S, then there is a topological ordering of strongly connected components of $G - S$ where some C_i appears last.*

Proof. For every $t \in T$ apply the following procedure. We assume that t appears in the last strongly connected component C of some topological ordering of strongly connected components of $G - S$. First we add one C_i with $C_i = \{t\}$. If this was not the right choice, we know that the connected component C must be 1-out-regular and contain only good arcs. We try to recover it by a branching procedure resulting in additional C_i's. During this branching procedure we also keep track of vertices we guess to be in S.

We start by setting $v_0 = t$. Notice that exactly one out-neighbor v_1 of v_0 belongs to C. Set $i = 0$ and notice that every out-neighbor of v_i other than v_{i+1} must be removed from the graph G as C is the last component in the topological ordering of $G-S$, there is no later component where those out-neighbors could go. This observation gives rise to a natural branching procedure: we guess the out-neighbor v_{i+1} of v_i that belongs to C and remove all other out-neighbors of v_i from the graph. We then repeat this branching step with $i \mapsto i + 1$ until we get back to the vertex t we started with. When we reach t, we add the currently recovered component as a new set C_i. This branching results in at least one deletion as long as v_i has out-degree at least two. If the out-degree of v_i is exactly one, then we simply proceed by setting $v_i := v_{i+1}$ (and increment i). In any case we stop early if (v_i, v_{i+1}) is a bad arc, as this arc may not be contained in a strongly connected component.

Recall that the vertices $t = v_0, v_1, \ldots$ must *not* belong to S, whereas the deleted out-neighbors of v_i must belong to S. From another perspective, the deleted out-neighbors of v_i must *not* belong to T. So once we reached back at the vertex $v_j = t$ for some $j \geq 1$, we have indeed found the component C that we were looking for.

Let us shortly analyze the run-time of the branching step. As for each vertex v_i, we have to remove all its out-neighbors from G except one and include them into the hypothetical solution S of size at most k, we immediately know that the degree of v_i in G can be at most $k + 1$. Otherwise, we have to include v_i into S. Therefore, there are at most $k + 1$ branches to consider in order to identify the unique out-neighbor v_{i+1} of v_i in C_ℓ. So for each vertex v_i with out-degree at least two we branch into at most $k + 1$ ways, and do so for at most k vertices, yielding at most $(k + 1)^k$ branches for every $t \in \mathcal{T}$. As $|T| \leq k + 1$, we produce at most $(k + 1)^{k+1}$ sets. Each of the steps (merging steps where we do not delete anything) takes polynomial time. So the overall run-time is $(k + 1)^{k+1} \operatorname{poly}(n) = 2^{\mathcal{O}(k \log k)} \operatorname{poly}(n)$. □

Among the subgraphs we branch which of these is a strongly connected component C appearing last in a topological ordering of strongly connected components of $G - S$. We proceed by deleting C's out-neighbors as they must belong to S. By starting again from the "shadow covering" we obtain more vertices of S until a complete solution is recovered or none is found as the instance has none. This procedure is described in the next section.

5.1.5 Disjoint 1-Out-Regular Vertex Deletion Compression Algorithm

We now give an algorithm for Disjoint 1-Out-Regular Vertex Deletion Compression hinted at in the last section.

Lemma 5.13. *There is an algorithm solving an instance (G, T, k) of Disjoint 1-Out-Regular Vertex Deletion Compression in time $2^{\mathcal{O}(k^3)} \operatorname{poly}(n)$.*

Proof. Given an instance we start with $S_{\mathsf{partial}} = \emptyset$ and apply a branching algorithm as follows. If S_{partial} is a solution, i.e. $|S_{\mathsf{partial}}| \leq k$ and every connected component C of $G - S_{\mathsf{partial}}$ is r_C-out-regular for some $r_C \leq 1$, we return S_{partial}. If instead $|S_{\mathsf{partial}}| > k$, we give up on this branch. So assume that none of the cases occurred. Then we try to complete S_{partial} to a solution S. Applying Lemma 5.6 to $G - S_{\mathsf{partial}}$ we create $2^{\mathcal{O}(k^2)} \log^2 n$ instances of the form $I_j = (G - S_{\mathsf{partial}}, T, Z_j, k - |S_{\mathsf{partial}}|)$, such that:

- the I_j are Shadow-Covered 1-Out-Regular Vertex Deletion instances,
- any solution to one of the I_j instances is a solution to (G, T, k), and
- if (G, T, k) admits a solution then one of the I_j has a solution.

We create a branch for every I_j. If (G, T, k) has a solution $S \supseteq S_{\mathsf{partial}}$, than at least one of the branches has a solution. Conversely, any solution S' of some I_j completes S_{partial} to a solution $S = S_{\mathsf{partial}} \cup S'$ of (G, T, k).

For each branch we continue by creating the Good 1-Out-Regular Vertex Deletion instance $(\mathsf{torso}(G - S_{\mathsf{partial}}, Z_j), T, k - |S_{\mathsf{partial}}|)$. By combining Lemma 5.8 and Lemma 5.9 we have that the solutions to these instances stay the same. Using Lemma 5.12 we create sets $C_1, \ldots, C_r$ with $r \leq (k+1)^{k+1}$ such that if $(\mathsf{torso}(G - S_{\mathsf{partial}}, Z_j), T, k - |S_{\mathsf{partial}}|)$ has a solution S', then there is a topological ordering of strongly connected components of $\mathsf{torso}(G - S_{\mathsf{partial}}, Z_j) - S'$ where one of the C_j's appears last. We branch again, creating a branch for every choice of C_j.

For each of these branches we have by Lemma 5.11, that C_j has at least one out-neighbor in $\mathsf{torso}(G - S_{\mathsf{partial}}, Z_j)$. If we made the right choice, these out-neighbors S_{out} must belong to a solution S of $(\mathsf{torso}(G - S_{\mathsf{partial}}, Z_i), T, k - |S_{\mathsf{partial}}|)$. By Lemma 5.9 S_{out} must then also belong to a solution of I_j, which in turn belongs to a solution of (G, T, k) if we made the right choice of I_j. Thus, we have that $S_{\mathsf{partial}} \cup S_{\mathsf{out}}$ is contained in a solution of (G, T, k) for the right branch, if (G, T, k) has a solution at all. So we add S_{out} to S_{partial} in each of the branches and continue our branching algorithm from the beginning with an S_{partial} of increased size.

Note that this procedure stops as we either find a solution and stop early or after every branching step involving Z_i and C_j our set S_{partial} increases by at least one element, and we stop once it contains more than k elements. Each branching step involving I_j and C_j creates at most $2^{\mathcal{O}(k^2)} \log^2 n \cdot \mathcal{O}(k \cdot (k+1)^k)$ new branches. Hence, the total number of nodes in the branching tree is

$$\begin{aligned}\left(2^{\mathcal{O}(k^2)} \log^2 n\right)^k \cdot \mathcal{O}\left(k \cdot (k+1)^k\right) &= \left(2^{\mathcal{O}(k^2)}\right)^k \left(\log^2 n\right)^k \cdot \mathcal{O}\left((k+1)^{k+1}\right) \\ &= 2^{\mathcal{O}(k^3)} \cdot n,\end{aligned}$$

where we used that $\log^{2k} n \in 2^{\mathcal{O}(k \log k)} + n$ by Lemma 2.10. For the run-time note that the creation of the I_j takes time $2^{\mathcal{O}(k^2)} \operatorname{poly}(n)$ for $2^{\Omega(k^2)} \log^2 n$ candidate sets and the creation of of the C_j takes time $2^{\mathcal{O}(k \log k)} \operatorname{poly}(n)$ for $(k+1)^{k+1} \in 2^{\Omega(k \log k)})$ candidate sets. As all other operation on a branch node can be done in time $\operatorname{poly}(n)$, this means that the run-time is at most $\operatorname{poly}(n)$ times the number of branches in the worst case. So our run-time is bound by $2^{\mathcal{O}(k^3)} \operatorname{poly}(n)$. □

Combining Lemma 5.13 and Lemma 5.4 proves our main result, Theorem 5.2.

5.2 Polynomial Parameter Transformation from Arc to Vertex Version

In this section we prove the existence of a polynomial parameter transformation from 1-Out-Regular Arc Deletion to 1-Out-Regular Vertex Deletion. Note that this reduction is parameter preserving.

Lemma 5.14. *Given an instance (G, k) of 1-Out-Regular Arc Deletion we can compute in polynomial time an equivalent instance (G', k') of 1-Out-Regular Vertex Deletion with $k' = k$.*

Proof. Let G' be the directed line graph of G, that is G' has an vertex v_a for every arc $a \in A(G)$ and the arc (v_a, v_b) exists in G' if and only if $a = (u, v) \in A(G)$ and $b = (v, w) \in A(G)$ for some $u, v, w \in V(G)$.

Obviously, there is a one-to-one correspondence between arcs in G and vertices in G'. This also holds if there is a set S of arcs in G and S' its corresponding set of vertices in G' for the directed graphs $G - S$ and $G' - S'$. The correspondence also holds for non-trivial strongly connected components as a closed walk on vertices $v_1, \ldots, v_t$ and arcs $a_1, \ldots, a_t$ corresponds to a closed walk on the vertices $v_{a_1}, \ldots, v_{a_t}$ in G'. It remains to show the equivalence of the instances.

For the forward direction, let S be a solution to (G, k). Let $S' = \{v_a \mid a \in S\}$. As $|S'| = |S| \leq k$, our candidate fulfills the size bound. Let now X' be a strongly connected component of $G' - S'$. Assume for contradiction that X' is neither trivial nor 1-out-regular. By above correspondence there is a non-trivial strongly connected component X in $G - S$ that has the arcs which X' possesses as vertices. As S is a solution to (G, k), X is 1-out-regular (as it is not trivial). Therefore, $G[X]$ forms a cycle O. This cycle has a corresponding cycle O' in $G'[X']$. Since O visits all arcs of $G[X]$, O' is a Hamiltonian cycle for $G'[X']$. As $G'[X']$ is not 1-out-regular, there must be an arc $(v_a, v_b) \in E(G'[X'])$ which is not part of O'. This arc means that the arcs a and b share a vertex v in $G[X]$ albeit being not adjacent in O. Thus, v has out-degree at least two in $G[X]$, a contradiction. Therefore, S' is a solution to (G', k).

For the reverse direction, let S' be a solution to (G', k). We consider as solution candidate for (G, k) the set $S = \{a \in A(G) \mid v_a \in S'\}$. Again we have $|S| = |S'| \leq k$ and thus the size bound fulfilled. Let X be a strongly connected component in $G - S$. Assume, for sake of contradiction, that X is neither trivial nor 1-out-regular. This means that $G[X]$ contains a cycle O_X and a (possibly closed) walk P with both endpoints on O_X and its interior disjoint of it. Let x be the start vertex of P and a the first arc of P. Furthermore, let $b = (v, x)$ and $c = (x, w)$ be the arcs adjacent to x on O. Then $G' - S'$ contains the vertices v_a, v_b, v_c, and by preservation of strongly connected connectivity they are in the same connected component of $G' - S'$. But by choice of a, b, c the arcs (v_b, v_a) and (v_b, v_c) exist in $G' - S'$. This means that v_b has out-degree at least two in its strongly connected component in $G' - S'$, a contradiction. In conclusion, S must be a solution for (G, k). □

5.3 Hardness of Eulerian Strongly Connected Component Vertex Deletion

In this section we prove Theorem 5.3, by showing NP-hardness and W[1]-hardness of the Eulerian Strongly Connected Component Vertex Deletion problem.

Before the hardness proof we recall an equivalent characterization of Eulerian directed graphs that uses the concept of balance. A directed graph G is balanced if and only if we have for every node that in-degree equals out-degree, i.e., $|\delta^+(v)| = |\delta^-(v)|$ for every $v \in V(G)$.

Lemma 5.15 (folklore)**.** *Let G be a weakly connected directed graph. Then*

$$G \text{ is Eulerian} \quad \Leftrightarrow \quad G \text{ is balanced.}$$

This shows that Eulerian Strongly Connected Component Vertex Deletion can be reformulated to the requirement that every strongly connected component of $G - S$ needs to be balanced. The vertex deletion problem where we require the whole graph to be balanced instead has already been considered by Cygan et al. [CMP+14]. Note that in this case also arcs between different strongly connected components are counted.

Directed Balanced Vertex Deletion	
Instance:	A directed graph G and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq V(G)$ of size at most k such that $G - S$ is balanced or decide that no such set exists.

Cygan et al. prove this problem to be NP-hard and W[1]-hard for parameter k.

Theorem 5.16 ([CMP+14, Theorem 7])**.** *Directed Balanced Vertex Deletion is* NP*-hard and* W[1]*-hard with parameter k.*

We will prove the hardness of Eulerian Strongly Connected Component Vertex Deletion for $(k+1)$-strongly connected, directed graphs by adding vertices ensuring this connectivity.

Theorem 5.3. *Eulerian Strongly Connected Component Vertex Deletion is* NP*-hard and* W[1]*-hard parameterized by solution size k, even for $(k+1)$-strongly connected, directed graphs.*

Proof. We give a polynomial reduction from Directed Balanced Vertex Deletion. Let (G, k) an instance of Directed Balanced Vertex Deletion. Let G' arise from G by adding vertices $z_1, \ldots, z_{k+1}$ and arcs $(z_i, v), (v, z_i)$ for all $v \in V(G)$ and all $i \in \{1, \ldots, k+1\}$. This construction obviously can be made to run in polynomial time. Moreover, G' is $(k+1)$-strongly connected as one needs to delete at least all z_i to disconnect two vertices.

All we have to show is that (G, k) has a solution as instance of Directed Balanced Vertex Deletion if and only if (G', k) has a solution as instance of Eulerian Strongly Connected Component Vertex Deletion.

Let S' be a solution to (G', k) as instance of Eulerian Strongly Connected Components Vertex Deletion. As G' is $(k+1)$-strongly connected, $G' - S'$ is strongly connected. Moreover, S' is a solution, so $G' - S'$ is Eulerian (because it is the only strongly connected component). Therefore, by Lemma 5.15 every vertex of $G' - S'$ is balanced.

Deleting the remaining vertices of $\{z_1, \ldots, z_{k+1}\}$ does not harm the balance of the remaining vertices, as for each $v \in V(G)$ and z_i we delete one outgoing and one incoming arc of v. Thus $G' - (S' \cup \{z_1, \ldots, z_{k+1}\}) = G - (S' \setminus \{z_1, \ldots, z_{k+1}\})$ is balanced. Hence, $S' \setminus \{z_1, \ldots, z_{k+1}\}$ is a solution to (G, k) as instance of Directed Balanced Vertex Deletion.

Let S be a solution to (G, k) as instance of Directed Balanced Vertex Deletion. Then $G - S$ is balanced and by construction $G' - S$ as well. Furthermore, $G' - S$ is strongly connected and thus by Lemma 5.15 also Eulerian. Hence, the only strongly connected component of $G' - S$ is Eulerian and therefore S is a solution to (G', k) as instance of Eulerian Strongly Connected Component Vertex Deletion. □

Chapter 6

Negative Cycle Deletion

In this chapter we analyze the Negative Cycle Deletion problem. In contrast to other problems studied in this thesis, this problem is defined on *weighted* graphs, i.e. graphs G in combination with a weight functions $w : A(G) \to \mathbb{Z}$ on the arcs. The problem is to remove all cycles of negative total weight from the graph by deleting at most k vertices (vertex deletion version) or arcs (arc deletion version).

Negative Cycle Deletion	
Instance:	A graph G, a weight function $w : A(G) \to \mathbb{Z}$ and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq V(G)$ (vertex deletion version) or a set $S \subseteq A(G)$ (arc deletion version) such that • the set S has size at most k and • every directed cycle of $G - S$ has non-negative total weight or decide that no such set exists.

The codomain $\mathbb{Z}$ of the weight function here is arbitrary. As we will see later, there is always an integer weight function that leads to the same structural instance and has its absolute values bounded in some function of $|A(G)|$. Alas, this bound is in general not polynomial. However, the encoding length of this weight function is bounded by $\text{poly}(|G|)$, thus leading to an instance of essentially the same encoding length. See Section 6.3 for details.

The main motivation for studying this problem comes from the area of linear programming. One of the main tasks in linear programming is to decide whether a system of linear inequalities $(a_i \cdot x \leq b_i)_{i \in \{1,\dots,m\}}$ has a feasible solution $x = x^\star$, where the a_i and $x^\star$ are n-dimensional vectors and the b_i are some scalars. However, such inequality systems may not always have feasible solution. In practice measurement or modeling errors are a common problem rendering inequality systems infeasible. An algorithmical approach to deal with these infeasible systems would be to find a set $S \subseteq \{1, \dots, m\}$ of minimum size such that $(a_i \cdot x \leq b_i)_{i \in \{1,\dots,m\} \setminus S}$ has a feasible solution. In the case where all inequalities have the form $x_i - x_j \leq b_{i,j}$, this directly corresponds to the arc deletion version of Negative Cycle Deletion (see Section 6.2).

Thus, we study Negative Cycle Deletion in contrast to the previous chapters mainly in terms of the arc deletion version. We will however see that for many of

our considered parameters there is a polynomial parameter transformation between the arc and vertex deletion version (see Section 6.7). In particular, the arc deletion version is algorithmically at least as hard as the vertex deletion version for all parameter combinations.

The general Negative Cycle Deletion problem is NP-hard as setting $w \equiv -1$ is exactly the Directed Feedback Vertex Set or Directed Feedback Arc Set problem. We study a variety of parameters to solve the problem even for these NP-hard cases. A natural choice here is the parameter k (the size of the deletion set) as Directed Feedback Vertex Set is known to be fixed-parameter tractable for this parameter [CLL+08]. However, Negative Cycle Deletion is more general than this, and we can show W[1]-hardness when parameterized in k only. Thus, we add the parameter w_+, the number of arcs with positive weight, which for Directed Feedback Vertex Set is 0. For symmetry reasons, we also consider the parameter w_-, the number of arcs with negative weight. Lastly, we see whether the structure of the graph can help us to solve this problem. For this we study the treewidth $\mathrm{tw}(G)$, pathwidth $\mathrm{pw}(G)$ and treedepth $\mathrm{td}(G)$ of our graph, all defined by the underlying undirected graph (see Section 6.1). Recently, these parameters were also used to solve (integer) linear programming formulations [FLS+18, EHK+19, CCK+20]. These parameters in context of linear programming match our graph problem parameters when considering the special case where all inequalities have the form $x_i - x_j \leq b_{i,j}$.

A long term open question for linear programming is the existence of strongly polynomial algorithms solving arbitrary linear inequality systems, i.e. algorithms that only take polynomial time in the encoding of the input and not the actual numbers. That led us to consider also arc weights of the form $w : A(G) \to \{-1, 0, 1\}$. This corresponds to encoding the arc weights $w : A(G) \to \mathbb{Z}$ in unary encoding, as replacing an arc of weight $\pm w$ encoded in unary by a path of length w where all arcs have weight ± 1 yields an equivalent instance of the same encoding length.

For some of these choices we have a natural dominance. If we can solve instances with $w : A(G) \to \mathbb{Z}$, we can solve those with $w : A(G) \to \{-1, 0, 1\}$. Also $\mathrm{tw}(G) \leq \mathrm{pw}(G) \leq \mathrm{td}(G)$ holds. Last but not least, if $w_- \leq k$ holds, we can remove all of the w_--many negative arcs by removing at most w_- vertices/arcs from G resulting in a graph that contains no negative cycles. Thus, for all non-trivial cases we have $k \leq w_-$. Taking this dominance into consideration we don't have to consider parameter combinations, that dominate each other. For example the case of parameter $\mathrm{tw}(G) + \mathrm{pw}(G)$ is already covered by the $\mathrm{pw}(G)$ case for hardness results or the $\mathrm{tw}(G)$ case for algorithmic results. This way we are left with 48 cases, of which we determine 46 to be FPT or W[1]-hard (or even NP-hard for constant parameters). The remaining two cases both have weight functions of the form $w : A(G) \to \{-1, 0, 1\}$. We show both of them to be fixed-parameter tractable if there are no zero weight arcs present. See Table 6.1 in Section 6.4 for an overview of the results.

The results of this chapter are joint work with Kristóf Bérczi, Lydia Mirabel Mendoza Cadena and Matthias Mnich. An extended abstract of this work has previously appeared at IPEC 2019 [GMCM19].

6.1 Definitions

To better distinguish between the vertex and arc deletion version of Negative Cycle Deletion, we introduce the new names Negative Directed Feedback Vertex Set and Negative Directed Feedback Arc Set for them. This is based on the naming convention for Feedback Vertex Set and Feedback Arc Set. Formally they are defined as follows.

Negative Directed Feedback Vertex Set	
Instance:	A graph G, a weight function $w : A(G) \to \mathbb{Z}$ and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq V(G)$ of size at most k such that every directed cycle of $G - S$ has non-negative total weight or decide that no such set exists.

Negative Directed Feedback Arc Set	
Instance:	A graph G, a weight function $w : A(G) \to \mathbb{Z}$ and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq A(G)$ of size at most k such that every directed cycle of $G - S$ has non-negative total weight or decide that no such set exists.

For an instance (G, w, k) of Negative Directed Feedback Vertex Set or Negative Directed Feedback Arc Set we denote by A_-, A_0 and A_+ the sets $w^{-1}(\mathbb{Z}_{<0})$, $w^{-1}(0)$ and $w^{-1}(\mathbb{Z}_{>0})$, respectively. Moreover, we use the abbreviation $A_{\neq 0}$ for $A_- \cup A_+$. With this notation we can introduce the first set of parameters. Namely, we denote by $w_- = |A_-|$ the number of negative arcs and by $w_+ = |A_+|$ the number of positive arcs.

The remaining parameters treewidth, pathwidth and treedepth measure the structure of the graph. Treewidth and pathwidth measure how tree-like or path-like a graph is. Treedepth however measures how many recursive vertex deletions on connected components are necessary to delete the whole graph (see Theorem 6.2).

Definition 6.1. *Let G be an undirected graph. A* tree decomposition *of G is a tuple $(T, \mathcal{B})$ consisting of a tree T and a collection of vertex sets $\mathcal{B} = (B_x)_{x \in V(T)}$, one $B_x \subseteq V(G)$ for every $x \in V(T)$, that fulfills the following properties:*

1. *$V(G) = \bigcup_{x \in V(T)} B_x$, and*
2. *for all $v \in V(G)$ the graph $T[\{x \mid v \in B_x\}]$ is connected, and*
3. *for every $e \in E[G]$, there is a $x \in V(T)$ with $e \subseteq B_x$.*

The sets B_x are called bags *of the tree decomposition. The* width of a tree decomposition *is the maximum size of its bags minus one, i.e. $|(T, \mathcal{B})| = \max_{x \in V(T)} |B_x| - 1$. The* treewidth $\mathrm{tw}(G)$ *of G is the minimum width over all tree decompositions for G.*

A path decomposition *of G is a tree decomposition, where T is a path. Instead of specifying T, one can also enumerate the bags along the path, i.e. $B_1, \dots, B_t$. The* pathwidth $\mathrm{pw}(G)$ *of G is the minimum size over all path decompositions for G.*

Let F be a rooted arborescence. The height *of the arborescence is the length of a longest path in F originating at the root. The* closure $\mathrm{clos}(F)$ *of F is the graph on the vertices $V(F)$ that contains an edge $\{v, w\}$ if v is an ancestor of w in F. The* treedepth $\mathrm{td}(G)$ *of G is the minimum height of an arborescence F such that G is a subgraph of* $\mathrm{clos}(F)$.

For a directed graph G the treewidth/pathwidth/treedepth of G is defined as the treewidth/pathwidth/treedepth of the underlying undirected graph of G.

A more useful definition to compute the treedepth is the following recursive one. For a proof of the equivalence and a further introduction of treewidth/pathwidth/treedepth we refer the reader to the textbook by Nešetřil and de Mendez [NDM06].

Theorem 6.2. *Let G be an undirected graph. Then*

$$\mathrm{td}(G) = \begin{cases} 1 & \text{, if } |G| = 1 \\ 1 + \min_{v \in V(G)} \mathrm{td}(G - v) & \text{, if } G \text{ is connected and } |G| > 1 \\ \max_{\text{connected component } C} \mathrm{td}(G[C]) & \text{, otherwise.} \end{cases}$$

Maybe the most prominent property of bounded-treedepth graphs is that their paths have bounded length. Naturally, this property also extends to cycles. We refer to the textbook by Nešetřil and de Mendez [NDM06] for similar stronger results implying these.

Lemma 6.3. *Let G be a graph, let P a path in G and let C be a cycle of G. Then $|P| \leq 2^{\mathrm{td}(G)} - 1$ and $|C| \leq 2^{\mathrm{td}(G)-1}$ holds.*

Finally, we introduce the concept of nice tree decompositions which are very useful for dynamic programs in graphs of bounded treewidth.

Definition 6.4. *Let G be an undirected graph. A tree decomposition $(T, \mathcal{B})$ is called* nice *if T is a rooted tree with root r, $B_r = \emptyset$ and every $x \in V(T)$ has one of the following four types:*

Leaf node: *x is a leaf of T and $B_x = \emptyset$.*

Introduce node: *x has exactly one child y in T and $B_x = B_y \uplus \{v\}$ for some $v \in V(T)$.*

Forget node: *x has exactly one child y in T and $B_x \uplus \{v\} = B_y$ for some $v \in V(T)$.*

Join node: *x has exactly two children y_1 and y_2 and $B_x = B_{y_1} = B_{y_2}$.*

For $x \in V(G)$, we denote by T_x the subtree of T rooted at x. Moreover, we define G_x to be the graph $G[\bigcup_{y \in T_x} B_y]$.

To make use of this concept, we need an algorithm to find such a nice tree decomposition. To compute a nice tree decomposition of minimum width, one can use the following result by Bodlaender.

Theorem 6.5 ([Bod96])**.** *Given a graph G with n vertices, there is an algorithm that constructs a nice tree decomposition of size* $\mathrm{tw}(G)$ *with $\mathcal{O}(n)$ nodes in run-time*

$$2^{\mathcal{O}(\mathrm{tw}(G)^3 \log \mathrm{tw}(G))} n.$$

For our usage, it is enough to have a nice tree decomposition that is within a constant factor of minimum width. This allows us to use the following faster algorithm by Bodlander et al.

Theorem 6.6 ([BDD$^+$16])**.** *Given a graph G with n vertices, there is an algorithm that constructs a nice tree decomposition of size $5\,\mathrm{tw}(G)+4$ with $\mathcal{O}(n)$ nodes in $2^{\mathcal{O}(\mathrm{tw}(G))}n$.*

The last notion we need is that of feasible potentials.

Definition 6.7. *Let (G, w) a weighted graph. A vertex function $\pi : V(G) \to \mathbb{R}$ is called* feasible potential *if for all arcs $a = (u, v) \in A(G)$ we have $\pi(u) - \pi(v) + w(a) \geq 0$.*

The following folklore result describes the connection between feasible potentials and solutions of Negative Directed Feedback Arc Set.

Theorem 6.8. *A weighted graph (G, w) contains no cycle of negative total weight if and only if it has a feasible potential.*

By this theorem for every solution S of a Negative Directed Feedback Arc Set instance (G, w, k) there is a feasible potential π_S for $G - S$. We often call any feasible potential of $G - S$, a feasible potential of S. Such a feasible potential also certifies that S is indeed a solution.

6.2 Relation to Linear Programming

In this section we describe the connection of Negative Directed Feedback Arc Set to linear programming. For this consider the following problem naturally arising in the area of linear programming.

Minimum Feasibility Blocker	
Instance:	A system of linear inequalities $(a_i \cdot x \leq b_i)_{i\in\{1,\dots,m\}}$ and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq \{1, \dots, m\}$ of size at most k such that $(a_i \cdot x \leq b_i)_{i\in\{1,\dots,m\}\setminus S}$ has a feasible solution or decide that no such set exists.

We consider the special case of this problem, where all inequalities have the form $x_i - x_j \leq b_{i,j}$. This type of inequalities is called difference constraints.

Theorem 6.9. *The Negative Directed Feedback Arc Set problem and the Minimum Feasibility Blocker problem for difference constraints are equivalent with the equivalence computable in polynomial time. Additionally, there is a one-to-one correspondence between constraints and arc weights with $b_a = w(a)$.*

Proof. Let (G, w, k) be a Negative Directed Feedback Arc Set instance, and let $n = |V(G)|$ and $m = |A(G)|$. Fix an arbitrary order $v_1, \dots, v_n$ of the vertices of G and $\alpha_1, \dots, \alpha_m$ of the arcs of G. Define the inequality system by setting for every $\alpha_i \in A(G)$ the entry $a_{i,j} = +1$ for $\alpha_i \in \delta^-(v_j)$, $a_{i,j} = -1$ for $\alpha_i \in \delta^+(v_j)$, and $a_{i,j} = 0$ otherwise. Furthermore, let $b_i = w(\alpha_i)$. The resulting inequality system $((a_i \cdot x \leq b_i)_{i \in \{1,\dots,m\}}, k)$ is an instance of the Minimum Feasibility Blocker problem all inequalities being difference constraints.

The construction is bijective by the following reverse construction: Define a directed graph on n vertices $v_1, \dots, v_n$, then for every constraint $a_{i,\bullet} \cdot x \leq b_i$ add an arc as follows: Let j^- be the unique index with $a_{i,j^-} = -1$, and let j^+ be the unique index with $a_{i,j^+} = +1$. Add an arc $\alpha = (v_{j^+}, v_{j^-})$ with weight $w(\alpha) = b_i$ to the current directed graph. Let G be the resulting directed graph after all arcs are added. Then (G, w, k) is the constructed Negative Directed Feedback Arc Set instance. It is easy to verify that this indeed reverses the first construction.

Now we want to compare solutions of both problems. Intuitively, deleted constraints and arcs have an one to one correspondence, but we will formally prove the equivalence here.

In the following, for each $X \subseteq A(G)$ denote by $X_{\mathcal{I}}$ the corresponding indices of the constraints and vice-versa. Then the following equivalences hold:

$(G - X, w)$ contains no negative cycles with respect to w.

$\Leftrightarrow$ $(G - X, w)$ has a feasible potential $\pi : V(G) \to \mathbb{R}$.

$\Leftrightarrow$ There is a $\pi : V(G) \to \mathbb{R}$ such that $\pi(u) \leq \pi(v) + w(\alpha)$, $\forall \alpha = (u, v) \in A(G) \setminus X$.

$\Leftrightarrow$ There is an $x \in \mathbb{R}^{V(G)}$ such that $x_u - x_v \leq w(\alpha)$ for all $\alpha = (u, v) \in A(G) \setminus X$.

$\Leftrightarrow$ There is an $x \in \mathbb{R}^n$ such that $a_{i,\bullet} \cdot x \leq b_i$ for all $i \in \{1, \dots, m\} \setminus X_{\mathcal{I}}$.

Furthermore, as X and $X_{\mathcal{I}}$ have the same cardinality, the last statement is equivalent to "X is a solution to (G, w, k) if and only if $X_{\mathcal{I}}$ is a solution to (A, b, k)". □

6.3 Integral Weights

Here we discuss whether the constraint of having integral weights has an effect on the problem. The main result of this section is that for general weights there is always an equivalent instance with integer weights of essentially the same encoding length. However, it is not clear how to find such an equivalent instance in polynomial time. For parameters w_+ and w_- however, there is a fixed-parameter algorithm computing such an equivalent instance.

We assume that the weights are given via some oracle that can compute whether $w(A')$ is negative (< 0), zero (0) or positive (> 0). In particular, this allows us to identify the set of zero weight arcs A_0 and non-zero arcs $A_{\neq 0}$. Fix now a weighted graph (G, w). The results of this section are based on the following linear inequality system that has a variable x_a for every $a \in A_{\neq 0}$.

$$\sum_{a \in A'} x_a \leq -1 \qquad \forall A' \subseteq A_{\neq 0} \text{ with } w(A') < 0$$

$$\sum_{a \in A'} x_a = 0 \qquad \forall A' \subseteq A_{\neq 0} \text{ with } w(A') = 0$$

$$\sum_{a \in A'} x_a \geq 1 \qquad \forall A' \subseteq A_{\neq 0} \text{ with } w(A') > 0$$

Denote by $P_{(G,w)}$ the polyhedron defined by this inequality system. The first observation is that every $x^\star \in P_{(G,w)}$ defines an equivalent weight function for G by setting $w^\star(a) = x^\star_a$. The equivalence follows from the fact that for every arc set A' (and thus the arc set of every cycle) the total weight $w^\star(A')$ has the same sign as total weight of the original weight $w(A')$ by the corresponding inequality. We now want to show that $P_{(G,w)}$ always contains an integer point whose size is bounded in $|A_{\neq 0}|$. A natural candidate for this is an extreme point/vertex of $P_{(G,w)}$.

Lemma 6.10. *For every weighted graph (G, w) the polyhedron $P_{(G,w)}$ has a vertex.*

Proof. Recall that an n-dimensional polyhedron P has no vertex if and only if there is no $v \in P$ and $d \in \mathbb{R}^n \setminus \{0\}$ such that $v + \lambda d \in P$ for all $\lambda \in \mathbb{R}$. Assume for contradiction that there exists such $v \in P_{(G,w)}$ and $d \in \mathbb{R}^{|A_{\neq 0}|} \setminus \{0\}$ for $P_{(G,w)}$. As $d \neq 0$, there is an $a \in A_{\neq 0}$ such that $d_a \neq 0$. Now choose $\lambda = -\frac{v_a}{d_a}$. Then we have that $v + \lambda d \in P_{(G,w)}$ and $(v + \lambda d)_a = v_a - \frac{v_a}{d_a} d_a = 0$. But as $a \in A_{\neq 0}$, we have that for $P_{(G,w)}$ either the inequality $x_a \leq -1$ or $x_a \geq 1$ holds. Thus, $v + \lambda d \notin P_{(G,w)}$, a contradiction. □

We now want to argue about the encoding length of such a vertex. For this we want to define a binary encoding of our numbers and a function size(.) denoting the length of such an encoding. In the case of an integer $n \in \mathbb{Z}$ we use $\text{size}(n) := 1 + \lceil \log(|n|+1) \rceil$ bits to encode the sign and absolute value of this number. For rational numbers $r := \frac{p}{q} \in \mathcal{Q}$ we have $\text{size}(r) = \text{size}(p) + \text{size}(q)$. Finally, for vectors $x \in \mathbb{Q}^n$ and matrices $A \in \mathbb{Q}^{n \times m}$ we have $\text{size}(x) := n + \sum_{i=1}^{n} \text{size}(x_i)$ and $\text{size}(A) := nm + \sum_{i,j} \text{size}(a_{i,j})$.

Recall that any vertex of an n-dimensional polyhedron $\{x \mid Ax \leq b\}$ can be written as unique solution to $A'x = b'$, where $A'x \leq b'$ is a subsystem of $Ax \leq b$ with n inequalities. Using Cramer's rule this implies that any vertex v can be represented using $\text{size}(v) \in \mathcal{O}(\text{size}(A) + \text{size}(b))$ bits. Moreover, for rational A and b, any solution obtained this way is rational. As our polyhedron $P_{(G,w)}$ is defined by A and b having only $\{-1, 0, 1\}$ entries, we get the following result.

Corollary 6.11. *For every weighted graph (G, w) there is a rational $x^\star \in P_{(G,w)}$ with $\text{size}(x^\star) \in \mathcal{O}(|A_{\neq 0}|^2)$ and all denominators of the $x^\star_i$ being equal.*

We now use this rational solution to construct an integer solution of bounded size.

Lemma 6.12. *For every weighted graph (G, w) there is an integral $x_I \in P_{(G,w)}$ with $\text{size}(x^\star) \in \mathcal{O}(|A_{\neq 0}|^2)$.*

Proof. Let $x^\star \in P_{(G,w)}$ be a rational vector with size$(x^\star) \in \mathcal{O}(|A_{\neq 0}|^2)$ as in Corollary 6.11. We multiply each entry of $x^\star$ by the common denominator $d^\star$ to obtain our vector $x_I = d^\star x^\star$. We have size$(x_I) \leq$ size$(x^\star)$ as this multiplication can be reflected in the encoding by removing the denominator. It now remains to prove that $x_I \in P_{(G,w)}$. For this note that $d^\star \geq 1$. Now, for any $d \geq 1$ and every $A' \subseteq A_{\neq 0}$ we have that $\sum_{a \in A'} dx_a$ fulfills the inequality $\leq -1/= 0/\geq 1$ if this was already the case for $\sum_{a \in A'} x_a$. Therefore, from $x^\star \in P_{(G,w)}$ it follows that $x_I \in P_{(G,w)}$, concluding the proof. □

Using these results, we get that there is always an equivalent instance with integral weights (and the same graph). Note that for parameters w_+ and w_-, the size of our inequality system is bounded in $|A_{\neq 0}| = w_+ + w_-$. Thus, for these parameters, we can enumerate all potential solutions of the given size and test in fixed-parameter time, whether they form a feasible solution.

Theorem 6.13. *Let (G, w, k) be a* Negative Directed Feedback Arc Set *instance with general weights w. Then there are integral weights w_I with* size$(w_I) \in$ poly$(|G|)$ *such that (G, w_I, k) is an equivalent instance.*

Moreover, such an instance can be found in time $2^{\mathcal{O}((w_+ + w_-)\log(w_+ + w_-))}\,\mathrm{poly}(n)$.

6.4 Overview of the Results

To obtain algorithmic or hardness results for all possible parameter combinations, we do not have to consider each case separately. As mentioned before, there is a natural dominance between parameters. The advantages are twofold. First, we do not need to consider cases with parameters that dominate each other. Second, we do not have to show results for all of the 48 remaining cases, as some results imply others by this dominance. The dominances are as follows. Obviously, the cost functions of the form $w : A(G) \to \mathbb{Z}$ generalize those of the form $w : A(G) \to \{-1, 0, 1\}$. Bodlaender et al. have shown relations between treewidth, pathwidth and treedepth.

Theorem 6.14 ([BGHK95])**.** *Let G be a graph. Then it holds that*

$$\mathrm{tw}(G) \leq \mathrm{pw}(G) \leq \mathrm{td}(G) - 1.$$

Thus, the graph class of graph with bounded treewidth is more general than those with bounded pathwidth which in turn is more general than the class of bounded treedepth graphs. The last dominance is more specialized to the Negative Directed Feedback Arc Set problem. Namely, all instances of Negative Directed Feedback Arc Set with $w_- \leq k$ are solvable in polynomial time.

Theorem 6.15. *Let (G, w, k) a* Negative Directed Feedback Arc Set *instance with $w_- \leq k$. Then (G, w, k) has a solution. Moreover, we can check for this condition and compute a solution in linear time.*

Proof. Iterate over $A(G)$ and collect the set A_- of all arcs with negative weight. Then $G - A_-$ contains no negative cycles as it contains no arcs of negative weight. Moreover, we can check whether $|A_-| = w_- \leq k$ and possibly return A_- as solution. □

$w : A(G) \to \mathbb{Z}$

	-		tw(G) or pw(G)		td(G)	
	-	w_+	-	w_+	-	w_+
-	NP-hard	NP-hard	NP-hard	NP-hard Thm. 6.42	W[1]-hard	W[1]-hard Thm. 6.43
k	W[1]-hard	W[1]-hard	W[1]-hard	W[1]-hard Thm. 6.44	FPT Thm. 6.18	FPT
w_-	W[1]-hard	FPT Thm. 6.23	W[1]-hard Thm. 6.47	FPT	FPT	FPT

(a) Algorithmic and Hardness Results for $w : A(G) \to \mathbb{Z}$

$w : A(G) \to \{-1, 0, 1\}$

	-		tw(G) or pw(G)		td(G)	
	-	w_+	-	w_+	-	w_+
-	NP-hard	NP-hard Thm. 6.40	W[1]-hard	FPT Thm. 6.31	FPT Thm. 6.32	FPT
k	W[1]-hard	open	W[1]-hard Thm. 6.45	FPT	FPT	FPT
w_-	open	FPT	FPT Thm. 6.30	FPT	FPT	FPT

(b) Algorithmic and Hardness Results for $w : A(G) \to \{-1, 0, 1\}$

Table 6.1: Result overview for Negative Directed Feedback Arc Set.
Results are split by type of weight function ($w : A(G) \to \mathbb{Z}$ vs. $w : A(G) \to \{-1, 0, 1\}$). The rows are indexed by parameters $\{\text{-}, k, w_-\}$, where k denotes the size of the deletion set and w_- the number of arcs with negative weight. - means that no other parameter of this set is taken. Note that for non-trivial cases $k < w_-$ holds, as we can otherwise delete all negative arcs. The columns are double indexed by parameters in $\{\text{-}, \text{tw}(G), \text{pw}(G), \text{td}(G)\}$ and $\{\text{-}, w_+\}$. Here tw(G)/pw(G)/td(G) denote the treewidth/pathwidth/treedepth of the graph, respectively. The parameter w_+ denotes the number of arcs with strictly positive weight. Again, - means that no other parameter of this set is taken. It is well-known, that $\text{tw}(G) \leq \text{pw}(G) \leq \text{td}(G)$ holds. The columns of tw(G) and pw(G) are merged as they share the same algorithmic and hardness results. That is, algorithmic results hold also if only the parameter tw(G) is used and hardness results hold if only the parameter pw(G) is used. Cells with a light shade are implied by the cells with a darker shade. For the latter, we also list the theorem showing this result.

Using the above results, we can split our parameters into three groups. For each group we select at most one parameter, combine all of them and investigate the resulting parameter combination for its parameterized complexity. The three groups are $\{\text{-}, k, w_-\}$, $\{\text{-}, \mathrm{tw}(G), \mathrm{pw}(G), \mathrm{td}(G)\}$ and $\{\text{-}, w_+\}$, where - stands for the choice of no parameter. Moreover, we have a choice between the two types of weight functions $w : A(G) \to \mathbb{Z}$ and $w : A(G) \to \{-1, 0, 1\}$. For each of the resulting 48 cases we aim to show one of the following results:

- A fixed-parameter algorithm (i.e. membership in FPT), or
- a W[1]-hardness result (no fixed-parameter algorithm unless FPT = W[1]), or
- an NP-hardness result for constant parameters (no fixed-parameter algorithm unless P = NP).

We are able to resolve all but two of the cases this way (see Table 6.1). For the two remaining cases the weight function has the form $w : A(G) \to \{-1, 0, 1\}$. In these cases we are able to obtain a fixed-parameter algorithm for the even more restrictive weight function type $w : A(G) \to \{-1, 1\}$. However, this is makes for a good indicator as all of the remaining results for $w : A(G) \to \{-1, 0, 1\}$ weight functions also hold if restricted to $w : A(G) \to \{-1, 1\}$.

As we have argued before, it is not necessary to show a result as above for every parameter combination, as dominance between parameters implies several results. Overall, we obtain seven unique hardness results and five algorithms (seven if we account for the $w : A(G) \to \{-1, 1\}$ cases). These results are marked in a darker shade in Table 6.1.

6.5 Algorithmic Results

6.5.1 Verifying a Solution

We shortly recall how we can verify that a graph $G-S$ has indeed no negative cycle. For this we use the Moore-Bellman-Ford algorithm for shortest paths. Originally, it only finds all shortest $s \to v$-paths from a single vertex s in a graph G, given that G contains no negative cycle (reachable from s). However, it can additionally detect whether there is such a negative cycle. In particular, it detects a cycle that minimizes the vertex distance to the start vertex s. By running the Moore-Bellman-Ford algorithm once from every vertex we get the following result.

Theorem 6.16. *For a weighted graph (G, w) we can detect in time $\mathcal{O}(n^2 m)$ whether G contains a negative cycle. Moreover, we can recover in the same time a length-minimal negative cycle if it exists.*

Running this algorithm on $(G - S, w|_{V(G)\setminus S})$ and additionally checking whether $|S| \leq k$, allows us to verify a solution to some Negative Directed Feedback Arc Set instance (G, w, k).

6.5.2 Algorithm for Bounded Treedepth and Solution Size

The main observation to solve Negative Directed Feedback Arc Set on graphs of bounded treedepth is that on these graphs all cycles have bounded length (see Lemma 6.3).

Thus, if we have as additional parameter the size of our solution, we can iteratively detect negative cycles in our graph and branch on which arc of the cycle is contained in some solution (if there is any).

We will now proof our main algorithm of this section. Its run-time depends on the maximum length of a negative cycle in the given graph. We will revisit it later on, when we derive other bounds on the length of negative cycles in G depending on the parameter choices.

Lemma 6.17. *There is an algorithm solving a Negative Directed Feedback Arc Set instance (G, w, k) in time $\mathcal{O}(L^k n^2 m)$, where L is an upper bound on the length of any negative cycle in G.*

Proof. We recursively call the following procedure with some potential partial solution $S \subseteq V(G)$ with $|S| \leq k$. The initial call is done with $S = \emptyset$. First we check in time $\mathcal{O}(n^2 m)$ whether there is a negative cycle C in $G - S$ and recover it via Theorem 6.16. If there is no negative cycle and $|S| \leq k$, we return S' as solution. If there is a negative cycle C and $|S| = k$, we give up on this branch. Otherwise, there is a negative cycle C and $|S| < k$. Then for every $v \in V(C)$ we make a subroutine call with $S_v = S \cup \{v\}$. If our initial call with $S = \emptyset$ did not return a solution for any branch, we report that the instance has no solution. This finishes the description of our algorithm.

As we only make subroutine calls in the case of $|S| < k$, we add only one vertex to S, and we start with $|\emptyset| = 0 \leq k$, one can show by induction that all our calls indeed fulfill $|S| \leq k$. Now we argue for the correctness. For this we have to show that if there is a solution, we do indeed return a solution. As we only return sets S such that $|S| \leq k$ and $G - S$ has no negative cycles, we just have to make sure that we return a set. We consider the variant of our algorithm, where we do not return a solution early, but rather save it and return it at the end of the algorithm. Let $S^\star$ be an inclusion-wise minimal solution to (G, w, k). We are going to reconstruct a possible subroutine call sequence $\emptyset = S_0 \subsetneq S_1 \subsetneq \ldots \subsetneq S_{|S^\star|} = S^\star$ that is called by our algorithm. As at least in this branch our algorithm considers a solution, it will return a solution.

We will do an induction over i and show that S_i appears in one branch of our algorithm. We start with $S_0 = \emptyset$ which is our initial subroutine call. As long as $i < |S^\star|$ we have that $S_i \subsetneq S^\star$ and thus, by inclusion-wise minimality of $S^\star$, $G - S_i$ contains at least one negative cycle C_i. Now $S^\star$ is a solution, so we know that there is a $v_i \in S^\star \cap V(C_i)$. By C_i being disjoint of S_i, we have that $S_{i+1} = S_i \cup \{v_i\}$ is a strict superset of S_i. Thus we make a subroutine call with this S_{i+1}. This shows by induction that our modified algorithm considers the set $S^\star$ and our original algorithm returns a solution.

For the run-time note, that at each sub-routine call we branch into $|C| \leq L$ many branches. We start with $|S| = 0$ and in each recursive subroutine call $|S|$ increases by one. As we stop once $|S| = k$, this means our recursion nests at most k levels deep. This results in L^k subroutine calls. In each call we need $\mathcal{O}(nm)$ time to check for negative cycles and $\mathcal{O}(n)$ to iterate over the vertex set. □

We will now use that the cycle length is bounded in the treedepth of a graph.

Theorem 6.18. *There is an algorithm solving* Negative Directed Feedback Arc Set *in time* $\mathcal{O}(2^{k\,\mathrm{td}(G)}n^2m)$.

Proof. Use Lemma 6.17 with Lemma 6.3 as upper bound on the length of negative cycles. □

6.5.3 Algorithm for Bounded Number of Non-Zero Arcs

In this section we are going to derive an algorithm for Negative Directed Feedback Arc Set parameterized in w_+ and w_-. As w_- is an upper bound on k for all non-trivial cases (Theorem 6.15), we may also use the parameter k. The high-level idea is as follows. In a first step we guess the intersection between a solution and the set of non-zero arcs. Afterwards we focus on the graph induced by the zero-weight arcs only. By using that after the deletion of a solution our graph has a feasible potential, we get that the endpoints of the non-zero arcs are ordered by this potential. We use this observation to show that solutions can be obtained by solving a Skew Cut problem in the zero-arc graph. As we do not know the feasible potential or the exact Skew Cut instance, our algorithm guesses among all potential orderings of the endpoints of the non-zero arcs.

The main graph we work with is constructed from G by removing the arcs of $A_{\neq 0}$ and splitting their endpoints.

Definition 6.19. *Let (G, w) be an integer-weighted directed graph and denote by Z the set of vertices $z \in V(G)$ with $(\delta^+(z) \cup \delta^-(z)) \cap A_{\neq 0}(G) \neq \emptyset$. The* zero-weight propagation graph *of G is the graph $\overrightarrow{G}_0$ obtained from G by deleting $A_{\neq 0}$ and splitting every $z \in Z$ into two vertices z^+ and z^-, where z^+ inherits the outgoing arcs and z^- inherits the incoming arcs of z. In this context we denote for every subset $Y \subseteq Z$ by Y^+ and Y^- the set of all z^+'s and z^-'s with $z \in Y$, respectively.*

As memory aid think that Z^+ contains the vertices with $\delta^+(z) \neq \emptyset$ and Z^- those with $\delta^-(z) \neq \emptyset$ in $\overrightarrow{G}_0$. To understand the structure of our solutions, we rely on the Skew Cut problem, which we will define in the following.

Definition 6.20. *Let G be a directed graph, let $p \in \mathbb{Z}_{\geq 0}$ be a non-negative integer, and let $X_1, \ldots, X_p, Y_1, \ldots Y_p \subseteq V(G)$ be pairwise disjoint vertex sets of G. An* $(X_1, \ldots, X_p) \to (Y_1, \ldots, Y_p)$-skew cut *is an arc set $S \subseteq A(G)$ such that there is no $X_i \to Y_j$-path in $G - S$ for any $1 \leq j < i \leq p$.*

Skew Cut	
Instance:	A graph G, vertex sets $X_1, \ldots, X_p, Y_1, \ldots Y_p \subseteq V(G)$ and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find an $(X_1, \ldots, X_p) \to (Y_1, \ldots, Y_p)$-skew cut of size at most k or decide that no such skew cut exists.

The Skew Cut problem has already been solved by Chen et al. as subroutine for their Directed Feedback Vertex Set algorithm. In fact, they solve the vertex deletion variant (called Skew Separator), but their algorithm can be applied to the arc deletion variant by subdividing arcs with vertex and introducing $k+1$ copies of all original vertices.

Theorem 6.21 ([CLL+08])**.** *An instance $(G, (X_1, \ldots, X_p), (Y_1, \ldots, Y_p), k)$ of Skew Cut can be solved in time $\mathcal{O}(4^k kn^3)$.*

We are now able to state our main observation of this chapter.

Lemma 6.22. *Let (G, w, k) be a Negative Directed Feedback Arc Set instance and let $S \subseteq A_0(G)$ be a negative directed feedback arc set for (G, w). Let Z be the endpoints of arcs in $A_{\neq 0}(G)$ and let $\overrightarrow{G}_0$ be the zero-weight propagation graph of G. Then there is an ordered partition $(Z_1, \ldots, Z_p)$ of Z such that*

1. *S is a $(Z_1^+, \ldots, Z_p^+, \emptyset) \to (\emptyset, Z_1^-, \ldots, Z_p^-)$-skew cut in $\overrightarrow{G}_0$, and*
2. *any $(Z_1^+, \ldots, Z_p^+, \emptyset) \to (\emptyset, Z_1^-, \ldots, Z_p^-)$-skew cut in $\overrightarrow{G}_0$ is a solution for (G, w, k).*

Proof. As S is a negative directed feedback arc set for (G, w), the graph $G' = G - S$ has a feasible potential π with respect to $w|_{A(G')}$. We define an ordered partition $(Z_1, \ldots, Z_p)$ of Z by using the potential π. Two vertices z and z' belong to the same Z_i if and only if $\pi(z) = \pi(z')$. This assigns every Z_i a unique potential $\pi(Z_i) = \pi(z)$ with $z \in Z_i$. The Z_i are then ordered by decreasing $\pi(Z_i)$.

We now have to verify the two theorem statements. First we check, whether S is a $(Z_1^+, \ldots, Z_p^+, \emptyset) \to (\emptyset, Z_1^-, \ldots, Z_p^-)$-skew cut in $\overrightarrow{G}_0$. Assume for contradiction that it is not. Then there is an $x^+ \to y^-$-path in $\overrightarrow{G}_0$ with $x^+ \in Z_i^+$ and $y^- \in Z_j^-$ such that Z_j^- is at an earlier or the same position in $(\emptyset, Z_1^-, \ldots, Z_p^-)$ as Z_i^+ is in $(Z_1^+, \ldots, Z_p^+, \emptyset)$. By the index shift caused by the addition of the two $\emptyset$, we know that $j < i$. This implies $\pi(y) > \pi(x)$, by choice of the Z_i.

Consider now the P as $x \to y$-path P in $G - S$, where x and y are the original vertices of x^+ and y^- before they got split. Note that by construction of $\overrightarrow{G}_0$ our path P contains only arcs of weight zero. As π is a feasible potential, we have that for all arcs $a = (u, v) \in A(P)$, we have $\pi(u) - \pi(v) = \pi(u) - \pi(v) + w(a) \geq 0$. Summing this up over all arcs in P, we get $\pi(x) - \pi(y) = \sum_{a=(u,v)\in A(P)} \pi(u) - \pi(v) \geq 0$. In other words $\pi(y) \leq \pi(x)$, in contradiction to the inequality $\pi(y) > \pi(x)$ we derived earlier.

Now it remains to check that any $(Z_1^+, \ldots, Z_p^+, \emptyset) \to (\emptyset, Z_1^-, \ldots, Z_p^-)$-skew cut in $\overrightarrow{G}_0$ is a negative directed feedback arc set for (G, w). Let $S^\star$ be such a skew cut. We try to prove that $S^\star$ is a negative directed feedback arc set for (G, w) by constructing a feasible potential $\pi^\star$ of $G^\star = G - S^\star$ with respect to $w|_{A(G^\star)}$. In the following we assume that $\max_{v\in V(G)} \pi(v) \leq 0$, by possibly shifting all vertex potentials by the same value. We define $\pi^\star$ as

$$\pi^\star(v) = \begin{cases} \pi(v) & \text{, if } v \in Z \\ \min\{\pi(z) \mid \text{ there is a } z^+ \to v\text{-path in } \overrightarrow{G}_0 - S^\star\} & \text{, otherwise} \end{cases}$$

where we define the minimum to be 0 if there is no $z^+ \to v$-path in $\overrightarrow{G}_0 - S^\star$ for any $z^+ \in Z^+$.

We now check for all arcs in $G^\star$ that $\pi^\star$ is a feasible potential. For arcs $a = (u, v) \in A_{\neq 0}(G^\star)$ we have that $A_{\neq 0}(G^\star) = A_{\neq 0}(G)$ and therefore $u, v \in Z$. This implies $\pi^\star(u) - \pi^\star(v) + w(a) = \pi(u) - \pi(v) + w(a) \geq 0$. Hence, it remains to check the arcs $a = (u, v) \in A_0(G^\star) = A_0(G) \setminus S^\star$.

If $u \in Z$, then the arc (u^+, v^-) or (u^+, v) exists in $\overrightarrow{G}_0 - S^\star$. In the former case we again have $u, v \in Z$ and the potential is feasible for this arc. Otherwise, we have that there is an $u^+ \to v$-path in $\overrightarrow{G}_0 - S^\star$ and thus $\pi^\star(v) \leq \pi(u) = \pi^\star(u)$. As $w(a) = 0$, this implies $\pi^\star(u) - \pi^\star(v) + w(a) \geq 0$.

So we can assume that $u \notin Z$. If additionally u does not have a $z^+ \to u$-path in $\overrightarrow{G}_0 - S^\star$, then $\pi^\star(u) = 0$ and by $\max_{v \in V(G)} \pi(v) \leq 0$, we have that $\pi^\star(v) \leq 0 = \pi^\star(u)$. Thus, also in this case, we have $\pi^\star(u) - \pi^\star(v) + w(a) \geq 0$.

Now we know, that $u \notin Z$ and there is a $z^+ \to u$-path in $\overrightarrow{G}_0 - S^\star$ for some $z \in Z$. We distinguish between the two cases $v \notin Z$ and $v \in Z$. If $v \notin Z$, then the arc (u, v) exists in $\overrightarrow{G}_0 - S^\star$. Hence, any $z^+ \to u$-path can be prolonged to a $z^+ \to v$-path by adding (u, v) at the end. Thus, $\pi^\star(v) \leq \pi^\star(u)$, and again $\pi^\star(u) - \pi^\star(v) + w(a) \geq 0$.

In the remaining case $v \in Z$, we have that the arcs (u, v^-) exists in $\overrightarrow{G}_0 - S^\star$. Again any $z^+ \to u$-path can be prolonged to a $z^+ \to v^-$-path. By definition of our Skew Cut instance, we have that for all such paths in $\overrightarrow{G}_0 - S^\star$, we have $\pi(z) \geq \pi(v)$. This implies $\pi^\star(v) \leq \pi^\star(u)$, and thus for the last type of arcs $\pi^\star(u) - \pi^\star(v) + w(a) \geq 0$. Hence, $\pi^\star$ is a feasible potential for $G - S^\star$ and thus $S^\star$ is a negative directed feedback arc set for G. □

We are now going to use this result to give an algorithm solving the Negative Directed Feedback Arc Set problem parameterized by w_+ and w_-.

Theorem 6.23. *There is an algorithm that solves a Negative Directed Feedback Arc Set instance (G, w, k) in time $2^{\mathcal{O}((w_+ + w_-)\log(w_+ + w_-))} \operatorname{poly}(n)$.*

Proof. If $w_- \leq k$, return the set $A_-(G)$ of negative arcs. Otherwise, for every subset $A' \subseteq A_{\neq 0}(G)$ of size at most k do the following. For the graph $G' = G - A'$ with Z' being endpoints of non-zero arcs $A_{\neq 0}(G')$ and every ordered partition $(Z_1, \ldots, Z_p)$ of Z', we call the algorithm for Skew Cut (see Theorem 6.21) on

$$(\overrightarrow{G'}_0, (Z_1^+, \ldots, Z_p^+, \emptyset), (\emptyset, Z_1^-, \ldots, Z_p^-), k - |A'|).$$

If it returns a set S', we check whether $A' \cup S'$ is a negative directed feedback arc set for G and if it is, we return it. If the algorithm does not find a negative directed feedback arc set for any A' and ordered partition $(Z_1, \ldots, Z_p)$ of Z' this way, we return that it has no negative directed feedback arc set.

As we check any solution we return for correctness, we just have to verify that we always return a negative directed feedback arc set, if there is any. Let S be a negative directed feedback arc set for (G, w). For $A' = S \cap A_{\neq 0}(G)$ we know that $S' = S \setminus A'$ is a negative directed feedback arc set for $G' = G - A'$, as $(G - A') - S' = G - S$ which contains no negative cycles. Now $S' \cap A_{\neq 0}(G') = \emptyset$, and thus by Lemma 6.22 there is an ordered partition $(Z_1, \ldots, Z_p)$ of Z' such that S' is a $(Z_1^+, \ldots, Z_p^+, \emptyset) \to (\emptyset, Z_1^-, \ldots, Z_p^-)$-skew cut in $\overrightarrow{G'}_0$. Thus, we know that $(\overrightarrow{G'}_0, (Z_1^+, \ldots, Z_p^+, \emptyset), (\emptyset, Z_1^-, \ldots, Z_p^-), k - |A'|)$ is a "yes"-instance. Moreover, Lemma 6.22 tells us that any solution $S^\star$ to this Skew Cut instance our algorithm call finds, is a negative directed feedback arc set for G'.

Thus, $A' \cup S^\star$ is a negative directed feedback arc set of G, as $G - (A' \cup S^\star) = G' - S^\star$. Also, $A' \cup S^\star$ has size at most k. Hence, our algorithm finds a negative directed feedback arc set of the given size if it exists.

For the run-time note that by returning the trivial solution A_- if $w_- \leq k$, we have either a linear run-time for constructing A_- or $k < w_-$. Now we have at most $2^{w_+ + w_-}$ possible subsets A' of $A_{\neq 0}$. The set Z' contains at most $2w_+ + 2w_-$ vertices and its ordered partitions can thus be bounded by $(2w_+ + 2w_-)! \cdot 4^{w_+ + w_-}$ (orders of Z' times the choice for every element whether to start a new subset there). Thus, the overall run-time is

$$\mathcal{O}(2^{w_+ + w_-} \cdot (2w_+ + 2w_-)! \cdot 4^{w_+ + w_-} \cdot (4^k k n^3 + \text{poly}(n))),$$

which using $k < w_-$ we can rewrite as $2^{\mathcal{O}((w_+ + w_-) \log(w_+ + w_-))} \text{poly}(n)$. □

6.5.4 Normalized Arc Weights and Feasible Potentials

This section is dedicated to Negative Directed Feedback Arc Set with arc weights in $\{-1, 0, 1\}$ and how the number of positive or negative arcs influences the feasible potential of the solution. The findings are that for both parameters w_- and w_+, the solution must have an integral feasible potential in $[0, w_-]$ and $[0, w_+]$, respectively. Unfortunately, the latter only holds for graphs that are strongly connected after removing the solution. Moreover, we get that for the parameter treedepth $\text{td}(G)$ the solution has a feasible potential in $[0, 2^{\text{td}(G)}]$

We first state a folklore technique for constructing a feasible potential of a graph.

Lemma 6.24. *Let (G, w) a weighted directed graph that contains no negative cycles. Let $G^\star$ be the graph G where we introduced a new vertex $s^\star$ that is connected to every original vertex $v \in V(G)$ by an arc $(s^\star, v)$ of weight 0. For every $v \in V(G^\star)$ let $\pi^\star(v)$ the weight of a minimum weight $s^\star \to v$-path P_v. Then $\pi = \pi^\star|_{V(G)}$ is a feasible potential for G.*

Proof. We claim that $\pi^\star$ is a feasible potential for $G^\star$. Assume that it is not, then there is an arc $a = (u, v)$ for which $\pi^\star(u) - \pi^\star(v) + w(a) < 0$. In other words, we have that $w(P_u \circ a) < w(P_v)$. As P_v is a minimum weight $s^\star \to v$-path, we have that $P_u \circ a$ is only an $s^\star \to v$-walk. Moreover, $P_u \circ a$ has to contain a negative closed walk, as otherwise it would contain a $s^\star \to v$-path of weight less than P_v. As P_u is a path, this closed walk is indeed a cycle O containing the arc a. Now $s^\star$ has only outgoing arcs, thus the cycle lies in $G^\star - s^\star = G$, a contradiction to G having no negative cycles. So $\pi^\star$ is a feasible potential for $G^\star$. By $G = G^\star - s^\star$ being a subgraph of $G^\star$, we get that the function $\pi = \pi^\star|_{V(G)}$ is a feasible potential for G. □

With this construction in place, we can make our observations about integral feasible potentials with few distinct values.

Lemma 6.25. *Let G be a directed graph, and $w : A(G) \to \{-1, 0, 1\}$ arc weights for it. If G contains no negative cycles, then there is a feasible potential $\pi : V(G) \to [0, w_-]$, where $w_- = |w^{-1}(-1)|$.*

Proof. First we use Lemma 6.24 to construct the auxiliary graph $G^\star$ with its function $\pi^\star$. Now note that any $s^\star \to v$-path can contain at most w_--many negative arcs and each of those has weight -1. Thus, we have that $\pi^\star(v) = w(P_v) \geq -w_-$ for all $v \in V(G^\star)$. Moreover, the arc $(s^\star, v)$ always forms an $s^\star \to v$-path of weight 0. By all arcs weights being integral, we have that $\pi^\star$ is an integer function with values in $[-w_-, 0]$. Now, by Lemma 6.24, $\pi = \pi^\star|_{V(G)}$ is a feasible potential for G, which has only integer values in $[-w_-, 0]$. Shifting all values by w_- shows the theorem. □

Lemma 6.26. *Let G be a strongly connected, directed graph, and $w : A(G) \to \{-1, 0, 1\}$ arc weights for it. If G contains no negative cycles, then there is a feasible potential $\pi : V(G) \to [0, w_+]$, where $w_+ = |w^{-1}(1)|$.*

Proof. First we use Lemma 6.24 to construct the auxiliary graph $G^\star$ with its function $\pi^\star$. Assume now for contradiction, that any $s^\star \to v$-path P_v has weight less than $-w_+$. Let P be the $u \to v$-subpath of P_v that contains everything but the first arc $a = (s^\star, u)$ of P_v. As $w(a) = 0$ we know that $w(P) < -w_+$ and moreover P exists in G. Now, G is strongly connected, and therefore there is a $v \to u$-path Q in G. G contains only w_+-many positive arcs and all of them have weight 1, implying $w(Q) \leq w_+$. Hence, $P \circ Q$ is a closed walk of weight $w(P) + w(Q) < w_+ - w_+ = 0$. This negative closed walk contains a negative cycle, in contradiction to G having none of those. Thus, we have that $\pi^\star(v) = w(P_v) \geq -w_+$ for all $v \in V(G^\star)$. Moreover, the arc $(s^\star, v)$ always forms an $s^\star \to v$-path of weight 0. By all arcs weights being integral, we have that $\pi^\star$ is an integer function with values in $[-w_+, 0]$. Now, by Lemma 6.24, $\pi = \pi^\star|_{V(G)}$ is a feasible potential for G, which has only integer values in $[-w_+, 0]$. Shifting all values by w_+ shows the theorem. □

Lemma 6.27. *Let G be a directed graph, and $w : A(G) \to \{-1, 0, 1\}$ arc weights for it. If G contains no negative cycles, then there is a feasible potential $\pi : V(G) \to [0, 2^{\mathrm{td}(G)}]$.*

Proof. First we use Lemma 6.24 to construct the auxiliary graph $G^\star$ with its function $\pi^\star$. By Lemma 6.3 we have that any path in G contains at most $2^{\mathrm{td}(G)}$ many arcs. Moreover, any $s^\star \to v$-path consists of an arc $(s^\star, u)$ of weight 0, followed by an $u \to v$-path in G. Thus, any $s^\star \to v$-path can contain at most $2^{\mathrm{td}(G)}$ many negative arcs and each of those has weight -1. Thus, we have that $\pi^\star(v) = w(P_v) \geq -2^{\mathrm{td}(G)}$ for all $v \in V(G^\star)$. Moreover, the arc $(s^\star, v)$ always forms an $s^\star \to v$-path of weight 0. By all arcs weights being integral, we have that $\pi^\star$ is an integer function with values in $[-2^{\mathrm{td}(G)}, 0]$. Now, by Lemma 6.24, $\pi = \pi^\star|_{V(G)}$ is a feasible potential for G, which has only integer values in $[-2^{\mathrm{td}(G)}, 0]$. Shifting all values by $2^{\mathrm{td}(G)}$ shows the theorem. □

6.5.5 Dynamic Program for Treewidth and Bounded Feasible Potentials

We now want to apply our findings on feasible potentials with few different values (see last chapter) to a dynamic program utilizing the treewidth. The overall approach is computing a nice treewidth decomposition of our graph (see Definition 6.4 and Theorem 6.5) and then guessing via a dynamic program the feasible potential of some solution on each bag of the tree decomposition. The deleted arcs are then exactly those that violate the guessed potential. However, when we parameterize in the number of positive arcs, we are only guaranteed a feasible potential with few different values for

every strongly connected component of $G - S$. Therefore we also have to guess a topological order of the strongly connected components of $G - S$ when restricted to the bags of the tree decomposition. To handle both cases simultaneously, we introduce a set $\mathcal{C}$ which contains ordered partitions of $V(G)$ that represent components for which the guessed potential could be feasible. For the parameter w_- the single partition consisting of all vertices suffices as set $\mathcal{C}$. In the w_+ case, $\mathcal{C}$ has to contain a topological order of the strongly connected components of $G - S$. In this case we choose $\mathcal{C}$ as all ordered partitions of $V(G)$. We generalize the properties we need for $\mathcal{C}$ and our feasible potential to unify both choices.

Definition 6.28. *Let G be a directed graph and $w : A(G) \to \mathbb{Z}$ a weight function on its arcs. We call an arc set $S \subseteq A(G)$ $((C_1, \ldots, C_t), \pi)$-*feasible *for some ordered partition $(C_1, \ldots, C_t)$ of $V(G)$ and some $\pi : V(G) \to \mathbb{Z}$, if for all arcs $a = (p, q) \in A(G) \setminus S$ with $p \in C_i$ and $q \in C_j$ we have either $i < j$ or $i = j$ and $\pi(p) - \pi(q) + w(a) \geq 0$.*

For an ordered partition $(C_1, \ldots, C_t)$ of $V(G)$ and some $U \subseteq V(G)$, we call an ordered partition $(C'_1, \ldots, C'_{t'})$ of U the projection of $(C_1, \ldots, C_t)$ on U*, denoted by $(C_1, \ldots, C_t)|_U$, if for all $u \in C_i \cap C'_p$ and $v \in C_j \cap C'_q$ we have $i < j$ if and only if $p < q$. For a set $\mathcal{C}$ of ordered partitions of $V(G)$, we denote by $\mathcal{C}|_U$ the set $\{C \mid_U \mid C \in \mathcal{C}\}$.*

Similarly, for some $U \subseteq V(G)$ and a pair (C, π) consisting of an ordered partition C of $V(G)$ and $\pi : V(G) \to \mathbb{Z}$, we call (C', π') projection of (C, π) on U *for C' being the projection of C on U and $\pi' = \pi|_U$.*

*Let T be a tree decomposition of G, i.e. a tree decomposition of the underlying undirected graph of G. We say that a set $\mathcal{C}$ of ordered partitions of $V(G)$ is T-*compatible*, if for every $C, C' \in \mathcal{C}$ with $C|_{B_x} = C'|_{B_x}$, we have that there is a $C^\star \in \mathcal{C}$, with $C^\star|_{V(G_x)} = C|_{V(G_x)}$ and $C^\star|_{V(G)\setminus(V(G_x)\setminus B_x)} = C'|_{V(G)\setminus(V(G_x)\setminus B_x)}$.*

We are now able to state our general algorithm.

Lemma 6.29. *Let (G, w, k) be a* Negative Directed Feedback Arc Set *instance with weights $w : A(G) \to \mathbb{Z}$. Given a nice tree decomposition T of G with vertex bags $(B_x)_{x \in V(T)}$ of width $\mathcal{O}(\mathrm{tw}(G))$ and a set $\mathcal{C}$ of ordered partition of $V(G)$ compatible with T and two integers $a \leq b$, there is an algorithm that in time $f(\mathcal{C}, T)2^{\mathcal{O}(\mathrm{tw}(G)\log(b-a))} \cdot (n + m)$ computes a minimum size negative directed feedback arc set that is (C, π)-feasible for some $C \in \mathcal{C}$ and $\pi : V(G) \to \mathbb{Z} \cap [a, b]$. Here $f(\mathcal{C}, T)$ is the maximum time needed to enumerate $\mathcal{C}|_{B_x}$ for any $x \in V(T)$.*

Proof. We compute a dynamic programming table D via dynamic program on $V(T)$ from the leaves upwards. The dynamic program table has an entry $D[x, \kappa]$ for every $x \in V(T)$ and κ, where κ consists of an ordered partition $C \in \mathcal{C}|_{B_x}$ and an integer-valued function $\pi : B_x \to \mathbb{Z} \cap [a, b]$. The entry $D[x, \kappa]$ contains an arc set $S_{x,\kappa} \subseteq A(G_x)$ with G_x being the graph induced by the subtree decomposition of T rooted at x. Our dynamic program is motivated by the following observation:

Claim 1. *Let $S \subseteq A(G_x)$ be a minimum size set that is (C, π)-feasible for some $C \in \mathcal{C}|_{V(G_x)}$ and $\pi : V(G_x) \to \mathbb{Z} \cap [a, b]$. Let $((C^x_1, \ldots, C^x_{t^x}), \pi^x)$ be the projection of (C, π) to B_x. Then S contains exactly those arcs $a = (p, q) \in A(G[B_x])$ with $p \in C^x_i$ and $q \in C^x_j$ such that either $j < i$ or $i = j$ and $\pi^x(p) - \pi^x(q) + w(a) < 0$.*

Proof of Claim 1. It has to contain these arcs as otherwise it would not be (C, π)-feasible. If it contains more than these arcs, removing the rest would not lead to a violation of the (C, π)-feasibility and obtaining a smaller set S, a contradiction. ■

Here is how the entries are computed for $x \in V(T)$ depending on its node type:

leaf node As $B_x = \emptyset$, the only choice for κ are the empty partition () and π being the empty function ε and we set $D[x, ((), \varepsilon)] = \emptyset$.

introduce nodes Let v be the newly introduced vertex to the bag B_x and let y be the child of x in T. Let κ consist of an ordered partition $(C_1, \dots C_t) \in \mathcal{C}|_{B_x}$ and an integer-valued function $\pi : B_x \to \mathbb{Z} \cap [a, b]$. Define S_v to be the arcs $a = (p, q) \in \delta(v)$ with $p \in C_i$ and $q \in C_j$ such that either $j < i$ or $i = j$ and $\pi(p) - \pi(q) + w(a) < 0$. Then set

$$D[x, \kappa] = S_v \cup D[y, \kappa'],$$

where κ' is the projection of κ to B_y.

forget nodes Let y be the child of x in T. Let κ consist of an ordered partition $C \in \mathcal{C}|_{B_x}$ and an integer-valued function $\pi : B_x \to \mathbb{Z} \cap [a, b]$. Denote by $K(y, \kappa)$ the set of all $\kappa^\star$ consisting of ordered partitions $C^\star \in \mathcal{C}|_{B_y}$ and $\pi^\star : B_y \to \mathbb{Z} \cap [a, b]$, whose projection to B_x is κ. Then set

$$D[x, \pi] = \operatorname*{argmin}_{\substack{S = D[y, \kappa^\star] \\ \kappa^\star \in K(y, \kappa)}} |S|.$$

merge nodes Let y_1 and y_2 the two children of x. Then for every possible choice of κ, we set

$$D[x, \kappa] = D[y_1, \kappa] \cup D[y_2, \kappa].$$

We claim that at the root r of T, the unique entry $D[r, ((), \varepsilon)]$, where ε denotes the empty function, contains a negative directed feedback arc set for G of minimum size. First note, that this is really the unique entry as $B_r = \emptyset$.

We are going to prove the following stronger statement: the entry $D[x, \kappa]$ contains a minimum-size set $S \subseteq A(G_x)$ that is (C, π)-feasible for some $C \in \mathcal{C}|_{V(G_x)}$ and $\pi : V(G_x) \to \mathbb{Z} \cap [a, b]$ with the projection of (C, π) to B_x being κ. That is, we have to prove, that the $D[x, \kappa]$'s indeed contain a set that is (C, π)-feasible as above and that it has minimum size among those. We first focus on the (C, π)-feasibility, which we prove by induction from leaf vertices to the root. Let $x \in V(T)$ be a node, such that the feasibility property holds for all vertices y that are below x in T.

leaf node As G_x is the empty graph, the statement holds trivially.

introduce node For the keys κ, we enumerate all ordered partitions $C \in \mathcal{C}|_{B_x}$. Let $C^\uparrow \in \mathcal{C}$ be an ordered partition that projects to C in B_x. The set $D[y, \kappa']$ we chose is by induction $(C'|_{V(G_y)}, \pi')$-feasible on G_y. By T-compatibility of $\mathcal{C}$ we have that there is a $C^\star \in \mathcal{C}$, with $C^\star|_{V(G_y)} = C'|_{V(G_y)}$ and $C^\star|_{V(G) \setminus (V(G_y) \setminus B_y)} = C^\uparrow|_{V(G) \setminus (V(G_y) \setminus B_y)}$. Thus, by construction our set is $(C^\star|_{V(G_x)}, \pi^\star|_{V(G_x)})$-feasible, where $\pi^\star(v) = \pi(v)$ and $\pi^\star(u) = \pi'(u)$ for all $u \in V(G_x) \setminus \{v\}$.

forget node As $G_x = G_y$ the statement holds trivially.

merge nodes By induction, let $D[y_i, \kappa]$ be $(C_i|_{V(G_{y_i})}, \pi_i)$-feasible. By T-compatibility of $\mathcal{C}$ we have that there is a $C^\star \in \mathcal{C}$ such that

$$C^\star|_{V(G_x)} = C_1|_{V(G_x)} \text{ and } C^\star|_{V(G)\setminus(V(G_x)\setminus B_x)} = C_2|_{V(G)\setminus(V(G_x)\setminus B_x)}.$$

With $\pi^\star(u) = \pi_1(u)$ for all $u \in V(G_{y_1})$ and $\pi^\star(u) = \pi_2(u)$ otherwise, we get that our chosen set is $(C^\star|_{V(G_x)}, \pi^\star|_{V(G_x)})$-feasible.

Now assume for contradiction that our set is not the minimum choice among the (C, π)-feasible ones. Then there is a node $x \in V(T)$ such that our statement holds for all nodes y in the subtree of T rooted at x but not x (with possibly x being a leaf and the set of other nodes being empty). In particular, we made the minimum choice for all these nodes y and all κ's. We make a case distinction based on the type of x.

leaf node G_x is empty and thus the empty set is the right choice of $D[x, ((), \varepsilon)]$.

introduce node Assume there is a set of smaller size $S^\star$ that is $(C^\star, \pi^\star)$-feasible for some $C^\star = (C_1^\star, \dots C_t^\star) \in \mathcal{C}|_{V(G_x)}$ and $\pi^\star : V(G_x) \to \mathbb{Z} \cap [a, b]$ with $(C^\star, \pi^\star)$'s projection to B_x being κ. Choose $S^\star$ smallest possible among all such choices. By Claim 1, we have that $S^\star \cap A(G[B_x])$ contains exactly those arcs $a = (p, q) \in A(G[B_x])$ with $p \in C_i^\star$ and $q \in C_j^\star$ such that either $j < i$ or $i = j$ and $\pi^\star(p) - \pi^\star(q) + w(a) < 0$. In particular, $S^\star \cap \delta(v)$ and $D[x, \kappa] \cap \delta(v)$ are identical by $(C^\star, \pi^\star)$ projecting to κ. Thus, $S^\star \setminus \delta(v)$ is $(C^\star|_{V(G_y)}, \pi^\star|_{V(G_y)})$-feasible by $G_y - (S^\star \setminus \delta(v))$ being a subgraph of $G_x - S^\star$ and projects down to the same κ' as κ in B_y. But $S^\star \setminus \delta(v)$ is of size smaller than $D[y, \kappa|_{B_y}]$, a contradiction to $D[y, \kappa|_{B_y}]$ containing the smallest such arc set.

forget nodes We have that $G_x = G_y$ and thus any candidate for $D[x, \kappa]$ is also a candidate for $D[y, \kappa^\star]$ by extending the $\kappa^\star$ in a way that matches the candidate. By taking the minimum over the possible choices of $\kappa^\star$ and the claim holding for y we get that $D[x, \pi]$ is indeed such a set of minimum size.

merge nodes Assume there is a set of smaller size $S^\star$ that is $(C^\star, \pi^\star)$-feasible for some $C^\star = (C_1^\star, \dots C_{t^\star}^\star) \in \mathcal{C}|_{V(G_x)}$ and $\pi^\star : V(G_x) \to \mathbb{Z} \cap [a, b]$ with $(C^\star, \pi^\star)$'s projection to B_x being κ. Choose $S^\star$ smallest possible among all such choices. By Claim 1, we have that $S^\star \cap A(G[B_x])$ contains exactly those arcs $a = (p, q) \in A(G[B_x])$ with $p \in C_i^\star$ and $q \in C_j^\star$ such that either $j < i$ or $i = j$ and $\pi^\star(p) - \pi^\star(q) + w(a) < 0$. In particular, we have that $S^\star \cap A(G[B_x])$, $D[y_1, \kappa] \cap A(G[B_x])$ and $D[y_2, \kappa] \cap A(G[B_x])$ are identical (as $B_x = B_{y_1} = B_{y_2}$). Thus, from $|S^\star| < |D[x, \kappa]|$, we have that $|S^\star \cap A(G_{y_i})| < |D[y_i, \kappa]|$ for some $i \in \{1, 2\}$. As $S^\star \cap A(G_{y_i})$ is a candidate for $D[y_i, \kappa]$, this is a contradiction to the minimality of $D[y_i, \kappa]$.

Thus, our algorithm computes a minimum size negative directed feedback arc set for G. For the run-time note that for every of the $\mathcal{O}(n)$ nodes of the tree decomposition T, we make a number of computations that is bounded by the number of κ's for the node itself and its up to two children. Any of those computations is polynomial in the size of the arc sets of the bag. Now, the number of κ's is bounded by the number of possible functions π and the time needed to enumerate $\mathcal{C}|_{B_x}$ for any $x \in V(T)$. The number of functions π is bounded by $(b - a + 1)^{\mathcal{O}(\mathrm{tw}(G))} = 2^{\mathcal{O}(\mathrm{tw}(G) \log(b-a))}$, showing the overall run-time. □

We can deduce three of our main results from this general algorithm.

Theorem 6.30. *Let (G, w, k) be a* Negative Directed Feedback Arc Set *instance with weights $w : A(G) \to \{-1, 0, 1\}$. Then we can find in time $2^{\mathcal{O}(\mathrm{tw}(G) \log w_-)}(n+m)$ a minimum solution to (G, w, k), where $w_- = |w^{-1}(-1)|$.*

Proof. Let $\mathcal{C}$ consist of the single partition $(V(G))$ of $V(G)$. Compute a nice tree decomposition T of G. Note that $\mathcal{C}$ is T-compatible. We know that any (C, π)-feasible set S for $C \in \mathcal{C}$ is a negative directed feedback arc set with π being a feasible potential of $G - S$. By Lemma 6.25, we know that for any negative directed feedback arc set S of (G, k), $G - S$ has a feasible potential $\pi : V(G) \to \mathbb{Z} \cap [0, w_-]$. Thus, we call Lemma 6.29 with T, $\mathcal{C}$, $a = 0$ and $b = w_-$ and get a negative directed feedback arc set S' of minimum size. We then check whether $|S'| \leq k$ or not and return the corresponding answer.

For the run-time note that by Theorem 6.6 we can compute a nice tree decomposition of width $\mathcal{O}(\mathrm{tw}(G))$ in $2^{\mathcal{O}(\mathrm{tw}(G))}(n + m)$ time. Moreover, the sets of the form $C|_{B_x}$ for any $x \in V(T)$ can be enumerated in constant time as they are exactly the set (B_x). □

Theorem 6.31. *Let (G, w, k) be a* Negative Directed Feedback Arc Set *instance with weights $w : A(G) \to \{-1, 0, 1\}$. Then we can find in time*

$$2^{\mathcal{O}(\mathrm{tw}(G)(\log \mathrm{tw}(G) + \log w_+))}(n + m)$$

a minimum solution to (G, w, k), where $w_+ = |w^{-1}(1)|$.

Proof. Let $\mathcal{C}$ consist of all ordered partitions of $V(G)$. Compute a nice tree decomposition T of G. By properties of tree decompositions we have that for any $x \in V(T)$, the sets $V(G_x)$ and $V(G) \setminus (V(G_x) \setminus B_x)$ only intersect in B_x. Thus, $\mathcal{C}$ is T-compatible as we can recombine any two ordered partitions that match on a subset B_x.

We know that any (C, π)-feasible set S for $C = (C_1, \ldots, C_t) \in \mathcal{C}$ is a negative directed feedback arc set with π being a feasible potential for every $G[C_i] - S$ and $(C_1, \ldots, C_t)$ is a (super-)partition of the strongly connected components of $G - S$. By Lemma 6.26, we know that for any negative directed feedback arc set S of (G, k), the strongly connected components of $G - S$ have a feasible potential $\pi : V(G) \to \mathbb{Z} \cap [0, w_+]$. Thus, we call Lemma 6.29 with T, $\mathcal{C}$, $a = 0$ and $b = w_+$ and get a negative directed feedback arc set S' of minimum size. We then check whether $|S'| \leq k$ or not and return the corresponding answer.

For the run-time note that by Theorem 6.6 we can compute a nice tree decomposition of width $\mathcal{O}(\mathrm{tw}(G))$ in $2^{\mathcal{O}(\mathrm{tw}(G))}(n + m)$ time. Moreover, the sets of the form $C|_{B_x}$ for any $x \in V(T)$ can be enumerated in time $2^{\mathcal{O}(\mathrm{tw}(G) \log \mathrm{tw}(G))}$ by taking every ordered partition of B_x. □

Theorem 6.32. *Let (G, w, k) be a* Negative Directed Feedback Arc Set *instance with weights $w : A(G) \to \{-1, 0, 1\}$. Then we can find in time $2^{\mathcal{O}((\mathrm{td}(G))^2)}(n+m)$ a minimum solution to (G, w, k).*

Proof. Let $\mathcal{C}$ consist of the single partition $(V(G))$ of $V(G)$. Compute a nice tree decomposition T of G. Note that $\mathcal{C}$ is T-compatible. We know that any (C, π)-feasible set S for $C \in \mathcal{C}$ is a negative directed feedback arc set with π being a feasible potential

of $G - S$. By Lemma 6.27, we know that for any negative directed feedback arc set S of (G, k), $G - S$ has a feasible potential $\pi : V(G) \to \mathbb{Z} \cap [0, 2^{\mathrm{td}(G)}]$. Thus, we call Lemma 6.29 with T, $\mathcal{C}$, $a = 0$ and $b = 2^{\mathrm{td}(G)}$ and get a negative directed feedback arc set S' of minimum size. We then check whether $|S'| \leq k$ or not and return the corresponding answer.

For the run-time note that by Theorem 6.6 we can compute a nice tree decomposition of width $\mathcal{O}(\mathrm{tw}(G))$ in $2^{\mathcal{O}(\mathrm{tw}(G))}(n+m)$ time. Moreover, the sets of the form $C|_{B_x}$ for any $x \in V(T)$ can be enumerated in constant time as they are exactly the set (B_x). Last but not least, we have $\mathrm{tw}(G) \leq \mathrm{td}(G)$ and thus an overall run-time of

$$2^{\mathcal{O}(\mathrm{td}(G)\log(2^{\mathrm{td}(G)}))}(n+m) = 2^{\mathcal{O}((\mathrm{td}(G))^2)}(n+m).$$

□

6.5.6 Algorithm for $\{-1, 1\}$ Weights with Few Negatives

In this section we give an algorithm for Negative Directed Feedback Arc Set when parameterized by w_-. However, this algorithm works only for weights of the form $w : A(G) \to \{-1, 1\}$. As stated in the results overview (Section 6.4), this parameter is open for weights of the form $w : A(G) \to \{-1, 0, 1\}$ and W[1]-hard for general integral weights. In the case of $\{-1, +1\}$-weights finding a negative directed feedback arc set however becomes easy by the following observation.

Lemma 6.33. *Let G be a directed graph with arc weights $w : A(G) \to \{-1, 1\}$. Then any negative cycle of G has length at most $2w_-$.*

Proof. Any cycle C of length more than $2w_-$ contains at least $(w_- + 1)$-many arcs of weight $+1$ and at most $-w_-$-many of weight -1. Thus, it has weight at least $+1$ and thus non-negative weight. □

Now recall Lemma 6.17.

Lemma 6.17. *There is an algorithm solving a Negative Directed Feedback Arc Set instance (G, w, k) in time $\mathcal{O}(L^k n^2 m)$, where L is an upper bound on the length of any negative cycle in G.*

Combining the two statements above leads to the following theorem.

Theorem 6.34. *There is an algorithm solving Negative Directed Feedback Arc Set with arc weights $w : A(G) \to \{-1, 1\}$ in time $\mathcal{O}((2w_-)^k n^2 m)$.*

Proof. Use Lemma 6.17 with Lemma 6.33 as upper bound on the length of negative cycles. □

Note that the run-time can be improved to $\mathcal{O}((k+1)w_-^k n^2 m)$ by guessing the number of -1 arcs in our solution first and enumerating all sets of that size. Then the algorithm of Lemma 6.17 can be modified to only make subroutine calls on arcs of weight $+1$, leading to an improved run-time.

Moreover, note that we can assume $k \leq w_-$ by Theorem 6.15, which shows that Negative Directed Feedback Arc Set with arc weights $w : A(G) \to \{-1, 1\}$ is fixed-parameter tractable in w_-.

6.5.7 Algorithm for $\{-1, 1\}$ Weights with Few Positives

We will now study Negative Directed Feedback Arc Set with weight functions $w : A(G) \to \{-1, +1\}$ when parameterized by $k + w_+$. As stated in the results overview Section 6.4 this parameter is open for weights of the form $w : A(G) \to \{-1, 0, 1\}$ and W[1]-hard for general integral weights.

The main observation for our algorithm is made in the following lemma:

Lemma 6.35. *Let G be a directed graph with arc weights $w : A(G) \to \{\pm 1\}$. Then either G has a negative cycle of length at most $2(w_+)^2 + 2w_+$, or every negative cycle C has some arc $a \in A(C)$ that lies only on negative cycles of G.*

Proof. First, if G has no negative cycles then we are done. We are also done if there is a negative cycle of length at most $2(w_+)^2 + 2w_+$. Hence, we can assume that G has a negative cycle and all negative cycles have length at least $2(w_+)^2 + 2w_+ + 1$

Suppose, for sake of contradiction, that each arc a in a negative cycle lies on a cycle C_a of non-negative weight. Let C be a shortest negative cycle in G. By the argumentation above, C has length at least $2(w_+)^2 + 2w_+ + 1$. In particular, C has length at least $w_+(w_+ + 1) + 1$ and contains at most w_+ arcs of weight $+1$; all other arcs of C have weight -1. By pigeonhole principle, there must be a segment P of C consisting of $w_+ + 1$ arcs of weight -1.

By assumption, for every arc $a = (v, w) \in A(P)$ we have a $w \to v$-path R_a of length at most $|C_a| - 1$. As $w(C_a) \geq 0$ and the C_a's contain at most w_+ arcs of weight $+1$ and all other arcs have weight -1 the cycles have length at most $2w_+$. Thus, the reverse paths R_a have length $|R_a| \leq |C_a| - 1 \leq 2w_+ - 1$. Say P is an $s \to t$-path, then by concatenating the R_a's with $a \in P$ we get an $t \to s$-walk R' which contains a $t \to s$-path R. The path R contains at most w_+ arcs of weight $+1$ thus $w(R) \leq w_+$.

Consider the closed walk $O = P \circ R$. Then $w(O) = w(P) + w(R) = -(w_+ + 1) + w(R) \leq -1$. Thus, O contains a negative cycle C'. Eventually,

$$\begin{aligned} |\ell(C')| &\leq |O| \\ &= |P| + |R| \\ &\leq |P| + \sum_{e \in E(P)} |R_e| \\ &\leq 2w_+(w_+ + 1) \\ &< 2(w_+)^2 + 2w_+ + 1 \leq |\ell(C)| \end{aligned}$$

yields a contradiction to the fact that C was a shortest negative cycle. □

This lemma forms the basis of our algorithm, Algorithm 1. First, the algorithm checks for negative cycles with up to $2(w_+)^2 + 2w_+$ arcs. It then guesses the arc contained in a solution like the algorithm in Lemma 6.17. Afterwards, we are left with a directed graph without short negative cycles. We now identify the set U of arcs which are not part of a non-negative cycle. Then, we know that $G - S$ may not contain a cycle on which an arc of U lies, as this cycle would be negative by definition of U. Likewise, any negative cycle in G has some arc in U by the previous lemma. For general sets U, this is exactly the Directed Subset Feedback Arc Set problem.

Directed Subset Feedback Arc Set	
Instance:	A graph G, an arc set $U \subseteq A(G)$ and an integer $k \in \mathbb{Z}_{\geq 0}$.
Task:	Find an arc set $X \subseteq A(G)$ of size at most k such that every cycle of $G - X$ is disjoint of U or decide that no such set exists.

The Directed Subset Feedback Arc Set problem was shown to be fixed-parameter tractable for parameter k by Chitnis et al. [CCHM15].

Theorem 6.36 ([CCHM15])**.** *Directed Subset Feedback Arc Set is solvable in time* $2^{\mathcal{O}(k^3)} \operatorname{poly}(n)$.

Input : A directed graph G with arc weights $w : A(G) \to \mathbb{Z}$ and $k \in \mathbb{Z}_{\geq 0}$.
Output: A set $S \subseteq A(G)$ of at most k arcs such that $G - S$ has no negative cycle, or `false` if no such set exists.

if $k < 0$ **then**
  return `false`.
if *there is some negative cycle* C *of length at most* $2(w_+)^2 + 2w_+$ *in* G **then**
  Branch on deleting an arc of C and try to solve with $k - 1$ by recursion.
else
  Identify the set U of all arcs which do not lie on a non-negative cycle.
  return `DirectedSubsetFeedbackArcSet` (G, U, k).

Algorithm 1: NegativeCycleDeletion

Before we can prove correctness and run-time we have to show how we can detect the set U of all arcs which lie only on negative cycles. We first argue that this problem is NP-hard even for weights $w : A(G) \to \{-1, +1\}$. To this end, we provide a reduction from the Hamiltonian s-t-Path problem, which for a directed graph H and vertices $s, t \in V(H)$ asks for an $s \to t$-path in H visiting each vertex of H exactly once. Its NP-hardness was shown by Karp [Kar72]. The reduction works as follows: Take the original directed graph H and two vertices $s, t \in V(H)$ which we want to test for the existence of a Hamiltonian path starting in s and ending in t. Add a path P of length $n - 1$ from t to s to the graph. Assign weight $+1$ to each arc of H, and weight -1 to each arc of P. Then an arc of P lies on a cycle of non-negative length if and only if there is a Hamiltonian $s \to t$-path in H.

However, for this construction of weights w we have $w_+ \in \Omega(n)$. We will now show that the task is indeed fixed-parameter tractable when parameterized by w_+. For that, the main observation is that every non-negative cycle has length at most $2w_+$. We now consider the Weighted ℓ-Path problem: given a directed graph G with arc weights $w : A(G) \to \mathbb{R}$ and numbers $W \in \mathbb{R}, \ell \in \mathbb{Z}_{\geq 0}$, the task is to find a path of length exactly ℓ and weight at least W in G. Zehavi [Zeh15] gave a fast algorithm for Weighted ℓ-Path, based on color coding-related techniques and representative sets.

Theorem 6.37 ([Zeh15])**.** *Weighted* ℓ*-Path can be solved in time* $2^{\mathcal{O}(\ell)} \cdot \mathcal{O}(m \log n)$.

So given an arc $a = (s, t)$ one can enumerate all path sizes ℓ from 1 to $2w_+ - 1$ and ask whether there is a $t \to s$-path of length ℓ of weight at least $-w(a)$. This way one can detect a non-negative cycle containing a.

Corollary 6.38. *Let G be a directed graph, let $w : A(G) \to \{\pm 1\}$ and $(s, t) \in A(G)$. Then one can detect in time $2^{\mathcal{O}(w_+)} \cdot m \log n$ if $a = (s, t)$ is part of some non-negative cycle C.*

Finally, we argue the correctness and run-time of Algorithm 1, proving the following:

Theorem 6.39. *Algorithm 1 is correct and solves an instance of* Negative Directed Feedback Arc Set *with $w : A(G) \to \{-1, 1\}$ in time $2^{\mathcal{O}(k^3 + w_+ + k \log w_+)} \operatorname{poly}(n)$.*

Proof. Obviously, we return correctly "false" if our set should contain less than zero items. Otherwise, we detect with help of Lemma 6.16 in time $\mathcal{O}(n^2 m)$ whether there is a negative cycle in G, and if the minimum-length negative cycle C has length at most $2(w_+)^2 + 2(w_+)$.

If such a cycle C exists, we split into $|\ell(C)|$ instances, one for each $a \in A(C)$, calling our algorithm recursively with parameter k decreased by one on $G - a$. This is correct, as one of the arcs of C needs to be deleted, and we just search for every arc if there is a solution containing this arc.

Otherwise, all negative cycles have length at least $2(w_-)^2 + 2(w_-) + 1$, and by Lemma 6.35 every negative cycle must have an arc not contained in a non-negative cycle. Choose one arc of each such cycle and gather them in the set $U_{<0}$. By definition of the set U, in our algorithm we have $U_{<0} \subseteq U$. Subset DFAS for (G, W, k) now asks for a set S of size at most k such that $G - S$ has no cycle containing an arc of W. As $U_{<0} \subseteq U$, we get that a solution for (G, U, k) is also a solution for $(G, U_{<0}, k)$. We now want to show that also the reverse direction holds and this solves our original problem. For this, let $S_{<0}$ be a solution for $(G, U_{<0}, k)$ and C a cycle in $G - S_{<0}$. If there is no such cycle, we are done. Otherwise, we know that C cannot be a cycle of negative weight as every cycle of negative weight has an arc in $U_{<0}$. But as $w(C) \geq 0$, our cycle C cannot contain an arc of U as those are not contained in non-negative cycles. Thus, $S_{<0}$ is a solution for (G, U, k). Also, all cycles in $G - S_{<0}$ are non-negative, and therefore $S_{<0}$ is also a solution to our Negative DFAS instance (G, k).

This shows the correctness of Algorithm 1. Its run-time can be bounded as follows: Detecting cycles in line 4 can be done in time $\mathcal{O}((2(w_+)^2 + 2w_+)nm)$. The branching step then creates up to $2(w_+)^2 + 2w_+$ instances with the parameter k decreased by one. As there is no other recursive call to this algorithm, we have at most $(2(w_+)^2 + 2w_+ + 1)^k$ instances for which this algorithm is called. In the end, we call the algorithm from Lemma 6.38 to compute the set U which takes time $2^{\mathcal{O}(w_+)} \cdot m^2 \log n$, as it is called for every arc. By Proposition 6.36, the final call to the Directed Subset Feedback Arc Set oracle takes time $2^{\mathcal{O}(k^3)} \operatorname{poly}(n)$.

Thus, we obtain an overall run-time of

$$(2w_+^2 + 2w_+ + 1)^k \cdot \left[(w_+^2 + 2w_+) \cdot \mathcal{O}(nm) + 2^{\mathcal{O}(w_+)} \cdot \mathcal{O}(m^2 \log n) + 2^{\mathcal{O}(k^3)} \operatorname{poly}(n)\right],$$

which simplifies to $(w_+)^{\mathcal{O}(k)} \cdot 2^{\mathcal{O}(k^3 + w_+)} \operatorname{poly}(n) = 2^{\mathcal{O}(k^3 + w_+ + k \log w_+)} \operatorname{poly}(n)$. □

6.6 Hardness Results

6.6.1 NP-Hardness for Number of Positive Arcs

For completeness of the hardness results, this section contains a short observation about the equivalence of Directed Feedback Arc Set and Negative Directed Feedback Arc Set instances where all arc weights are -1. This implies that Negative Directed Feedback Arc Set is NP-hard even in the case where all arc weights are -1. This implies the hardness result for w_+ and arc weights $w : A(G) \to \{-1, 0, 1\}$.

Theorem 6.40. *Negative Directed Feedback Arc Set is NP-hard even if all arc weights are -1.*

Proof. We show the theorem by a reduction from Directed Feedback Arc Set, which is NP-hard. Let (G, k) be an instance of Directed Feedback Arc Set. We claim that (G, w, k) with $w \equiv -1$ is an equivalent instance of Negative Directed Feedback Arc Set. Indeed, for every $S \subseteq A(G)$ there is a cycle C in $G - S$ if and only if $G - S$ with weights -1 contains the cycle C of weight $-|C| < 0$. □

6.6.2 NP-Hardness for Constant Pathwidth

In this section we show that Negative Directed Feedback Arc Set is NP-hard even for graphs of pathwidth 6. We show this hardness by reduction from Partition.

Partition	
Instance:	A set $\mathcal{A} = \{a_1, \ldots, a_n\}$ of positive integers.
Task:	Find a subset $\mathcal{A}'$ such that $\sum_{a_i \in \mathcal{A}'} a_i = \sum_{a_i \in \mathcal{A} \setminus \mathcal{A}'} a_i$ or decide that no such subset exists.

Using $A = \sum_{i=1}^{n} a_i$, we can reformulate Partition as the problem of finding a subset $\mathcal{A}'$ such that $\mathcal{A}'$ and $\mathcal{A} \setminus \mathcal{A}'$ each sum up to $\frac{A}{2}$ (or no such subset exists).

Karp [Kar72] showed that Partition is NP-complete.

Theorem 6.41 (Karp 1972)**.** *Partition is NP-complete.*

Now we are ready to state our hardness result.

Theorem 6.42. *Negative Directed Feedback Arc Set is NP-hard even for graphs of pathwidth 6 and one arc of positive weight.*

Proof. Let $\mathcal{A}$ be an instance of Partition and $A = \sum_{a_i \in \mathcal{A}} a_i$. For every number $a_i \in \mathcal{A}$ we construct a gadget G_i as follows (see Fig. 6.1 for an illustration). Let $V^{(i)} = \{s_i^{(j)}, t_i^{(j)}, x_i^{(j)}, y_i^{(j)} |\ j = 1, 2\}$ be the set of vertices in G_i. There are three different types of arcs. The first arc set $A_1^i = \{(x_i^{(j)}, y_i^{(j)}) |\ j = 1, 2\}$ with arc weight $-a_i$, contains the arcs we will consider for deletion later. The second arc set $A_2^i = \{(y_i^{(j)}, x_i^{(j+1)}), (y_i^{(j)}, x_i^{(j-1)}) |\ j = 1, 2\}$ with arc weight 0, contains arcs which enforce the deletion of arcs from the first arc set by inducing negative cycles. The last arc set $A_3^i = \{(s_i^{(j)}, t_i^{(j)}), (s_i^{(j)}, x_i^{(j)}), (y_i^{(j)}, t_i^{(j)}) |\ j = 1, 2\}$ with arc weight 0, contains arcs that connect the vertices $s_i^{(j)}$ and $t_i^{(j)}$ to the rest of the gadget.

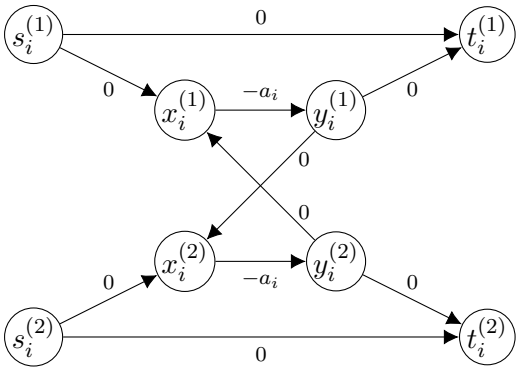


Figure 6.1: The gadget graph G_i.

The whole gadget G_i is then defined as $(V^{(i)}, A_1^{(i)} \cup A_2^{(i)} \cup A_3^{(i)})$. Out of these gadgets we construct the graph (G, w) of our Negative Directed Feedback Arc Set instance by taking the union of all gadgets G_i for $1 \leq i \leq n$, where we identify $t_i^{(j)} = s_{i+1}^{(j)}$ for $j = 1, 2$ and $i \in \{1, \ldots, n-1\}$. Additionally, we add two vertices s and t with the arcs $(s, s_1^{(j)})$ and $(t_n^{(j)}, t)$ for $j = 1, 2$ of weight 0. Finally, we add the arc (t, s) of weight $\frac{A}{2}$. We then choose (G, w, k) with $k = n$ as our Negative Directed Feedback Arc Set instance.

Next we show that $\mathcal{A}$ has a solution if and only if (G, w, n) has one.

Claim 1. *If $\mathcal{A}$ has a solution $\mathcal{A}'$, then there is a solution $S^\star$ to (G, w, n).*

Proof of Claim 1. For every $i \in \{1, \ldots, n\}$, we define an $s_i \in S^\star$ by

$$s_i = \begin{cases} (x_i^{(1)}, y_i^{(1)}) & \text{, if } a_i \in \mathcal{A}' \\ (x_i^{(2)}, y_i^{(2)}) & \text{, if } a_i \notin \mathcal{A}'. \end{cases}$$

Assume for contradiction that $G - S^\star$ contains a negative cycle C. Observe that for each $i \in \{1, \ldots, n\}$ the graph $G_i - s_i$ contains no negative cycle. Moreover, any gadget G_i can only be entered through one $s_i^{(j)}$ and left through $t_i^{(j)} = s_{i+1}^{(j)}$ with the same j. Thus, C has to start in s, pass through the gadgets G_i in order either on the $j = 1$ or the $j = 2$ side, then go to t and eventually use the backward arc (t, s). As the backward arc has weight $\frac{A}{2}$, C contains an $s \to t$-path P of weight less than $\frac{A}{2}$. Furthermore, P uses either only vertices with index $j = 1$ or $j = 2$. The only arcs of negative weight with index $j = 1$ in $G - S^\star$ are those $(x_i^{(1)}, y_i^{(1)})$ with $a_i \notin \mathcal{A}'$. For these we have $\sum_{a_i \notin \mathcal{A}'} w((x_i^{(1)}, y_i^{(1)})) = \sum_{a_i \notin \mathcal{A}'} -a_i = -\frac{A}{2}$, as $\mathcal{A}'$ is a solution to $\mathcal{A}$. Likewise, the only arcs of negative weight with index $j = 2$ in $G - S^\star$ are those $(x_i^{(2)}, y_i^{(2)})$ with $a_i \in \mathcal{A}'$. For these we have $\sum_{a_i \in \mathcal{A}'} w((x_i^{(2)}, y_i^{(2)})) = -\sum_{a_i \in \mathcal{A}'} a_i = -\frac{A}{2}$, as $\mathcal{A}'$ is a solution to $\mathcal{A}$. Thus, in any case, P has weight at least $-\frac{A}{2}$, a contradiction. ■

Claim 2. *If (G, w, n) has a solution $S^\star$ then $\mathcal{A}$ has a solution $\mathcal{A}'$.*

Proof of Claim 2. We choose $\mathcal{A}' = \{a_i \mid (x_i^{(1)}, y_i^{(1)}) \in S^\star\}$. Assume for contradiction that $\mathcal{A}'$ is not a solution. Then we have that either $\sum_{a_i \in \mathcal{A}'} a_i > \frac{A}{2}$ or $\sum_{a_i \notin \mathcal{A}'} a_i > \frac{A}{2}$.

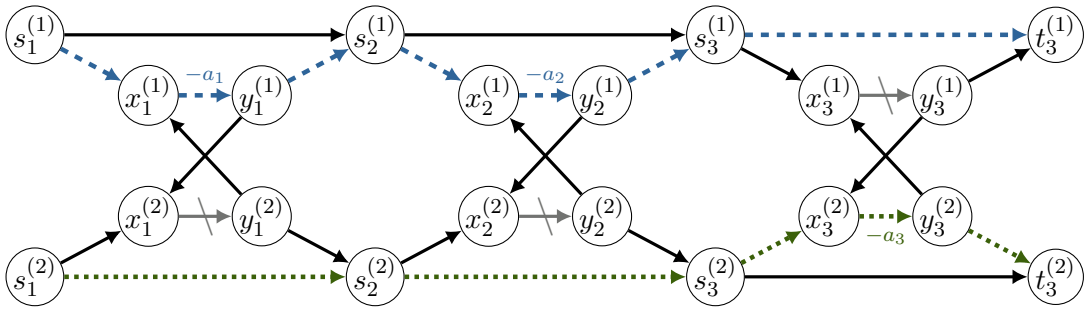


Figure 6.2: Union of three gadgets after the deletion of negative cycles. Deleted arcs are shown in gray with a backslash. The dashed blue path shows the shortest path from $s_1^{(1)}$ to $t_3^{(1)}$, and the dotted green path shows the shortest path from $s_1^{(2)}$ to $t_3^{(2)}$.

We consider now a minimum weight $s_1^{(j)} \to t_n^{(j)}$-path P_j in $G - S^\star$ for $j = 1, 2$ as depicted in Fig. 6.2. Note that for every gadget G_i, our solution $S^\star$ has to contain one of the arcs $(x_i^{(1)}, y_i^{(1)})$, $(x_i^{(2)}, y_i^{(2)})$, $(y_i^{(1)}, x_i^{(2)})$ and $(y_i^{(2)}, x_i^{(1)})$. As $S^\star$ contains only n elements (one for each gadget), the arcs $(s_i^{(j)}, t_i^{(j)})$ are always available and have weight 0. Moreover, if $(x_i^{(j)}, y_i^{(j)})$ is undeleted, the $s_i^{(j)} \to t_i^{(j)}$-path $(s_i^{(j)}, x_i^{(j)}) \circ (x_i^{(j)}, y_i^{(j)}) \circ (y_i^{(j)}, t_i^{(j)})$ of weight $-a_i$ exists. Thus, a minimum weight $s_1^{(j)} \to t_n^{(j)}$-path P_j has weight at most $-\sum_{a_i \in \mathcal{A}_j} a_i$ where $\mathcal{A}_j$ is the set of a_i's for which $(x_i^{(j)}, y_i^{(j)})$ is undeleted. By choice of $\mathcal{A}'$, we have that $\mathcal{A}_1 = \mathcal{A} \setminus \mathcal{A}'$ and $\mathcal{A}_2 \supseteq \mathcal{A}'$. So, if $\sum_{a_i \in \mathcal{A}'} a_i > \frac{A}{2}$, we have that $w(P_2) \leq -\sum_{a_i \in \mathcal{A}_2} a_i \leq -\sum_{a_i \in \mathcal{A}'} a_i < -\frac{A}{2}$. If instead $\sum_{a_i \notin \mathcal{A}'} a_i > \frac{A}{2}$, we have that $w(P_1) \leq -\sum_{a_i \in \mathcal{A}_1} a_i \leq -\sum_{a_i \notin \mathcal{A}'} a_i < -\frac{A}{2}$. Thus, in any case we have an $s_1^{(j)} \to t_n^{(j)}$-path P_j in $G - S^\star$ of weight less than $-\frac{A}{2}$ for some $j \in \{1, 2\}$.

Again, our solution $S^\star$ contains exactly one arc of every gadget and thus no arc incident to s or t. So, we can complete this path P_j to a cycle $(s, s_1^{(j)}) \circ P_j \circ (t_n^{(j)}, t) \circ (t, s)$ in $G - S^\star$ of weight $w(P_j) + \frac{A}{2} < 0$, a contradiction to $G - S^\star$ containing no negative cycles. ■

This completes the reduction from Partition to Negative DFAS and therefore shows the NP-hardness of Negative DFAS. It only remains to bound the pathwidth of the generated instances.

We show that the underlying graph of G has pathwidth at most 6, by providing a path decomposition $(P, \mathcal{B})$ of G of width 6. Let P be the path on $2n + 1$ vertices, corresponding to bags $B_1, ..., B_{2n+1} \in \mathcal{B}$:

- $B_{2i-1} = \{s, s_i^{(1)}, s_i^{(2)}, x_i^{(1)}, x_i^{(2)}, y_i^{(1)}, y_i^{(2)}\}$ for $i = 1, \ldots, n$
- $B_{2i} = \{s, s_i^{(1)}, s_i^{(2)}, y_i^{(1)}, y_i^{(2)}, t_i^{(1)}, t_i^{(2)}\}$ for $i = 1, \ldots, n$
- $B_{2n+1} = \{s, t_n^{(1)}, t_n^{(2)}, t\}$

Also, the arc (t, s) is the only arc of positive weight. □

6.6.3 W[1]-hardness for Treedepth and Few Positive Arcs

In this section we prove W[1]-hardness for Negative Directed Feedback Arc Set when parameterized by treedepth and number of positive arcs.

Theorem 6.43. *Negative Directed Feedback Arc Set is* W[1]*-hard when parameterized in the treedepth and number of positive arcs.*

Proof. We prove the theorem by reduction from Clique. The input of Clique consists of an undirected graph G and an integer k. The question is, whether there is a vertex set $X \subseteq V(G)$ of at least k vertices such that $G[X]$ forms a complete graph. Clique is $W[1]$-hard parameterized by k.

First we introduce a meta gadget that allows us to model different choices as minimum deletion sets influencing the minimum weight of a path between prescribed pairs of vertices. See Fig. 6.3 for an illustration.

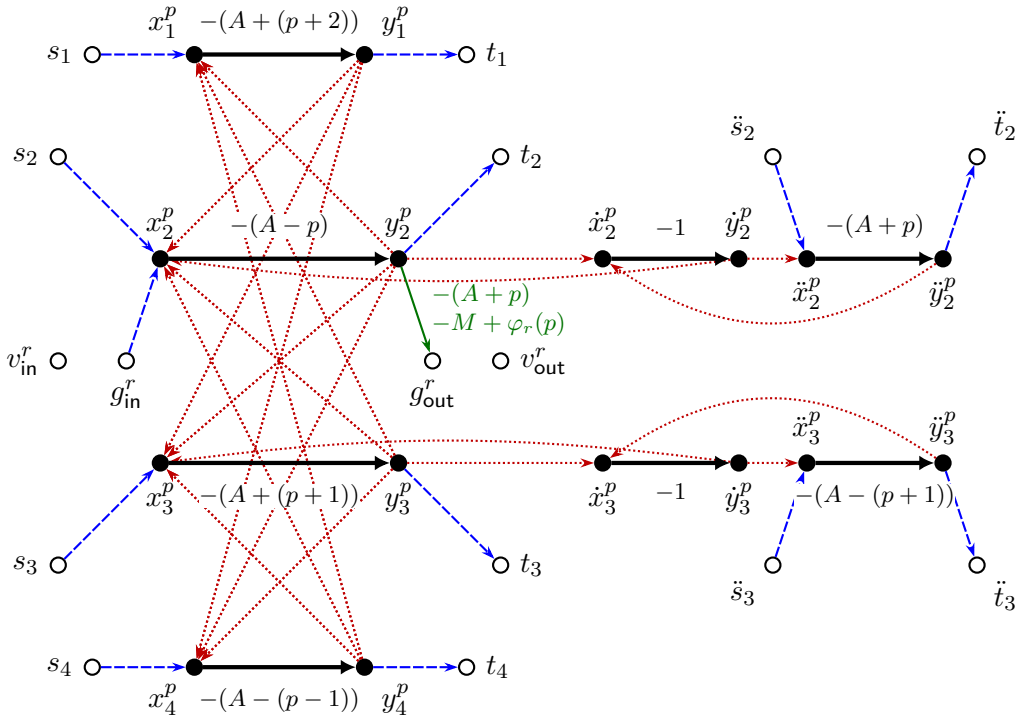
Let $a, b \in \mathbb{Z}_{>0}$ be positive integers. For every $r \in \{1, \dots, b\}$, let $\varphi_r : \{1, \dots, a\} \to \mathbb{Z} \cap [-M, M]$ be some function. We define the gadget $R^{a,b}$ in the following way: Let $A = a + 4$. For every $p \in \{-1, 0, \dots, a+1\}$ we have a smaller gadget consisting of the vertices x_i^p, y_i^p for $i \in \{1, 2, 3, 4\}$ and $\dot{x}_i^p, \dot{y}_i^p, \ddot{x}_i^p, \ddot{x}_i^p$ for $i \in \{2, 3\}$ that are interconnected by the following arcs:

- the arc (x_1^p, y_1^p) of weight $-(A + (p + 2))$, and
- the arc (x_2^p, y_2^p) of weight $-(A - p)$, and
- the arc (x_3^p, y_3^p) of weight $-(A + (p + 1))$, and
- the arc (x_4^p, y_4^p) of weight $-(A - (p - 1))$, and
- the arc $(\dot{x}_i^p, \dot{y}_i^p)$ of weight -1 for $i \in \{2, 3\}$, and
- the arc $(\ddot{x}_2^p, \ddot{y}_2^p)$ of weight $-(A + p)$, and
- the arc $(\ddot{x}_3^p, \ddot{y}_3^p)$ of weight $-(A - (p + 1))$, and
- the arcs (y_i^p, x_j^p) for any distinct $i, j \in \{1, 2, 3, 4\}$ all of weight -1, and
- the arcs $(y_i^p, \dot{x}_i^p), (\dot{y}_i^p, x_i^p), (\dot{y}_i^p, \ddot{x}_i^p), (\ddot{y}_i^p, \dot{x}_i^p)$ for $i \in \{2, 3\}$ all of weight -1.

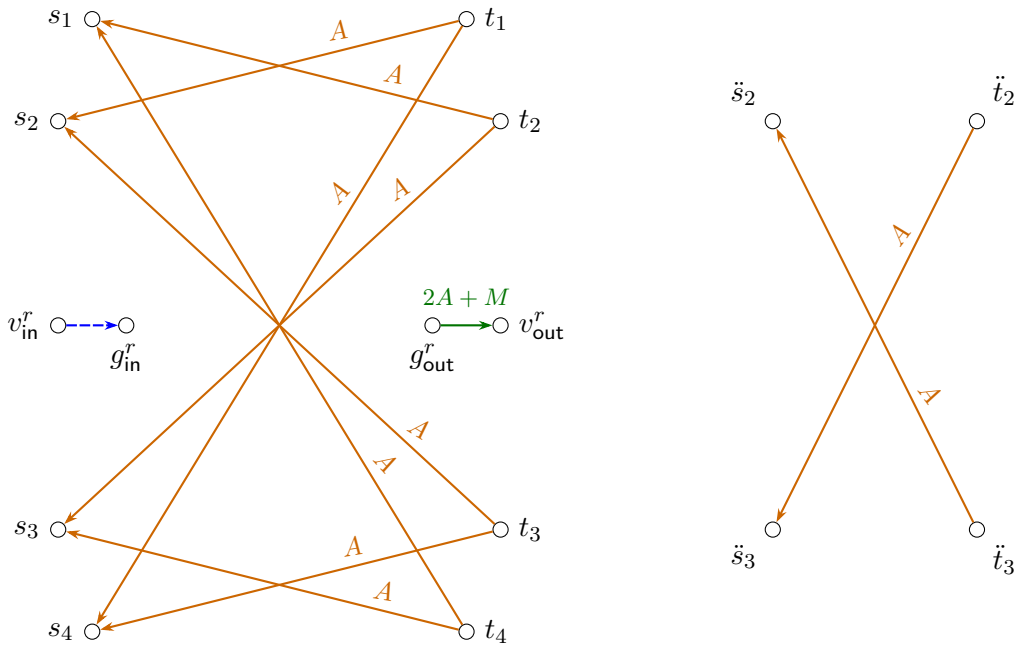
For $p = -1$ we duplicate the arc (x_1^{-1}, y_1^{-1}) and for $p = a + 1$ we duplicate the arc (x_4^{a+1}, y_4^{a+1}). Moreover, we duplicate the arcs $(\dot{y}_i^p, \ddot{x}_i^p), (\ddot{y}_i^p, \dot{x}_i^p)$ for $i \in \{2, 3\}$.

Shared by all these mini gadgets, there are vertices s_i, t_i for $i \in \{1, 2, 3, 4\}$. These are connected to the previous vertices by the arcs (s_i, x_i^p) and (y_i^p, t_i) for $i \in \{1, 2, 3, 4\}$, all of weight 0. The following arcs run between the shared vertices: (t_i, s_j) for $i, j \in \{1, 2, 3, 4\}$ with $|i - j|$ odd, all of weight A. Moreover there are vertices $\ddot{s}_i, \ddot{t}_i$ for $i \in \{2, 3\}$. These are adjacent to the arcs $(\ddot{s}_i, \ddot{x}_i^p)$ and $(\ddot{y}_i^p, \ddot{t}_i)$ of weight 0 for $i \in \{2, 3\}$, as well as to the arcs $(\ddot{t}_2, \ddot{s}_3)$ and $(\ddot{t}_3, \ddot{s}_2)$, both of weight A.

This defines the functional part of the gadget. To extract the information we want, we add for every $r \in \{1, \dots, b\}$ the vertices g_{in}^r and g_{out}^r to gather the information of the mini gadgets. For every $p \in \{1, \dots, a\}$ these vertices are connected to the mini-gadgets by an arc $(g_{\mathsf{in}}^r, x_2^p)$ of weight 0 and an arc $(y_2^p, g_{\mathsf{out}}^r)$ of weight $-(A + p) - M + \varphi_r(p)$.



(a) Meta Gadget (sub-level view)



(b) Meta Gadget (top-level view)

Figure 6.3: Overview of construction of a meta gadget. The dotted red arcs have weight -1, the dashed blue arcs have weight 0, the solid orange arcs have weight A. The black arcs model the choice we make for each gadget and the green arcs are used to extract information from this choice. For both of these arc types the weight is individual per arc.

Finally, for every $r \in \{1, \dots, b\}$, the gadget $R^{a,b}$ contains the vertices v^r_{in} and v^r_{out} that defines the interface to the outside. These are connected to g^r_{in} and g^r_{out} by an arc $(v^r_{\mathsf{in}}, g^r_{\mathsf{in}})$ of weight 0 and an arc $(g^r_{\mathsf{out}}, v^r_{\mathsf{out}})$ of weight $2A + M$.

We prove some claims about these meta-gadgets first:

Claim 1. *Every solution to $R^{a,b}$ as* Negative Directed Feedback Arc Set *instance has size at least $5(a+3)$. Moreover, any solution of size at most $5(a+3)$ deletes*

- *exactly three of the four arcs (x^p_i, y^p_i) with $i \in \{1,2,3,4\}$,*
- *exactly one of the two arcs $(\dot{x}^p_2, \dot{y}^p_2)$ and $(\ddot{x}^p_2, \ddot{y}^p_2)$, and*
- *exactly one of the two arcs $(\dot{x}^p_3, \dot{y}^p_3)$ and $(\ddot{x}^p_3, \ddot{y}^p_3)$.*

Proof of Claim 1. We prove that any solution to $R^{a,b}$ as Negative Directed Feedback Arc Set instance has to delete at least five arcs from every mini-gadget. For any $p \in \{-1, 0, \dots, a+1\}$ and consider the following cycles. For distinct $i, j \in \{1,2,3,4\}$ there is a cycle

$$(x^p_i, y^p_i) \circ (y^p_i, x^p_j) \circ (x^p_j, y^p_j) \circ (y^p_j, x^p_i),$$

which has negative weight as all arcs are negative. Moreover, there are the cycles

$$(\ddot{x}^p_i, \ddot{y}^p_i) \circ (\ddot{y}^p_i, \dot{x}^p_i) \circ (\dot{x}^p_i, \dot{y}^p_i) \circ (\dot{y}^p_i, \ddot{x}^p_i)$$

for $i \in \{2,3\}$, both of which are negative as all their arcs are negative. Now, any set of four arcs is disjoint from at least one of these cycles. So every solution deletes at least five arcs from every mini-gadget. As there are $a+3$ arc-disjoint mini-gadgets, every solution has size at least $5(a+3)$.

Note, any solution that deletes exactly five arcs from a mini-gadget must use at least two of these arcs to hit cycles involving the dotted vertices. Thus, there are only three arcs to intersect the cycles on non-dotted vertices. As the cycles for (i,j) and (j,i) are distinct (but not disjoint), these three arcs have the form (x^p_i, y^p_i) as otherwise one of the non-dotted cycles is not hit. To hit the two cycles involving dotted vertices, a solution has to spend exactly one arc on every cycle. As the arcs $(\dot{y}^p_i, \ddot{x}^p_i)$ and $(\ddot{y}^p_i, \dot{x}^p_i)$ are doubled, the solution must delete either $(\dot{x}^p_i, \dot{y}^p_i)$ or $(\ddot{x}^p_i, \ddot{y}^p_i)$ for $i \in \{2,3\}$. ■

Claim 2. *Consider $R^{a,b}$ as* Negative Directed Feedback Arc Set *instance. Then for every $p^\star \in \{1, \dots, a\}$ there is a solution $S_{p^\star}$ of size $5(a+3)$ such any $v^r_{\mathsf{in}} \to v^r_{\mathsf{out}}$-path in $R^{a,b} - S_{p^\star}$ has weight $\varphi^r(p^\star)$ for any $r \in \{1, \dots, b\}$.*

Proof of Claim 2. Let $S_{P^\star}$ consist of the following arcs with non-dotted endpoints:

- for $-1 \le p < p^\star - 1$, the arcs (x^p_2, y^p_2), (x^p_3, y^p_3), and (x^p_4, y^p_4),
- for $p^\star - 1$ the arcs $(x^{p^\star-1}_1, y^{p^\star-1}_1)$, $(x^{p^\star-1}_2, y^{p^\star-1}_2)$ and $(x^{p^\star-1}_4, y^{p^\star-1}_4)$,
- for $p^\star$ the arcs $(x^{p^\star}_1, y^{p^\star}_1)$, $(x^{p^\star}_3, y^{p^\star}_3)$ and $(x^{p^\star}_4, y^{p^\star}_4)$,
- for $p^\star < p \le a+1$, the arcs (x^p_1, y^p_1), (x^p_2, y^p_2), and (x^p_3, y^p_3).

Additionally, $S_{p^\star}$ contains the following arcs with dotted endpoints:

- for $-1 \leq p < p^\star - 1$, the arcs $(\ddot{x}_2^p, \ddot{y}_2^p)$ and $(\ddot{x}_3^p, \ddot{y}_3^p)$,
- for $p^\star - 1$ the arcs $(\ddot{x}_2^{p^\star-1}, \ddot{y}_2^{p^\star-1})$ and $(\dot{x}_3^{p^\star-1}, \dot{y}_3^{p^\star-1})$,
- for $p^\star$ the arcs $(\dot{x}_2^{p^\star}, \dot{y}_2^{p^\star})$ and $(\ddot{x}_3^{p^\star}, \ddot{y}_3^{p^\star})$,
- for $p^\star < p \leq a + 1$, the arcs $(\ddot{x}_2^p, \ddot{y}_2^p)$ and $(\ddot{x}_3^p, \ddot{y}_3^p)$.

We have to check that $R^{a,b} - S_{p^\star}$ has no cycle of negative weight. Note that the vertices x_i^p where (x_i^p, y_i^p) is deleted have out-degree zero and thus are not part of any cycles. The same holds for the vertices y_i^p where (x_i^p, y_i^p) is deleted as they have in-degree zero. The argument also holds for $\ddot{x}_i^p$ and $\ddot{y}_i^p$ with $(\ddot{x}_i^p, \ddot{y}_i^p)$ deleted. In addition, $\dot{x}_i^p$ and $\dot{y}_i^p$ are never part of any cycle, as for any p either $(\dot{x}_i^p, \dot{y}_i^p)$ is deleted, and they have either in- or out-degree zero, or (x_i^p, y_i^p) and $(\ddot{x}_i^p, \ddot{y}_i^p)$ are deleted, of which at least one is part of any cycle involving $\dot{x}_i^p$ and $\dot{y}_i^p$. Moreover, the vertices v_{in}^r, v_{out}^r, g_{in}^r and g_{out}^r are never part of any cycle in $R^{a,b}$. So the only vertices on a cycle are

- the vertices s_i, t_i (non-dotted, top level),
- the vertices x_i^p, y_i^p where (x_i^p, y_i^p) is not deleted, (non-dotted, mini-gadget),
- the vertices $\ddot{s}_i, \ddot{t}_i$ (dotted, top level),
- the vertices $\ddot{x}_i^p, \ddot{y}_i^p$ where $(\ddot{x}_i^p, \ddot{y}_i^p)$ is not deleted, (dotted, mini-gadget).

Note that as $\dot{x}_i^p$ and $\dot{y}_i^p$ are never part of any cycle, there is no cycle involving both the non-dotted and the dotted part of our gadget. Thus, we can analyze them separately.

First for the non-dotted part: By leaving out the vertices that do not form a cycle (see above), the mini-gadgets form an $s_i \to t_i$-path for certain i's. The only remaining arcs (those that are not part of these paths) are the backwards arcs (t_i, s_j) on the top-level. We analyze the $s_i \to t_i$-paths first. As we are interested only whether there are negative cycles and not their exact weight, we may restrict ourselves to minimum weight paths of this kind. Observe that in $R^{a,b} - S_{p^\star}$

- any $s_1 \to t_1$-path has weight at least $-(A + p^\star)$,
- the only $s_2 \to t_2$ path has weight $-(A - p^\star)$,
- the only $s_3 \to t_3$ path has weight $-(A + p^\star)$, and
- any $s_4 \to t_4$-path has weight at least $-(A + p^\star)$.

The backward arcs (t_i, s_j) have weight A and exist only for $|i - j|$ odd. Therefore, on every cycle we have that $s_i \to t_i$-paths of weight at least $-(A - p^\star)$ and of weight at least $-(A + p^\star)$ alternate. Also, between them lies an arc of weight A. To return to the starting point a walk has to consist of only such pairs (as the backward arcs only exist for $|i - j|$ odd. Each such pair has weight at least $-(A - p^\star) + A - (A + p^\star) + A = 0$ and thus every closed walk in the non-dotted part of $R^{a,b} - S_{p^\star}$ has non-negative weight.

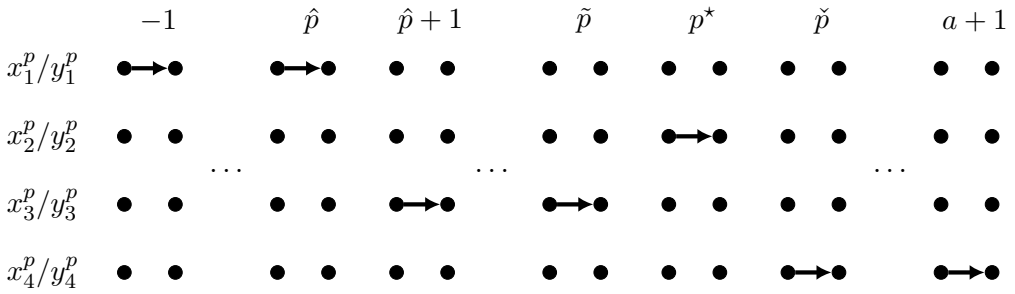


Figure 6.4: Structure of undeleted arcs (x_i^p, y_i^p) as proven in Claim 3.

For the dotted part of the gadget, note that of the arcs of the form $(\ddot{x}_i^p, \ddot{y}_i^p)$, only $(\ddot{x}_2^{p^\star}, \ddot{y}_2^{p^\star})$ and $(\ddot{x}_3^{p^\star-1}, \ddot{y}_3^{p^\star-1})$ are undeleted. So the only cycle in the dotted part is $\ddot{s}_2, \ddot{x}_2^{p^\star}, \ddot{y}_2^{p^\star}, \ddot{t}_2, \ddot{s}_3, \ddot{x}_3^{p^\star-1}, \ddot{y}_3^{p^\star-1}, \ddot{t}_3, \ddot{s}_2$ of weight $-(A+p^\star)+A-(A-(p^\star-1+1))+A=0$. Thus, there are no negative cycles in $R^{a,b}-S_{p^\star}$ and $S_{p^\star}$ is a solution of size $5(a+3)$.

It remains to show that any $v_{\mathsf{in}}^r \to v_{\mathsf{out}}^r$-path has weight $\varphi^r(p^\star)$ for any $r \in \{1,\dots,b\}$. Note that any $v_{\mathsf{in}}^r \to v_{\mathsf{out}}^r$-path in $R^{a,b}$ has to start with a subpath $v_{\mathsf{in}}^r, g_{\mathsf{in}}^r, x_2^p, y_2^p$ and to end with a (potentially overlapping) subpath $x_2^{p'}, y_2^{p'}, g_{\mathsf{out}}^r, v_{\mathsf{out}}^r$ for some (not necessarily distinct) $p, p' \in \{-1, 0, \dots, a+1\}$. As (x_2^p, y_2^p) is contained in $S_{p^\star}$ for any $p \neq p^\star$, we have that the only $v_{\mathsf{in}}^r \to v_{\mathsf{out}}^r$-path is $v_{\mathsf{in}}^r, g_{\mathsf{in}}^r, x_2^p, y_2^p, g_{\mathsf{out}}^r, v_{\mathsf{out}}^r$. It's weight is

$$-(A-p^\star)-(A+p^\star)-M+\varphi_r(p^\star)+2A+M=\varphi_r(p^\star). \qquad \blacksquare$$

Claim 3. *Let S be a solution to $R^{a,b}$ as* Negative Directed Feedback Arc Set *instance of size $5(a+3)$. Then there is a $p^\star \in \{1,\dots,a\}$ such that any $v_{\mathsf{in}}^r \to v_{\mathsf{out}}^r$-path in $R^{a,b}-S$ has weight $\varphi^r(p^\star)$ for any $r \in \{1,\dots,b\}$.*

Proof of Claim 3. We want to prove that S has the structure of undeleted arcs that is show in Fig. 6.4.

By Claim 1 our solution S has to delete exactly three arcs of type (s_i^p, t_i^p) in every mini-gadget, which is equivalent to leaving exactly one arc $(x_{i_p}^p, y_{i_p}^p)$ undeleted for every $p \in \{0,1,\dots,a+1\}$ with $i_p \in \{1,2,3,4\}$. Note that by doubling (x_1^{-1}, y_1^{-1}) and (x_4^{a+1}, y_4^{a+1}), these are the undeleted arcs for $p \in \{-1, a+1\}$.

For ease of notation we let $w_i(p)$ denote the weight of the arc (x_i^p, y_i^p) for any $i \in \{1,2,3,4\}$ and any $p \in \{-1,0,\dots,a+1\}$. Then for any two distinct indices $p, p' \in \{-1,0,\dots,a+1\}$ with $|i_p - i_{p'}|$ odd, we have that the cycle

$$(s_{i_p}, x_{i_p}^p) \circ (x_{i_p}^p, y_{i_p}^p) \circ (y_{i_p}^p, t_{i_p}) \circ (t_{i_p}, s_{i_{p'}}) \circ (s_{i_{p'}}, x_{i_{p'}}^{p'}) \circ (x_{i_{p'}}^{p'}, y_{i_{p'}}^{p'}) \circ (y_{i_{p'}}^{p'}, t_{i_{p'}}) \circ (t_{i_{p'}}, s_{i_p})$$

exists in $R^{a,b}-S$. This cycle we denote by $C_{p,p'}$. The weight of this cycle is $2A + w_{i_p}(p) + w_{i_{p'}}(p')$. As S is a solution to $R^{a,b}$ as Negative Directed Feedback Arc Set instance, these cycles always have non-negative weight.

Let $\hat{p}$ be the maximum p such that (x_1^p, y_1^p) is undeleted, i.e. $\hat{p} = \max\{p \mid i_p = 1\}$. Analogously, let $\check{p}$ be the minimum p such that (x_4^p, y_4^p) is undeleted, i.e. $\check{p} = \min\{p \mid i_p = 4\}$. Note that both exist, since we doubled the arcs (x_1^{-1}, y_1^{-1}) and (x_4^{p+1}, y_4^{p+1}).

So the cycle $C_{\check{p},\hat{p}}$ exists and has non-negative weight $0 \leq 2A-(A+(\hat{p}+2))-(A-(\check{p}-1)) = \check{p}-\hat{p}-3$. This is equivalent to $\hat{p}+3 \leq \check{p}$. So for every index $\hat{p} < p < \check{p}$ both arcs (x_1^p, y_1^p) and (x_4^p, y_4^p) are deleted and there are at least two such indices.

Next we show that $(x_2^{\hat{p}+1}, y_2^{\hat{p}+1})$ is also deleted. Otherwise, we have that $i_{\hat{p}+1} = 2$ and thus the cycle $C_{\hat{p},\hat{p}+1}$ has weight $2A - (A + (\hat{p}+2)) - (A - (\hat{p}+1)) = -1$, which is a contradiction to $C_{\hat{p},\hat{p}+1}$ having non-negative weight. So we have that $i_{\hat{p}+1} = 3$. Now define $\tilde{p}$ to be the maximum p such that (x_3^p, y_3^p) is undeleted, i.e. $\tilde{p} = \max\{p \mid i_p = 3\}$. By the previous argument we know that this maximum exists and $\hat{p}+1 \leq \tilde{p}$.

Now consider the cycle $C_{\check{p},\tilde{p}}$. Since this cycle again has non-negative weight, we have $0 \leq 2A - (A + (\check{p}-1)) - (A - (\tilde{p}+1)) = \check{p} - \tilde{p} - 2$. This is equivalent to $\tilde{p}+2 \leq \check{p}$.

Note that for the index $\tilde{p}+1$, we have that $\hat{p}, \tilde{p} < \tilde{p}+1 < \check{p}$. That means the only arc that can be undeleted for $\tilde{p}+1$ is $(x_2^{\tilde{p}+1}, y_2^{\tilde{p}+1})$, i.e. $i_{\tilde{p}+1} = 2$. We want to show that $p^\star = \tilde{p}+1$ is indeed the only index p with $i_p = 2$. For this let $\overleftarrow{p}^\star = \min\{p \mid i_p = 2\}$ and $\overrightarrow{p}^\star = \max\{p \mid i_p = 2\}$. By the weight of $C_{\overleftarrow{p}^\star,\tilde{p}}$ being non-negative, we get $0 \leq 2A - (A - \overleftarrow{p}^\star) - (A + (\tilde{p}+1)) = \overleftarrow{p}^\star - \tilde{p} - 1$, i.e. $\tilde{p}+1 \leq \overleftarrow{p}^\star$.

Now consider any $i \in \{2,3\}$ and any $p \in \{-1, 0, \ldots, a+1\}$. By Claim 1, we have that S contains either $(\dot{x}_i^p, \dot{y}_i^p)$ or $(\ddot{x}_i^p, \ddot{y}_i^p)$, but none of the arcs $(y_i^p, \dot{x}_i^p)$ and $(\dot{y}_i^p, x_i^p)$. Thus, if (x_i^p, y_i^p) is undeleted, by the negative cycle $(x_i^p, y_i^p) \circ (y_i^p, \dot{x}_i^p) \circ (\dot{x}_i^p, \dot{y}_i^p) \circ (\dot{y}_i^p, x_i^p)$, we have that $(\dot{x}_i^p, \dot{y}_i^p)$ is deleted and thus $(\ddot{x}_i^p, \ddot{y}_i^p)$ is undeleted. So for $\tilde{p}$, the arc $(\ddot{x}_3^{\tilde{p}}, \ddot{y}_3^{\tilde{p}})$ is undeleted, and for $\overrightarrow{p}^\star$ the arc $(\ddot{x}_2^{\overrightarrow{p}^\star}, \ddot{y}_2^{\overrightarrow{p}^\star})$ is undeleted. By Claim 1, we also get that none of the other arcs of the following cycle $\ddot{C}_{\overrightarrow{p}^\star,\tilde{p}}$ are deleted:

$$(\ddot{s}_2, \ddot{x}_2^{\overrightarrow{p}^\star}) \circ (\ddot{x}_2^{\overrightarrow{p}^\star}, \ddot{y}_2^{\overrightarrow{p}^\star}) \circ (\ddot{y}_2^{\overrightarrow{p}^\star}, \ddot{t}_2) \circ (\ddot{t}_2, \ddot{s}_3) \circ (\ddot{s}_3, \ddot{x}_3^{\tilde{p}}) \circ (\ddot{x}_3^{\tilde{p}}, \ddot{y}_3^{\tilde{p}}) \circ (\ddot{y}_3^{\tilde{p}}, \ddot{t}_3) \circ (\ddot{t}_3, \ddot{s}_2).$$

As this cycle exists in $R^{a,b} - S$, it has non-negative weight, which implies

$$0 \leq -(A + \overrightarrow{p}^\star) + A - (A - (\tilde{p}+1)) + A = \tilde{p} - \overrightarrow{p}^\star + 1.$$

This is equivalent to $\overrightarrow{p}^\star \leq \tilde{p}+1$. Overall we have $\tilde{p}+1 \leq \overleftarrow{p}^\star \leq \overrightarrow{p}^\star \leq \tilde{p}+1$. So indeed the only index p with (x_2^p, y_2^p) undeleted is $p^\star = \tilde{p}+1$.

Now we want to consider the $v_{\mathsf{in}}^r \to v_{\mathsf{out}}^r$-paths in $R^{a,b} - S$ for $r \in \{1, \ldots, b\}$. Note that each such path starts with the subpath $v_{\mathsf{in}}^r, g_{\mathsf{in}}^r, x_2^p, y_2^p$ and ends with the (potentially overlapping) subpath $x_2^{p'}, y_2^{p'}, g_{\mathsf{out}}^r, v_{\mathsf{out}}^r$ for $p, p' \in \{-1, 0, \ldots, a+1\}$. As the only index with (x_2^p, y_2^p) undeleted is $p^\star = \tilde{p}+1$, we have that the only $v_{\mathsf{in}}^r \to v_{\mathsf{out}}^r$-path in $R^{a,b} - S$ has the form $v_{\mathsf{in}}^r, g_{\mathsf{in}}^r, x_2^{p^\star}, y_2^{p^\star}, g_{\mathsf{out}}^r, v_{\mathsf{out}}^r$. This path has weight

$$-(A - p^\star) - (A + p^\star) - M + \varphi_r(p^\star) + 2A + M = \varphi_r(p^\star).$$

It only remains to show that $p^\star \in \{1, \ldots, a\}$. For this note that $-1 \leq \hat{p} \leq \tilde{p} - 1$ which is equivalent to $1 \leq \tilde{p} + 1 = p^\star$, and that $p^\star = \tilde{p} + 1 \leq \check{p} - 1 \leq a$. ■

After we have proven what minimum solutions look like for our meta gadget we can now give the overall reduction from Clique. Let $v_1, \ldots, v_n$ be an arbitrary ordering of $V(G)$ and $e_1, \ldots, e_m$ be an arbitrary ordering of $E(G)$. For every $i \in \{1, \ldots, k\}$, we introduce a vertex gadget H^i which is a copy of the meta gadget $R^{a,b}$ with $a = n$, $b = 2$, $M = n$ and the functions $\varphi_1(p) = p$ and $\varphi_2(p) = -p$. Moreover, we rename the vertices v_{in}^1 and v_{out}^1 of H_i to s and z_i^+ and the vertices v_{in}^2 and v_{out}^2 of H_i to s and z_i^-. For every $i, j \in \{1, \ldots, k\}$ with $i < j$, we introduce an edge gadget $H^{i,j}$ which is a copy of the meta gadget $R^{a,b}$ with $a = m$, $b = 4$, $M = n$ and the functions defined as follows. For every $e_\ell = \{v_p, v_q\} \in E(G)$ with $p \leq q$ we let $\varphi_1(\ell) = -p$, $\varphi_2(\ell) = p$, $\varphi_3(\ell) = -q$ and $\varphi_4(\ell) = q$.

Moreover, we rename the vertices in the following way:

- v^1_{in} and v^1_{out} of $H_{i,j}$ become z^+_i and t,
- v^2_{in} and v^2_{out} of $H_{i,j}$ become z^-_i and t,
- v^3_{in} and v^3_{out} of $H_{i,j}$ become z^+_j and t,
- v^4_{in} and v^4_{out} of $H_{i,j}$ become z^+_j and t,

Any renamed vertices sharing the same name are identified with each other. Finally, we add an arc (t, s) of weight 0. Call the resulting graph H. We claim that H as Negative Directed Feedback Arc Set instance has a solution of size at most $d = k \cdot 5(n+3) + \frac{1}{2}k(k+1) \cdot 5(m+3)$ if and only if G has a clique of size k.

For the forward direction let S be any solution to H as Negative Directed Feedback Arc Set instance of size at most d. By Claim 1, we know that S restricted to any H^i has size at least $5(n+3)$ and restricted to any $H^{i,j}$ has size at least $5(m+3)$. By d matching exactly the total of these numbers, we know that S has exactly these sizes in the gadgets. Moreover, (t, s) is not contained in S.

Using Claim 3, we get that for every H^i there is a $p_i \in \{1, \ldots, n\}$ such that any $s \to z^+_i$-path has weight p_i and any $s \to z^-_i$-path has weight $-p_i$. Also, we get that for every $H^{i,j}$ there is an $\ell_{i,j}$ such that for $e_{\ell_{i,j}} = \{v_p, v_q\}$ we have that any $z^+_i \to t$-path has weight $-p$, any $z^-_i \to t$-path has weight p, any $z^+_j \to t$-path has weight $-q$ and any $z^-_j \to t$-path has weight q. Let now

$$V' = \{v_{p_i} \mid i \in \{1, \ldots, k\}\} \subseteq V(G) \text{ and } E' = \{e_{\ell_{i,j}} \mid i, j \in \{1, \ldots, k\}, i < j\} \subseteq E(G).$$

We claim that (V', E') is a clique in G of size k. As V' and E' have exactly the right cardinality, we only have to proof that for any unordered pair $v_{p_i}, v_{p_j} \in V'$ there is an edge between them. Without loss on generality we can assume $i < j$, otherwise swap the indices. Moreover, let $e_{\ell_{i,j}} = \{v_p, v_q\}$.

Consider the cycle consisting of the arc (t, s), the $s \to z^+_i$-path in $H^i - S$ and the $z^+_i \to t$-path in $H^{i,j} - S$. This cycle is non-negative and has weight $p_i - p$, thus $p \leq p_i$. By considering the cycle consisting of the arc (t, s), the $s \to z^-_i$-path in $H^i - S$ and the $z^-_i \to t$-path in $H^{i,j} - S$, we get a non-negative cycle of weight $-p_i + p$. Thus, $p \geq p_i$ and combined with the former inequality, $p = p_i$. Analogously, by using the paths starting and ending in z^+_j and z^-_j instead, we get that $p_j = q$. Thus, we get that indeed the edge $\{v_{p_i}, v_{p_j}\} = \{v_p, v_q\} = e_{\ell_{i,j}} \in E(G)$. So G contains the clique (V', E') of size k.

For the backward direction consider any clique on vertices $v_{p_1}, \ldots, v_{p_k}$ in G. Choose by Claim 2 a solution S^i of size $5(n+3)$ for every gadget H^i such that any $s \to z^+_i$-path has weight p_i and any $s \to z^-_i$-path has weight $-p_i$. Analogously, by Claim 2 choose a solution $S^{i,j}$ of size $5(n+3)$ for every gadget $H^{i,j}$ such that any $z^+_i \to t$-path has weight $-p_i$, any $z^-_i \to t$-path has weight p_i, any $z^+_j \to t$-path has weight $-p_j$ and any $z^-_j \to t$-path has weight p_j. Note that the latter solution can be chosen that way, because $\{v_{p_i}, v_{p_j}\} \in E(G)$ and $i < j$.

We claim that $S = \bigcup_i S^i \cup \bigcup_{i<j} S^{i,j}$ is a solution to H of size d. The size bound follows from adding up the sizes of the individual solutions. It remains to check that $H - S$ contains no negative cycles. As S constraint to a single gadget was chosen as

a solution, the only negative cycles that can exists use some high-level vertices. There are two different types of such cycles. The first type uses the arc (t, s), followed by an $s \to z_i^+$-path in H^i for some i and an $z_i^+ \to t$-path in some $H^{i,j}$ or $H^{j,i}$. Our partial solutions were chosen such that these cycles have weight $0 + p_i - p_i$. The other type uses the arc (t, s), followed by an $s \to z_i^-$-path in H^i for some i and an $z_i^- \to t$-path in some $H^{i,j}$ or $H^{j,i}$. But by choice of S these paths have weight $0 - p_i + p_i$. So $H - S$ contains no negative cycles.

Bounding of Parameters: It remains to show that the treedepth and the number of arcs with positive weight is bounded. First we will derive bounds for these parameters on the meta gadgets $R^{a,b}$. For the treedepth we will use the recursive formula of Theorem 6.2. Consider the set

$$D = \{s^1, s^2, s^3, s^4, t^1, t^2, t^3, t^4, \ddot{s}_2, \ddot{s}_3, \ddot{t}_2, \ddot{t}_3\} \cup \{g^r_{\mathsf{in}}, g^r_{\mathsf{out}}, v^r_{\mathsf{in}}, v^r_{\mathsf{out}} \mid r \in \{1, \dots, b\}\}.$$

Note that $R^{a,b} - D$ consists only of the mini-gadgets, which are connected components of size 16 each. Thus, $\mathrm{td}(R^{a,b}) \leq |D| + 16 = 28 + 4b$.

For the number of arcs with positive weight note that A and M are chosen such that the only positive arc weights are A and $2A + M$. Of the former there are ten and of the later there are b many in every meta gadget. Thus, the number of arcs with positive weight in $R^{a,b}$ is bounded by $10 + b$.

For the graph H we can combine the above bounds. For treedepth note that for $D' = \{s, t\} \cup \{z_i^+, z_i^- \mid i \in \{1, \dots, k\}\}$, the graph $H - D'$ consists of connected components that only form subgraphs of the H^i and $H^{i,j}$. By Theorem 6.2:

$$\begin{aligned}\mathrm{td}(H) &\leq |D'| + \mathrm{td}(H - D') \leq |D'| + \sum_{i=1}^{k} \left(\mathrm{td}(H^i) + \sum_{j=i+1}^{k} \mathrm{td}(H^{i,j}) \right) \\ &\leq |D'| + 36k + 22k(k+1) \leq 22k^2 + 60k + 2.\end{aligned}$$

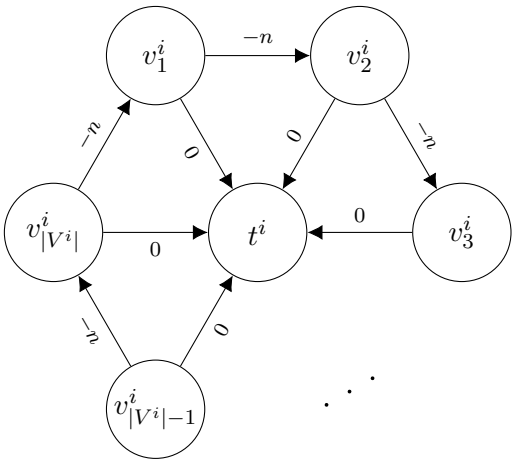
For the number of positive arcs note that we did not add any new arc of positive weight. Thus, the number of positive arcs in H is bounded by $12k + 7k(k + 1)$. □

6.6.4 W[1]-hardness for Pathwidth, Deletion Size and Few Positive Arcs

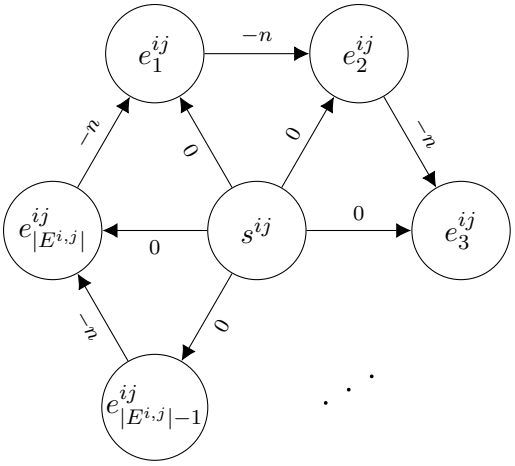
In this section we will show W[1]-hardness for Negative Directed Feedback Arc Set with general integral weights when parameterized in $\mathrm{pw}(G) + k + w_+$. We will see how this implies W[1]-hardness for Negative Directed Feedback Arc Set parameterized in $\mathrm{pw}(G) + k$ for instances with weights of the form $w : A(G) \to \{-1, 0, 1\}$.

Theorem 6.44. *Negative Directed Feedback Arc Set is* W[1]*-hard when parameterized in the pathwidth, deletion size and number of positive arcs. This still holds for instances (G, w, k), where $w : A(G) \to \mathbb{Z} \cap [-|V(G)|^2, |V(G)|^2]$.*

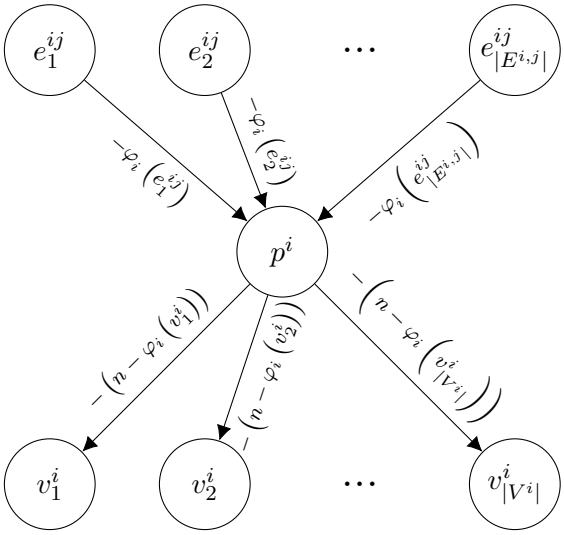
Proof. We prove W[1]-hardness by doing a parameterized reduction from Multicolored Clique. In Multicolored Clique, we are given an undirected graph G, a non-negative integer k, and a k-partition $\dot{\bigcup}_{i=1}^{k} V^i$ of $V(G)$. The task is to decide whether G contains a clique with k vertices $v_1, \dots, v_k$ where $v_i \in V^i$. This problem is W[1]-hard (cf. [CFK+15, Section 13.2]).



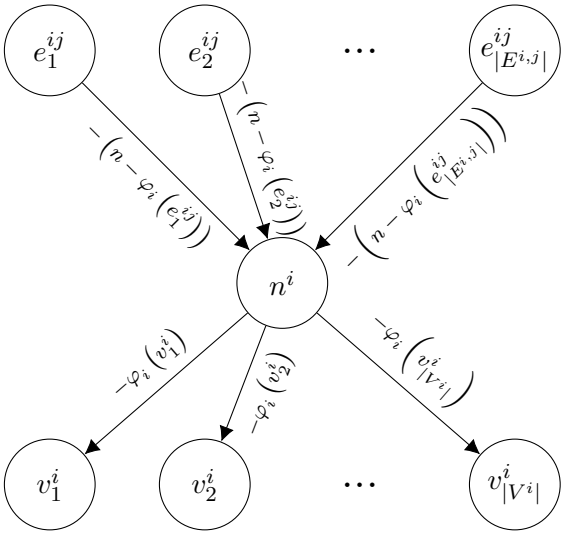
(a) The vertex gadget H^i



(b) The edge gadget $H^{i,j}$



(c) The connection gadget of p^i



(d) The connection gadget of n^i

Figure 6.5: Gadgets used in the construction of Theorem 6.44

For our reduction let (G, k) an instance of Multicolored Clique with $V(G) = \dot{\bigcup}_{i=1}^{k} V^i$. For ease of notation define for $1 \leq i < j \leq k$ the set $E^{i,j}$ to be the set of edges of type $\{v_i, v_j\}$ with $v_i \in V^i$ and $v_j \in V^j$. Moreover, we abuse this notation slightly by defining $E^{j,i} = E^{i,j}$ but with the order of vertices switched, i.e. $\{v_i, v_j\} \in E^{i,j}$ but $\{v_j, v_i\} \in E^{j,i}$.

We are going to construct an equivalent instance (H, w, d) of Negative Directed Feedback Arc Set. See Fig. 6.5 for an illustration. The graph H will have three types of gadgets:

1. a vertex gadget H^i representing V^i for every $i \in \{1, \ldots, k\}$,
2. an edge gadget $H^{i,j}$ representing $E^{i,j}$ for every $1 \leq i < j \leq k$, and
3. a consistency gadget C^i for every $i \in \{1, \ldots, k\}$.

The gadgets H^i representing $V^i = \{v_1^i, \ldots, v_{|V^i|}^i\}$ consist of one vertex t^i and the vertices of V^i. H^i contains two types of arcs. It has an arc (v_r^i, t^i) of weight 0 for every $v_r^i \in V^i$ and an arc (v_r^i, v_{r+1}^i) of weight $-n$ for every $r \in \{1, \ldots, |V^i|\}$, where we define $v_{|V^i|+1}^i = v_1^i$.

The gadgets $H^{i,j}$ representing $E^{i,j} = \{e_1^{i,j}, \ldots, e_{|E^{i,j}|}^{i,j}\}$ consist of one vertex $s^{i,j}$ and one vertex for every $e_r^{i,j} \in E^{i,j}$. $H^{i,j}$ contains two types of arcs. It has an arc $(s^{i,j}, e_r^{i,j})$ of weight 0 for every $e_r^{i,j} \in E^{i,j}$ and an arc $(e_r^{i,j}, e_{r+1}^{i,j})$ of weight $-n$ for every $r \in \{1, \ldots, |E^{i,j}|\}$, where we define $e_{|E^{i,j}|+1}^{i,j} = e_1^{i,j}$.

Finally, the consistency gadgets C_i consist of the vertices n^i and p^i and interconnect the existing gadgets as follows. First there is an arc $(t^i, s^{i,j})$ of weight $n(|V^i| + |E^{i,j}| - 1)$ for every $j \in \{1, \ldots, k\} \setminus \{i\}$. Then we need a bijective function $\varphi_i : V^i \to \{1, \ldots, |V^i|\}$ (for example $\varphi_i(v_r^i) = r$ will do). We abuse the notation and that every edge of E has at most one endpoint in every V^i and define for $e^{i,j} = \{v^i, v^j\} \in E^{i,j}$ that $\varphi_i(e^{i,j}) = \varphi_i(v^i)$. With this definition we add the following arcs to C^i:

- $(e^{i,j}, n^i)$ of weight $-(n - \varphi_i(e^{i,j}))$ for every $j \in \{1, \ldots, k\} \setminus \{i\}$ and $e^{i,j} \in E^{i,j}$,
- $(e^{i,j}, p^i)$ of weight $-\varphi_i(e^{i,j})$ for every $j \in \{1, \ldots, k\} \setminus \{i\}$ and $e^{i,j} \in E^{i,j}$,
- (n^i, v^i) of weight $-\varphi_i(v^i)$ for every $v^i \in V^i$, and
- (p^i, v^i) of weight $-(n - \varphi_i(v^i))$ for every $v^i \in V^i$.

We call the resulting graph consisting of all the gadgets H and the corresponding weights w. By choosing $d = k + \binom{k}{2}$, we get our Negative Directed Feedback Arc Set instance (H, w, d).

Negative Directed Feedback Arc Set to Multicolored Clique: Let S be an solution to (H, w, d). The gadgets H^i and $H^{i,j}$ each contain a negative cycle consisting of the arcs (v_r^i, v_{r+1}^i) or $(e_r^{i,j}, e_{r+1}^{i,j})$, respectively. As there are exactly $k + \binom{k}{2}$ such gadgets and their cycles are disjoint, S has size exactly d and contains exactly one arc $(v_{r_i}^i, v_{r_i+1}^i)$ for every $i \in \{1, \ldots, k\}$ and one arc $(e_{r_{i,j}}^{i,j}, e_{r_{i,j}+1}^{i,j})$ for every $1 \leq i < j \leq k$. Define $K = (\{v_r^i \mid i \in \{1, \ldots, k\}, r \in \{2, \ldots, |V^i| + 1\}, (v_{r-1}^i, v_r^i) \in S\}, \{e_r^{i,j} \mid 1 \leq i < j \leq k, r \in \{1, \ldots, |E^{i,j}|\}, (e_r^{i,j}, e_{r+1}^{i,j}) \in S\})$. We claim that K is a solution to (G, k) i.e. a multicolored clique in G. As K contains exactly one element of every V^i and $E^{i,j}$, we only have to check that the edges really connect to the vertices of K.

Assume for contradiction that this is not the case, i.e. there is a vertex $v^i \in V^i \cap V(K)$ and an edge $e^{i,j} \in E^{i,j} \cap E(K)$ for some $1 \leq i, j \leq k, i \neq j$, such that $e^{i,j} = \{u^i, w^j\}$ with $u^i \neq v^i$. For $e^{i,j}$ we have deleted only the outgoing arc in gadget $H^{i,j}$, so there is an $s^{i,j} \to e^{i,j}$-path P of length $|E^{i,j}|$ that uses one arc of weight 0 and $(|E^{i,j}| - 1)$ many arcs of weight $-n$. So $w(P) = -n(|E^{i,j}| - 1)$.

Likewise, we deleted only the incoming arc of v^i in the gadget H^i. Therefore, there is an $v^i \to t^i$-path Q of length $|V^i|$ that uses one arc of weight 0 and $(|V^i| - 1)$ many arcs of weight $-n$. Thus, $w(Q) = -n(|V^i| - 1)$. We link P and Q together with the arc $(t^i, s^{i,j})$ of weight $n(|V^i| + |E^{i,j}| - 1)$ to an overall $v_i \to e^{i,j}$-path $R = Q \circ (t^i, s^{i,j}) \circ P$ of weight $n(|V^i| + |E^{i,j}| - 1) - n(|E^{i,j}| - 1) - n(|V^i| - 1) = n$.

Let us now consider the unique $e^{i,j} \to v^i$-paths

$$W^+ = \{(e^{i,j}, p^i), (p^i, v^i)\} \text{ and } W^- = \{(e^{i,j}, n^i), (n^i, v^i)\}$$

in the consistency gadget C_i. These are using only the internal vertices p^i or n^i, respectively. Their weights are

$$\begin{aligned} w(W^+) &= -(n - \varphi_i(v^i)) - \varphi_i(e^{ij}) = (\varphi_i(v^i) - \varphi_i(u^i)) - n \qquad \text{and} \\ w(W^-) &= -(n - \varphi_i(e^{ij})) - \varphi_i(v^i) = (\varphi_i(u^i) - \varphi_i(v^i)) - n. \end{aligned}$$

If we join these two paths with R, we get two cycles with weights

$$\begin{aligned} w(W^+ \circ R) &= (\varphi_i(v^i) - \varphi_i(u^i)) - n + n = \varphi_i(v^i) - \varphi_i(u^i) \qquad \text{and} \\ w(W^- \circ R) &= (\varphi_i(u^i) - \varphi_i(v^i)) - n + n = \varphi_i(u^i) - \varphi_i(v^i). \end{aligned}$$

As $v^i \neq u^i$ and φ_i is bijective, we have that either $w(W^+ \circ R)$ or $w(W^- \circ R)$ is negative. Thus, $R \circ W^+$ or $R \circ W^-$ is a negative cycle, a contradiction to $H - S$ containing no negative cycles. Hence, K must be consistent and a clique in G.

Multicolored Clique to Negative Directed Feedback Arc Set: Now suppose that K is a multicolored clique on k vertices in G, i.e. a solution to (G, k). We choose S to be the set $\{(v^i_{r_i - 1}, v^i_{r_i}) \mid i \in \{1, \ldots, k\}, v^i_{r_i} \in V^i \cap V(K)\} \cup \{(e^{i,j}_{r_{i,j}}, e^{i,j}_{r_{i,j}+1}) \mid 1 \leq i < j \leq k, e^{i,j}_{r_{i,j}} \in E^{i,j} \cap E(K)\}$. Note that now every gadget H^i and $H^{i,j}$ is acyclic on its own. Furthermore, the whole graph H without our set S and the arcs $(t^i, s^{i,j})$ is acyclic.

Claim 1. *Every $s^{i,j} \to t^{i'}$-path in $H - S$ has weight at least $-n(|V^{i'}| + |E^{i,j}| - 1)$.*

Proof of Claim 1. Assume for contradiction that there is a $s^{i,j} \to t^{i'}$-path in $H - S$ of weight lower than $-n(|V^{i'}| + |E^{i,j}| - 1)$. Let P the shortest among them with respect to the number of arcs. Then P does not use any arc of type $(t^p, s^{p,q})$, as they have weight $n(|V^p| + |E^{p,q}| - 1)$ and therefore either $w(P[s^{i,j}, t^p]) < -n(|V^p| + |E^{i,j}| - 1)$ or $P[s^{p,q}, t^{i'}] < -n(|V^{i'}| + |E^{p,q}| - 1)$ and those have fewer arcs. But as P doesn't use any arc of type $(t^p, s^{p,q})$ it can only go from the gadget $H^{i,j}$ to $H^{i'}$ once with the help of $C^{i'}$ and does not use other gadgets. Moreover, we have that $i' \in \{i, j\}$. W.l.o.g. we can assume $i' = i$. Then P leaves $H^{i,j}$ exactly once and enters H^i exactly once. Let $e^{i,j}_a$ be the last vertex of $H^{i,j}$ and v^i_b the first vertex of H^i our path P visits. Note that $w(P[s^{i,j}, e^{i,j}_a])$ and $w(P[v^i_b, t^i])$ are always multiples of n and have weight at least $-n(|E^{i,j}| - 1)$ and $-n(|V^i| - 1)$, respectively. Also, this weight is only attained if and only if $a = r_{i,j}$ or $b = r_i$, respectively.

In any case, $P[e_a^{i,j}, v_b^i]$ consists of three vertices with n^i or p^i being the middle one. By choice of weights in C^i, the lowest weight $P[e_a^{i,j}, v_b^i]$ can achieve is $1 - |V^i| - n > -2n$. Also, if $a = r_{i,j}$ and $b = r_i$ the weight is $-n$. So in any case the composition $P = P[s^{i,j}, e_a^{i,j}] \circ P[e_a^{i,j}, v_b^i] \circ P[v_b^i, t^i]$ has weight at least $-n(|V^i| + |E^{i,j}| - 1)$ — a contradiction to the choice of P. ■

As every cycle in $H - S$ has to use an arc $(t^i, s^{i,j})$ of weight $n(|V^i| + |E^{i,j}| - 1)$ and every path completing this arc to a cycle has weight at least $-n(|V^{i'}| + |E^{i,j}| - 1)$, the graph $H - S$ contains no negative cycle.

For the size bound note that S contains exactly one arc from every H^i and $H^{i,j}$. Thus, it has size $k + \binom{k}{2} = d$. So S is a solution to (H, w, d).

Bounding the pathwidth of H: The pathwidth of H is defined as the pathwidth of the underlying undirected graph $\bar{H}$. We will show that $\mathrm{pw}(\bar{H}) \in \mathcal{O}(k^2)$.

Note that, if for some set $U \subseteq V(\bar{H})$ we get a path decomposition of $\bar{H} - U$ with bag size at most b, we get a path decomposition for $\bar{H}$ with bag sizes at most $b + |U|$ by adding U to every bag. Thus, $\mathrm{pw}(\bar{H}) \leq \mathrm{pw}(\bar{H} - U) + |U|$. Let

$$U = \bigcup_{i=1}^{k} \{t^i, n^i, p^i, v^i_{|V^i|}\} \cup \bigcup_{1 \leq i < j \leq k} \{s^{i,j}, e^{i,j}_{|E^{i,j}|}\}.$$

This set U has size $4k + k(k-1) = k^2 + 3k$. Observe that the graph $\bar{H} - U$ consist exactly of paths of the following form. Each vertex gadget H^i turned into a path consisting of the edges $\{v^i_\ell, v^i_{\ell+1}\}$ for $\ell \in \{1, \ldots |V^i| - 2\}$. Meanwhile, each edge gadget $H^{i,j}$ turned into a path consisting of the edges $\{e^{i,j}_\ell, v^{i,j}_{\ell+1}\}$ for $\ell \in \{1, \ldots |E^{i,j}| - 2\}$.

Furthermore, note that if we have a path decomposition for every connected component, then we can concatenate them to one path decomposition. Thus, the pathwidth of $\bar{H} - U$ is the maximum pathwidth of its connected components. But these are all paths and have pathwidth one. Thus, $\mathrm{pw}(\bar{H}) \leq \mathrm{pw}(\bar{H} - U) + |U| = k^2 + 3k + 1 \in \mathcal{O}(k^2)$.

Overall we have proven that our reduction produces an instance whose pathwidth and deletion size are bounded by some function in the parameter of the original Multicolored Clique instance. Moreover, by choice of φ, the only arcs of positive weight are the arcs of type $(t^i, s^{i,j})$ of which only $k(k-1)$ many exist. So we have a parameterized reduction from Multicolored Clique with parameter k to Negative Directed Feedback Arc Set with parameters pathwidth, deletion size and number of positive arcs.

To see that the weights have the claimed form, first note that all of them are integral. Moreover, the highest absolute weight have the arcs $(t^i, s^{i,j})$, which have weight $n(|V^i| + |E^{i,j}| - 1)$. As $|V(H)| \geq |V^i| + |E^{i,j}|$ and $|V(H)| \geq n$, we have that indeed $w : A(H) \to \mathbb{Z} \cap [-|V(H)|^2, |V(H)|^2]$, proving the theorem. □

From this theorem we are also able to infer W[1]-hardness for Negative Directed Feedback Arc Set parameterized in $\mathrm{pw}(G) + k$ on instances with weights restricted to $w : A(G) \to \{-1, 0, 1\}$.

Theorem 6.45. *Negative Directed Feedback Arc Set is* W[1]*-hard when parameterized in the pathwidth and deletion size, even on instances with weights of the form* $w : A(G) \to \{-1, 0, 1\}$.

Proof. We proof the theorem by showing how an instance described by Theorem 6.44 can be turned into one with its pathwidth increased by two and weights of the form $w : A(G) \to \{-1, 0, 1\}$ while keeping the deletion size the same. In the process, it loses the property of having only a bounded number of positive arcs. Let (G, w, k) be a Negative Directed Feedback Arc Set instance as described by Theorem 6.44. In particular, we have $w : A(G) \to \mathbb{Z} \cap [-|V(G)|^2, |V(G)|^2]$. From this instance we obtain an instance (G', w', k) by replacing every non-zero arc $a \in A(G)$ by a path P_a of $|w(a)|$-many arcs all having weight 1 if $w(a) > 0$ or -1 if $w(a) < 0$. We denote by (G', w') the resulting graph and weights. It remains to show that (G, w, k) and (G', w', k') are equivalent and $\mathrm{pw}(G') \leq \mathrm{pw}(G) + 2$.

To see the equivalence of the instances, note that any cycle in G' uses a newly created path P_a either as a whole or not at all. Thus deleting an arbitrary arc of a path P_a is equivalent to deleting the whole path and this is equivalent to deleting the arc a in G. For the pathwidth note that replacing arcs by paths can be incorporated into the path-decomposition as follows. Let B_i the bag that contains the arc a in the path decomposition of G that defines $\mathrm{pw}(G)$. Let $x_1, \ldots, x_{|w(a)|-1}$ the internal vertices of P_a. Then we modify the path decomposition of G by replacing B_i with the sequence $B_i, B_i \cup \{x_1, x_2\}, \ldots, B_i \cup \{x_{|w(a)|-2}, x_{|w(a)|-1}\}, B_i$. This covers all newly created arcs for P_a. By doing this replacement for all arcs a (while using only B_i of the original decomposition), this yields a new path-decomposition for G', where the maximum bag size increased by at most two. This shows that $\mathrm{pw}(G') \leq \mathrm{pw}(G) + 2$, concluding the proof of the theorem. □

6.6.5 W[1]-hardness for Pathwidth, Deletion Size and Few Negative Arcs

In this section we are going to prove W[1]-hardness for Negative Directed Feedback Arc Set parameterized in w_-, k and $\mathrm{pw}(G)$ for general arc weights. We will do this by doing an intermediate step showing W[1]-hardness for the Bounded Edge Directed (s,t)-Cut problem in DAGs when parameterized in k and $\mathrm{pw}(G)$.

Bounded Edge Directed (s,t)-Cut	
Instance:	A graph G, vertices $s, t \in V(G)$ and two integers $k, \ell \in \mathbb{Z}_{\geq 0}$.
Task:	Find a set $S \subseteq V(G)$ of size at most k such that every $s \to t$-path of $G - S$ has length more than ℓ or decide that no such set exists.

For the undirected Bounded Edge (s,t)-Cut Bentert et al. showed W[1]-hardness for the parameters maximum vertex degree and pathwidth [BHK19, Theorem 1]. By inspecting their reduction closely, one easily sees that their proof actually shows hardness for parameters maximum vertex degree, pathwidth and deletion size. The hardness for the directed case follows from this by replacing every edge by a forward and backward arc. For completeness (and bound on the pathwidth) we choose to do a direct reduction to the directed version here. It has the nice property that it produces acyclic directed graphs, a property we cannot achieve by replacing arcs with a *cycle* of length two. However, note that all important ideas are already present in the paper by Bentert et al. [BHK19].

Lemma 6.46. *The* Bounded Edge Directed (s,t)-Cut *problem parameterized in the pathwidth* $\mathrm{pw}(G)$*, deletion size* d *and maximum degree of the underlying undirected graph* $\Delta(G)$ *is* $\mathsf{W}[1]$*-hard, even when the graph is restricted to be acyclic.*

Proof. We start our reduction from the Clique problem. The input of Clique consists of an undirected graph G and an integer k. The question is, whether there is a vertex set $X \subseteq V(G)$ of at least k vertices such that $G[X]$ forms a complete graph. Clique is $W[1]$-hard parameterized by k.

Let now (G,k) be a Clique instance. We want to construct an equivalent instance (H,d,Λ) of Bounded Edge Directed (s,t)-Cut whose parameters $\mathrm{pw}(H)$, d and $\Delta(G)$ are bounded by a function of k only. For notation, we fix some arbitrary ordering $v_1,\ldots,v_n$ of the vertices $V(G)$. For the edges $e \in E(G)$ we define their canonical representation $\{v_i,v_j\}$ to be the order of vertices with $i \leq j$. Setting this canonical representation then defines an ordering $e_1 \prec \ldots \prec e_m$ of the edges $E(G)$ by $e = \{v_i,v_j\} \preceq e' = \{v_{i'},v_{j'}\}$ iff $(i,j) \preceq_{\mathsf{lex}} (i',j')$, where $\preceq_{\mathsf{lex}}$ is the lexicographical ordering on vectors of length two.

As usual for reduction from cliques we construct two kinds of gadgets, vertex gadgets H^i and edge gadgets $H^{i,j}$. The vertex gadgets each choose a vertex in the graph and the edge gadgets check whether there is indeed an edge between them. For all these constructions we will use paths of different lengths. For ease of notation we will replace these paths by arcs with positive integral weights. We make sure that these weights are bounded by some function of $n+m$ and that parallel arcs have weight at least two. This way replacing an arc of weight w by a directed path of length w gives us a still polynomial sized simple graph for Bounded Edge Directed (s,t)-Cut. Also, deletion of an arc of a path whose internal vertices have in-degree and out-degree one and are disjoint of s and t, means that no arc on this path is part of an $s \to t$-path anymore. So a solution does not delete more paths by choosing several arcs on the same replaced path. This preserves equivalence, between the weighted and the replaced graph.

An important role for the weights plays an integer M which we will choose later. Our choice of Λ will depend on M, so we also defer its choice until later. For know think of M as a big integer.

Gadgets: Before introducing the different vertex and edge gadgets, we discuss some gadget that will be the foundation for both. For any positive integers $a,b \in \mathbb{Z}_{>0}$ and any set $\{\varphi^q : \{1,\ldots,b\} \to \mathbb{Z} \cap [-M,M] \mid q \in \{1,\ldots,a\}\}$ of functions, we introduce the gadget $R^{a,b}$. $R^{a,b}$ consists of a many disjoint paths $P^1,\ldots,P^a$ of length $2b+2$. For each path P^q with $q \in \{1,\ldots,a\}$ all arcs of P^q have weight 1, and we denote the vertices of P^q in order of there appearance by $p^q_{\mathsf{in}}, p^q_1, \hat{p}^q_1, p^q_2, \hat{p}^q_2 \ldots, p^q_{b+1}, p^q_{\mathsf{out}}$. We add the following "detours" along the paths P^q. For every $r \in \{1,\ldots,b\}$ there is an arc (p^q_r, p^q_{r+1}) of weight $5M+\varphi^q(r)$. Moreover, we interlink the paths P^q and P^{q+1} for every $q \in \{1,\ldots,a-1\}$ by the following arcs. For every $r \in \{1,\ldots,b\}$ there are arcs $(\hat{p}^q_r, p^{q+1}_{r+1})$ and $(\hat{p}^{q+1}_r, p^q_{r+1})$ of weight $3M$ each. Finally, the gadget $R^{a,b}$ contains the vertices s and t. To connect those vertices to the rest of the graph, for every $q \in \{1,\ldots,a\}$, there are two parallel arcs (s,p^q_1) of weight 2 and two parallel arcs (p^q_{b+1},t) of weight $\Lambda - 4M$. This concludes the construction of our gadget. Note that for $M \geq 1$ the graph has positive arc weights and for $\Lambda \geq 4M+2$ all parallel arcs are of weight at least 2.

We now prove the crucial properties of this gadget:

Claim 2. *Let $M \geq 2b+2$ and $\Lambda \geq 4M+2$ be integers. For every set $S \subseteq A(R^{a,b})$ such that $R^{a,b} - S$ contains no $s \to t$-path of weight $\leq \Lambda$, we have that $|S| \geq a$.*

Moreover, for every such set $S \subseteq A(R^{a,b})$ of size $|S| = a$, there exists a unique $x \in \{1,\dots,b\}$ such that for every $q \in \{1,\dots,a\}$ any minimum-weight $s \to p^q_{\mathsf{out}}$-path has weight $5M + 2b + 1 + \varphi^q(x)$ and any minimum-weight $p^q_{\mathsf{in}} \to t$-path has weight $\Lambda + M + 2b - 1 + \varphi^q(x)$.

Also, for every $x \in \{1,\dots,b\}$, we can choose a set $S \subseteq A(R^{a,b})$ of size a such that $R^{a,b} - S$ contains no $s \to t$-path of weight $\leq \Lambda$, any minimum-weight $s \to p^q_{\mathsf{out}}$-path has weight $5M + 2b + 1 + \varphi^q(x)$ and any minimum-weight $p^q_{\mathsf{in}} \to t$-path has weight $\Lambda + M + 2b - 1 + \varphi^q(x)$.

Proof of Claim 2. Consider for every $q \in \{1,\dots,a\}$ the path $s, p^q_1, \hat{p}^q_1, \dots, p^q_{b+1}, t$. This path has weight $2 + 2b + \Lambda - 4M \leq \Lambda$. Thus, any set S as in the theorem statement has to contain an arc of all of these $s \to t$-paths. As these paths are arc-disjoint, we have that all sets have size at least a.

Now we turn to the sets S of size exactly a. Consider the paths from above again. As there are two parallel arcs for each of (s, p^q_1) and (p^q_{b+1}, t), our set S has to intersect each P^q with exactly one edge that is not $(p^q_{\mathsf{in}}, p^q_1)$ or $(p^q_{b+1}, p^q_{\mathsf{out}})$.

We claim that there is an $x \in \{1,\dots,b\}$ such that $S = \{(p^q_x, \hat{p}^q_x) \mid q \in \{1,\dots,a\}\}$. Suppose that not, then for some pair (p,q) with $q \in \{1,\dots,a-1\}$ and $y \in \{1,\dots,n\}$ we have that either $P^q[p^q_1, \hat{p}^q_y]$ and $P^{q+1}[p^{q+1}_{y+1}, p^{q+1}_{b+1}]$ are disjoint from S or $P^{q+1}[p^{q+1}_1, \hat{p}^{q+1}_y]$ and $P^q[p^q_{y+1}, p^q_{b+1}]$ are disjoint from S. In the former case, the path

$$(s, p^q_1) \circ P^q[p^q_1, \hat{p}^q_y] \circ (\hat{p}^q_y, p^{q+1}_{y+1}) \circ P^{q+1}[p^{q+1}_{y+1}, p^{q+1}_{b+1}] \circ (p^{q+1}_{b+1}, t)$$

is an $s \to t$-path in $R^{a,b} - S$. In the latter case, the path

$$(s, p^{q+1}_1) \circ P^{q+1}[p^{q+1}_1, \hat{p}^{q+1}_y] \circ (\hat{p}^{q+1}_y, p^q_{y+1}) \circ P^q[p^q_{y+1}, p^q_{b+1}] \circ (p^q_{b+1}, t)$$

is an $s \to t$-path in $R^{a,b} - S$. So one of these $s \to t$-paths exists in $R^{a,b} - S$. Note that both paths have weight

$$2 + (2(y-1)+1) + 3M + 2(b-y) + (\Lambda - 4M) = 2b + 1 + 3M + \Lambda - 4M \leq \Lambda,$$

which is a contradiction to the choice of S.

So, if there is a set S such that $R^{a,b} - S$ contains no $s \to t$-path of weight $\leq \Lambda$ that has size a, it has the form $\{(p^q_x, \hat{p}^q_x) \mid q \in \{1,\dots,a\}\}$. Indeed, for such a set S, we have that any $s \to t$-path in $R^{a,b} - S$ has to use one of the detour arcs (p^q_x, p^q_{x+1}) of weight $5M + \varphi^q(x) \geq 4M$. Moreover, any $s \to t$-path has to use an arc incident to s and t of weight 2 and $\Lambda - 4M$ respectively. So for any $x \in \{1,\dots,b\}$, each $s \to t$-path in $R^{a,b} - \{(p^q_x, \hat{p}^q_x) \mid q \in \{1,\dots,a\}\}$ has overall weight at least $2 + 4M + \Lambda - 4M > \Lambda$.

It only remains to prove, that for these solutions $S = \{(p^q_x, \hat{p}^q_x) \mid q \in \{1,\dots,a\}\}$, we have that in $R^{a,b} - S$ any minimum-weight $s \to p^q_{\mathsf{out}}$-path has weight $5M + 2b + 1 + \varphi^q(x)$ and any minimum-weight $p^q_{\mathsf{in}} \to t$-path has weight $\Lambda + M + 2b - 1 + \varphi^q(x)$.

Any $s \to p^q_{\mathsf{out}}$-path uses exactly one incident edge of s (which has weight 2) and p^q_{out} (which has weight 1). So, for these paths it is enough to prove that any $p^h_1 \to p^q_{b+1}$-path in $R^{a,b} - S$ for $h \in \{1,\dots,a\}$ has weight at least $5M + 2(b-1) + \varphi^q(x)$ and one path achieves that weight.

Moreover, any $p^q_{\mathsf{in}} \to t$-path uses exactly one incident edge of p^q_{in} (which has weight 1) and t (which has weight $\Lambda - 4M$). Thus, for these paths it is enough to prove that any $p^q_1 \to p^g_{b+1}$-path in $R^{a,b} - S$ for $g \in \{1, \dots, a\}$ has weight at least $5M + 2(b-1) + \varphi^q(x)$ and one path achieves that weight. Together it suffices to show that any $p^h_1 \to p^g_{b+1}$-path in $R^{a,b} - S$ has weight at least $5M + 2(b-1) + \varphi^q(x)$ and there is a $p^q_1 \to p^q_{b+1}$-path that achieves this weight.

Notice that any $p^h_1 \to p^g_{b+1}$-path contains a detour arc (p^r_x, p^r_{x+1}) of weight $5M + \varphi^h(x) \geq 4M$. If a $p^h_1 \to p^g_{b+1}$-path contains another detour arc or an interconnecting arc, it occurs an additional weight of at least $3M$. So any $p^h_1 \to p^g_{b+1}$-path that has uses arcs other than those of a single P^q and the arc (p^q_x, p^q_{x+1}) has weight at least $8M$. But $5M + 2b + 1 + \varphi^q(x) \leq 7M < 8M$. So it is enough to show that there is a $p^q_1 \to p^q_{b+1}$-path of weight $5M + 2(b-1) + \varphi^q(x)$.

Notice that $P^q[p^q_1, p^q_x] \circ (p^q_x, p^q_{x+1}) \circ P^q[p^q_{x+1}, p^q_{b+1}]$ has weight $2(x-1) + 5M + \varphi^q(x) + 2(b-x) = 5M + 2(b-1) + \varphi^q(x)$ and thus is exactly such a path. ■

We are now ready to define the vertex gadgets and edge gadgets. All vertex gadgets will be identical and all edge gadgets will be identical, just the interconnection between them differs.

For all $i \in \{1, \dots, k\}$, we introduce a vertex gadget H^i by setting $a = 2$, $b = n$, $\varphi^1(x) = x$ and $\varphi^2(x) = -x$. Note that the image of φ^1 and φ^2 lies in $\mathbb{Z} \cap [-M, M]$ for $M \geq n$. Thus, the graph $R^{2,n}$ is well-defined. To obtain our gadget H^i from this we forget about the vertices p^1_{in} and p^2_{in}. Furthermore, we rename the vertices p^1_{out} and p^2_{out} to v^i_+ and v^i_-.

For all $i, j \in \{1, \dots, k\}$ with $i < j$, we introduce an edge gadget $H^{i,j}$. For this we set $a = 4$, $b = m$ and for every $x \in \{1, \dots, m\}$ and edge e_x with canonical representation (v_g, v_h), we set $\varphi^1(x) = -g$, $\varphi^2(x) = g$, $\varphi^3(x) = -h$ and $\varphi^4(x) = h$. Again the image of $\varphi^1, \dots, \varphi^4$ lies in $\mathbb{Z} \cap [-M, M]$ for $M \geq m$. Thus, the graph $R^{4,m}$ is well-defined. By forgetting the vertices p^q_{out} and renaming p^1_{in}, p^2_{in}, p^3_{in} and p^4_{in} to v^i_+, v^i_-, v^j_+ and v^j_-.

From these gadgets we obtain our complete graph H by taking one copy of each gadget and identifying all vertices of the same name (s, t, v^i_+ and v^i_-). Finally, we set $d = 2n + 2n(n-1)$, $M = 2(n+m) + 2$ and $\Lambda = 10M + 2(n-1) + 2(m-1) - 1$. This completes our construction.

Correctness: We now want to prove that (G, k) as instance of Clique is a "yes"-instance if and only if (H, d, Λ) as instance of Bounded Edge Directed (s,t)-Cut is a "yes"-instance.

For the forward direction let $U \subseteq V(G)$ be a clique in G with corresponding edge set F. Let $u_1, \dots, u_k$ be the indices of the vertices of $v_1, \dots, v_n$ corresponding to the vertices of U in increasing order. Also, let $f_{i,j}$ be the index of the edge $\{v_{u_i}, v_{u_j}\}$ in the fixed ordering of $E(G)$. By Claim 2, there is a solution S^i of size two to each of the H^i's such that each $s \to v^i_+$-path has weight at least $5M + 2(n-1) + u_i$ and each $s \to v^i_-$-path has weight at least $5M + 2(n-1) - u_i$. Moreover, again by Claim 2, there is a solution $S^{i,j}$ of size four to each of the $H^{i,j}$'s such that each $v^i_+ \to t$-path has weight at least $5M + 2(m-1) - u_i$, each $v^i_- \to t$-path has weight at least $5M + 2(m-1) + u_i$, each $v^j_+ \to t$-path has weight at least $5M + 2(m-1) - u_j$ and each $v^j_- \to t$-path has weight at least $5M + 2(m-1) + u_j$. We define our solution S as $\bigcup_{i=1}^k S^i \cup \bigcup_{1 \leq i < j \leq k} S^{i,j}$.

By definition our S is a solution when restricted to a single vertex gadget or a single edge gadget. The only $s \to t$-paths that are not contained in a single gadget consist of an $s \to v^i_+$-path in some vertex gadget and a $v^i_+ \to t$-path in some edge gadget or of an $s \to v^i_-$-path in some vertex gadget and a $v^i_- \to t$-path in some edge gadget. In each case, as seen above, these paths have length at least $5M+2(n-1)+u_i+5M+2(m-1)-u_i = \Lambda+1$. It only remains to check the size of S which is exactly $2k + 4\frac{k(k-1)}{2} = d$.

For the backwards direction, let S be a solution to (H, d, Λ). From Claim 2, we get that any solution to a vertex gadget has size at least two and any solution to an edge gadget has size at least four. As a solution must be a solution to every single gadget and our gadgets are arc disjoint, we have that any solution must have size at least $2k+4\frac{k(k-1)}{2} = d$. From this it follows that our solution must have size exactly two in every vertex gadget and size exactly four in every edge gadget. Claim 2 states that the structure of these solutions in the vertex gadgets must be such that in $H^i - S$ any minimum weight $s \to v^i_+$-path in $V^i - S$ has weight $5M+2(n-1)+u_i$ and any minimum weight $s \to v^i_-$-path in V^i has weight $5M + 2(n-1) - u_i$ for some $u_i \in \{1, \ldots, n\}$. We define the vertex set $U \subseteq V(G)$ to be the set $\{v_{u_i} \mid i \in \{1, \ldots, k\}\}$.

We can apply Claim 2 also to the structure of S with respect to $H^{i,j} - S$. This tells us that for every $1 \leq i < j \leq k$ there is an index $f_{i,j} \in \{1, \ldots, m\}$ such that for the canonical representation $\{v_g, v_h\}$ of $e_{f_{i,j}}$, in $H^{i,j} - S$ we have that any minimum weight $v^i_+ \to t$-path has weight $5M+2(m-1)-g$, any minimum weight $v^i_- \to t$-path has weight $5M + 2(m-1) + g$, any minimum weight $v^j_+ \to t$-path has weight $5M + 2(m-1) - h$ and any minimum weight $v^j_- \to t$-path has weight $5M+2(m-1)+h$. If we concatenate a minimum weight $s \to v^i_+$-path in $V^i - S$ with a minimum weight $v^i_+ \to t$-path in $H^{i,j} - S$, we get an $s \to t$-path of weight $10M + 2(n-1) + 2(m-1) + u_i - g$. As S is a solution to (H, d, Λ), this path has length more than Λ, which is equivalent to $u_i + 1 > g$ or $u_i \geq g$. By applying the same argument to minimum weight paths that concatenate at v^i_- instead of v^i_+, we get the inequality $u_i \leq g$. Together this yields $u_i = g$. If we swap H^i for H^j and v^i_+ and v^i_- for v^j_+ and v^j_-, we get $u_j = h$ in the same way. So the edge $\{v_{u_i}, v_{u_j}\} = \{v_g, v_h\} = e_{f_{i,j}}$ exists in G. Since this is true for all $1 \leq i < j \leq k$, U is indeed the vertex set of a clique.

Bound on Parameters and Acyclicity: For pathwidth consider the vertex set $U = \{s, t\} \cup \bigcup_{i=1}^{k} \{v^i_+, v^i_-\}$. This is a set of $2(k+1)$ vertices and will be contained in every bag of our path decomposition. Note that all vertices of $H - U$ have no neighbors outside the gadget they are contained in. So it suffices to find a path decomposition for every gadget (minus vertices in U), concatenate them and add U to every set. The pathwidth of this decomposition is then bounded by $|U|$ plus the maximum pathwidth of any gadget path decomposition. We do this path decomposition on the meta-gadget $R^{a,b}$. For every $r \in \{1, \ldots, b\}$ let $X_{2r-1} = \{p^q_r, \hat{p}^q_r \mid q \in \{1, \ldots, a\}\}$ and for every $r \in \{1, \ldots, b-1\}$ let $X_{2r} = \{\hat{p}^q_r, p^q_{r+1} \mid q \in \{1, \ldots, a\}\}$. Then every arc of $R^{a,b} - S$ is contained in one of the bags X_i and each vertex in at most two consecutive bags. Thus, $X_1, \ldots, X_{2b-1}$ is a path decomposition. The maximum size of a bag is $2a$. For our specific gadgets a is at most four. Thus, we get that the pathwidth of H is at most $8 + |U| = 2k + 10$. The maximum vertex degree inside a gadget is 6 for the p^q_r vertices. However, the vertices v^i_+, v^i_-, s and t have neighbors in several gadgets. v^i_+, v^i_- have degree $k + \frac{k(k-1)}{2}$, s has degree $4k$ and t has degree $16\frac{k(k-1)}{2}$. In any case the maximum is bounded by a function of k. Last but not least the deletion size is $d = 2k + 4\frac{k(k-1)}{2}$.

For the acyclicity note that the gadgets $R^{a,b}$ are acyclic (as all arcs only run to increasing positions along the paths P^q). So it only remains to prove that the boundary vertices s, t, v^i_+ and v^i_- are connected in a way that preserves acyclicity. s has only outgoing arcs and t has only incoming arcs. Moreover, all v^i_+ and v^i_- the ingoing arcs come exactly from the vertex gadgets and the outgoing arcs come exactly form the edge gadgets and these gadgets are not connected by any other vertices. □

Now we use Lemma 6.46 to show hardness for Negative Directed Feedback Arc Set by weighting all arcs of an Bounded Edge Directed (s,t)-Cut instance with weight one and introducing $(k+1)$-many (t,s) arcs of weight $-(\ell+1)$.

Theorem 6.47. *Negative Directed Feedback Arc Set is* W[1]*-hard when parameterized in the pathwidth, deletion size and number of negative arcs.*

Proof. We show the theorem by reduction from Bounded Edge Directed (s,t)-Cut on DAGs parameterized by pathwidth and deletion size, which is W[1]-hard by Lemma 6.46. Let (G,s,t,k,ℓ) be an Bounded Edge Directed (s,t)-Cut instance with G being a DAG. We construct a equivalent instance (H,w,k) of Negative Directed Feedback Arc Set as follows. To construct H, give all arcs in G weight 1 and introduce $k+1$ arcs (t,s) of weight $-(\ell+1)$. This completes the construction of (H,w,k). We now prove the equivalence.

Bounded Edge Directed (s,t)-Cut to Negative Directed Feedback Arc Set: Let S be a solution to (G,s,t,k,ℓ). We claim that S is also a solution to (H,w,k). The set S trivially fulfills the size bound of k. Assume now for contradiction that there is a negative cycle C in $H-S$. As the only negative arcs of H are those from t to s, C contains exactly one of these. So $C[s,t]$ is an $s \to t$-path in $G-S$ and has weight less than $-w((t,s)) = \ell+1$. Now, all weights in G are 1 and thus $C[s,t]$ is an $s \to t$-path in $G-S$ of length at most ℓ, a contradiction to S being a solution to (G,s,t,k,ℓ). Thus, S is a solution to (H,w,k).

Negative Directed Feedback Arc Set to Bounded Edge Directed (s,t)-Cut: Let S be an inclusion-wise minimal solution to (H,w,k). As there are $(k+1)$-many parallel arcs (t,s), one of them does not lie in S. Any cycle in H can use at most one of these arcs and thus as S is inclusion-wise minimal, S contains non of them, i.e. it contains only arcs of G. We claim that S is a solution to (G,s,t,k,ℓ). It trivially fulfills the size bound. Now assume for contradiction that $G-S$ contains an $s \to t$-path P of length at most ℓ. Then $P \circ (t,s)$ is a cycle of weight $w(P) - (\ell+1) < 0$ in $H-S$, which is a contradiction to S being a solution to (H,w,k).

Bounding the parameters: The deletion size of our new instance stays the same and thus is bounded by the deletion size of the old instance. The same holds for the number of negative arcs, as we only introduced $k+1$ of them. Finally, note that we can turn any pathwidth decomposition of G into one of H by adding $\{s,t\}$ to all bags. This shows $\mathrm{pw}(H) \leq \mathrm{pw}(G)+2$. □

6.7 Reductions between Arc and Vertex Deletion Version

In this section we will see parameterized reductions between Negative Directed Feedback Arc Set and Negative Directed Feedback Vertex Set. For a formal definition of those two problems refer to Section 6.1.

Theorem 6.48. *Given an instance (G, w, k) of* Negative Directed Feedback Arc Set *one can in* $\mathrm{poly}(n)$ *time construct an equivalent instance (G', w', k') of* Negative Directed Feedback Vertex Set*, such that*

$$k' = k \quad \mathrm{td}(G') \leq (k+1)\,\mathrm{td}(G)+2 \;\Big|\; w'_+ = w_+ \quad \mathrm{pw}(G') \leq (k+1)\,\mathrm{pw}(G)+2 \;\Big|\; w'_- = w_- \quad \mathrm{tw}(G') \leq (k+1)\,\mathrm{tw}(G)+2$$

and $w'(A(G')) = w(A(G)) \cup \{0\}$.

Proof. We construct G' as follows. The vertex set $V(G')$ consist of $k+1$ many vertices $x_v^1, \ldots, x_v^{k+1}$ for every $v \in V(G)$ as well as two vertices x_a^1 and x_a^2 for every $a \in A(G)$. The arc set $A(G')$ then has for every arc $a = (u, v) \in A(G)$ and every $i \in \{1, \ldots, k+1\}$ two arcs (x_u^i, x_a^1) and (x_a^2, x_v^i) of weight 0 and one arc (x_a^1, x_a^2) of weight $w(a)$. By setting $k' = k$, this completes the construction of our Negative Directed Feedback Vertex Set instance.

We now show the equivalence of the instances. Let $S \subseteq A(G)$ be a solution to (G, w, k). We claim that $S' = \{x_a^1 \mid a \in S\}$ is a solution to (G', w', k'). The size bound holds trivially. Assume for contradiction that $G' - S'$ contains a cycle C' of negative weight. As the x_v^i only have neighbors of type x_a^1 and x_a^2 and those have to be traversed in this order, every cycle in G' visits subsequent path segments of the form $x_u^i, x_a^1, x_a^2, x_v^i$ for some $a = (u, v) \in A(G)$. Let C be the ordered (multi-)subset of vertices v that appear in C' as some x_v^i and order them by their appearances in C'. By choice of S', C is a closed walk in $G - S$. Moreover, we have by choice of arc weights, that C has negative weight. Therefore, it contains a negative cycle in $G - S$, a contradiction to S being a solution.

For the backward direction of the equivalence, we consider a solution $S' \subseteq V(G')$ of (G', w', k'). Let $S = \{a \in A(G) \mid \{x_a^1, x_a^2\} \cap S' \neq \emptyset\}$. We claim that S is a solution to (G, w, k). The size bound holds trivially. Assume for contradiction that $G - S$ contains a cycle C of negative weight. Let $i \in \{1, \ldots, k+1\}$ be an index such that $\{x_v^i \mid v \in V(G)\} \cap S' = \emptyset$. As $|S'| \leq k$, such an index exists. Then, by choice of S, the cycle C induces a cycle C' in $G' - S'$ by replacing the sequence u, v by $x_u^i, x_{(u,v)}^1, x_{(u,v)}^2, x_v^i$. Moreover, $w'(C') = w(C) < 0$, a contradiction to S' being a solution.

It remains to check the claimed bound on the (potential) parameters. We have $k' = k$ by construction and $w'_+ = w_+$, $w'_- = w_-$ as every arc $a \in A(G)$ corresponds to the arc $(x_a^1, x_a^2) \in A(G')$ of the same weight and all other arcs have weight 0. This also shows $w'(A(G')) = w(A(G)) \cup \{0\}$. Now we show the remaining bounds on treedepth, pathwidth and treedepth. We do this by showing how a decomposition corresponding to the parameter of the original graph G can be adapted to form a decomposition for G'. In particular, we show how the different parts of the construction of G' can be reflected in the decompositions. First we want to split vertices into $k+1$ vertices (the x_v^i's). For treedepth, we replace the appearance of v in the rooted tree by a path of length $k+1$ containing the x_v^i's. This leads to a factor $k+1$ increase in the threedepth. For pathwidth and treewidth, we replace v in the bags by the x_v^i's also leading to a factor $k+1$ increase. Afterwards, we want to subdivide a set of arcs twice (creating the x_a^i's). For treedepth, we can add the new vertices as child of the leaf of a root-leaf-path where the subdivided arc appears on. For pathwidth/treewidth, we create a new bag for each arc originating from the bag that contained it and add the subdivision vertex. By this method, subdividing a set of arcs increases these parameters by 1. Thus creating the x_a^i's increases these parameters by at most 2. Putting both modifications together leads to the claimed results. □

Theorem 6.49. *Given a* Negative Directed Feedback Vertex Set *instance* (G, w, k)*, one can in* $\mathrm{poly}(n)$ *time construct an equivalent instance* (G', w', k') *of* Negative Directed Feedback Arc Set*, such that*

$$\begin{array}{c|c|c} k' = k & w'_+ = w_+ & w'_- = w_- \\ \mathrm{td}(G') \leq 2\,\mathrm{td}(G) & \mathrm{pw}(G') \leq 2\,\mathrm{pw}(G) & \mathrm{tw}(G') \leq 2\,\mathrm{tw}(G) \end{array}$$

and $w'(A(G')) = w(A(G)) \cup \{0\}$.

Proof. To construct G', split every vertex $v \in V(G)$ into v^- inheriting the incoming arcs of v and v^+ inheriting the outgoing arcs of v. Then add an arc (v^-, v^+) of weight 0. Setting $k' = k$ completes the construction of our Negative Directed Feedback Arc Set instance.

We now prove the equivalence of the instances. Let $S \subseteq V(G)$ be a solution to (G, w, k). We claim that $S' = \{(v^-, v^+) \mid v \in S\}$ is a solution to (G', w', k'). Assume for contradiction that $G' - S'$ contains a cycle C' of negative weight. As G' is bipartite with the v^+'s and the v^-'s forming the bipartition we have that in C' the vertices appear alternating as u^+ and v^-. Moreover, as the arc (u^-, v^+) only exists for $u = v \notin S$, we have that C' has pairs of the same vertex appearing after another and contracting them to the original vertex results in a cycle C in $G - S$. However, this contraction only eliminates arc of weight 0, and thus C is a cycle of negative weight in $G - S$. This is a contradiction to S being a solution.

For the backward direction of the equivalence, we consider a solution $S' \subseteq A(G')$ of (G', w', k'). Choose $S = \{v \in V(G) \mid (v^-, v^+) \in S' \vee (u^+, v^-) \in S'\}$. We claim that S is a solution to (G, w, k). Again, the size bound holds trivially. Assume for contradiction that $G - S$ contains a cycle C of negative weight. Then neither the arcs (v^-, v^+) with $v \in V(C)$ are in S' nor the arcs (u^+, v^-) with $(u, v) \in A(C)$ are in S'. Thus applying the construction of G' to C yields a cycle C' in $G' - S'$. As the only newly introduced arcs have weight 0, this cycle also has negative weight. This is a contradiction to S' being a solution.

It remains to check the claimed bound on the (potential) parameters. We have $k' = k$ by construction and $w'_+ = w_+$, $w'_- = w_-$ as the only newly introduced arcs have weight 0. For treedepth, pathwidth and treewidth note that treating v^+ and v^- as single vertex for construction of the corresponding decomposition and replacing it either by a path (treedepth) or adding both to the bag increases the decomposition size by a factor of 2. □

Chapter 7

Conclusion

In this thesis we have seen algorithms and hardness results for various $\mathcal{C}$-Vertex Deletion problems. We now summarize these results and discuss possible directions for future research.

In Chapter 3 we considered the Directed Long Cycle Hitting Set problem, i.e. the problem of removing all directed cycles above a certain length ℓ from the graph by deleting a given number k of vertices. We obtained a fixed-parameter algorithm for the combined parameters $k+\ell$. A NP-hardness reduction shows that none of these can be omitted.

Here, we introduced the new notion of important ranged deletion separators (c.f. Section 3.1.4). These generalize the concept of important separators to further consider the structure of an hereditary $\mathcal{C}$-Vertex Deletion problem. As this tool of important ranged deletion separators was formulated independently of the Directed Long Cycle Hitting Set problem, we hope that it can be applied to other hereditary $\mathcal{C}$-Vertex Deletion problems.

Furthermore, we obtained a result about a small representative set of paths in graphs of bounded circumference (Theorem 3.4) that may be of interest for other fixed-parameter algorithms on graphs of bounded circumference. Or it may be used in combination with our main result of this section to first delete a certain number of vertices to reach bounded circumference (if the studied problem permits) and afterwards use the representative set of paths result on the remaining graph. Finally, this result might be a foundation for further research on hereditary $\mathcal{C}$-Vertex Deletion problems in the following way: Neogi, Ramanujan, Saurabh, and Sharma have shown that $\mathcal{C}$-Vertex Deletion is fixed-parameter tractable in the deletion size for certain families $\mathcal{C}$ that are defined by a set $\mathcal{H}$ of forbidden subgraphs [NRSS20]. In particular, they achieve a fixed-parameter algorithm for the cases where $\mathcal{H}$ either contains only arborescences (rooted directed graphs) or contains at least one directed path. If the set of forbidden subgraphs is the set $\mathcal{H}_\ell$ consisting of all cycles of length at least $\ell+1$, then we obtain exactly the Directed Long Cycle Hitting Set problem. By further research one might wish to obtain a full characterization of forbidden families $\mathcal{H}$ for which $\mathcal{C}$-Vertex Deletion is fixed-parameter tractable on directed graphs, akin to the results accomplished by Robertson and Seymour in their graph minor series for undirected graphs and forbidden minors [RS95, RS04].

We studied in Chapter 4 the Bounded Size Strongly Connected Component Vertex Deletion problem, i.e. the problem of deleting a given number k of

vertices such that every strongly connected component of the remaining graph has size at most s. We gave a fixed-parameter algorithm for combined parameter $k+s$ with run-time $2^{\mathcal{O}((k+s)(\log k+\log s))}\operatorname{poly}(n)$. Bang-Jensen, Eiben, Gutin, Wahlström, and Yeo built upon our result by replacing our subroutine for guessing the bounded size components by a randomized algorithm which – using standard techniques – can be derandomized to obtain an improved run-time of $2^{\mathcal{O}(k(\log k+\log s))}\operatorname{poly}(n)$ [BJEG+20]. The same run-time was also obtained by Neogi, Ramanujan, Saurabh, and Sharma with their algorithms for $\mathcal{H}$-free strongly connected components, where in this case $\mathcal{H}$ contains all arborescences on $s+1$ vertices [NRSS20]. This run-time dependence on k is unlikely to be improved without improving the algorithm for Directed Feedback Vertex Set first, as this is the special case of $s=1$. Whether an algorithm with run-time $2^{o(k\log k)}\operatorname{poly}(n)$ exists, is still a major open question. Improving the run-time dependence on s is unlikely as well because Drange, Dregi, and van 't Hof have shown that even for the undirected version there is no algorithm with run-time $2^{o(k\log s)}\operatorname{poly}(n)$ unless the exponential time hypothesis fails [DDv16].

In Chapter 5, we proved that 1-out-regular Vertex Deletion, i.e. the problem of deleting a given number k of vertices such that every strongly connected component C is r_C-out-regular for an $r_C \leq 1$, is fixed-parameter tractable when parameterized in k. In particular, we gave an algorithm with run-time $2^{\mathcal{O}(k^3)}\operatorname{poly}(n)$. This run-time was later improved by Neogi, Ramanujan, Saurabh, and Sharma to $2^{\mathcal{O}(k\log k)}\operatorname{poly}(n)$, matching the best known run-time for Directed Feedback Vertex Set [NRSS20].

It would be of interest to see generalizations of this results to higher degrees of regularity. But as already extending the maximum regularity from 1 to 2 breaks the hereditary structure of $\mathcal{C}$, this requires new techniques. Also results on Eulerian Strongly Connected Component Vertex Deletion, which gave the original motivation for considering this problem, would be interesting. But again, this goes beyond the space of hereditary graph classes $\mathcal{C}$.

Finally, we considered in Chapter 6 the Negative Directed Feedback Arc Set problem, i.e. the problem of deleting a given number k of arcs from a weighted graph such that no cycle negative weight remains. We considered the parameters deletion size (k), number of positive (w_+) and negative (w_-) arcs as well as treewidth (tw), pathwidth (pw) and treedepth (td) of the underlying undirected graph. For these parameters we gave an almost complete FPT/W[1]-hard/NP-hard dichotomy which can be found in Table 6.1 (see Section 6.4). The main open question is whether the two gaps in this dichotomy can be filled. For further research directions in this area, the original motivation of removing a minimum number of inequalities from an infeasible linear program such that it becomes feasible should be considered. The Negative Directed Feedback Arc Set only addresses the scenario where all constraints are difference constraints, i.e. inequalities of the form $x_i - x_j \leq b_{i,j}$. Here two natural questions arise. First, can we extend the algorithmic findings to inequalities with two arbitrary non-zero coefficients? When formulating this as a problem in graphs, the feasibility checking becomes the so called Generalized Max Flow problem, which is an active research area. See e.g. [OV20] for a strongly polynomial algorithm. Second, it would be interesting to extend the result to inequalities with three variables, as any linear program can be written in this form. However this seems challenging, as it requires going from directed graphs to directed hypergraphs.

Bibliography

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.

[BBF99] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[BJEG+20] Jørgen Bang-Jensen, Eduard Eiben, Gregory Gutin, Magnus Wahlström, and Anders Yeo. Component order connectivity in directed graphs. In *15th International Symposium on Parameterized and Exact Computation (IPEC)*, pages 33–48, 2020.

[BHK19] Matthias Bentert, Klaus Heeger, and Dušan Knop. Length-bounded cuts: Proper interval graphs and structural parameters. *arXiv:1910.03409*, 2019.

[Bod96] Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[BDD+16] Hans L Bodlaender, Pål Grønås Drange, Markus S Dregi, Fedor V Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A cˆkn 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.

[BGHK95] Hans L Bodlaender, John R Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995.

[BL11] Paul Bonsma and Daniel Lokshtanov. Feedback vertex set in mixed graphs. In *Workshop on Algorithms and Data Structures (WADS)*, pages 122–133, 2011.

[CCL15] Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set: New measure and new structures. *Algorithmica*, 73(1):63–86, 2015.

[CCK+20] Timothy FN Chan, Jacob W Cooper, Martin Koutecký, Daniel Král', and Kristýna Pekárková. Matrices of optimal tree-depth and row-invariant parameterized algorithm for integer programming. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2020.

[CLL+08] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5):Article 21, 2008.

[CCHM15] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(4):1–28, 2015.

[CHM13] Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM Journal on Computing*, 42(4):1674–1696, 2013.

[CFK+15] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015.

[CMP+14] Marek Cygan, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Ildikó Schlotter. Parameterized complexity of Eulerian deletion problems. *Algorithmica*, 68(1):41–61, 2014.

[DF12] Rodney G Downey and Michael R Fellows. *Parameterized complexity*. Springer, 2012.

[DDv16] Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1201, 2016.

[EHK+19] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. An algorithmic theory of integer programming. *arXiv:1904.01361*, 2019.

[FGPR08] Fedor V Fomin, Serge Gaspers, Artem V Pyatkin, and Igor Razgon. On the minimum feedback vertex set problem: Exact and enumeration algorithms. *Algorithmica*, 52(2):293–307, 2008.

[FLPS14] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative sets of product families. In *European Symposium on Algorithms (ESA)*, pages 443–454, 2014.

[FLS14] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 142–151, 2014.

[FLS+18] Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, Michał Pilipczuk, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms*, 14(3):1–45, 2018.

[GMM19] Alexander Göke, Dániel Marx, and Matthias Mnich. Parameterized algorithms for generalizations of directed feedback vertex set. In *International Conference on Algorithms and Complexity (CIAC)*, pages 249–261, 2019.

[GMM20a] Alexander Göke, Dániel Marx, and Matthias Mnich. Hitting long directed cycles is fixed-parameter tractable. In *International Colloquium on Automata, Languages and Programming (ICALP)*, 2020.

[GMM20b] Alexander Göke, Dániel Marx, and Matthias Mnich. Hitting long directed cycles is fixed-parameter tractable. *arXiv:2003.05267*, 2020.

[GMM20c] Alexander Göke, Dániel Marx, and Matthias Mnich. Parameterized algorithms for generalizations of directed feedback vertex set. *arXiv:2003.02483*, 2020.

[GMCM19] Alexander Göke, Lydia Mirabel Mendoza Cadena, and Matthias Mnich. Resolving infeasibility of linear systems: A parameterized approach. In *International Symposium on Parameterized and Exact Computation (IPEC)*, 2019.

[Kar72] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[KS73] Daniel J Kleitman and Joel Spencer. Families of k-independent sets. *Discrete Mathematics*, 6:255–262, 1973.

[Mar06] Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

[Mar09] Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009.

[MR14] Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM Journal on Computing*, 43(2):355–388, 2014.

[Mon85] B. Monien. How to find long paths efficiently. *Annals of Discrete Mathematics*, 25:239–254, 1985.

[NRSS20] Rian Neogi, MS Ramanujan, Saket Saurabh, and Roohani Sharma. On the parameterized complexity of deletion to $\mathcal{H}$-free strong components. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2020.

[NDM06] Jaroslav Nešetřil and Patrice Ossona De Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *European Journal of Combinatorics*, 27(6):1022–1041, 2006.

[OV20] Neil Olver and László A Végh. A simpler and faster strongly polynomial algorithm for generalized flow maximization. *Journal of the ACM (JACM)*, 67(2):1–26, 2020.

[RSS06] Venkatesh Raman, Saket Saurabh, and CR Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Transactions on Algorithms*, 2(3):403–415, 2006.

[RSV04] Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[RS95] Neil Robertson and Paul D Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[RS04] Neil Robertson and Paul D Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[SZ16] Hadas Shachnai and Meirav Zehavi. Representative families: a unified tradeoff-based approach. *Journal of Computer and System Sciences*, 82(3):488–502, 2016.

[Zeh15] Meirav Zehavi. Mixing color coding-related techniques. In *European Symposium on Algorithms (ESA)*, pages 1037–1049, 2015.

[Zeh16] Meirav Zehavi. A randomized algorithm for long directed cycle. *Information Processing Letters*, 116(6):419–422, 2016.