Wannabe Bounded Treewidth Graphs Admit a Polynomial Kernel for DFVS

Daniel Lokshtanov[1], M. S. Ramanujan[2], Saket Saurabh[3], Roohani Sharma[3], and Meirav Zehavi[4]

[1] University of California, Santa Barbara, USA daniello@ucsb.edu
[2] University of Warwick, UK R.Maadapuzhi-Sridharan@warwick.ac.uk
[3] Institute of Mathematical Sciences, HBNI, India {saket,roohani}@imsc.res.in
[4] Ben-Gurion University of the Negev, Israel meiravze@bgu.ac.il

Abstract. In the Directed Feedback Vertex Set (DFVS) problem, given a digraph D and $k \in \mathbb{N}$, the goal is to check if there exists a set of at most k vertices whose deletion from D leaves a directed acyclic graph. Resolving the existence of a polynomial kernel for DFVS parameterized by the solution size k is a central open problem in Kernelization. In this paper, we give a polynomial kernel for DFVS parameterized by k *plus* the size of a treewidth-η modulator (of the underlying undirected graph), where η is any fixed positive integer. Our choice of parameter strictly encompasses previous positive kernelization results on DFVS. Our main result is based on a novel application of the tool of *important separators* embedded in state-of-the-art machinery such as protrusion decompositions.

Keywords: DFVS · Kernel · Important Separator · Treewidth.

1 Introduction

Feedback Set problems are fundamental combinatorial optimization problems. Typically, in these problems, we are given a (directed or undirected) graph G and an integer k, and the objective is to select at most k vertices, edges or arcs to hit all cycles of the input graph. Feedback Set problems are among Karp's 21 NP-complete problems and have been a subject of active research from algorithmic [3, 5, 6, 11–13, 15, 16, 18, 19, 25, 32, 35, 33, 37, 45, 49] as well as structural point of view [24, 34, 36, 44, 46–48]. In particular, such problems constitute one of the most important topics of research in parameterized algorithms [11, 13, 15, 16, 18, 19, 35, 33, 37, 45, 49], spearheading the development of several new techniques.

In this paper, we study the Directed Feedback Vertex Set (DFVS) problem, whose input consists of a digraph D on n vertices and m arcs, and an integer k that is the parameter. The goal is to check whether there exists a vertex subset of size at most k that intersects every directed cycle in D. In other words, we ask whether there exists a set of vertices S of size at most k such that $F = D - S$ is a directed acyclic graph (DAG). For over a decade, resolving the parameterized complexity of DFVS was considered one of the most important open problems in parameterized complexity. In fact, this question was

posed as an open problem in the first few papers on fixed-parameter tractability (FPT) [21, 22]. In a breakthrough paper, DFVS was shown to be fixed-parameter tractable by Chen et al. [15] in 2008, who gave an algorithm that runs in time $\mathcal{O}(4^k \cdot k! \cdot k^4 \cdot n^4)$. Subsequently, it was observed that, in fact, the running time of this algorithm is $\mathcal{O}(4^k \cdot k! \cdot k^4 \cdot nm)$ (see, e.g., [17]). Since this breakthrough, the techniques used to solve DFVS have found numerous applications. However, apart from the design of a parameterized algorithm for DFVS with a linear dependency on $m + n$ [41], the question of the existence of a polynomial kernel for the same problem has seen close to no progress. To be specific, the following fundamental question about the problem remains open:

Does DFVS *admit a polynomial kernel?*

That is, does there exist a polynomial-time algorithm (called a *kernelization algorithm*) that, given an instance (D, k) of DFVS, returns an equivalent instance (D', k') (called a *kernel*) of DFVS whose size is bounded by a polynomial function of k? We refer the reader to the surveys [29, 31, 38, 40], as well as the books [28, 17, 23, 26, 43], for a detailed treatment of the area of kernelization.

The lack of progress on the kernelization complexity of DFVS has led to the study of this problem on restrictive input instances. In particular, we know of polynomial kernels for DFVS when the input digraph is a tournament or even various other generalizations of it [1, 4, 20, 39]. However, the existence of a polynomial kernel for DFVS is open even when the input digraph is a planar digraph. Recently, in order to shed some light on the kernelization complexity of DFVS, the following two directions have been proposed.

1. Study the kernelization complexity of DFVS where, in addition to the solution size, we parameterize by a structural parameter (such as the size of a modulator to a graph of constant treewidth). Throughout the paper, by treewidth of a digraph we refer to the *treewidth of its underlying undirected graph*.
2. Study the kernelization complexity of DFVS with an additional restriction on the resulting DAG $F = D - S$.

In this paper, we aim to significantly broaden the scope of both directions as much as possible—our efforts are mostly aimed at the first approach, but we also deal with the second one. Towards our contribution for the first direction, we give a polynomial kernel for DFVS parameterized by the solution size plus the size of a treewidth-η modulator. Formally, for a directed graph D, a subset $M \subseteq V(D)$ is called a *treewidth η-modulator* if $D - M$ has treewidth at most η. We consider the following parameterized problem parameterized by $k + \ell$.

DFVS/DFVS+Treewidth-η Modulator (Tw-η Mod)
Input: A digraph D, $k \in \mathbb{N}$, $M \subseteq V(D)$ where $|M| = \ell$ and $D - M \in \mathcal{F}_\eta$.
Output: Is there $S \subseteq V(D)$ where $|S| \leq k$ and $D - S$ is a DAG?

Observe that DFVS/DFVS+Tw-η Mod is the same problem as DFVS with just a different parameter. Our main contribution is the following theorem.

Theorem 1.1. DFVS/DFVS+Tw-η Mod *admits a kernel of size* $(k \cdot \ell)^{\mathcal{O}(\eta^2)}$.

Notably, our result can be viewed as a proof that DFVS parameterized *only* by k, admits a polynomial kernel on the class of *all* graphs whose treewidth can be made constant by the removal of $k^{\mathcal{O}(1)}$ vertices. Yet another justification for our choice of parameter is the following. Parameterized by k alone, the problem has been open for a very long time. On the other hand, parameterized by ℓ alone, it can be easily seen that the problem does not exhibit a polynomial kernel (by a reduction from Vertex Cover parameterized by the size of a treewidth-2 modulator) unless NP $\subseteq$ coNP/poly. Thus, $k+\ell$ is a natural parameter to explore.

We also remark that the proof of Theorem 1.1 required the development of a novel use of important separators, among other ideas for finding protrusions and using the state-of-the-art protrusion machinery. Thus, as a side reward, the ideas developed in this article may be insightful, helping to design reduction rules for a polynomial kernel of DFVS. Lastly, our result encompasses the recent result of Bergougnoux et al. [7], where they studied DFVS parameterized by the feedback vertex set number of the underlying undirected graph, and gave a polynomial kernel for this problem. Specifically, they gave a kernel of size $\mathcal{O}(\mathbf{fvs}^4)$, where **fvs** is the feedback vertex set number of the underlying undirected graph of D. Note that our parameter $k + \ell$ is not only upper bounded by $\mathcal{O}(\mathbf{fvs})$, but it can be arbitrarily smaller than **fvs**. We also remark that DFVS has already been parameterized by treewidth in the literature (not for kernelization purposes)—recently, Bonamy et al. [10] showed that DFVS parameterized by the treewidth of the input graph, t, can be solved in time $2^{\mathcal{O}(t \log t)} n^{\mathcal{O}(1)}$, and that unless the Exponential Time Hypothesis fails, it cannot be solved in time $2^{o(t \log t)} n^{\mathcal{O}(1)}$.

We now consider our contribution towards the second direction viz. understanding the kernelization complexity of DFVS with an additional restriction on the resulting DAG $F = D - S$. This direction was proposed by Mnich and van Leeuwen [42]. Essentially, the basic philosophy of their program is the following: What happens to the kernelization complexity of DFVS when we consider deletion to subclasses of DAGs? Specifically, Mnich and van Leeuwen [42] obtained polynomial kernels for the classes of out-forests, out-trees and directed pumpkins. Note that for all these families, the treewidth of the graph obtained after deleting the solution is constant. In a follow-up paper [2], the kernel sizes given by Mnich and van Leeuwen [42] were reduced. Towards our contribution, we begin by considering a broad family of graphs: for a *fixed* positive integer η, let $\mathcal{F}_\eta$ be the family of digraphs of treewidth at most η. The corresponding problem, named $\mathcal{F}_\eta$-Vertex Deletion Set, is defined as follows. Given a digraph D and an integer k, determine whether there exists a set S of size at most k such that $F - S$ is a DAG in $\mathcal{F}_\eta$. Observe that this problem is different from DFVS as the deletion set is required to bring in more structure to the resulting graph. Towards resolving the existence of a polynomial kernel for this problem (parameterized by k), we observe that the existing machinery can already be harnessed to resolve this question affirmatively. Nevertheless, as this finding already generalizes several results in the literature, we give an outline of this result in Appendix F.

Given a choice of which amongst the two directions may bring us closer to the resolution of the kernelization complexity of DFVS, we believe that studying

DFVS parameterized by a non-trivial structural parameter larger than k has a major advantage over studying the DFVS problem by restricting the resulting DAG—the study of a larger parameter does not alter the problem at hand, that is, the focus is still aimed at DFVS itself rather than at a variant of it. In fact, the second approach may derail us from the track of resolving the kernelization complexity of DFVS as each restriction of the output DAG results in its own definition of a variant of DFVS that may have its own properties. Thus, if the ultimate goal is to design a polynomial kernel for DFVS itself (or prove that such a kernel does not exist), we find the first approach more suitable. Nevertheless, it is also important to note that the questions raised by the second approach, namely, the study of the variants of DFVS, may be interesting in their own right.

Proof Idea of Theorem 1.1. Our kernelization algorithm can be divided into three main phases. We give a brief summary of each phase here.

1. Computing a zone decomposition of the directed graph: We first compute a decomposition of D into three components: the vertex set M (modulator), a collection of $\mathcal{O}(k\ell^2)$ vertex sets $\mathcal{Z}$ (zones), and a vertex set R (remainder) of size $\mathcal{O}(k\ell^2)$. All of these sets are pairwise disjoint and form a partition of $V(D)$. The aim of this decomposition is to achieve a few properties with respect to each zone $Z \in \mathcal{Z}$, which we will later exploit to design reduction rules to bound the size of each zone. Since the number of zones in the decomposition and the size of R is $\mathcal{O}(k\ell^2)$, in order to get the desired kernel, it would be enough to bound the size of each zone $Z \in \mathcal{Z}$ by $k\ell^{\mathcal{O}(1)}$, after such a decomposition is constructed.

Let us mention three important properties of a zone $Z \in \mathcal{Z}$ that this decomposition achieves, and which play a critical role in helping us bound the size of Z. The first property is that if D has a directed feedback vertex set of size at most k, then there exists a directed feedback vertex set, say S, in D of size at most k, whose intersection with Z is of constant size. The second property is that the neighborhood of Z is entirely contained in $M \cup R$ and the size of the neighborhood of Z in R is bounded by some constant. Finally, for any two vertices in the neighborhood of Z in M, the maximum value of a directed flow from one to the other is either extremely high or zero. We will exploit the first property to mark a "small" set of vertices in Z that in some sense "represents" all partial solutions in Z. Such a set, which is called Γ_{DFVS}, is then used to design reduction rules that eliminate arcs between $Z \setminus \Gamma_{\text{DFVS}}$ and M. The third property is critically used in these reduction rules. Then, from the second property, all the vertices in $Z \setminus \Gamma_{\text{DFVS}}$ have a constant-sized neighborhood outside Z. Having this information at hand, we further partition Z into small slices, each of which is then replaced by constant sized sets by using protrusion machinery.

Though at first glance this decomposition of the graph may look very similar to the near-protrusion decomposition of [27], it is not a near-protrusion decomposition. In fact, for this problem we cannot find a near-protrusion decomposition.

2. Computing a k-DFVS Representative Set in Z: A k-DFVS representative in a zone Z is a subset of vertices of Z (say, Γ_{DFVS}) with the following property: If D has a directed feedback vertex set of size at most k, then there is a directed feedback vertex set S in D of size at most k and $S \cap Z \subseteq \Gamma_{\text{DFVS}}$. We

aim to compute such a set whose size is bounded by some polynomial in k and ℓ. For this purpose, we first revisit the relation between our problem and Skew Multicut. In particular, we see that for any directed feedback vertex set$S$ D, $S \cap Z$ is a solution to an "appropriate" instance of the Skew Multicut problem. Thus, if we can compute solutions to all possible appropriate instances of Skew Multicut, then we can set Γ_{DFVS} to be the union of all these solutions. In this overview, we prefer to keep the notion of an appropriate instance abstract.

Unfortunately, a single instance of our problem gives rise to a huge number of appropriate instances. In particular, if we naively construct Γ_{DFVS} by *individually* computing a solution for each possible choice for an appropriate instance, we do not obtain a set whose size is bounded by a polynomial function in k and ℓ. So, in the second step, we invest significant efforts to construct a set Γ_{DFVS} of the desired small size, which contains a solution for each possible choice of an appropriate instance. To this end, we observe that if such a set of the desired size exists, then solutions of "many" possible appropriate instances intersect a lot. Very roughly speaking, we aim to identify a small set of vertices that is guaranteed to be contained in solutions of "many" instances. If we can identify such a set, then we delete it from all appropriate instances in which it is guaranteed to be present in some solution, and recurse on the resulting instances. (Here, only one recursive call is performed.) From the properties of a zone decomposition, we are able to derive that there is a solution to our original problem whose intersection with Z is small, which in turn leads us to the observation that we can only focus on small solutions for each appropriate instance. Hence, we can bound the depth of the recursion. Though this description roughly conveys the broad picture, the implementation of these ideas is significantly more complex. For example, we are unable to find a small set of vertices that is contained in some solution for "many" instances. Instead, we find a *collection* of small sets such that at least one among them is the set that we want, though we do not know which one.

These abstract ideas are materialized with the help of important separators (defined in Section 2), the Pushing Lemma for Skew Multicut and a new (simple) lemma, which we call the *Important Separator Preservation (via Small Sink Set) Lemma*. This lemma says that if S is an important (X, Y)-separator of size α in some digraph, then S is also an important (X, Y')-separator for some subset Y' of Y of size at most $\alpha + 1$, where $Y \setminus Y'$ is removed from that digraph. We mainly use this lemma in situations (that arise when we try to compute the collection of sets mentioned above) that require guessing the set Y when X is given, so that we can compute important (X, Y)-separators. In these situations, since it is enough to guess Y' to compute all important (X, Y)-separators (from the Important Separator Preservation Lemma), the fact that Y' is small significantly reduces the search space for Y' compared to that for Y.

3. Reduction Rules for bounding the size of each zone Z: After computing a small set that is a k-DFVS representative in a zone in our zone decomposition, our final objective is to bound its size. To this end, we design reduction rules that decompose a zone Z into a "small" number of *protrusions*. (Roughly speaking, a protrusion in a graph G is an induced subgraph $G[U]$ of G for a subset $U \subseteq V(G)$

that has constant treewidth and only a constant number of vertices with neighbors in $G-U$.) More precisely, we first design a set of reduction rules that only bounds the size of the neighborhood of every zone Z (with Γ_{DFVS} removed) outside Z by a constant. Then, by computing a nice tree decomposition of $D[Z]$ and relying on properties of an LCA-closure in that tree, we decompose the set Z as $Z = \widetilde{\Gamma}_{\text{DFVS}} \uplus \biguplus_{U \in \mathcal{U}} U$ such that $\Gamma_{\text{DFVS}} \subseteq \widetilde{\Gamma}_{\text{DFVS}}$, the size of $\mathcal{U}$ is "small", and each set $U \in \mathcal{U}$ induces a protrusion. We then replace each protrusion $D[U]$ by a "small" digraph such that the resulting digraph is a "minor" of the original digraph, and the input modulator M is also a treewidth-η modulator in the resulting digraph. This concludes our kernelization algorithm.

We remark that the operations of the last step of our kernelization algorithm ensure that the input modulator M remains a modulator in the final returned kernel. If we allow the modulator in the returned instance to be of size larger than $|M|$, then we can bypass all of these reduction rules and the protrusion machinery, and directly create the kernelized instance by taking the torso of the set S that is the union of M, R and a k-DFVS representative set in each Z. Here, by torso we mean that for every two vertices $u, v \in S$ with a directed path from u to v whose internal vertices do not belong to S, we add an arc from u to v. Since the set S is of small size (polynomial in k and ℓ), we directly obtain a kernel by omitting the vertices outside S. However, when we perform the torso operation, we lose the property that M is a modulator for the final instance, which means that the parameter can increase to be of the magnitude of the entire kernel.

Road-map. In this extended abstract, we only present a high-level overview of our approach. (All details can be found in the appendix, with pointers provided in relevant sections.) In particular, our focus is to convey the main ideas of Step 2 above. In Section 3, we present a short description of our zone decomposition (Step 1). In Section 4, we describe the main difficulties, and the insights incorporated to overcome them, with respect to the design a procedure to find a k-DFVS representative set for any single zone (Step 2). In Section 5, we recall our final objective, that is, to bound the size of each zone (Step 3).

2 Preliminaries

Standard definitions are relegated to Appendix A. Throughout the paper, parenthesis (resp. braces) notation denote ordered (resp. unordered) sets. For a digraph D and subsets $X, Y \subseteq V(D)$, an (X, Y)-*separator* in D is a set $S \subseteq V(D) \setminus (X \cup Y)$ such that there is no path from any vertex in X to any vertex in Y in $D - S$. (The separator should have an empty intersection with $X \cup Y$.) When we consider a topological ordering of a DAG, suppose that no arc is directed from a vertex v to a vertex u that occurs *before* v in the ordering. Given a topological ordering π of a DAG D and $X \subseteq V(D)$, we say that π_X is *induced by* π if the vertices of X appear in the same order in π_X and in π. By the treewidth $\mathbf{tw}(D)$ of a digraph D and a (nice) tree decomposition of D, we refer to the treewidth and a (nice) tree decomposition of the underlying undirected graph of D, respectively. For any $X \subseteq V(D)$, we say that X is a η-*treewidth modulator* in D if $\mathbf{tw}(D - X) \leq \eta$.

Definition 2.1 (Important Separators). *Let D be a digraph and $X, Y \subseteq V(D)$. Let $S \subseteq V(D) \setminus (X \cup Y)$ be an (X,Y)-separator and let R be the set of vertices reachable from X in $D - S$. We say that S is an* important (X,Y)-separator *if it is inclusion-wise minimal and there is no (X,Y)-separator $S' \subseteq V(D) \setminus (X \cup Y)$ such that $|S'| \leq |S|$ and $R \subsetneq R'$, where R' is the set of vertices reachable from X in $D - S'$.*

Proposition 2.1 ([14]). *Let D be a digraph, $X, Y \subseteq V(D)$ and $k \in \mathbb{N} \cup \{0\}$. Then, D has at most 4^k important (X,Y)-separators of size at most k, and the set of all of them can be constructed in time $\mathcal{O}(4^k \cdot k^2 \cdot (n+m))$.*

3 Decomposing the Graph

Given an instance (D, k, M) of DFVS/DFVS+Tw-η Mod, the goal of this section is to compute a decomposition of D consisting of three components: the vertex set M (modulator), a collection of vertex sets $\mathcal{Z}$ (zones), and a vertex set R (remainder). All of these sets would be pairwise disjoint. The crux is to "divide-and-conquer" D so that each zone—that is, a set $Z \in \mathcal{Z}$—would correspond to a subproblem that is easier to solve than (D, k, M) because (1) the intersection of a minimum solution with Z would be necessarily small, and (2) the interaction of Z is "well-structured" with respect to M, "limited" with respect to R , and "non-existent" with respect to any other zone. Towards the computation of R, we compute three sets: (i) a solution S in $D - M$; (ii) a set F to separate vertices in M that have low-flow; (iii) an LCA-closure of bags derived from $S \cup F$. The arguments given on the way to construct these sets will only partially prove that we have derived the desired decomposition. At the end, we complete the proof by focusing on the property regarding the intersection of a minimum solution with each zone. For lack of space, we defer these details to Appendix B. We summarize the properties of our decomposition and its construction as follows.

Definition 3.1. *Let (D, k, M) be an instance of* DFVS/DFVS+Tw-η Mod *. A partition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$ is a* zone-decomposition *if:*

1. *$D - (M \cup R)$ is a DAG.*
2. *For all $Z \in \mathcal{Z}$, we have $N(Z) \subseteq M \cup R$, and $|N(Z) \cap R| \leq 2(\eta + 1)$.*
3. *For all $(u, v) \in M \times M \setminus E(D)$, either there is no path from u to v in the digraph $D - ((M \cup R) \setminus \{u, v\})$, or there are at least $k + 1$ internally vertex-disjoint paths from u to v in D.(For $u = v$, having no path refers to having no path on at least two vertices.)*

Lemma 3.1. *There is a polynomial-time algorithm that, given an instance (D, k, M) of* DFVS/DFVS+Tw-η Mod*, either correctly decides that (D, k, M) is a* NO*-instance, or constructs a zone-decomposition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$ with $|\mathcal{Z}| \leq 6k(\ell^2 + 1)$ and $|R| \leq 2(\eta + 1)k(\ell^2 + 1)$.*

We now argue that if (D, k, M) is a YES-instance, then the size of the intersection of each minimum(-size) solution with each zone is only a constant. Formally, we have the following lemma (proved in Appendix B).

Lemma 3.2. *Let (D, k, M) be a* YES*-instance of* DFVS/DFVS+Tw-η Mod *with zone-decomposition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$. For any minimum-sized directed feedback vertex set S of D, we have $|S \cap Z| \leq |N(Z) \cap R| \leq 2(\eta+1)$ for all $Z \in \mathcal{Z}$.*

4 Reducing Each Part: k-DFVS Representative Marking

For an instance (D, k, M) of DFVS/DFVS+Tw-η Mod, the kernelization algorithm starts by applying Lemma 3.1 and either concludes that (D, k, M) is a NO-instance, or obtains a zone decomposition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$ with the properties in Lemma 3.1. In this section, we fix an arbitrary zone $Z \in \mathcal{Z}$ and give a polynomial time algorithm (Lemma 4.1) to mark a small set of vertices in Z which in some sense "represents" all partial solutions in Z. Such a set will then be used to design reduction rules that bound the degree of the vertices in Z that are not in the representative, which will further be useful to decompose Z into a small number of protrusions. We now formally define the desired set.

Definition 4.1 (k-DFVS Representative in Z). *For a digraph D, $Z \subseteq V(D)$ and an integer k, we say that $\Gamma_{\text{DFVS}} \subseteq Z$ is a* k-DFVS representative in Z *if the following holds. If D has a directed feedback vertex set of size at most k, then it also have a directed feedback vertex set S of size at most k where $S \cap Z \subseteq \Gamma_{\text{DFVS}}$.*

Lemma 4.1 (k-DFVS Representative Marking Lemma). *There is an algorithm that given a digraph D, $Z \subseteq V(D)$ and an integer k, runs in time $2^{\mathcal{O}(\eta^2)} \cdot (k\ell)^{\mathcal{O}(\eta^2)} \cdot (n+m)$, and returns a set $\Gamma_{\text{DFVS}} \subseteq Z$ of size $(k\ell)^{\mathcal{O}(\eta^2)}$ such that Γ_{DFVS} is a k-DFVS representative in Z.*[5]

We prove Lemma 4.1 in two parts. In Section 4.1, we revisit the relation between DFVS and Skew Multicut (defined later). Using this relation, we conclude that Γ_{DFVS} can be computed by taking the union of skew multicuts of "appropriate" instances of Skew Multicut. The problem at this stage stems from the fact that the set of appropriate instances that we need to consider is not polynomially bounded, and hence a naive approach of finding a solution to each of these instances and taking their union does not work. This issue is tackled in Section 4.2. Statements in this section marked by an * are proved in Appendix C.

4.1 Revisiting the Relation with the Skew Multicut Problem

Towards the definition of Skew Multicut, we first define a skew multicut in a digraph. Let D be a digraph, and $\mathcal{P} = ((s_1, t_1), \ldots, (s_p, t_p))$ be an *ordered* set of pairs of vertices (called *terminals*) of D. A *skew multicut* in D with respect to $\mathcal{P}$ is a set S of non-terminal vertices of D such that for all $i, j \in \{1, \ldots, p\}$ with $i \leq j$, there is no path from s_i to t_j in $D - S$. [6] In Skew Multicut (SMC), the

[5] Throughout the paper, we do not hide constants that depend on η in the $\mathcal{O}$ notation.

[6] In the standard definition of a skew multicut, the latter condition is replaced by the following symmetric condition: For all $i, j \in \{1, \ldots, p\}$ with $j \leq i$, there is no path from s_i to t_j in $D - S$. Here, it is convenient to use $i \leq j$ rather than $j \leq i$.

input is a digraph D, an ordered set $\mathcal{P} = ((s_1, t_1), \ldots, (s_p, t_p))$ and an integer k. The goal is to decide whether D has a skew multicut of size at most k with respect to $\mathcal{P}$. In order to state the relation between our problem and Skew Multicut, the following notation will come handy. Informally, for a digraph D and a subset $B \subseteq V(D)$, we "split" each vertex in B into two distinct vertices, and thereby define the digraph $D_{\dagger B}$. The two fractions of each split vertex will latter correspond to a terminal pair. Formally, we construct $D_{\dagger B}$ as follows.

Definition 4.2. *Let D be a digraph and $B \subseteq V(D)$. The digraph $D_{\dagger B}$ is obtained from D as follows. Replace each vertex $v \in B$ by two new vertices v^{out} and v^{in}, add the arc (v^{in}, v^{out}) and replace each arc $(u, v) \in E(D)$ by (u, v^{in}) and each arc $(v, u) \in E(D)$ by (v^{out}, u).*

Lemmas 4.2 and 4.3 show that any DFVS solution restricted to Z is a skew multicut solution to an appropriate instance of Skew Multicut and vice-versa.

Lemma 4.2 (*). *Let D be a digraph, $Z \subseteq V(D)$ and $k \in \mathbb{Z}$. Let S be a directed feedback vertex set in D of size at most k. Let $B = N(Z) \setminus S$, and let $\pi_B = (v_1, \ldots, v_b)$ be an ordering of B induced by a topological ordering π of $D - S$. Denote $D' = D[Z \cup B]_{\dagger B}$, $\mathcal{P} = ((v_1^{out}, v_1^{in}), \ldots, (v_b^{out}, v_b^{in}))$ and $k' = |S \cap Z|$. Then, there is a skew multicut in D' with respect to $\mathcal{P}$ of size at most k', that is, $(D', \mathcal{P}, k')$ is a* YES*-instance of* Skew Multicut.

Lemma 4.3 (*). *Let D be a digraph, $Z \subseteq V(D)$ and $k \in \mathbb{Z}$. Let S be a directed feedback vertex set in D of size at most k. Let $B = N(Z) \setminus S$ and let $\pi_B = (v_1, \ldots, v_b)$ be an ordering of the vertices of B induced by a topological ordering π of $D - S$. Denote $D' = D[Z \cup B]_{\dagger B}$, and $\mathcal{P} = ((v_1^{out}, v_1^{in}), \ldots, (v_b^{out}, v_b^{in}))$. Let S' be any skew multicut in D' with respect to $\mathcal{P}$. Then, $S^* = (S \setminus Z) \cup S'$ is a directed feedback vertex set in D.*

The number of guesses for the Skew Multicut instance for which the intersection of a potential DFVS solution with Z is a skew multicut solution is $2^{|N(Z)|} \cdot |N(Z)|^{|N(Z)|}$. This is not polynomially bounded in k and ℓ. In the next section, we see how to compute a set containing some skew multicut solution to each of these instances without having to go over the instances individually.

4.2 Computing Solutions for All Instances of Skew Multicut

We formalize the notion of "all possible choices for the appropriate instance of Skew Multicut", by defining a family of instances of Skew Multicut denoted by $\mathcal{F}_{\text{SMC}}$. To simplify notation, for a digraph D and a (not necessarily ordered) set $\mathcal{P}$ of terminal pairs, let $D - \mathcal{P}$ be the digraph obtained from D by deleting all terminals in $\mathcal{P}$. Similarly, for a subset $X \subseteq V(D)$, let $X - \mathcal{P}$ be the set of vertices obtained from X by deleting all terminals in $\mathcal{P}$. To improve readability, *unordered* sets of terminal pairs will be denoted by $\mathcal{Q}$ rather than $\mathcal{P}$. We also stress that in what follows, k should be thought of as a small constant, because here it does not refer to the original k in the input instance of DFVS, but to the parameter set up when we construct an instance of Skew Multicut.

Definition 4.3. *Given a digraph D, an* unordered *set $\mathcal{Q} = \{(s_i, t_i) : i \in \{1, \ldots, p\}, s_i, t_i \in V(D)\}$ and an integer k, $\mathcal{F}_{\text{SMC}}(D, \mathcal{Q}, k)$ is a family of instances of* Skew Multicut *such that for each $P^* \subseteq \{1, \ldots, p\}$, for each ordering π of P^* and for each $k' \leq k$, the instance $(D - (\mathcal{Q} \setminus \mathcal{P}^*), \mathcal{P}^*, k')$ belongs to $\mathcal{F}_{\text{SMC}}(D, \mathcal{Q}, k)$ where $\mathcal{P}^* = ((s_{\pi(i)}, t_{\pi(i)}) : i \in P^*)$.*

We clarify that the above notation $\mathcal{P}^* = ((s_{\pi(i)}, t_{\pi(i)}) : i \in P^*)$ means that for all $i, j \in P^*$, we have that $(s_{\pi(i)}, t_{\pi(i)})$ is ordered before $(s_{\pi(j)}, t_{\pi(j)})$ if and only if $i < j$. Similar to the notion of a k-DFVS representative of Z, we first define the notion of a k-SMC representative. The construction of a set that, for any instance in $\mathcal{F}_{\text{SMC}}$, contains some solution for that instance, is captured by the Lemma 4.4.

Definition 4.4 (k-SMC Representative). *Given a digraph D, a set $\mathcal{Q} = \{(s_i, t_i) : i \in \{1, \ldots, p\}, s_i, t_i \in V(D)\}$ and an integer k, a k-SMC representative in D with respect to $\mathcal{Q}$ is a subset $\Gamma_{\text{SMC}} \subseteq D$ such that each* YES*-instance in the family $\mathcal{F}_{\text{SMC}}(D, \mathcal{Q}, k)$ has a solution that belongs to Γ_{SMC}.*

Lemma 4.4 (k-SMC Representative Marking Lemma). *There is an algorithm that, given a digraph D $p, k \in \mathbb{N}$ and $\mathcal{Q} = \{(s_i, t_i) : i \in \{1, \ldots, p\}, s_i, t_i \in V(D)\}$, runs in time $p^{\mathcal{O}(k^2)} \cdot (n + m)$ time and outputs a k-SMC representative in D with respect to $\mathcal{Q}$ of size at most $k^2(k+1)^k \cdot p^{k(k+2)} \cdot 4^{k^2}$.*

The rest of this section concerns the proof of Lemma 4.4. We first explain (intuitively) how the algorithm of Lemma 4.4 works. Since $|\mathcal{F}_{\text{SMC}}(D, \mathcal{Q}, k)|$ is exponential in p, if a k-SMC representative in D with respect to $\mathcal{Q}$, say Γ_{SMC}, of the desired size exists, then the solutions of "many" instances in $\mathcal{F}_{\text{SMC}}(D, \mathcal{Q}, k)$ intersect "a lot". This is exactly what we want to exploit. Roughly speaking, we want to recursively apply the following step. In each recursive call, partition the instances of $\mathcal{F}_{\text{SMC}}(D, \mathcal{Q}, k)$ into $p^{\mathcal{O}(k)}$ classes, and for each class find a set that is guaranteed to be contained in some solution for each of the instances in the class. Delete this set from the instances in the class, and recurs. Note that we keep track of deleted vertices, since they are precisely the vertices that will form Γ_{SMC}. Since we are looking for a k-sized solution in each instance in $\mathcal{F}_{\text{SMC}}(D, \mathcal{P}, k)$, the depth of the recursion is at most k and hence, we can form the set Γ_{SMC} of the desired size (i.e. $\mathcal{O}(p^{g(k)})$ for some function g of k).

Now, we try to formalize this approach; depending on the obstacles faced, we add layers and machinery to the outline above. First consider the step of partitioning the instances of $\mathcal{F}_{\text{SMC}}(D, \mathcal{Q}, k)$ into some $p^{\mathcal{O}(k)}$ classes, such that for each class there is a set guaranteed to be contained in some solution for each of the instances in the class. To this end, we first try the power of the Pushing Lemma for Skew Multicut, defined below. Roughly speaking, this lemma states that any YES-instance $(D, ((s_i, t_i) : i \in \{1, \ldots, a\}), k)$ of Skew Multicut (for instances in $\mathcal{F}_{\text{SMC}}(D, \mathcal{Q}, k)$, we have $a \leq p$) has a solution of size at most k that contains an important $(\{s_1\}, \{t_1, \ldots, t_a\})$-separator of size at most k.

Proposition 4.1 (Pushing Lemma for Skew Multicut, [14]). *For a* YES*-instance $(D, \mathcal{P} = ((s_1, t_1), \ldots, (s_p, t_p)), k)$ of* Skew Multicut*, there is a solution S^* containing an important $(\{s_1\}, \{t_1, \ldots, t_p\})$-separator in D of size at most k.*

Now, consider any $P^* \subseteq \{1, \ldots, p\}$. Assume w.l.o.g. that $P^* = \{1, \ldots, p^*\}$. Consider all YES-instances in $\mathcal{F}_{\text{SMC}}(D, \mathcal{P}, k)$ where the first terminal pair is (s_1, t_1) and the other terminal pairs are $\{(s_2, t_2), \ldots, (s_{p^*}, t_{p^*})\}$ in some order. Then, by Proposition 4.1, for each of these instances there exists a solution containing some important $(\{s_1\}, \{t_1, \ldots, t_{p^*}\})$-separator of size at most k. This is not exactly what we wanted (since we do not obtain a *single* set that is contained in some solution for each of the instances), but we can still work with this as the number of important separators of size at most k is at most 4^k (from Proposition A.4). Then, we can branch on which important separator to add to Γ_{SMC}. Thus, Proposition 4.1 seems to give a way to go about constructing Γ_{SMC}.

However, we are not done yet because if we naively utilize the Pushing Lemma approach, we need to partition $\mathcal{F}_{\text{SMC}}(D, \mathcal{P}, k)$ into $2^p \cdot p$ classes. Indeed, we have 2^p possibilities to choose a subset $P^* \subseteq \{1, \ldots, p\}$ (which captures the indices of the terminals pairs in $\mathcal{P}$ that should not be deleted), and $p^* \leq p$ choices for which is the index in P^* of the first terminal pair (from which we push our solution as described above). For us, $2^p \cdot p$ is a huge number. To handle this issue, we introduce another tool, called the *Important Separator Preservation (via Small Sink Set) Lemma* (formally defined later). Intuitively, this lemma says that if I is an important (X, Y)-separator, then I is also an important (X, Y')-separator for some $Y' \subseteq Y$ where the size of Y' is at most the size of I plus one.

Lemma 4.5 (*, Important Separator Preservation (via Small Sink Set) Lemma). *Let D be a digraph with $X, Y \subseteq V(D)$ and an important (X, Y)-separator $S \subseteq V(D)$ of size α. There is $Y' \subseteq Y$ of size $\alpha + 1$ such that S is an important (X, Y')-separator in $D - (Y \setminus Y')$.*

The observation that we can exploit this lemma in our setting is a crucial insight in the design of our kernel. Recall that by Proposition 4.1, we can conclude that for some class of instances, the following property holds: There exists a pair (X, Y), where $X = \{s_i\}$ and Y is some set of terminals t_j, such that there is an important (X, Y)-separator of size at most k that is contained in some solution for each of the instances in the class. Since the number of important (X, Y)-separators of size at most k is small, we could branch on them. Basically, we combine Lemma 4.5 with Proposition 4.1 to add another layer of branching. Below, we briefly discuss the meaning of this extra layer.

Here, we partition our instances into p classes: All instances that have (s_i, t_i) as the first terminal pair (recall that the set of terminal pairs in Skew Multicut is ordered) belong to the same class. While before we had a refined partition with $2^p \cdot p$ classes, here we only have p classes, but which *at first glance* seem non-informative. However, we show that (by Lemma 4.5) not much additional information is needed. More precisely, we argue that for all YES-instances in the same class of our rough partition, there exists some $p^{\mathcal{O}(k)}$ sized collection of pairs $\{(X_i, Y_i) : i \in p^{\mathcal{O}(k)}\}$ with the following property: For any instance in the class, there exists a pair in this collection, say (X_i, Y_i), such that there exists an important (X_i, Y_i)-separator of size at most k that is contained in some solution of that instance. Since the size of the collection is $p^{\mathcal{O}(k)}$, and for each pair in it

there are at most 4^k important separators of size at most k, we branch into at most $p^{\mathcal{O}(k)} \cdot 4^k$ branches for each class. Since $p^{\mathcal{O}(k)} \cdot 4^k$ is small enough to obtain a kernel—recall that in Skew Multicut, k is small (constant) but p is large—let us move ahead to see how we obtain the collection $\{(X_i, Y_i) : i \in p^{\mathcal{O}(k)}\}$.

We claim that for any class, whose first terminal pair is some (s_i, t_i), the collection $\{(s_i, T) : T \subseteq \{t_1, \ldots, t_p\}, |T| \leq k+1\}$ is precisely that collection that we want. To see this, consider any YES-instance $(D, \mathcal{P}^*, k)$ whose first terminal pair is (s_i, t_i). Let P^* denote the set of indices of the pairs in $\mathcal{P}^*$. By the Pushing Lemma for Skew Multicut, there exists a solution to this instance that contains some important $(\{s_i\}, \{t_j \mid j \in P^*\})$-separator of size at most k. In turn, by the Important Separator Preservation Lemma, there exists $T \subseteq \{t_j \mid j \in P^*\}$ of size at most $k+1$, such that any important $(\{s_i\}, \{t_j \mid j \in P^*\})$-separator is also an important $(\{s_i\}, T)$-separator! Thus, we conclude that for each YES-instance in $\mathcal{F}_{\text{SMC}}(D, \mathcal{P}, k)$ whose first terminal pair is (s_i, t_i), there exists a pair in the collection $\{(s_i, T) : T \subseteq \{t_1, \ldots, t_p\}\}$ such that one of the important separators of size at most k of this pair is contained in some solution for this instance.

We formalize the approach above in the proof of Lemma 4.4 in Appendix C.4. Having this lemma at hand, we give a short proof for Lemma 4.1 .

5 Reduction Rules

In this section we give reduction rules to reduce the size of each "zone". More precisely, we first apply the algorithm of Lemma 3.1 which either correctly decides that (D, k, M) is a NO-instance, or constructs a zone-decomposition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$ with $|\mathcal{Z}| \leq 6k(\ell^2+1)$ and $|R| \leq 2(\eta+1)k(\ell^2+1)$. For a fixed zone $Z \in \mathcal{Z}$, we concentrate on reducing the size of Z. Once we are able to bound the size of each zone by a polynomial function of k and ℓ, we obtain a polynomial kernel for our problem. Thus, from now onwards we concentrate on bounding the size of a single zone Z.

For the rest of this section, consider the zone-decomposition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$ computed by the algorithm of Lemma 3.1 on input (D, k, M). Let $Z \in \mathcal{Z}$ be an arbitrarily fixed zone and let Γ_{DFVS} be a k-DFVS representative in Z, computed using the algorithm of Lemma 4.1. We bound the size of Z in a two step procedure. In the first step, described in Appendix D.1, we design reduction rules that remove all the arcs between M and $Z \setminus \Gamma_{\text{DFVS}}$ (at the cost of adding arcs between M and Γ_{DFVS}). Once this is done, we have that Z interacts with the "outside world" in a limited fashion via Γ_{DFVS} alone. After we have achieved this, in the second step (described in Appendix D.2), we will be able to partition $Z \setminus \Gamma_{\text{DFVS}}$ into a "small" number of slices such that each slice has treewidth at most η and has at most $\mathcal{O}(\eta)$ neighbors outside (that is, the slice is an $\mathcal{O}(\eta)$-protrusion). Each such slice can then be replaced by a constant size equivalent slice using the protrusion replacement machinery. For the implementation of this approach, see Appendix D (concluded with the proof of Theorem 1.1).

References

1. Abu-Khzam, F.: A kernelization algorithm for d-HS. JCSS **76**(7), 524–531 (2010)
2. Agrawal, A., Saurabh, S., Sharma, R., Zehavi, M.: Kernels for deletion to classes of acyclic digraphs. JCSS **92**, 9–21 (2018)
3. Bafna, V., Berman, P., Fujito, T.: A 2-approximation algorithm for the undirected feedback vertex set problem. SIAM J. Discrete Math. **12**(3), 289–297 (1999)
4. Bang-Jensen, J., Maddaloni, A., Saurabh, S.: Algorithms and kernels for feedback set problems in generalizations of tournaments. Algorithmica pp. 1–24 (2015)
5. Bar-Yehuda, R., Geiger, D., Naor, J., Roth, R.M.: Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. SIAM J. Comput. **27**(4), 942–959 (1998)
6. Becker, A., Geiger, D.: Optimization of pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. AI **83**, 167–188 (1996)
7. Bergougnoux, B., Eiben, E., Ganian, R., Ordyniak, S., Ramanujan, M.: Towards a polynomial kernel for directed feedback vertex set. In: LIPIcs. vol. 83 (2017)
8. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM Journal on computing **25**(6), 1305–1317 (1996)
9. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (Meta) kernelization. J. ACM **63**(5), 44:1–44:69 (2016)
10. Bonamy, M., Kowalik, Ł., Nederlof, J., Pilipczuk, M., Socała, A., Wrochna, M.: On directed feedback vertex set parameterized by treewidth. In: WG. pp. 65–78 (2018)
11. Cao, Y., Chen, J., Liu, Y.: On feedback vertex set new measure and new structures. In: SWAT. Lecture Notes in Computer Science, vol. 6139, pp. 93–104 (2010)
12. Chekuri, C., Madan, V.: Constant factor approximation for subset feedback set problems via a new LP relaxation. In: SODA. pp. 808–820 (2016)
13. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for feedback vertex set problems. JCSS **74**(7), 1188–1198 (2008)
14. Chen, J., Liu, Y., Lu, S.: An improved parameterized algorithm for the minimum node multiway cut problem. Algorithmica **55**(1), 1–13 (2009)
15. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. J. ACM **55**(5) (2008)
16. Chitnis, R.H., Cygan, M., Hajiaghayi, M.T., Marx, D.: Directed subset feedback vertex set is fixed-parameter tractable. ACM TALG **11**(4), 28 (2015)
17. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015)
18. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: FOCS. pp. 150–159 (2011)
19. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Subset feedback vertex set is fixed-parameter tractable. SIDMA **27**(1), 290–309 (2013)
20. Dom, M., Guo, J., Hüffner, F., Niedermeier, R., Truß, A.: Fixed-parameter tractability results for feedback set problems in tournaments. JDA **8**(1), 76–86 (2010)
21. Downey, R.G., Fellows, M.R.: Fixed-parameter intractability. In: CCC. pp. 36–49 (1992)
22. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness I: basic results. SIAM J. Comput. **24**(4), 873–921 (1995)
23. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer (2013)

24. Erdős, P., Pósa, L.: On independent circuits contained in a graph. Canad. J. Math **17**, 347–352 (1965)
25. Even, G., Naor, J., Schieber, B., Sudan, M.: Approximating minimum feedback sets and multicuts in directed graphs. Algorithmica **20**(2), 151–174 (1998)
26. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin (2006)
27. Fomin, F.V., Lokshtanov, D., Misra, N., Saurabh, S.: Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In: FOCS. pp. 470–479 (2012), see http://www.ii.uib.no/ daniello/papers/PFDFullV1.pdf for the fullversion.
28. Fomin, F.V., Lokshtanov, D., Saurabh, S., Zehavi, M.: Kernelization: theory of parameterized preprocessing. Cambridge University Press (2018)
29. Fomin, F.V., Saurabh, S.: Kernelization methods for fixed-parameter tractability. In: Tractability, pp. 260–282. Cambridge Univ. Press (2014)
30. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. CJM **8**(3), 399–404 (1956)
31. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. SIGACT News **38**(1), 31–45 (2007)
32. Guruswami, V., Lee, E.: Inapproximability of H-transversal/packing. In: APPROX/RANDOM. LIPIcs, vol. 40, pp. 284–304 (2015)
33. Kakimura, N., Kawarabayashi, Kobayashi, Y.: Erdös-Pósa property and its algorithmic applications: parity constraints, subset feedback set, and subset packing. In: SODA. pp. 1726–1736 (2012)
34. Kakimura, N., ichi Kawarabayashi, K., Marx, D.: Packing cycles through prescribed vertices. JCTB **101**(5), 378–381 (2011)
35. Kawarabayashi, K., Kobayashi, Y.: Fixed-parameter tractability for the subset feedback fet problem and the S-cycle packing problem. JCTB **102**, 1020–1034 (2012)
36. Kawarabayashi, K., Král, D., Krcál, M., Kreutzer, S.: Packing directed cycles through a specified vertex set. In: SODA. pp. 365–377 (2013)
37. Kociumaka, T., Pilipczuk, M.: Faster deterministic feedback vertex set. IPL **114**(10), 556–560 (2014)
38. Kratsch, S.: Recent developments in kernelization. Bulletin EATCS **113** (2014)
39. Le, T., Lokshtanov, D., Saurabh, S., Thomassé, S., Zehavi, M.: Subquadratic kernels for implicit 3-hitting set and 3-set packing problems. In: SODA. pp. 331–342 (2018)
40. Lokshtanov, D., Misra, N., Saurabh, S.: Kernelization-preprocessing with a guarantee. In: The Multivariate Algorithmic Revolution and Beyond. pp. 129–161 (2012)
41. Lokshtanov, D., Ramanujan, M.S., Saurabh, S.: When recursion is better than iteration. In: SODA. pp. 1916–1933 (2018)
42. Mnich, M., van Leeuwen, E.J.: Polynomial kernels for deletion to classes of acyclic digraphs. Discrete Optimization **25**, 48–76 (2017)
43. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxf. Uni. Pr. (2006)
44. Pontecorvi, M., Wollan, P.: Disjoint cycles intersecting a set of vertices. JCTB **102**(5), 1134–1141 (2012)
45. Raman, V., Saurabh, S., Subramanian, C.R.: Faster fixed parameter tractable algorithms for finding feedback vertex sets. ACM TALG **2**(3), 403–415 (2006)
46. Reed, B.A., Robertson, N., Seymour, P., Thomas, R.: Packing directed circuits. Combinatorica **16**(4), 535–554 (1996)
47. Seymour, P.: Packing directed circuits fractionally. Combin. **15**(2), 281–288 (1995)
48. Seymour, P.: Packing circuits in eulerian digraphs. Combin. **16**(2), 223–231 (1996)
49. Wahlström, M.: Half-integrality, LP-branching and FPT algorithms. In: SODA. pp. 1762–1781 (2014)

A Preliminaries (Full)

Throughout the paper, parenthesis notation denote ordered sets, and braces notation denote unordered sets.

Graphs. We consistently use D to refer to a digraph with vertex set $V(D)$ and arc set $E(D)$. Moreover, n denotes the number of vertices of D. For any $u, v \in V(D)$, $(u, v) \in E(D)$ refers to the arc directed from u to v. We say a directed graph is *acyclic* if it has no directed cycles. For a directed graph D, by the *underlying undirected graph* of D we refer to the simple, undirected graph with vertex set $V(D)$ and arc set $E(D)$. For a directed graph D, by the *connected components* of D, we refer to the connected components of the underlying undirected graph of D. For a path P from u to v, the set of *internal vertices* of P refers to the set of vertices in the path P except for u and v. For any $X, Y \subseteq V(D)$, a path from X to Y refers to a path from some vertex in X to some vertex in Y. A collection $\mathcal{P}$ of paths from X to Y are called *internally vertex-disjoint* if all the sets of internal vertices of the paths in $\mathcal{P}$ are pairwise disjoint.For any $X \subseteq V(D) \times V(D)$, $D \cup X$ refers to the directed graph with vertex set $V(D)$ and arc set $E(D) \cup X$. Given $X \subseteq V(D)$, $D[X]$ is the subgraph of D induced by X, and $D - X$ refers to $D[V(D) \setminus X]$. For any $X, Y \subseteq V(D)$, an (X, Y)-*separator* in D is a set of vertices, say $S \subseteq V(D) \setminus (X \cup Y)$, such that there is no path from any vertex in X to any vertex in Y in $D - S$. Notice that the separator should have an empty intersection with $X \cup Y$. For a (di)graph D and $v \in V(D)$, by $N_G(v)$ we denote the set of neighbors of v in G. For $C \subseteq V(G)$, $N_G(C) = \cup_{v \in C} N(v) \setminus C$, and $N_G[C] = \cup_{v \in C} N(v) \cup C$.

Let D be a directed graph and let $V(D) = \{v_1, \ldots, v_n\}$. A *topological ordering* of D is an ordering π of $V(D)$ such that if $(v_{\pi(i)}, v_{\pi(j)}) \in E(D)$, then $i > j$. **We stress that here we suppose that no arc is directed from a vertex v to a vertex u that occurs before v, while it might be more standard to consider the symmetric condition.** It is well known that a digraph has a topological ordering if and only if it is acyclic. A set $S \subseteq V(D)$ whose deletion makes the digraph acyclic is called a *directed feedback vertex set* of D. Given a topological ordering π of D, for any $X \subseteq V(D)$, we say π_X is an ordering induced by π if the vertices of X appear in the same order in π_X and in π.

LCA-closure. For a rooted tree T and a subset $M \subseteq V(T)$, the least common ancestor-closure (LCA-closure) **LCA-closure**(M) is obtained by the following process. Initially, set $M' = M$. As long as there are vertices x and y in M' whose least common ancestor w is not in M', add w to M'. When the process terminates, output M' as the LCA-closure of M.

Proposition A.1 (Lemma 1, [27]). *Let T be a rooted tree, $M \subseteq V(T)$ and $M' =$ **LCA-closure**(M). Then $|M'| \leq 2|M|$ and for every connected component C of $T - M'$, $|N(C)| \leq 2$.*

Tree Decomposition and Treewidth. Roughly speaking, the *treewidth* of an undirected graph is a structural parameter indicating how much a graph resembles a tree. To define treewidth formally, we first need to define the concept of a tree decomposition.

Definition A.1. *A* tree decomposition *of a graph G is a pair (T, β), where T is a tree and $\beta : V(T) \to 2^{V(G)}$, that satisfies the following properties:*

1. **Edge Covering Property:** *For any edge $\{u, v\} \in E(G)$ there exists a node $t \in V(T)$ such that $x, y \in \beta(t)$, and*
2. **Connectivity Property:** *For any vertex $u \in V(G)$, the subgraph of T induced by the set $T_u = \{t \in V(T) : u \in \beta(t)\}$ is a non-empty tree.*

The width *of (T, β) is $\max_{t \in V(T)}\{|\beta(t)|\} - 1$. The* treewidth *of G, denoted by $\mathbf{tw}(G)$, is the minimum width over all tree decompositions of G.*

The following proposition gives an algorithm to compute a tree decomposition of a particular width of a graph.

Proposition A.2 (Computing a Tree decomposition, [8]). *Given a graph G and $t \in \mathbb{N}$, there is an $\mathcal{O}(t^{\mathcal{O}(t^3)} \cdot n)$-time algorithm that computes a tree decomposition (T, β) of G of treewidth at most t (if such a decomposition exists). Moreover, $|V(T)| = \mathcal{O}(|V(G)|)$.*

We now define a special kind of tree decomposition, called a *nice tree decomposition*.

Definition A.2 (Nice Tree Decomposition). *A tree decomposition (T, β) of a graph G is* nice *if T is a rooted, binary tree with root r, such that $\beta(r) = \emptyset$ and every node $t \in V(T)$ is of the one of the following types.*

- **Leaf***: t is a leaf in T and $\beta(t) = \emptyset$.*
- **Forget***: t has exactly one child, say t', and $\beta(t) = \beta(t') \setminus \{v\}$, where $v \in \beta(t')$.*
- **Introduce***: t has exactly one child, say t', $\beta(t) = \beta(t') \cup \{v\}$, where $v \notin \beta(t')$.*
- **Join***: t has exactly two children, say t_1 and t_2, and $\beta(t) = \beta(t_1) = \beta(t_2)$.*

Whenever we work with a tree decomposition of a graph, we will work with the nice tree decomposition, mainly because the tree T in the nice tree decomposition (T, β) is a rooted, binary tree. Given a tree decomposition (T, β) of a graph G, Bodlaender [8] showed how to construct a *nice* tree decomposition of G of the same width as (T, β).

Proposition A.3 ([8]). *Given a tree decomposition (T, β) of the graph G of width t, a nice tree decomposition (T', β') of G on at most $\mathcal{O}(t \cdot |V(G)|)$ nodes and also of width at most t, can be computed in time $\mathcal{O}(t^2 \cdot \max\{|V(T)|, |V(G)|\})$.*

For a directed graph D, by treewidth of D ($\mathbf{tw}(D)$) and by a (nice) tree decomposition of D, we will refer to the treewidth of the underlying undirected graph of D and a (nice) tree decomposition of the underlying undirected graph of D, respectively. For any $X \subseteq V(D)$, we say that X is a *η-treewidth modulator* if $\mathbf{tw}(D - X) \leq \eta$.

Important Separators. Roughly speaking, an important separator for sets $X, Y \subseteq V(D)$ is an (X, Y)-separator that cannot be "pushed" towards Y without increasing its size. Formally, this notion is defined as follows.

Definition A.3 (Important Separators). *Let D be a directed graph and $X, Y \subseteq V(D)$. Let $S \subseteq V(D) \setminus (X \cup Y)$ be an (X,Y)-separator and let R be the set of vertices reachable from X in $D - S$. We say that S is an* important (X,Y)-separator *if it is inclusion-wise minimal and there is no (X,Y)-separator $S' \subseteq V(D) \setminus (X \cup Y)$, such that $|S'| \leq |S|$ and $R \subset R'$, where R' is the set of vertices reachable from X in $D - S'$.*

Proposition A.4 ([14]). *Let D be a directed graph, $X, Y \subseteq V(D)$ and $k \in \mathbb{N} \cup \{0\}$. Then, D has at most 4^k important (X,Y)-separators of size at most k. In fact, the set of all important (X,Y)-separators in D can be constructed in time $\mathcal{O}(4^k \cdot k^2 \cdot (n+m))$.*

Kernelization. A *parameterization* of a problem is the association of an integer p with each input instance, which results in a *parameterized problem*. A parameterized problem is said to admit a *kernel* of size $f(p)$, for some function f that depends *only* on p, if there exists a polynomial-time algorithm, called a *kernelization algorithm*, that translates any input instance into an equivalent instance of the same problem whose size is bounded by $f(p)$ and such that the value of the parameter does not increase. In case the function f is polynomial in p, the problem is said to admit a *polynomial kernel*. While designing our kernelization algorithm, we might be able to determine whether the input instance is a YES-instance or a NO-instance. For the sake of clarity, in the first case, we simply return YES, and in second case, we simply return NO. To properly comply with the definition of a kernel, the return of YES and NO should be interpreted as the return of a trivial YES-instance and a trivial NO-instance, respectively. To design our kernelization algorithm, we rely on the notion of a *reduction rule*. A reduction rule is a polynomial-time procedure that replaces an instance (I,p) of a parameterized problem Π by a new instance (I',p') of Π. Roughly speaking, a reduction rule is useful when the instance I' is in some sense "simpler" than the instance I. In particular, it is desirable to ensure that $p' \leq p$. The rule is said to be *safe* if (I,p) is a YES-instance if and only if (I',p') is a YES-instance.

B Decomposing the Graph (Full)

Let (D,k,M) be an instance of DFVS/DFVS+Tw-η Mod. Informally, the objective of this section is to compute a decomposition of D that consists of three components: the vertex set M (modulator), a collection of vertex sets $\mathcal{Z}$ (zones), and a vertex set R (remainder). All of these sets would be pairwise disjoint. The crux is to "divide-and-conquer" D so that each zone—that is, a set $Z \in \mathcal{Z}$—would correspond to a subproblem that is easier to solve than (D,k,M) because (1) the intersection of a minimum solution with Z would be necessarily small, and (2) the interaction of Z is "well-structured" with respect to M, "limited" with respect to R , and "non-existent" with respect to any other zone.

Towards the computation of R, we need to compute three sets: (i) a solution S in $D - M$; (ii) a set F to separate vertices in M that have low-flow; (iii) an LCA-closure of bags derived from $S \cup F$. The arguments we provide along the

way to construct these sets will only partially prove that we have indeed derived the decomposition we desire. At the end, we complete the proof by focusing on the property regarding the intersection of a minimum solution with each zone. In what follows, we execute this plan.

Phase I: Solution S. The first phase of our proof is simple. Recall that the treewidth of $D - M$ is upper bounded by η. Furthermore, Bonamy et al. [10] have shown that, given a (di)graph D' of treewidth t, a smallest size set S' such that $D' - S'$ is a DAG can be computed in time $2^{\mathcal{O}(t \log t)} n^{\mathcal{O}(1)}$. In particular, this means that in time $2^{\mathcal{O}(\eta \log \eta)} n^{\mathcal{O}(1)}$—that is, polynomial time—we can compute a subset $S \subseteq V(D)$ of smallest size such that $D-(M \cup S)$ is a DAG. In case $|S| > k$, we directly conclude that (D, k, M) is a NO-instance of DFVS/DFVS+Tw-η Mod. Otherwise, we proceed as described below.

Phase II: Flow-Blocker F. The purpose of the second phase is to compute a subset $F \subseteq V(D)$ that governs the flow between *all* pairs of vertices in M. Having this subset at hand will be crucial when we later argue about the intersection of a minimum directed feedback vertex set with each zone. Specifically, for every pair of vertices in M, if we can easily separate them—we simply do that; otherwise, no solution of size at most k can separate these vertices, which is again beneficial for our arguments. Let us now proceed to the formal description of the computation of F.

Formally, for every ordered pair (u, v) of (not necessarily distinct) vertices $u, v \in M$ such that $(u, v) \notin E(D)$, we compute a subset $C_{(u,v)} \subseteq V(D) \setminus ((M \cup S) \setminus \{u, v\})$ of minimum size such that $D - ((M \cup S \cup C_{(u,v)}) \setminus \{u, v\})$ has no directed path from u to v (that consists of more than one vertex), that is, $C_{(u,v)}$ is a vertex cut. The computation of each such set $C_{(u,v)}$ can be executed in polynomial time by using, for example, Ford-Fulkerson algorithm [30]. Having all vertex cuts at hand, we define $F = \bigcup_{(u,v) \in M \times M \setminus E(D), |C_{(u,v)}| \leq k} C_{(u,v)}$. In words, we take the union of all cuts $C_{(u,v)}$ of size at most k. Note that $|F| \leq k\ell^2$.

Before we turn to further enriching the set $S \cup F$, let us formally summarize the structural properties already induced by this set on $D - (M \cup S \cup F)$. To this end, note that by the classic Menger's theorem, the size of $C_{(u,v)}$ for a pair $(u, v) \in M \times M \setminus E(D)$ equals the number of internally vertex-disjoint paths from u to v in $C_{(u,v)} \subseteq V(D) \setminus (M \cup S \cup \{u, v\})$. Thus, the correctness of our observation is an immediate consequence of the construction of S and F.

Observation B.1. *The digraph $D - (M \cup S \cup F)$ is a DAG such that for every pair $(u, v) \in M \times M \setminus E(D)$, either there is no path from u to v in the digraph $D - ((M \cup S \cup F) \setminus \{u, v\})$, or there are at least $k + 1$ internally vertex-disjoint paths from u to v in D.*

Phase III: LCA-Closure to Derive R. Having that the set $M \cup S \cup F$ is not sufficient for us—the vertices of $D - (M \cup S \cup F)$ can have many neighbors in $M \cup S \cup F$ while Observation B.1 provides us a handle only for those neighbors in M. However, we can compute a subset $R \subseteq V(D) \setminus M$ that contains $S \cup F$, such that no vertex of $D - (M \cup R)$ has many neighbors in R. In fact, we require a stronger claim—we will bound the neighborhood not only of each individual

vertex, but also of entire sets of vertices that will later be called zones. For this purpose, we rely on the fact that the treewidth of $D-M$ is at most η. Specifically, we will "highlight" a small set of bags (in a tree decomposition) that captures $S \cup F$, and since bags are separators and the size of each bag is small, we will be able to derive the desired set R.

Towards the computation of R, we first obtain a nice tree decomposition (T, β) of $D - M$ of width at most η. This step is done in time $\mathcal{O}(n)$ using the algorithms of Propositions A.2 and A.3 because $\eta = \mathcal{O}(1)$. Now, we highlight bags that capture the vertices in $S \cup F$ as follows. For every vertex $v \in S \cup F$, select (arbitrarily) a node $t_v \in V(T)$ such that $v \in \beta(t)$. Then, we define B as the set of all selected nodes, that is, $B = \{t_v : v \in S \cup F\}$. Next, keeping in mind that our eventual goal is to control sizes of neighborhoods, we define $B^\star$ as the LCA-closure of B in T. From Proposition A.1, $|B^\star| \leq 2 \cdot |B| \leq 2(|S| + |F|) \leq 2k(\ell^2 + 1)$. Having $B^\star$ at hand, we define R as the union of all bags corresponding to its vertices, that is, $R = \bigcup_{t \in B^\star} \beta(t)$. Then, the following observation is immediate.

Observation B.2. $|R| \leq 2k(\eta + 1)(\ell^2 + 1)$.

Decomposition. Having already computed R, it remains to partition $V(D) \setminus (M \cup R)$ into zones. To this end, we first define $\mathcal{C}$ as the set of connected components (subtrees) of the forest $T - B^\star$. Since (T, β) is a nice tree decomposition, the degree of every node is at most 3, which means that $|\mathcal{C}| \leq 3|B^\star| \leq 6k(\ell^2 + 1)$. Now, for every tree $C \in \mathcal{C}$, we define $Z_C = \bigcup_{t \in V(C)} \beta(t)$ and $Z_C^\star = Z_C \setminus R$. Finally, the collection of zones is given by $\mathcal{Z} = \{Z_C^\star : C \in \mathcal{C}\}$. Because (T, β) is a tree decomposition, for every vertex $v \in V(D)$, the set of nodes whose bags contain v induce a tree, and hence it is immediate that for all distinct $C, C' \in \mathcal{C}$, we have that $Z_C^\star \cap Z_{C'}^\star = \emptyset$. Furthermore, because (T, β) is a tree decomposition, for every arc $(u, v) \in E(D - M)$, there exists $t \in V(T)$ such that $u, v \in \beta(t)$, and hence it is also immediate that for all $Z_C^\star \in \mathcal{Z}$, $N(Z_C) \cap R$ is a subset of the bags of the nodes in $N(C)$ (that is, the neighbors of the nodes in C in T); because we have defined $B^\star$ as the LCA-closure of B, $|N(C)| \leq 2$ (see Proposition A.1). Specifically, this means that for all $Z_C^\star \in \mathcal{Z}$, $N(Z_C^\star) \subseteq M \cup R$ and $|N(Z_C^\star) \cap R| \leq 2(\eta + 1)$.

Let us summarize the properties of our decomposition, as well as its construction, in the following definition and lemma.

Definition B.1. *Let (D, k, M) be an instance of* DFVS/DFVS+Tw-η Mod. *A partition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$ is a* zone-decomposition *if:*

1. *$D - (M \cup R)$ is a DAG.*
2. *For all $Z \in \mathcal{Z}$, we have $N(Z) \subseteq M \cup R$, and $|N(Z) \cap R| \leq 2(\eta + 1)$.*
3. *For all $(u, v) \in M \times M \setminus E(D)$, the digraph $D - ((M \cup R) \setminus \{u, v\})$ either has at least $k + 1$ internally vertex-disjoint paths from u to v, or it has no path from u to v (that consists of more than one vertex).*

Lemma 3.1. *There is a polynomial-time algorithm that, given an instance (D, k, M) of* DFVS/DFVS+Tw-η Mod, *either correctly decides that (D, k, M) is a* NO*-instance, or constructs a zone-decomposition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$ with $|\mathcal{Z}| \leq 6k(\ell^2 + 1)$ and $|R| \leq 2(\eta + 1)k(\ell^2 + 1)$.*

Small Intersection Property. We are now ready to argue that if (D, k, M) is a YES-instance, then the size of the intersection of every minimum(-size) solution with every zone is merely a constant. Formally, we have the following lemma.

Lemma 3.2. *Let (D, k, M) be a* YES*-instance of* DFVS/DFVS+Tw-η Mod *with zone-decomposition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$. For any minimum-sized directed feedback vertex set S of D, we have $|S \cap Z| \leq |N(Z) \cap R| \leq 2(\eta+1)$ for all $Z \in \mathcal{Z}$.*

Proof. Let S be a directed feedback set of D of minimum size. Since (D, k, M) is a YES-instance, we have that $|S| \leq k$. Suppose, by way of contradiction, that there exists $Z \in \mathcal{Z}$ such that $|S \cap Z| > |N(Z) \cap R|$. Let $S^\star = (S \setminus Z) \cup (N(Z) \cap R)$. Clearly, $|S^\star| < |S|$. However, this means that $S^\star$ is not a directed feedback vertex set of D. Thus, there exists a directed cycle C in $D - S^\star$. Note that $V(C) \cap (N(Z) \cap R) = \emptyset$ because $N(Z) \cap R \subseteq S^\star$.

Since $D - S$ is a DAG and $S \setminus Z \subseteq S^\star$, we have that $V(C) \cap Z \neq \emptyset$. Moreover, $D[Z]$ is a DAG since $D - (M \cup R)$ is a DAG (by Definition B.1). Because $V(C) \cap (N(Z) \cap R) = \emptyset$ and $N(Z) \subseteq M \cup R$ (by Definition B.1), this means that $V(C) \cap (N(Z) \cap M) \neq \emptyset$. In turn, this means that the directed cycle C is of the form $C = u_1 - P_{u_1,u_2} - u_2 - P_{u_2,u_3} - \ldots - u_{r-1} - P_{u_{r-1}u_r} - u_r$ for some $r \geq 1$, such that $u_1 = u_r$, $u_1, \ldots, u_r \in N(Z) \cap M$, and for all $i \in \{1, \ldots, r\}$, $u_i - P_{u_i u_{i+1}} - u_{i+1}$ is a directed path from u_i to u_{i+1} in D whose internal vertices either all belong to Z or all belong to $V(D) \setminus (M \cup R \cup Z)$. Consider any non-empty path $P_{u_i u_{i+1}}$ where all the vertices belong to Z. By Definition B.1 (Condition 3), there are at least $k+1$ internally vertex-disjoint paths from u_i to u_{i+1}. Since $|S| \leq k$, there exists a path, say $u_i - P'_{u_i u_{i+1}} - u_{i+1}$, all of whose internal vertices do *not* belong to S. This means that we can replace (in C) each non-empty path $P_{u_i u_{i+1}}$ whose vertices belong to Z by the corresponding path $P'_{u_i u_{i+1}}$ as defined above, thereby getting a directed closed walk C' in $D - S$. As a directed closed walk contains a directed cycle, this contradicts the choice of S as a directed feedback set of D.

Important Note. From now onwards, we denote by $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$ a zone-decomposition as computed by the algorithm of Lemma 3.1. Recall that $|M| = \ell$, $|\mathcal{Z}| \leq 6k(\ell^2+1)$ and $|R| \leq 2(\eta+1)k(\ell^2+1)$, and thus to derive our polynomial kernel, it is next sufficient to upper bound the size of each set $Z \in \mathcal{Z}$ individually. For this purpose, from now onwards, we fix an arbitrary set $Z \in \mathcal{Z}$, and argue how the size of Z can be bounded. As the choice of Z is arbitrary, we can thus bound the size of every set in $\mathcal{Z}$. When we eventually formally prove Theorem 1.1, we shall zoom out of the view of a specific $Z \in \mathcal{Z}$, but until we reach this (short) proof, we suppose that Z is fixed.

C Proofs Omitted from Section 4

C.1 Proof of Lemma 4.2

We claim that $S' = S \cap Z$ is a solution to the Skew Multicut instance $(D', \mathcal{P}, k')$. Observe that the intersection of S' with the set of terminals in $\mathcal{P}$ is empty. Clearly,

$|S'| \leq k'$. Suppose, for the sake of contradiction, that S' is not a skew multicut in D' with respect to $\mathcal{P}$. Then, there exist v_i^{out} and v_j^{in}, $i < j$, such that there is a path from v_i^{out} to v_j^{in} in $D' - S'$. From the construction of D', there is a path from v_i to v_j in $D - S$. This is a contradiction because π_B is induced by some topological ordering of $D - S$, and hence there cannot be a path from v_i to v_j, $i < j$, in $D - S$. □

C.2 Proof of Lemma 4.3

Suppose, for the sake of contradiction, that S^* is not a directed feedback vertex set in D. Then, there exists a directed cycle C in $D - S^*$. Since $D[Z]$ is acyclic (from Lemma 3.1) and $S^* \setminus Z = S \setminus Z$, we have that $C \cap B \neq \emptyset$ and $C \cap Z \neq \emptyset$. Let us first consider the case where $|C \cap B| = 1$. In this case, let $C \cap B = \{v_i\}$. Since C is a cycle in $D - S^*$ and $S^* \cap Z = S'$, we observe (from the construction of D') that there is path from v_i^{out} to v_i^{in} in $D' - S'$. This is a contradiction to the assumption that S' is a skew multicut in D' with respect to $\mathcal{P}$. Henceforth, we assume that $|C \cap B| \geq 2$.

Let P be some directed path of the cycle C such that the endpoints of P are in $B = \{v_1, \ldots, v_b\}$ and all its internal vertices are in Z. Let the first vertex of P be v_i and the last vertex of P be v_j. Since S' is a skew multicut solution of the instance stated in the lemma, and because $S' \subseteq S^*$, we have that $j < i$. Since P is a path of the cycle C, there exists another path P' from v_j to v_i in $D - S^*$, such that the union of the sets of arcs of P and P' forms the cycle C. We will show that the existence of the path P' from v_j to v_i in $D - S^*$ (with $j < i$) implies that there exists a path P'' from some v_q to some v_r with $q < r$, such that either *(i)* all the internal vertices of P'' belong to $Z \setminus S^*$ (which includes the case where P'' has no internal vertices), or *(ii)* all the internal vertices of P'' belong to $(V(D) \setminus Z) \setminus S^*$. Let us first show why, to complete the proof, it suffices to prove the existence of such a path P''. In the first case, from the construction of D' and because $S^* \cap Z = S'$, we have that there is a path from v_q^{out} to v_r^{in} in $D' - S'$. Such a path cannot exist because S' is a skew multicut solution in D' with respect to $\mathcal{P}$. In the second case, since the ordering $\{v_1, \ldots, v_b\}$ is induced by some topological ordering of $D - S$ and $S^* \setminus Z = S \setminus Z$, the path P'' again should not exist. Thus, both cases lead to a contradiction.

Let us now describe how to obtain the path P'' from P'. Recall that P' is a path from v_j to v_i (with $j < i$) in $D - S^*$. Let v_r be the first internal vertex of P' such that $v_r \in B$ and $j < r$ (such a vertex exists because $v_i \in B$ and $j < i$). Now consider this sub-path of P' from v_j to v_r. Let v_q be the last vertex on this path that belongs to B and which is not equal to v_r (q could be equal to j). Now, consider the subpath of P' from v_q to v_r. We claim that this subpath is the desired path P''. Since v_q is the last vertex that belongs to B on the path from v_j to v_r that is not v_r itself, the set of internal vertices of P'' has an empty intersection with B. If there are at least two internal vertices in P'', say x, y, such that $x \in Z \setminus S^*$, $y \in (V(D) \setminus Z) \setminus S^*$ and there is a path from x to y in P'', then this path has to pass through a vertex of B (because $N(Z) \setminus S = B$ and $S^* \setminus Z = S \setminus Z$). Thus, we have reached a contradiction. The other case, where

$x \in (V(D) \setminus Z) \setminus S^*$ and $y \in Z \setminus S^*$, is symmetric. This shows that P'' must satisfy one of the conditions *(i)* and *(ii)*, and hence the proof is complete. □

C.3 Proof of Lemma 4.5

Let R be the set of vertices reachable from X in $D - S$, and denote $R_S = R \cup S$. Let $S^* \subseteq V(D)$ be such that $S^* \cap R_S \subsetneq S$, $S^* \cap Y = \emptyset$ and there is no path from $R_S \setminus S^*$ to Y in $D - S^*$. Since S is an important (X, Y)-separator of size α and $R \subsetneq R_S$, any such S^* has size at least $\alpha + 1$. From Menger's Theorem, there are at least $\alpha + 1$ internally vertex disjoint paths from $R_S \setminus S^*$ to Y, all of whose internal vertices are in $D - ((R_S \setminus S^*) \cup Y)$. Consider an arbitrary collection of exactly $\alpha + 1$ of these paths. Let Y' be the subset of Y which contains the endpoints in Y of the $\alpha + 1$ paths in this collection. Observe that these $\alpha + 1$ paths exist even in $D - (Y \setminus Y')$. Since there are $\alpha + 1$ internally vertex disjoint paths from $R_S \setminus S^*$ to Y' in $D - (Y \setminus Y')$, from Menger's Theorem, any set S^* with $S^* \cap R_S \subsetneq S$ and $S^* \cap Y' = \emptyset$, that kills all paths from $R_S \setminus S^*$ to Y' in $D - (Y \setminus Y')$, is of size $\alpha + 1$.

Suppose, for the sake of contradiction, that S is not an important (X, Y')-separator in $D - (Y \setminus Y')$. Then there exists S' such that *(i)* $|S'| \leq |S| = \alpha$, *(ii)* S' is an (X, Y')-separator in $D - (Y \setminus Y')$, and *(iii)* $R \subsetneq R'$ (where R is the set of vertices reachable from X in $D - S$ and R' is the set of vertices reachable from X in $D - S'$). Note that $S' \cap R_S \subsetneq S$ (because $R \subsetneq R'$ and $S \neq S'$) and $S' \cap Y' = \emptyset$. Also, S' kills all paths from $R_S \setminus S'$ to Y' in $D - (Y \setminus Y')$. But this is a contradiction because we have already proved that any such S' has size at least $\alpha + 1$. □

C.4 Proof of Lemma 4.4

Description of the Algorithm. Our algorithm is a branching (recursive) algorithm. (Here, by branching tree we refer to the tree whose root is the initial call to the algorithm, and which described the relationships between the calls to the algorithm as excepted, that is, the children of a node correspond to the recursive calls made by that node.) Each node of the branching tree corresponds to a triple $(D, \mathcal{Q}, k)$ where D is a directed graph, $\mathcal{Q}$ is an *unordered* set of vertex pairs in D, and k is an integer. The *measure* to analyze the size of the branching tree is k. The algorithm initializes $\mathit{\Gamma}_{\mathrm{SMC}} = \emptyset$ and updates $\mathit{\Gamma}_{\mathrm{SMC}}$ at the end of each branch. The base case occurs when $k \leq 0$. Whenever $k > 0$, the algorithm branches into the branches described below and updates $\mathit{\Gamma}_{\mathrm{SMC}}$.

Consider a node of the branching tree labeled by $(D, \mathcal{Q}, k)$. We fix some notation before we proceed further. Let $\mathcal{Q} = \{(s_i, t_i) : i \in \{1, \ldots, p\}\}$. By T we denote the set $T = \{t_i : i \in \{1, \ldots, p\}\}$. For any $P' \subseteq \{1, \ldots, p\}$, by $T_{P'}$ we denote the set $T_{P'} = \{t_i : i \in P'\}$, and by $\mathcal{Q}_{P'}$ we denote $\mathcal{Q}_{P'} = \{(s_i, t_i) : i \in P'\}$. We will now describe the branches of the algorithm at the node $(D, \mathcal{Q}, k)$ of the branching tree. For each $i \in \{1, \ldots, p\}$, for each $P' \subseteq \{1, \ldots, p\}$ of size at most $k + 1$ and for each $I \neq \emptyset$ which is an important $(s_i, T_{P'})$-separator in $D - (\mathcal{Q} \setminus \mathcal{Q}_{P' \cup \{i\}})$ of size at most k, there is a branch that corresponds to the

triple (i, P', I). A branch (i, P', I) at a node $(D, \mathcal{Q}, k)$ results in a node that corresponds to the triple $(D - I, \mathcal{Q} \setminus \{(s_i, t_i)\}, k - |I|)$. Also, at the end of this branch, Γ_{SMC} is updated as $\Gamma_{\text{SMC}} = \Gamma_{\text{SMC}} \cup I$. Observe that the measure in each branch drops by at least 1 because $I \neq \emptyset$.

The size of the branching tree. Since the number of important separators of size at most k is at most 4^k (Proposition A.4), observe that at any node of the branching tree, the number of branches, $w \leq p \cdot \sum_{i=0}^{k+1} \binom{p}{i} \cdot 4^k \leq (k+1) \cdot p^{k+2} \cdot 4^k$. Since the measure of the branching tree is k, the branching algorithm halts when $k \leq 0$ and in each branch the measure strictly decreases, the depth of the branching tree is at most k. Thus, the number of nodes in the branching tree is at most $\sum_{i \in \{1,\ldots,k\}} w^i \leq k \cdot w^k \leq k(k+1)^k \cdot p^{k(k+2)} \cdot 4^{k^2}$. Since, at each node of the branching tree, the size of Γ_{SMC} increases by at most k, $|\Gamma_{\text{SMC}}| = k^2(k+1)^k \cdot p^{k(k+2)} \cdot 4^{k^2}$.

Running Time. At any node $(D, \mathcal{Q}, k)$, for each $i \in \{1, \ldots, p\}$ and $P' \subseteq \{1, \ldots, p\}$ of size at most $k+1$, we need to compute the set of appropriate important separators of size at most k. From Proposition A.4, this takes time, say $t = p \cdot \sum_{i=0}^{k+1} \binom{p}{i} \cdot \mathcal{O}(4^k \cdot k^2 \cdot (n+m)) = \mathcal{O}(p^{k+2} \cdot 4^k \cdot k^3 \cdot (n+m))$. Recall that at any node of the branching tree, the number of branches is $w \leq (k+1) \cdot p^{k+2} \cdot 4^k$. Let $T(k)$ be the time taken by the algorithm when the input triple $(D, \mathcal{Q}, k)$ has measure k. Then, we have the following recurrence relation: if $k \leq 0$, $T(k) = 1$; otherwise, $T(k) \leq wT(k-1) + t$. Solving this recurrence, we get that $T(k) \leq w^k \cdot t = \mathcal{O}(p^{(k+1)(k+2)} \cdot 4^{k(k+1)} \cdot (k+1)^k \cdot k^3 \cdot (n+m))$.

Correctness. Suppose that the algorithm takes as initial input the instance $(D, \mathcal{Q}, k)$. Consider an instance $(D, \mathcal{P}^*, k^*)$ in $\mathcal{F}_{\text{SMC}}(D, \mathcal{P}, k)$ which is a YES-instance of Skew Multicut. Without loss of generality, let $\mathcal{P}^* = ((s_i, t_i) : i \in \{1, \ldots, p^*\})$. We need to show that there exists a skew multicut solution of the instance $(D, \mathcal{P}^*, k^*)$ contained inside Γ_{SMC}. Let μ be the depth of the algorithm. For any $d \leq \mu$, let $\Gamma_{\text{SMC}d}$ be the partial set (subset of Γ_{SMC}) constructed by the algorithm at the end of the last branch at depth d of its branching tree. We will now prove the following claim by induction on d, and then later show how the correctness of the algorithm follows from this claim.

Claim. For any $d \leq \mu$, there exists $S_d \subseteq \Gamma_{\text{SMC}d}$ with the following properties.

1. There is a solution S to the instance $(D, \mathcal{P}^*, k^*)$ of Skew Multicut such that $S_d \subseteq S$.
2. For all $i \in \{1, \ldots, d\}$, $j \in \{1, \ldots, p^*\}$ with $j \geq i$, there is no path from s_i to t_j in $D - S_d$.
3. There is a path from the root of the branching tree to one of its nodes at depth d such that S_d is the union of the sets added to Γ_{SMC} at each node along this path.

Proof. We will prove this claim by induction on d, that is, the depth of the branching tree.

Base Case. Suppose $d = 1$. Since $(D, \mathcal{P}^*, k^*)$ is a YES-instance of Skew Multicut, from Proposition 4.1, there exists an important $(s_1, \{t_1, \ldots, t_{p^*}\})$-separator,

say I, of size at most k^* ($\leq k$) in D such that there is no path in $D - I$ from s_1 to t_j, for all $j \in P^*$, and there exists a solution S to the instance $(D, \mathcal{P}^*, k^*)$ such that $I \subseteq S$. We will now prove that $I \subseteq \Gamma_{\text{SMC}1}$. From Lemma 4.5, there exists $T \subseteq \{t_1, \ldots, t_{p^*}\}$ such that $|T| \leq k+1$ and I is an important (s_i, T)-separator in $D - (\mathcal{P}^* \setminus \{(s_i, t_i) : t_i \in T\})$ of size at most k. Consider the branch that corresponds to the triple $(i, \{j : t_j \in T\}, I)$. At the end of this branch, Γ_{SMC} is updated as $\Gamma_{\text{SMC}} = \Gamma_{\text{SMC}} \cup I$. This proves the base case.

Inductive Step: Suppose that the claim holds for $d-1$. By the induction hypothesis, there exists a set $S_{d-1} \subseteq \Gamma_{\text{SMC}d-1}$ with the properties mentioned in Claim C.4. Let $(D', \mathcal{Q}', k')$ be the triple corresponding to the node at level $d-1$ which is the end-point of the path in the branching tree along which S_{d-1} is constructed. Observe that $\mathcal{Q}' = \{(s_i, t_i) : i \in \{1, \ldots, p^*\}, i \geq d\}$, $D' = D - S_{d-1}$ and $k' = k - |S_{d-1}|$. We now prove the three properties stated in the claim.

Property 1. Let S' be a solution to the Skew Multicut instance $(D', ((s_i, t_i) : i \in \{1, \ldots, p^*\}, i \geq d), k')$. We first claim that $S_{d-1} \cup S'$ is a solution to the Skew Multicut instance $(D, \mathcal{P}^*, k^*)$. From the induction hypothesis, for all $i \in \{1, \ldots, d-1\}$, $j \in \{1, \ldots, p^*\}$ with $j \geq i$, there is no path from s_i to t_j in $D - S_{d-1}$. Since S' is a skew multicut solution to $(D', ((s_i, t_i) : i \in \{1, \ldots, p^*\}, i \geq d), k')$ and $D' = D - S_{d-1}$, for all $i, j \in \{1, \ldots, p^*\}$ with $j \geq i$, there is no path from s_i to t_j in $D - (S_{d-1} \cup S')$. We now need to show that $|S_{d-1} \cup S'| \leq k^*$, that is, $|S_{d-1}| + k' \leq k^*$. To prove this, note that, from the induction hypothesis, there exists a solution, say S, of $(D, \mathcal{P}^*, k^*)$, such that $S_{d_1} \subseteq S$. Since $k' = k - |S_{d-1}| \geq k^* - |S_{d-1}|$, we conclude that $k^* \geq k' + |S_{d-1}|$.

Thus, we have proved that there exists a solution, say S, to the Skew Multicut instance $(D, \mathcal{P}^*, k^*)$ such that $S = S_{d-1} \cup S'$, where S' is any solution to the Skew Multicut instance $(D', ((s_i, t_i) : i \in \{1, \ldots, p^*\}, i \geq d), k')$. From Proposition 4.1, we know that there exists a skew multicut solution S' for $(D', ((s_i, t_i) : i \in \{1, \ldots, p^*\}, i \geq d), k')$ which contains an important $(s_i, \{t_i : i \in \{1, \ldots, p^*\}, i \geq d\})$-separator, say I, in D'. Let S' be such a solution to $(D', ((s_i, t_i) : i \in \{1, \ldots, p^*\}, i \geq d), k')$. From Lemma 4.5, there exists $T \subseteq \{t_d \ldots, t_{p^*}\}$ of size at most $k+1$ such that I is an important (s_d, T)-separator in $D' - (\mathcal{Q}' \setminus \{(s_i, t_i) : i \in T\})$ of size at most k. Consider the branch at $(D', \mathcal{Q}', k')$ that corresponds to the triple $(d, \{(s_i, t_i) : i \in T\}, I)$. At the end of this branch, I is added to Γ_{SMC}. Let $S_d = S_{d-1} \cup I$.

Since $S_d \cup S'$ is a solution to the Skew Multicut instance $(D, \mathcal{P}^*, k^*)$, where S' is any solution of the Skew Multicut instance $(D', ((s_i, t_i) : i \in \{1, \ldots, p^*\}, i \geq d), k')$, and there exists a solution S' to the Skew Multicut instance $(D', ((s_i, t_i) : i \in \{1, \ldots, p^*\}, i \geq d), k')$ such that $I \subseteq S'$, we conclude that there exists a solution S to the Skew Multicut instance $(D, \mathcal{P}^*, k^*)$, such that $S_{d-1} \cup I \subseteq S$. Since $S_d = S_{d-1} \cup I$, we conclude that $S_d \subseteq S$.

Property 2. Since, for any $i \in \{1, \ldots, d-1\}$ and for all $j \in \{1, \ldots, p^*\}$ with $j \geq d-1$ there are no paths from s_i to t_j in $D - S_{d-1}$ (from induction hypothesis), and I is an $(s_d, \{t_d, \ldots, t_{p^*}\})$-separator in $D - S_{d-1}$, we have that for all $i \in \{1, \ldots, d\}$ and for all $j \in \{1, \ldots, p^*\}$ with $j \geq d$, there is no path from s_i to t_j in $D - S_d$.

Property 3. Consider the path in the branching tree along which S_{d-1} is constructed. Such a path exists because of induction hypothesis. Since $(D', \mathcal{Q}', k')$ is the last node on this path, the set I is added to Γ_{SMC} at one of the branches from $(D', \mathcal{Q}', k')$, we have that S_d is the union of the sets added to Γ_{SMC} along a path in the branching tree. This concludes the proof of the claim.

If μ is the depth of the branching tree of the algorithm, consider the set S_μ as defined in Claim C.4. Consider the path in the branching tree that corresponds to the construction of S_μ. Since μ is the depth of the branching tree of the algorithm and the algorithm terminates only when $k^* \leq 0$, $|S_\mu| \geq k^*$. From Claim C.4, there exists a solution S to the instance $(D^*, \mathcal{P}^*, k^*)$ such that $S_\mu \subseteq S$. Since $|S| \leq k^*$, $|S_\mu| \geq k^*$ and $S_\mu \subseteq S$, we have that $S_\mu = S$. Since $S_\mu \subseteq \Gamma_{\text{SMC}}$, we have that $S \subseteq \Gamma_{\text{SMC}}$. □

C.5 Proof of Lemma 4.1

Recall that the input for the algorithm is a digraph D, an integer k and a set Z. We run the algorithm of Lemma 4.4 on the input $(D[N[Z]]_{\dagger N(Z)}, \{(v^{out}, v^{in}) : v \in N(Z)\}, 2(\eta+1))$. Let Γ_{SMC} be the output of this algorithm. We claim that it is safe to set $\Gamma_{\text{DFVS}} = \Gamma_{\text{SMC}}$, that is, Γ_{SMC} is a k-DFVS representative in Z. First observe that, from Lemma 4.4, $|\Gamma_{\text{SMC}}| = |N(Z)|^{\mathcal{O}(\eta^2)} = |M \cup R|^{\mathcal{O}(\eta^2)} = (k \cdot \ell)^{\mathcal{O}(\eta^2)}$. Next, suppose that (D, k, M) is a YES-instance of DFVS/DFVS+Tw-η Mod. We now show that there is a solution S^* such that $S^* \cap Z \subseteq \Gamma_{\text{SMC}}$. Since (D, k, M) is a YES instance of DFVS/DFVS+Tw-η Mod, from Lemma 3.2, there exists a solution S such that $|S \cap Z| \leq 2(\eta+1)$. If $S \cap Z \subseteq \Gamma_{\text{SMC}}$, then $S = S^*$. Otherwise, let $B = N(Z) \setminus S$ and let $\pi_B = \{v_1, \ldots, v_b\}$ be an ordering of the vertices of B induced by some topological ordering of $D - S$. Consider the instance $(D[Z \cup B]_{\dagger B}, \{(v_i^{out}, v_i^{in}) : i \in \{1, \ldots, b\}\}), 2(\eta+1))$ of Skew Multicut. If this is a YES-instance, then there exists $S' \subseteq Z$ such that S' is a solution of the instance $(D[Z \cup B]_{\dagger B}, \{(v_i^{out}, v_i^{in}) : i \in \{1, \ldots, b\}\}), 2(\eta+1))$ and $S' \subseteq \Gamma_{\text{SMC}}$.

Since $|S \cap Z| \leq 2(\eta+1)$, from Lemmas 4.2 and 4.3, $(D[Z \cup B]_{\dagger B}, \{(v_i^{out}, v_i^{in}) : i \in \{1, \ldots, b\}\}), 2(\eta+1))$ is indeed a YES-instance of Skew Multicut and $S^* = (S \setminus Z) \cup S'$ is a solution to the instance (D, k, M) of directed feedback vertex set. Thus, we conclude that $S^* \cap Z \subseteq \Gamma_{\text{SMC}}$. □

D Reduction Rules (Continued)

In the upcoming subsections, we give several reduction rules. These rules are *applied in the given order*, exhaustively. That is, at any point of time we apply the lowest numbered reduction rule that is applicable. In all our reduction rules, we reduce an instance (D, k, M) to (D', k', M'), $M' \subseteq M$ and $k' \leq k$.

D.1 Limiting the Interaction Between M and Z

Recall that our goal is to design reduction rules that eventually remove all arcs between M and $Z \setminus \Gamma_{\text{DFVS}}$. This is achieved at the cost of adding arcs among the

vertices of M, and between vertices in M and vertices in Γ_{DFVS}. The first set of reduction rules ensures that the addition of this set of arcs is safe. Using this, we then design reduction rules that delete the arcs between M and $Z \setminus \Gamma_{\text{DFVS}}$.

Reduction Rule D.1. *If there exist $u, v \in N(Z) \cap M$ (here, u may be the same vertex as v) such that there is a path from u to v, all of whose internal vertices are in $Z \setminus \Gamma_{\text{DFVS}}$, then add the arc (u, v) to D, if it does not already exist. That is, reduce the instance (D, k, M) to $(D \cup \{(u, v)\}, k, M)$.*

Lemma D.1. *Reduction Rule D.1 is safe.*

Proof. The backward direction is trivial. For ease of notation, let us denote $D \cup \{(u, v)\}$ by D'. For the forward direction, first observe that M is also an η-treewidth modulator in the graph D', that is, $\mathbf{tw}(D' - M) \leq \eta$, because $D' - M = D - M$. If (D, k, M) is a YES instance, then from Lemma 4.1, there exists a directed feedback vertex set S in D of size at most k such that $S \cap Z \subseteq \Gamma_{\text{DFVS}}$. We claim that S is also a directed feedback vertex set in D'. Suppose not. Then there is a cycle C in $D' - S$ that uses the arc (u, v). Since there is a path from u to v, all of whose internal vertices are in $Z \setminus \Gamma_{\text{DFVS}}$ and the arc (u, v) is not present in D, there are at least $k+1$ internally vertex disjoint paths from u to v in D (from Lemma 3.1). Since $|S| \leq k$, there is a path from u to v in $D - S$. Replacing the arc (u, v) by this path in C, we get a closed walk in $D - S$, which contradicts the fact that S is a directed feedback vertex set in D.

Observe that when u and v are the same vertex in Reduction Rule D.1, the resulting digraph will have self-loops. The next reduction rule removes all self-loops in the digraph. It is easy to see that Reduction Rule D.2 is safe.

Reduction Rule D.2. *If there exists $v \in V(D)$ with a self-loop, then delete v and reduce k by 1. That is, reduce the instance (D, k, M) to $(D - \{v\}, k - 1, M \setminus \{v\})$.*

The next reduction rule gives a sufficient condition in which we can add an arc between a vertex in M and a vertex in Γ_{DFVS}.

Reduction Rule D.3. *Suppose that there exist $u \in M$ and $v \in Z \setminus \Gamma_{\text{DFVS}}$ such that $(u, v) \in E(D)$, and $x \in \Gamma_{\text{DFVS}}$ such that there exists a path from v to x, all of whose internal vertices are in $Z \setminus \Gamma_{\text{DFVS}}$. Let $D' = D \cup \{(u, x)\}$. Reduce (D, k, M) to (D', k, M).*

Lemma D.2. *Reduction Rule D.3 is safe.*

Proof. The backward direction is trivial. For the forward direction, first observe that, M is also a treewidth modulator for the graph D', that is, $\mathbf{tw}(D' - M) \leq \eta$, because $D' - M = D - M$. If (D, k, M) is a YES instance, from Lemma 4.1, there exists a directed feedback vertex set S in D of size at most k such that $S \cap Z \subseteq \Gamma_{\text{DFVS}}$. We claim that S is also a directed feedback vertex set in D'. Suppose not. Then there is a cycle C in $D' - S$ that uses the arc (u, x). Consider

the path from u to x in D, all of whose internal vertices are in $Z \setminus \Gamma_{\text{DFVS}}$. From the choice of S, such a path exists in $D - S$. Replacing the arc (u, x) in C by this path we get a closed walk in $D - S$, which contradicts the fact that S is a directed feedback vertex set in D.

Reduction Rule D.4. *Suppose there exist $u \in M$ and $v \in Z \setminus \Gamma_{\text{DFVS}}$ such that $(v, u) \in E(D)$, and $x \in \Gamma_{\text{DFVS}}$ such that there is a path from x to v, all of whose internal vertices are in $Z \setminus \Gamma_{\text{DFVS}}$. Let $D' = D \cup \{(x, u)\}$. Reduce (D, k, M) to (D', k, M).*

The proof of safeness of Reduction Rule D.4 is analogous to the proof of safeness of Reduction Rule D.3 and thus omitted.

The next reduction rules delete arcs between M and $Z \setminus \Gamma_{\text{DFVS}}$.

Reduction Rule D.5. *If there exists $u \in M$ and $v \in Z \setminus \Gamma_{\text{DFVS}}$ such that $(u, v) \in E(D)$, then reduce (D, k, M) to (D', k, M), where $D' = D \setminus \{(u, v)\}$.*

Lemma D.3. *Reduction Rule D.5 is safe.*

Proof. The forward direction is trivial. For the backward direction, let S be a directed feedback vertex set of D' of size at most k. We claim that S is also a directed feedback vertex set in D. Suppose not. Then there is a cycle C in $D - S$ that uses the arc (u, v). Since C uses the arc (u, v), C passes through Z. Let P be the unique path u to some vertex $w \in N(Z)$, all of whose internal vertices are in C and none in $N(Z)$. Observe that $u \in N(Z)$. If all the internal vertices of P are in $Z \setminus \Gamma_{\text{DFVS}}$, then, since Reduction Rule D.1 has been applied, there is an arc $(u, w) \in E(D)$. Consider the closed walk C' obtained from C after replacing the path P by (u, w). Since C' is a closed walk in $D' - S$, this contradicts that S is a directed feedback vertex set in D'. In the other case, there exists an internal vertex of P that belongs to Γ_{DFVS}. Let x be the first vertex of P that belongs to Γ_{DFVS}. Since Reduction Rule D.3 has been applied, there exists an arc $(u, x) \in E(D)$ and hence in $E(D' - S)$. Consider the sub-path P' of P from u to x. Replacing P' in C by the arc (u, x), we get a closed walk in $D' - S$, which contradicts that S is a directed feedback vertex set in D'.

Reduction Rule D.6. *If there exists $u \in M$ and $v \in Z \setminus \Gamma_{\text{DFVS}}$ such that $(v, u) \in E(D)$, then reduce (D, k, M) to (D', k, M), where $D' = D \setminus \{(v, u)\}$.*

The proof of safeness of Reduction Rule D.6 is symmetric to the proof of safeness of Reduction Rule D.5 and thus omitted. Observe that each of the Reduction Rules D.1-D.6 can be applied in polynomial time (by using the algorithms for computing shortest path in directed graphs). We conclude this subsection with the following observation.

Observation D.1. *If Reduction Rules D.5 and D.6 are no longer applicable, then $N(Z \setminus \Gamma_{\text{DFVS}}) \cap M = \emptyset$.*

D.2 Protrusion Replacement and Proof of Theorem 1.1

Recall that the goal now is to slice up $Z \setminus \Gamma_{\text{DFVS}}$ into pieces each of which has treewidth η and $\mathcal{O}(\eta)$ neighbors outside it. Such a slice is what we call an $\mathcal{O}(\eta)$-protrusion. For any positive integer r, the formal definition of an r-protrusion is given below.

Definition D.1 (r-Protrusion). *For any $r \in \mathbb{Z}^+$, an* r-protrusion *in a graph D is a set of vertices $X \subseteq V(D)$ such that $|N(X)| \leq r$ and $\mathbf{tw}(D[X]) \leq r$.*

We need the following lemma which says that if there exists a large enough protrusion, then it can be replaced with an equivalent one of constant size in linear time. As the proof of the lemma requires the introduction of several concepts only relevant to it, we present this proof separately in Appendix E.

Lemma D.4. *For every $t \in \mathbb{Z}^+$, there exists $c \in \mathbb{Z}^+$ (depending only on t), and an algorithm that, given an instance (D, k, M) of* DFVS/DFVS+Tw-η Mod *and a t-protrusion X in D such that $|X| > c$ and $X \cap M = \emptyset$, outputs, in time $\mathcal{O}(|X|)$, a digraph D' and integer k', such that:*

1. *$|V(D')| < |V(D)|$ and $k' \leq k$,*
2. *D' is obtained from D by deleting some vertices/arcs of X and/or contracting some edges of X, and*
3. *D has a directed feedback vertex set of size at most k if and only if D' has a directed feedback vertex set of size at most k'.*

Observe that the graph D' in Lemma D.4 is a minor of D (for a *directed* graph D, by a minor of D we refers to a minor in the underlying undirected graph of D). In fact, if M is a treewidth-η modulator in D, then M is also a treewidth-η modulator in D'. Note that Lemma D.4 is not replacing a big enough protrusion by any arbitrary smaller protrusion, rather the replacement is such that the resulting graph is a minor of the original graph. This is necessary to ensure that the size of the treewidth-η modulator (a component in the parameter of our problem) does not increase after this replacement.

Henceforth, c is the constant of Lemma D.4 when $t = 4(\eta + 1)$. For a given set $B \subseteq V(D)$, define X_B to be the set of vertices that are either in B or in some connected component of $D - B$ that has treewidth at most η. From Lemma 3.1, $\mathbf{tw}(D[Z]) \leq \eta$. Let (T, β) be a nice tree decomposition of $D[Z]$, computed using the algorithms of Propositions A.2 and A.3. Let $B_{\Gamma_{\text{DFVS}}} \subseteq V(T)$ be such that for each $u \in \Gamma_{\text{DFVS}}$, there exists $t \in B_{\Gamma_{\text{DFVS}}}$ such that $u \in \beta(t)$. Observe that $|B_{\Gamma_{\text{DFVS}}}| \leq |\Gamma_{\text{DFVS}}|$. Let $B'_{\Gamma_{\text{DFVS}}}$ be the LCA-closure of $B_{\Gamma_{\text{DFVS}}}$ in T. From Proposition A.1, $|B'_{\Gamma_{\text{DFVS}}}| \leq 2|B_{\Gamma_{\text{DFVS}}}| \leq 2|\Gamma_{\text{DFVS}}|$. Let $\mathcal{C}$ be the collection of connected components of $T - B'_{\Gamma_{\text{DFVS}}}$. Since (T, β) is a nice tree decomposition, for any $t \in V(T)$, the degree of t in T is at most 3. Hence, $|\mathcal{C}| \leq 3 \cdot |B'_{\Gamma_{\text{DFVS}}}| \leq 6|\Gamma_{\text{DFVS}}|$. Let $\widetilde{\Gamma}_{\text{DFVS}} = \bigcup_{t \in B'_{\Gamma_{\text{DFVS}}}} \beta(t)$. Observe that $\Gamma_{\text{DFVS}} \subseteq \widetilde{\Gamma}_{\text{DFVS}}$. For each $C_i \in \mathcal{C}$, define $U_i \subseteq Z \setminus \Gamma_{\text{DFVS}}$ as $U_i = \bigcup_{t \in C_i} \beta(t) \setminus \widetilde{\Gamma}_{\text{DFVS}}$. Since (T, β) is a tree decomposition of width at most η, $|\widetilde{\Gamma}_{\text{DFVS}}| \leq (\eta+1) \cdot |B'_{\Gamma_{\text{DFVS}}}| \leq 2(\eta+1) \cdot |\Gamma_{\text{DFVS}}|$.

Observe that $Z = \widetilde{\Gamma}_{\text{DFVS}} \uplus \bigcup_{i \in \{1,\ldots,|\mathcal{C}|\}} U_i$. From Proposition A.1, and the edge covering and connectivity properties of a tree decomposition, we have that for each $U_i \in \mathcal{C}$, $|N(U_i) \cap Z| \leq 2(\eta+1)$. Since $|N(Z) \cap R| \leq 2(\eta+1)$ (from Lemma 3.1), $|N(U_i) \cap R| \leq 2(\eta+1)$. Furthermore, from Observation D.1, $|N(U_i) \cap M| = 0$. Since $N(Z) \subseteq M \cup R$ (from Lemma 3.1) and $U_i \subseteq Z$, we conclude that $|N(U_i)| \leq 4(\eta+1)$. Also, since $\mathbf{tw}(D[Z]) \leq \eta$ and $U_i \subseteq Z$, we conclude that U_i is a $4(\eta+1)$-protrusion in D.

Reduction Rule D.7 (Protrusion Replacement Reduction Rule). *If there exists $i \in \{1,\ldots,|\mathcal{C}|\}$, such that $|U_i| > c$, then apply the algorithm of Lemma D.4 on the instance (D,k,M) along with the $4(\eta+1)$-protrusion U_i. Let D', k' be the digraph and integer, respectively, outputted by this algorithm. Then reduce (D,k,M) to (D',k',M).*

The safeness of Reduction Rule D.7 follows from Lemma D.4. Also, from the construction of U_i and Lemma D.4, Reduction Rule D.7 can be applied in polynomial time.

Lemma D.5. *When none of the reduction rules is applicable, $|Z| = (k\ell)^{\mathcal{O}(1)}$.*

Proof. Since $Z = \widetilde{\Gamma}_{\text{DFVS}} \uplus \bigcup_{i \in \{1,\ldots,|\mathcal{C}|\}} U_i$, $|\mathcal{C}| \leq 6|\Gamma_{\text{DFVS}}|$, $|\widetilde{\Gamma}_{\text{DFVS}}| \leq 2(\eta+1) \cdot |\Gamma_{\text{DFVS}}|$ and $|U_i| \leq c$ (after the application of Reduction Rule D.7), we have that $|Z| = (k\ell)^{\mathcal{O}(\eta^2)}$ (from Lemma 4.1).

Now that we have bounded the size of each zone Z, we are ready to prove Theorem 1.1.

Proof (Proof of Theorem 1.1). Let (D,k,M) be an instance to DFVS/DFVS+Tw-η Mod. Now we apply Lemma 3.1 and either correctly decide that (D,k,M) is a NO-instance, or construct a zone-decomposition $V(D) = M \uplus R \uplus (\biguplus_{Z \in \mathcal{Z}} Z)$ with $|\mathcal{Z}| \leq 6k(\ell^2+1)$ and $|R| \leq 2(\eta+1)k(\ell^2+1)$. For each $Z \in \mathcal{Z}$, we do the following. We compute a k-DFVS representative in Z and call it Γ_{DFVS}. We then apply all the reduction rules of Section 5 exhaustively. From Lemma D.5, when none of the reduction rules are applicable $|Z| = (k\ell)^{\mathcal{O}(\eta^2)}$. Thus, for each $Z \in \mathcal{Z}$, when none of the reduction rules are applicable, $|V(D)| \leq |M| + |R| + (k\ell)^{\mathcal{O}(\eta^2)} \cdot |\mathcal{Z}|$. Thus, from Lemma 3.1, $|V(D)| = (k \cdot \ell)^{\mathcal{O}(\eta^2)}$.

E Proof of Lemma D.4

In this section we give a proof of Lemma D.4. We start with some notations and definitions. The next two subsections are taken from [9].

E.1 Counting Monadic Second Order Logic and its properties

The syntax of Monadic Second Order Logic (MSO) of graphs includes the logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, variables for vertices, edges, sets of vertices, and sets of edges, the quantifiers $\forall$, $\exists$ that can be applied to these variables, and the following five binary relations:

1. $u \in U$ where u is a vertex variable and U is a vertex set variable;
2. $d \in D$ where d is an edge variable and D is an edge set variable;
3. $\mathbf{inc}(d, u)$, where d is an edge variable, u is a vertex variable, and the interpretation is that the edge d is incident with the vertex u;
4. $\mathbf{adj}(u, v)$, where u and v are vertex variables and the interpretation is that u and v are adjacent;
5. equality of variables representing vertices, edges, sets of vertices, and sets of edges.

In addition to the usual features of monadic second-order logic, if we have atomic sentences testing whether the cardinality of a set is equal to q modulo r, where q and r are integers such that $0 \leq q < r$ and $r \geq 2$, then this extension of the MSO is called the *counting monadic second-order logic*. Thus CMSO is MSO with the following atomic sentence for a set S:

$\mathbf{card}_{q,r}(S) = \mathbf{true}$ if and only if $|S| \equiv q \pmod{r}$.

The p-MIN-CMSO problem defined by formula ψ is denoted by p-MIN-CMSO$[\psi]$ and defined as follows.

p-MIN-CMSO$[\psi]$ *Input*: A graph G (or digraph D) and an integer k *Parameter:* k *Question*: Is there a subset $S \subseteq V(G)$ such that $\lvert S\rvert \leq k$ and $(G, S) \models \psi$?

In other words, p-MIN-CMSO$[\psi]$ is a subset Π of $\Sigma^* \times \mathbb{Z}$ where for every $(x, k) \in \Sigma^* \times \mathbb{Z}^+$, $(x, k) \in \Pi$ if and only if there exists a set $S \subseteq V$ where $|S| \leq k$ such that the graph G encoded by x together with S satisfy ψ, i.e., $(G, S) \models \psi$. For $(x, k) \in \Sigma^* \times \mathbb{Z}^-$ we know that $(x, k) \notin \Pi$.

E.2 Boundaried Graphs

Here we define the notion of *boundaried graphs* and various operations on them.

Definition E.1. [**Boundaried Graphs**] *A* boundaried graph *is a graph G with a set $B \subseteq V(G)$ of distinguished vertices and an injective labelling λ from B to the set $\mathbb{Z}^+$. The set B is called the* boundary *of G and the vertices in B are called* boundary vertices *or* terminals*. Given a boundaried graph G, we denote its boundary by $\delta(G)$, we denote its labelling by λ_G, and we define its* label set *by $\Lambda(G) = \{\lambda_G(v) \mid v \in \delta(G)\}$. Given a finite set $I \subseteq \mathbb{Z}^+$, we define $\mathcal{F}_I$ to denote the class of all boundaried graphs whose label set is I. Similarly, we define $\mathcal{F}_{\subseteq I} = \bigcup_{I' \subseteq I} \mathcal{F}_{I'}$. We also denote by $\mathcal{F}$ the class of all boundaried graphs. Finally we say that a boundaried graph is a* t-boundaried *graph if $\Lambda(G) \subseteq \{1, \ldots, t\}$.*

Definition E.2. [**Gluing by $\oplus$**] *Let G_1 and G_2 be two boundaried graphs. We denote by $G_1 \oplus G_2$ the graph (not boundaried) obtained by taking the disjoint union of G_1 and G_2 and identifying equally-labeled vertices of the boundaries of G_1 and G_2. In $G_1 \oplus G_2$ there is an edge between two labeled vertices if there is either an edge between them in G_1 or in G_2.*

Definition E.3. *Let $G = G_1 \oplus G_2$ where G_1 and G_2 are boundaried graphs. We define the* glued *set of G_i as the set $B_i = \lambda_{G_i}^{-1}(\Lambda(G_1) \cap \Lambda(G_2)), i = 1, 2$. For a vertex $v \in V(G_1)$ we define its* heir *$h(v)$ in G as follows: if $v \notin B_1$ then $h(v) = v$, otherwise $h(v)$ is the result of the identification of v with an equally labeled vertex in G_2. The* heir *of a vertex in G_2 is defined symmetrically. The* common boundary *of G_1 and G_2 in G is equal to $h(B_1) = h(B_2)$ where the evaluation of h on vertex sets is defined in the obvious way. The* heir *of an edge $\{u, v\} \in E(G_i)$ is the edge $\{h(u), h(v)\}$ in G.*

Let $\mathcal{G}$ be a class of (not boundaried) graphs. By slightly abusing notation we say that a boundaried graph *belongs to a graph class* $\mathcal{G}$ if the underlying graph belongs to $\mathcal{G}$.

E.3 Finite Integer Index

Definition E.4. [**Canonical equivalence on boundaried graphs.**] *Let Π be a parameterized graph problem whose instances are pairs of the form (G, k). Given two boundaried graphs $G_1, G_2 \in \mathcal{F}$, we say that $G_1 \equiv_\Pi G_2$ if $\Lambda(G_1) = \Lambda(G_2)$ and there exist a* transposition constant *$c \in \mathbb{Z}$ such that*

$$\forall (F, k) \in \mathcal{F} \times \mathbb{Z} \ \ (G_1 \oplus F, k) \in \Pi \Leftrightarrow (G_2 \oplus F, k + c) \in \Pi.$$

Note that the relation $\equiv_\Pi$ is an equivalence relation. Observe that c could be negative in the above definition. This is the reason we extended the definition of parameterized problems to include negative parameters also.

Next we define a notion of "transposition-minimality" for the members of each equivalence class of $\equiv_\Pi$.

Definition E.5. [**Progressive representatives, [9]**] *Let Π be a parameterized graph problem whose instances are pairs of the form (G, k) and let $\mathcal{C}$ be some equivalence class of $\equiv_\Pi$. We say that $J \in \mathcal{C}$ is a* progressive representative *of $\mathcal{C}$ if for every $H \in \mathcal{C}$ there exists $c \in \mathbb{Z}^-$, such that*

$$\forall (F, k) \in \mathcal{F} \times \mathbb{Z} \quad (H \oplus F, k) \in \Pi \Leftrightarrow (J \oplus F, k + c) \in \Pi. \tag{1}$$

The following lemma guaranties the existence of a progressive representative for each equivalence class of $\equiv_\Pi$.

Lemma E.1 ([9]). *Let Π be a parameterized graph problem whose instances are pairs of the form (G, k). Then each equivalence class of $\equiv_\Pi$ has a progressive representative.*

Definition E.6. [Finite Integer Index, [9, **?**]] *For a parameterized problem Π and two t-boundaried graphs G_1 and G_2, we say that $G_1 \equiv_\Pi G_2$ if there exists a constant c such that for every t-boundaried graph G and for every integer k, $(G_1 \oplus G_2, k) \in \Pi$ if and only if $(G_2 \oplus G, k + c) \in \Pi$. For every t, the relation $\equiv_\Pi$ on t-boundaried raphs is an equivalence relation, and we call $\equiv_\Pi$ the* canonical equivalence relation *of Π. We say that a problem Π has* Finite Integer Index *if for every t, $\equiv_\Pi$ has a finite index on t-boundaried graphs.*

Let $\mathcal{G}$ be a graph class. We say that $\mathcal{G}$ is CMSO-*definable* if there exist a sentence ψ on graphs such that $\mathcal{G} = \{G \mid G \models \psi\}$ and, in such a case, we say that ψ defines the class $\mathcal{G}$. Given a parameterized graph problem Π and a graph class $\mathcal{G}$, we denote by $\Pi \Cap \mathcal{G}$ the problem obtained by removing from Π all instances that encode graphs that do not belong to $\mathcal{G}$.

We would like to show that DFVS/DFVS+Tw-η Mod has Finite Integer Index (FII) property over $\mathcal{F}_\eta$. Observe that a digraph $D \in \mathcal{F}_\eta$ if and only if the underlying undirected graph has treewidth at most η. Thus, $\mathcal{F}_\eta$ can be characterized by finite forbidden set of minors. Since minor testing can be expressed into CMSO we have that $\mathcal{F}_\eta$ is CMSO-definable. Let Π denote the DFVS. The problem Π is p-min-CMSO[ψ] and is strongly monotone (see [9] for definition). A proof for this fact is analogous to the proof of [9, Lemma 8.4]. Thus, by [9, Lemmas 7.3 and 7.4] we get the following.

Lemma E.2. *$\Pi \Cap \mathcal{F}_\eta$ has FII. Here, Π is* DFVS/DFVS+Tw-η Mod.

E.4 Proof of Lemma D.4

Proof. The proof is essentially given in [9, Lemma 5.18]. We just adapt it here for our purposes. Our problem is $\Pi \Cap \mathcal{F}_\eta$ where Π is DFVS/DFVS+Tw-η Mod.

We denote by $\mathcal{S}_{\subseteq\{1,\ldots,2t+1\}}$ a set of (progressive) representatives for $\equiv_\Pi$ restricted to boundaried graphs with label sets from $\{1,\ldots,2t+1\}$. Let

$$c = \max\left\{|V(Y)| \;\middle|\; Y \in \mathcal{S}_{\subseteq\{1,\ldots,2t+1\}}\right\}.$$

Our algorithm has in its source code hard-wired a table that stores for each boundaried graph G_Y in $\mathcal{F}_{\subseteq\{1,\ldots,2t+1\}}$ on at most $2c$ vertices a boundaried graph $G'_Y \in \mathcal{S}_{\subseteq\{1,\ldots,2t+1\}}$ and a constant $\mu \leq 0$ such that $G_Y \equiv_\Pi G'_Y$, and specifically

$$\forall (F,k) \in \mathcal{F} \times \mathbb{Z} \;:\; (G_Y \oplus F, k) \in (\Pi \Cap \mathcal{F}_\eta) \iff (G'_Y \oplus F, k+\mu) \in (\Pi \Cap \mathcal{F}_\eta). \tag{2}$$

The existence of such a constant $\mu \leq 0$ is guaranteed by the fact that $\mathcal{S}_{\subseteq\{1,\ldots,2t+1\}}$ is a set of progressive representatives.

We now apply [9, Lemma 5.5] and find a $(2t+1)$-protrusion Y of D where $c < |Y| \leq 2c$. Split D into two boundaried graphs $D_Y = D[Y \cup N(Y)]$ and $D_R = D[(V(G) \setminus Y)]$ as follows. Both D_R and D_Y have boundary $N(Y)$, and since $|N(Y)| \leq 2t+1$ we may label the boundaries of D_Y and D_R with labels from $\{1,\ldots,2t+1\}$ such that $D = D_Y \oplus D_R$. As $c < |V(G_Y)| \leq 2c$ the algorithm can look up in its table and find a $D'_Y \in \mathcal{S}_{\subseteq\{1,\ldots,2t+1\}}$ and a constant μ such that $D_Y \equiv D'_Y$ and D_Y, D'_Y and μ satisfy Equation 2. Observe that since the initial protrusion X is disjoint from M we have that the vertex set of M remains in D'. The algorithm outputs

$$(D', k', M) = (D'_Y \oplus D_R, k+\mu, M).$$

Since $|V(D'_Y)| \leq c < |V(D_Y)|$ and $k' \leq k+\mu \leq k$ it remains to argue that the instances (D,k,M) and (D',k',M) are equivalent. However, this is directly

implied by Equation 2. In particular, by Equation 2 we have that D has a directed feedback vertex set of size at most k if and only if D' has directed feedback vertex set of size at most k'. Now we show that $D' - M$ has treewidth at most η. Let $D_R^M = D_R - M$. Now since, $D_Y \equiv D'_Y$ with respect to $\Pi \Cap \mathcal{F}_\eta$ we have that $(D_Y \oplus D_R^M, k^*)$ is in $\Pi \Cap \mathcal{F}_\eta$ if and only if $(D'_Y \oplus D_R^M, k^*)$ is in $\Pi \Cap \mathcal{F}_\eta$. Since $\mathbf{tw}(D_Y \oplus D_R^M) \leq \eta$ we have that $\mathbf{tw}(D'_Y \oplus D_R^M)$ has treewidth at most η. This completes the proof.

F An Outline of a Kernel for $\mathcal{F}_\eta$-Vertex Deletion Set

In this section we give an outline of a kernel for $\mathcal{F}_\eta$-Vertex Deletion Set. The kernelization algorithm follows along the lines of kernel for Planar-$\mathcal{F}$-Deletion given in [27]. We follow the steps of kernelization algorithm given for Planar-$\mathcal{F}$-Deletion in [27] to design a polynomial kernel for $\mathcal{F}_\eta$-Vertex Deletion Set.

Recall that for a positive integer $\eta > 0$, $\mathcal{F}_\eta$ denotes the family of digraphs of treewidth at most η. Let D be a digraph and let S be a vertex subset such that $D - S$ is a DAG and has treewidth at most η. Then, we call such S a *DAG-treewidth η-modulator* set. Towards kernelization, we begin by showing that any YES-instance (D, k) to $\mathcal{F}_\eta$-Vertex Deletion Set has a set X of $k^{\mathcal{O}(1)}$ vertices such that every connected component C of $D - X$ is a *near-protrusion*. Recall that a r-protrusion C in a graph D is a vertex set such that $|N(C)| \leq r$ and $\mathbf{tw}(G[C]) \leq r$. The components of $D - X$ will not necessarily be protrusions, however we will prove that there is a constant r such that if (D, k) is a yes instance, then for any DAG-treewidth η-modulator set S of size at most k, $C \setminus S$ is a r-protrusion of $D - S$.

For our kernelization algorithm we also need an approximation algorithm for $\mathcal{F}_\eta$-Vertex Deletion Set. To obtain a factor c-approximation algorithm for $\mathcal{F}_\eta$-Vertex Deletion Set we do as follows. Towards computing an approximate solution for $\mathcal{F}_\eta$-Vertex Deletion Set, we run the constant factor approximation algorithm for Treewidth-η-Modulator given in [27], on the underlying undirected graph of D. Let us recall that in the Treewidth-η-Modulator problem, we are given an undirected graph G and the objective is to find a minimum sized vertex set W such that $\mathbf{tw}(G - W) \leq \eta$. Let, W be a $c' \cdot \mathsf{OPT}$ sized approximation to the Treewidth-η-Modulator problem when run on the underlying undirected graph of D. Here, OPT is the size of a minimum sized vertex set W such that $\mathbf{tw}(D - W) \leq \eta$. Furthermore, let OPTdag be the size of a minimum sized set $W^\star$ such that $D - W^\star$ is a DAG in $\mathcal{F}_\eta$. Observe that $\mathsf{OPT} \leq \mathsf{OPTdag}$. Note that $D - W$ is in $\mathcal{F}_\eta$, though D_W may not be DAG. To make $D - W$ a DAG we now find a set of vertices such that it is a directed feedback vertex set. We compute a smallest size directed feedback vertex set, W_{dfvs}, in $D - W$ by calling the algorithm for DFVS parameterized by treewidth given in [10]. Since, an optimum solution to $\mathcal{F}_\eta$-Vertex Deletion Set is also a directed feedback vertex set we have that $|W_{\mathsf{dfvs}}| \leq \mathsf{OPTdag}$. Now observe that $W \cup W_{\mathsf{dfvs}}$ is a solution to $\mathcal{F}_\eta$-Vertex Deletion Set of size at most

$(c'+1)\mathsf{OPTdag}$. Thus, this gives a factor $c = (c'+1)$ approximation algorithm for $\mathcal{F}_\eta$-Vertex Deletion Set. Let this factor c approximation algorithm for $\mathcal{F}_\eta$-Vertex Deletion Set be called $\texttt{Approx-}\mathcal{F}_\eta\texttt{-VDS}$.

The kernelization algorithm begins by running $\texttt{Approx-}\mathcal{F}_\eta\texttt{-VDS}$, a c-approximation algorithm, for $\mathcal{F}_\eta$-Vertex Deletion Set. If the solution returned by the approximation algorithm is more than ck, the kernelization algorithm returns a trivial no instance. Otherwise, let W denote the approximate solution. Based on W, and exploiting the fact that $D - X$ has treewidth at most η, we are able to construct $R \subseteq (D - W)$ on $k^{\mathcal{O}(1)}$ vertices such that: (a) for every connected component C of $D \setminus (W \cup R)$, $|N(C) \cap R| \leq 2(\eta+1)$, and (b) for every connected component C of $D \setminus (W \cup R)$, and $u, v \in N(C) \cap W$ there are at least $k + \eta + 3$ vertex disjoint paths from u to v in the underlying undirected graph of D. Now we can combine these components (or rather near protrusions) to get the following partition of $V(D)$ satisfying the below stated properties. That is,

$$V(D) = W \uplus R \uplus \bigcup_{i \in \{1,\ldots,q\}} Z_i \tag{3}$$

The properties satisfied by the partition given in Equation 3 are as follows.

1. $|R| \leq k^{\mathcal{O}(1)}$.
2. for each $i \in \{1, \ldots, q\}$, $D[Z_i]$ is acyclic and $\mathbf{tw}(D[Z_i]) \leq \eta$,
3. for each $i \in \{1, \ldots, q\}$, $N(Z_i) \subseteq W \cup R$,
4. for each $i \in \{1, \ldots, q\}$, $|N(Z_i) \cap R| \leq 2(\eta + 1)$, and
5. for each $i \in \{1, \ldots, q\}$, Z_i is a $\mathcal{O}(\eta)$-near protrusion.

The proof for the partition given in Equation 3 is similar to the decomposition algorithm given in Lemma 3.1, and in fact identical to the proof of Lemma 25 of Fomin et al. [27].

The next step of the kernelization algorithm is to bound the number q of connected components of $D - (W \cup R)$ by a polynomial in k. Towards this we can proceed in a manner identical to Lemma 36 of Fomin et al. [27]: Here it is proved that, if $W \cup R$ satisfies properties 2-5 then one can mark in polynomial time a $k^{\mathcal{O}(1)}$ size subset of the connected components with the following properties.

For every connected component Z_i that is *not* marked, every subset $S \subseteq V(D)$ such that $|S| \leq k + 1$ and $\mathbf{tw}(D - (S \cup \{v\})) \leq \eta$, we have that $\mathbf{tw}(D - S) \leq \eta$.

In other words, all vertices in all of the unmarked connected components are *irrelevant* with respect to the treewidth of $D - S$ for a potential solution S. In our setting, such vertices v may still be relevant because $D - (S \cup \{v\})$ could be a DAG, while $D - S$ is not. However, this can be remedied by essentially the same strategy: For each pair u, v of vertices in $W \cup R$ (including the pairs where $u = v$), mark at most $k + 10$ connected components of $D - (W \cup R)$ that contain a directed path from an out-neighbor of u to an in-neighbor of v. This results in at most $k^{\mathcal{O}(1)}$ additional components being marked. A proof following the lines of the proof of Lemma 36 of Fomin et al. [27] (but simpler, because directed paths are easier than rooted minors) shows every vertex v in an unmarked component is irrelevant in the following sense: For every subset $S \subseteq V(D)$ such that $|S| \leq k+1$

and $D - S \cup \{v\} \in \mathcal{F}_\eta$ we have that $D - S \in \mathcal{F}_\eta$. This leads to a reduction rule that, provided the number of components of $D - (W \cup R)$ is more than $k^{\mathcal{O}(1)}$, selects any vertex v in an unmarked component and deletes it. After this rule is exhaustively applied we can also conclude that the number q of connected components of $D - (W \cup R)$ is at most $k^{\mathcal{O}(1)}$.

Having bounded the number of connected components of $D-(W\cup R)$, next we need to bound their size. However, instead of bounding the size of the connected components we instead bound the *bitsize* of a representation of the component. This leads to a compression rather than a kernel, but because the compression is into a problem in NP, we can use NP-completeness to reduce back to our problem and obtain a polynomial kernel.

For each component Z_i, define its boundary $B_i = N(Z_i) \setminus S$. We know that $|B_i| = \beta_i = \mathcal{O}(\eta)$ for any solution S of size at most k. Define D_i to be a boundaried graph $D[N[Z_i] \setminus S]$ with boundary $|N(Z_i) \setminus S|$. Since the class $\mathcal{F}_\eta$ has finite state (see e.g. Bodlaender et al. [9]) the canonical equivalence relation for $\mathcal{F}_\eta$ restricted to the set of β_i-boundaried graphs has a finite set of representatives. Let D'_i be the representative for D_i. We have that $(D - Z_i \cup S) \oplus D'_i$ is in $\mathcal{F}_\eta$ if and only if $D - S \in \mathcal{F}_\eta$.

We may think of the selection of D'_i given S as a function of B_i and $S_i = S \cap Z_i$, and write $D'_i(B_i, S_i)$ for this function. Similarly, $B_i(S) = N(Z_i) \setminus S$ and $S_i(S) = S \cap Z_i$ can be thought of as functions of S. The equivalence above shows that, for any two sets S^1 and S^2 of size at most k, if $S_1 \cap (W \cup R) = S_2 \cap (W \cup R)$ and for every i, $D'_i(B_i(S^1), S_i(S^1)) = D'_i(B_i(S^2), S_i(S^2))$ then $D - S_1 \in \mathcal{F}_\eta$ if and only if $D - S_2 \in \mathcal{F}_\eta$.

Thus, for every connected component Z_i it is sufficient to store for every $B_i \subseteq N(Z_i)$ of size $\mathcal{O}(\eta)$, for every representative D'_i of the canonical equivalence relation for $\mathcal{F}_\eta$ restricted to the set of $\mathcal{O}(\eta)$-boundaried graphs, the size of the smallest set S_i such that $D'_i(B_i, S_i) = D'_i$ (or ∞ if no such set exists or is larger than k). Hence, for each of the connected components Z_i we store a number between 0 and $k + 1$ for each of the $k^{\mathcal{O}(1)}$ choices of B_i and each of the $\mathcal{O}(1)$ choices of D'_i.