Kernels for Deletion to Classes of Acyclic Digraphs[*]

Akanksha Agrawal[1], Saket Saurabh[2], Roohani Sharma[3], and Meirav Zehavi[4]

1 **University of Bergen, Norway**
`akanksha.agrawal@uib.no`
2 **University of Bergen, Norway; and Institute of Mathematical Science, HBNI, India**
`saket@imsc.res.in`
3 **University of Bergen, Norway**
`roohani@imsc.res.in`
4 **University of Bergen, Norway**
`meirav.zehavi@uib.no`

Abstract

In the Directed Feedback Vertex Set (DFVS) problem, we are given a digraph D on n vertices and a positive integer k and the objective is to check whether there exists a set of vertices S of size at most k such that $F = D - S$ is a directed acyclic digraph. In a recent paper, Mnich and van Leeuwen [*STACS 2016*] considered the kernelization complexity of DFVS with an additional restriction on F, namely that F must be an out-forest (Out-Forest Vertex Deletion Set), an out-tree (Out-Tree Vertex Deletion Set), or a (directed) pumpkin (Pumpkin Vertex Deletion Set). Their objective was to shed some light on the kernelization complexity of the DFVS problem, a well known open problem in the area of Parameterized Complexity. In this article, we improve the kernel sizes of Out-Forest Vertex Deletion Set from $\mathcal{O}(k^3)$ to $\mathcal{O}(k^2)$ and of Pumpkin Vertex Deletion Set from $\mathcal{O}(k^{18})$ to $\mathcal{O}(k^3)$. We also prove that the former kernel size is tight under certain complexity theoretic assumptions.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases out-forest, pumpkin, parameterized complexity, kernelization

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.6

1 Introduction

Feedback Set problems form a family of fundamental combinatorial optimization problems. The input for Directed Feedback Vertex Set (DFVS) (Directed Feedback Edge Set (DFES)) consists of a directed graph (digraph) D and a positive integer k, and the question is whether there exists a subset $S \subseteq V(D)$ ($S \subseteq E(D)$) such that the graph obtained after deleting the vertices (edges) in S is a directed acyclic graph (DAG). Similarly, the input for Undirected Feedback Vertex Set (UFVS) (Undirected Feedback Edge

[*] The research leading to these results received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreements no. 306992 (S. Saurabh). Part of this work was done while M. Zehavi was visiting the Simons Institute for the Theory of Computing.

Set (UFES)) consists of an undirected graph G and a positive integer k, and the question is whether there exists a subset $S \subseteq V(G)$ ($S \subseteq E(G)$) such that the graph obtained after deleting the vertices (edges) in S is a forest.

All of these problems, excluding Undirected Feedback Edge Set, are NP-complete. Furthermore, Feedback Set problems are among Karp's 21 NP-complete problems and have been topic of active research from algorithmic [2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 18, 19, 20, 22, 24, 27, 32] as well as structural points of view [17, 21, 23, 26, 28, 29, 30]. In particular, such problems constitute one of the most important topics of research in Parameterized Complexity [6, 8, 9, 10, 12, 13, 22, 20, 24, 27, 32], spearheading development of new techniques. In this paper we study the parameterized complexity of restrictions of DFVS.

In Parameterized Complexity each problem instance is accompanied by a parameter k. A central notion in this field is the one of *fixed-parameter tractability (FPT)*. This means, for a given instance (I, k), solvability in time $f(k)|I|^{\mathcal{O}(1)}$ where f is some function of k. Another central notion is the one of *kernelization*. A parameterized problem is said to admit a *kernel* of size $f(k)$ for some function f of k if there is a polynomial-time algorithm, called a *kernelization algorithm*, that translates any input instance to an equivalent instance of the same problem whose size is bounded by $f(k)$. In case the function f is polynomial in k, the problem is said to admit a *polynomial kernel*. For more information on these concepts we refer the reader to monographs such as [16, 11].

In contrast to UFVS which admits a polynomial kernel, the existence of a polynomial kernel for DFVS is still an open problem. The lack of progress on this question led to the consideration of various restrictions on input instances. In particular, we know of polynomial kernels for DFVS in tournaments as well as various generalizations [1, 3, 15]. However, the existence of a polynomial kernel for DFVS is open even for planar digraphs. Recently, in a very interesting article, to make progress on this question Mnich and van Leeuwen [25] considered DFVS with an additional restriction on *the output rather than the input*. Essentially, the basic philosophy of their program is the following: What happens to the kernelization complexity of DFVS when we consider subclasses of DAGs?

Mnich and van Leeuwen [25] inspected this question by considering the classes of out-forests, out-trees and (directed) pumpkins. An *out-tree* is a digraph where each vertex has in-degree at most 1 and the underlying (undirected) graph is a tree. An *out-forest* is a disjoint union of out-trees. On the other hand, a digraph is a *pumpkin* if it consists of a source vertex s and a sink vertex t, $s \neq t$, together with a collection of internally vertex-disjoint induced directed paths from s to t. Here, all vertices except s and t have in-degree 1 and out-degree 1. The examination of the classes of out-forests and out-trees was also motivated by the corresponding questions of UFVS and Tree Deletion Set in the undirected settings. Formally, Mnich and van Leeuwen [25] studied the following problems.

Out-Forest Vertex Deletion Set (OFVDS) **Parameter:** k
Input: A digraph D and a positive integer k.
Question: Is there a set $S \subseteq V(D)$ of size at most k such that $F = D \setminus S$ is an out-forest?

Out-Tree Vertex Deletion Set (OTVDS) and Pumpkin Vertex Deletion Set (PVDS) are defined in a similar manner, where instead of an out-forest, F should be an out-tree or a pumpkin, respectively. Mnich and van Leeuwen [25] showed that OFVDS and OTVDS admit kernels of size $\mathcal{O}(k^3)$ and PVDS admits a kernel of size $\mathcal{O}(k^{18})$.

Our Results and Methods. The objective of this article is to give improved kernels for OFVDS and PVDS. In this context, we obtain the following results.

- OFVDS admits an $\mathcal{O}(k^2)$ kernel and PVDS admits an $\mathcal{O}(k^3)$ kernel. These results improve upon the best known upper bounds $\mathcal{O}(k^3)$ and $\mathcal{O}(k^{18})$, respectively.
- For any $\epsilon > 0$, OFVDS does not admit a kernel for of size $\mathcal{O}(k^{2-\epsilon})$ unless coNP $\subseteq$ NP /poly.

To get the improved kernel for OFVDS we incorporate the Expansion Lemma as well as a factor 3-approximation algorithm for OFVDS in the kernelization routine given in [25]. The significance of this improvement also lies in the fact that we show that it is essentially tight. Due to space constraints, the lower bound is omitted from this version of the paper.

The kernelization algorithm for PVDS given in [25] works roughly as follows. It has two phases: (a) first it gives an $\mathcal{O}(k^5)$ kernel for a variant of the problem where we know the source and the sink of the pumpkin obtained after deleting the solution vertices; and (b) in the second phase, it reduces PVDS to polynomially many instances of a variant of the problem mentioned in the item (a) and then composes these instances to get a kernel of size $\mathcal{O}(k^{18})$. In fact given an instance (D, k) of PVDS, the kernelization algorithm of [25] outputs an equivalent instance (D', k') such that $k' = \mathcal{O}(k^{18})$. We take a completely different route and use "sun-flower style" reduction rules together with a marking strategy to obtain an equivalent instance (D', k') such that $|V(D)| + |E(D)| = \mathcal{O}(k^3)$ and $k' \leq k$. We believe the method applied in this algorithm could be useful also in other kernelization algorithms.

2 Preliminaries

We denote the set of natural numbers from 1 to n by $[n]$, and we use standard terminology from the book of Diestel [14] for graph-related terms which are not explicitly defined here. A digraph D is a pair $(V(D), E(D))$ such that $V(D)$ is a set of vertices and $E(D)$ is a set of ordered pairs of vertices. The underlying undirected graph G of D is a pair $(V(G), E(G))$ such that $V(G) = V(D)$ and $E(G)$ is a set of unordered pairs of vertices such that $\{u, v\} \in E(G)$ if and only if either $(u, v) \in E(D)$ or $(v, u) \in E(D)$. Let D be a digraph. For any $v \in V(D)$, we denote by $N^-(v)$ the set of in-neighbors of v, that is, $N^-(v) = \{(u, v) \mid (u, v) \in E(D)\}$. Similarly, we denote by $N^+(v)$ the set of out-neighbors of v, that is, $N^+(v) = \{(v, u) \mid (v, u) \in E(D)\}$. We denote the in-degree of a vertex v by $d^-(v) = |N^-(v)|$ and its out-degree by $d^+(v) = N^+(v)$. We say that $P = (u_1, \ldots, u_l)$ is a directed path in the digraph D is $u_1, \ldots, u_l \in V(D)$ and for all $i \in [l-1]$, $(u_i, u_{i+1}) \in E(D)$. A *collision* is a triplet (u, w, v) of distinct vertices such that $(u, w), (v, w) \in E(D)$.

3 Improved Kernel for Out-Forest Vertex Deletion Set

The aim of this section is to present an $\mathcal{O}(k^2)$ kernel for OFVDS. In Section 3.1 we state definitions and results relevant to our kernelization algorithm. Next, in Section 3.2, we design an algorithm for OFVDS that outputs a 3-approximate solution, which will also be used by our kernelization algorithm. Finally, in Section 3.3, we present our kernelization algorithm.

3.1 Prerequisites

We start by giving the definition of a q-expansion and the statement of the Expansion Lemma.

▶ **Definition 1** (q-Expansion). For a positive integer q, a set of edges $M \subseteq E(G)$ is a *q-expansion of A into B* if (i) every vertex in A is incident to exactly q vertices in M, and (ii) M saturates exactly $q|A|$ vertices in B (i.e., there is a set of $q|A|$ vertices in B which are incident to edges in M).

▶ **Lemma 2** (Expansion Lemma [11, 31]). *Let q be a positive integer and G be an undirected bipartite graph with vertex bipartition (A, B) such that $|B| \geq q|A|$, and there are no isolated vertices in B. Then, there exist nonempty vertex sets $X \subseteq A$ and $Y \subseteq B$ such that there exists a q-expansion of X into Y, and no vertex in Y has a neighbor outside X (i.e., $N(Y) \subseteq X$). Furthermore, the sets X and Y can be found in time polynomial in the size of G.*

We will also need to rely on the well-known notion of l-flowers.

▶ **Definition 3** (l-Flower). An undirected graph G contains an *l-flower through v* if there is a family of cycles $\{C_1, \ldots, C_l\}$ in G such that for all distinct $i, j \in [l]$, $V(C_i) \cap V(C_j) = \{v\}$.

▶ **Lemma 4** ([11, 31]). *Given an undirected graph G and a vertex $v \in V(G)$, there is a polynomial-time algorithm that either outputs a $(k+1)$-flower through v or, if no such flower exists, outputs a set $Z_v \subseteq V(G) \setminus \{v\}$ of size at most $2k$ that intersects every cycle that passes through v in G.*

3.2 Approximation Algorithm for Out-Forest Vertex Deletion Set

This section presents a 3-factor approximation algorithm for OFVDS. Given an instance of OFVDS, let OPT be the minimum size of a solution. Formally, we solve the following.

3–Approximate Out-Forest Vertex Deletion Set (Approx-OFVDS)
Input: A DAG D.
Output: A subset $X \subseteq V(D)$ such that $D \setminus X$ is an out-forest and $|X| \leq 3 \cdot OPT$.

Given three distinct vertices $u_1, u_2, u_3 \in V(D)$, we say (u_1, u_2, u_3) is an *obstruction* if u_1 and u_2 are in-neighbors of u_3. Observe that any solution to OFVDS (and hence, Approx-OFVDS) must intersect any obstruction in at least 1 vertex. Moreover, it must intersect any cycle in at least 1 vertex. These observations form the basis of this algorithm.

▶ **Lemma 5.** Approx-OFVDS *can be solved in polynomial time.*

Proof. Given a digraph D, the algorithm first constructs (in polynomial time) a family $\mathcal{F}$ of obstructions and induced cycles in D such that the vertex sets of the entities in this family are pairwise disjoint. To this it, it initializes $\mathcal{F} = \emptyset$. Then, as long as there exists a vertex $v \in V(D)$ with at least two in-neighbors, u_1 and u_2, it inserts (v, u_1, u_2) into $\mathcal{F}$ and removes v, u_1 and u_2 from $\mathcal{F}$ (only for the purpose of the construction of $\mathcal{F}$). Once there is no vertex $v \in V(D)$ such that $d^-(v) \geq 2$, the digraph is a collection of directed vertex-disjoint cycles and paths. Each of these cycles is inserted into the family $\mathcal{F}$.

Let us now construct a solution, S_{app}, for Approx-OFVDS. For every obstruction in $\mathcal{F}$, we let S_{app} contain each of the three vertices of this obstruction. From every cycle C in $\mathcal{F}$ we pick an arbitrary vertex and insert it into S_{app}. Clearly, $|S_{app}| \leq 3|\mathcal{F}|$. It is now sufficient to prove is that $D \setminus S_{app}$ is an out-forest. Observe that no vertex v in $D \setminus S_{app}$ has in-degree at least 2, otherwise the obstruction consisting of v and two of its in-neighbors would have been inserted into $\mathcal{F}$ and hence also into S_{app}. Moreover, there is no directed cycle C in $D \setminus S_{app}$. Indeed, if the cycle C intersects an obstruction in $\mathcal{F}$, it is clear that it cannot exist in $D \setminus S_{app}$, and otherwise it would have been inserted into $\mathcal{F}$ and hence one of its vertices would have been inserted into S_{app}. We thus conclude that the theorem is correct. ◀

3.3 Kernelization algorithm for Out-Forest Vertex Deletion Set

We are are now ready to present our kernelization algorithm. Let (D, k) be an instance of OFVDS. We note that during the execution of our algorithm, D may become a multigraph.

Preprocessing. We start by applying the following reduction rules exhaustively, where a rule is applied only if its condition is true and the conditions of all of the preceding rules are false. Rule 4 is given in [25], and its correctness is proven in that paper. It will be clear that the other first five rules can be applied in polynomial time, while for the last rule, we call the algorithm given by Lemma 4. Moreover, it is straightforward to verify that each of these rules, except Rule 4, is safe (i.e., the instance it returns is equivalent to the input instance).

▶ **Reduction Rule 1.** *If there exists a vertex $v \in V(D)$ such that $d^+(v) = 0$ and $d^-(v) \leq 1$, remove v from D.*

▶ **Reduction Rule 2.** *If there exists a directed path $P = (w_0, w_1, \ldots, w_l, w_{l+1})$ in D such that $l \geq 2$ and for all $i \in [l]$, $d^-(w_i) = d^+(w_i) = 1$, remove each vertex in $\{w_1, \ldots, w_{l-1}\}$ from D and add the edge (w_0, w_l) to D.*

▶ **Reduction Rule 3.** *If there exists an edge $(u, v) \in E(D)$ with multiplicity at least 3, remove all but two copies of it.*

▶ **Reduction Rule 4.** *If there exist collisions $(u_1, w_1, v), \ldots, (u_{k+1}, w_{k+1}, v)$ that pairwise intersect only at v, remove v from D and decrease k by 1.*

▶ **Reduction Rule 5.** *If there exists a vertex $v \in V(D)$ such that $d^-(v) \geq k + 2$, remove v from D and decrease k by 1.*

▶ **Reduction Rule 6.** *Let G be the underlying graph of D. If there exists a vertex $v \in V(G)$ such that there is a $(k + 1)$-flower through v in G, remove v from D and decrease k by 1.*

Bounding Out-Degrees. Next, we aim to bound the maximum out-degree of a vertex in D. To this end, suppose that there exists a vertex $v \in V(D)$ with $d^+(v) \geq 16k + 1$. Let G be the underlying graph of D. Since Reduction Rule 6 is not applicable, we let Z_v be the set obtained by calling the algorithm given by Lemma 4. Moreover, we let S_{app} be a 3-factor approximate solution obtained be calling the algorithm given by Theorem 5. We can assume that $|S_{app}| \leq 3k$, since otherwise the input instance is a NO-instance. Denote $X_v = (S_{app} \cup Z_v) \setminus \{v\}$. Since $|Z_v| \leq 2k$, we have that $|X_v| \leq 5k$.

We proceed by examining the set $\mathcal{C}_v = \{C_1, C_2, \ldots, C_{|\mathcal{C}_v|}\}$ of the connected components in $G \setminus (X_v \cup \{v\})$. Since S_{app} is an approximate solution, each component $C_i \in \mathcal{C}_v$ is an out-tree. Moreover, for any component $C_i \in \mathcal{C}_v$, v has at most one neighbor in C_i, since otherwise there would have been cycle passing through v in $G \setminus Z_v$, contradicting the definition of Z_v. For each component $C_i \in \mathcal{C}_v$, let u_i be the root of C_i. Let $\mathcal{D}_v = \{C_i \mid C_i \in \mathcal{C}_v, (v, u_i) \in E(D)\}$ and $\tilde{\mathcal{D}}_v = \{C_i \mid C_i \in \mathcal{C}, (v, u) \in E(D), u \in C_i, u \neq u_i\}$. Observe that $d^+(v) \leq |\mathcal{D}_v| + |\tilde{\mathcal{D}}_v| + |X_v|$. Moreover, since Reduction Rule 4 is not applicable, $|\tilde{\mathcal{D}}_v| \leq k + 1$. Since $d^+(v) \geq 16k + 1$, we have that $|\mathcal{D}_v| \geq 10k$. Without loss of generality, let $\mathcal{D}_v = \{C_1, \ldots, C_p\}$ where $p = |\mathcal{D}_v|$. Since Reduction Rule 1 is not applicable, for any component $C_i \in \mathcal{D}_v$ there exists an edge in $E(G)$ with one endpoint in C_i and the other in X_v.

We now construct an auxiliary (undirected) bipartite graph H with bipartition (A, B), where $A = X_v$ and B is a set of new vertices denoted by $b_1, \ldots, b_p$. For any $u \in A$ and $b_i \in B$, $(u, b_i) \in E(H)$ if and only if there exists an edge in G between u and some vertex in C_i. Since $|B| \geq 2|A|$ and there are no isolated vertices in B, we can use the algorithm given by Lemma 2 to obtain nonempty vertex sets $X'_v \subseteq A$ and $Y'_v \subseteq B$ such that there is a 2-expansion of X'_v into Y'_v and $N(Y'_v) \subseteq X'_v$. Let $\mathcal{D}'_v = \{C_i \mid b_i \in Y'_v\}$.

▶ **Reduction Rule 7.** *Remove each of the edges in D between v and any vertex in a component in $\mathcal{D}'_v$. For every vertex $x_i \in X'_v$, insert two copies of the edge (v, x_i) into $E(D)$.*

▶ **Lemma 6.** *Reduction Rule 7 is safe.*

Proof. Let D' be the graph resulting from the application of the rule. We need to prove that (D,k) is a YES-instance if and only if (D',k) is a YES-instance.

Forward Direction. For the forward direction, we first claim that if (D,k) has a solution S such that $v \notin S$, then it has a solution S' such that $X'_v \subseteq S'$. To this end, suppose that (D,k) has a solution S such that $v \notin S$. Let $S' = (S \setminus \bigcup_{C_i \in \mathcal{D}'_v} V(C_i)) \cup X'_v$. It holds that $|S'| \leq |S|$ since for each $x \in X'_v \setminus S$, at least one vertex from at least one of the two components in its expansion set must belong to the solution. Suppose for the sake of contradiction that $F = D \setminus S'$ is not an out-forest. First, assume that there exists a vertex in F with in-degree at least 2. Note that $V(D) = \bigcup_{C_i \in \mathcal{C}_v} V(C_i) \cup X_v \cup \{v\}$. Recall that the neighborhood of each of the vertices in the connected components that belong to $\mathcal{D}'_v$ is contained in $\{v\} \cup X'_v$. Moreover, v only has out-neighbors in the components that belong to $\mathcal{D}'_v$ and each $C_i \in \mathcal{C}_v$ is an out-tree. Therefore, since $D \setminus S$ has no vertex of in-degree at least 2, so does $D \setminus S'$. Now, assume that there is a cycle C in F. Then, if $V(C) \cap (S \cap \bigcup_{C_i \in \mathcal{D}'} V(C_i)) = \emptyset$, then C is also a cycle in $D \setminus S$, which is a contradiction. Thus, $V(C) \cap (S \cap \bigcup_{C_i \in \mathcal{D}'} V(C)_i) \neq \emptyset$. However, any cycle that passes through a component in $\mathcal{D}'_v$ also passes through v and a vertex in X'_v. Since $X'_v \subseteq S'$, no such cycle exists. This finishes the proof of the claim.

Let S be a solution to (D,k). If $v \in S$, then it is clear that $D' \setminus S$ is an out-forest. Otherwise, if $v \notin S$, our claim implies that (D,k) has a solution S' such that $X'_v \subseteq S'$. Then, $D' \setminus S'$ is an out-forest.

Backward Direction. For the backward direction, let us prove the following claim. If (D',k) has a solution S such that $v \notin S$, then $X'_v \subseteq S$. Suppose, by way of contradiction, that the claim is incorrect. Then, there exists $x \in X'_v$ such that $x \notin S$. However, this implies that $D' \setminus S$ is not an out-forest as it contains the double edges (v, x_i).

Now, let S be a solution to (D',k), and denote $F = D' \setminus S$. Suppose $v \in S$. Then, $F = D \setminus S$ is an out forest and thus S is solution to (D,k). If $v \notin S$, then by our previous claim, $X'_v \subseteq S$. Observe that each vertex $u_i \notin S$ is a root in F. Moreover, each such vertex u_i and v belong to different out-trees of F. This implies that if we add (to D') the edges between v and each vertex u_i that have been removed by the application of the rule, F will remain an out-forest. Thus, S is a solution to (D,k). ◀

After an exhaustive application of Reduction Rule 7, the out-degree of each vertex in D is at most $16k$. However, since this rule inserts edges into $E(G)$, we need the following lemma.

▶ **Lemma 7** (*[1]). *The total number of applications of the reduction rules is bounded by a polynomial in the input size.*

Correctness. By relying on counting arguments as well as Lemmas 6 and 7, we obtain the main result of this section.

▶ **Theorem 8** (*). OFVDS *admits an $\mathcal{O}(k^2)$-kernel.*

We also prove that the size of the kernel given in Theorem 8 is tight, that is OFVDS does not admit an $\mathcal{O}(k^{2-\epsilon})$ size kernel unless coNP $\subseteq$ NP/poly. This result follows from an easy polynomial time parameter preserving transformation from the VERTEX COVER problem parameterized by the solution size to OFVDS.

[1] Due to space constraints, proofs of results marked with (*) were omitted.

4 Improved Kernel for Pumpkin Vertex Deletion Set

In this section we prove the following theorem.

▶ **Theorem 9.** PVDS *admits an* $\mathcal{O}(k^3)$*-vertex kernel.*

Let (D, k) be an instance of PVDS. We assume that $|V(D)| \geq k^3$, else we are done. Let $\mathsf{HO} = \{v \in V \mid d^+(v) \geq k+2\}$ and $\mathsf{HI} = \{v \in V \mid d^-(v) \geq k+2\}$. That is, HO and HI are vertices of high out-degrees and high in-degrees, respectively. Mnich and Leeuwen [25] proved that the following reduction rule is safe.

▶ **Reduction Rule 4.1.** *If* $|\mathsf{HO}| > k+1$ *or* $|\mathsf{HI}| > k+1$*, return that* (D, k) *is a* NO*-instance.*

For the sake of clarity, we divide the presentation of the kernelization algorithm into two subsections. At the end of Section 4.1, we will simplify the instance in a way that will allow us to assume that if there is a solution S, then *both the source and sink of the pumpkin* $D \setminus S$ *belong to* $\mathsf{HO} \cup \mathsf{HI}$ (Assumption 17). This assumption will be at the heart of the "marking approach" of Section 4.2, which will handle instances which have been reduced with respect to the reduction rules in Section 4.1. An intuitive explanation of the necessity of our marking process is given at the beginning of Section 4.2. Throughout this section, if k becomes negative, we return that (D, k) is a NO-instance, and if D becomes a pumpkin and k is positive or zero, we return that (D, k) is a YES-instance.

4.1 Simplification Phase

For any $v \in V(D)$, denote by X_v the set of in-neighbors of v, that is, $X_v = N^-(v)$ and by Y_v the set of every vertex $y \in V(D)$ for which there exists a vertex $x \in X_v$ such that $(x, y) \in E(D)$. Note that X_v and Y_v may or may not be disjoint sets. We now give a construction of an auxiliary graph that will be used to prove the safeness of the upcoming reduction rule. For this, consider a set Y'_v of new vertices such that there is exactly one vertex $y' \in Y'_v$ for any $y \in Y_v$. That is, Y'_v is a set containing a copy for each of the vertex in Y_v. By construction, X_v and Y'_v are disjoint sets. Let H^-_v be the (undirected) bipartite graph on the vertex set $X_v \cup Y'_v$ where for all $x \in X_v$ and $y' \in Y'_v$, $\{x, y'\} \in E(H^-_v)$ if and only if $(x, y) \in E(D)$. Let $\mathsf{match}^-(v)$ be the size of a maximum matching in H^-_v.

▶ **Reduction Rule 4.2.** *If there exists a vertex* $v \in V(D)$ *such that* $\mathsf{match}^-(v) > 2(k+1)$*, remove* v *from* D *and decrease* k *by* 1.

▶ **Lemma 10.** *Reduction Rule 4.2 is safe.*

Proof. For the backward direction, trivially if S is a pumpkin deletion set in $D \setminus \{v\}$ of size at most $k-1$, then $S \cup \{v\}$ is a pumpkin deletion set in D of size at most k. For the forward direction, it is *sufficient* to show that if (D, k) is a YES-instance then *every solution* S *contains* v. For a contradiction, assume that there exists a solution S that does not contain v. Let M be a maximum matching in the graph H^-_v. Observe that for every edge $\{x, y'\} \in M$ where $x \in X_v$, if x is not the source of the pumpkin $D \setminus S$, it holds that $|S \cap \{x, y\}| \geq 1$ (otherwise the pumpkin $D \setminus S$ contains a vertex, which is not its source, and has at least two out-neighbors). Moreover, for every edge $\{x, y'\} \in M$ where $x \in X_v$, if y is the source of the pumpkin $D \setminus S$, it holds that $x \in S$. We thus deduce that *for all but one of the edges* $\{x, y'\} \in M$*, we have that* $|S \cap \{x, y\}| \geq 1$. Since M is a matching, for every vertex $u \in S$, the vertex u can belong to at most one edge in M, and the vertex u' (if it belongs to Y'_v) can also belong to at most one edge in M. However, $|S| \leq k$, and therefore

$S \cup \{y' \in Y'_v : y \in S\}$ can intersect at most $2k$ edges in M. Since $S \cup \{y' \in Y'_v : y \in S\}$ must intersect all but one edge of M and $|M| > 2(k+1)$, we obtain a contradiction. ◀

Now, to present the symmetric rule, for any vertex $v \in V(D)$, denote by X_v the set of out-neighbors of v, that is, $X_v = N^+(v)$. Let Y_v be the set of vertices $y \in V(D)$ for which there exists a vertex $x \in X_v$ such that $(y, x) \in E(D)$. Let Y'_v be a set containing a copy y' of each vertex $y \in Y$. Let H^+_v be the bipartite graph on the vertex-set $X_v \cup Y'_v$ which for all $x \in X_v$ and $y' \in Y'_v$ contains the edge $\{x, y'\}$ if and only if $(y, x) \in E(D)$. Let $\mathsf{match}^+(v)$ be the size of a maximum matching in H^+_v. Then, the following reduction rule is safe.

▶ **Reduction Rule 4.3.** *If there exists a vertex $v \in V(D)$ such that $\mathsf{match}^+(v) > 2(k+1)$, remove v from D and decrement k by 1.*

We also need the following rule, proved by Mnich and Leeuwen [25].

▶ **Reduction Rule 4.4.** *Let $P = (w_0, \cdots, w_\ell)$ be an induced directed path, that is for all $i \in [l-1]$ $d^-(w_i) = d^+(w_i) = 1$, with $\ell > k+2$ in D. Then, delete w_1 from D and add the edge (w_0, w_2).*

Consider some hypothetical solution S (if such a solution exists). Let s and t denote the source and sink, respectively, of the pumpkin $D \setminus S$. Let A (or B) denote the set of out-neighbors (resp. in-neighbors) of s (resp. t) in the pumpkin. Clearly, $|A| = |B|$. Let $C = V(D) \setminus (S \cup A \cup B \cup \{s, t\})$. Next, we prove a series of useful claims relating to S.

▶ **Lemma 11 (*).** (i) *Every vertex in $\{s\} \cup A \cup B \cup C$ has in-degree (in D) at most $k+1$, and* (ii) *every vertex in $\{t\} \cup A \cup B \cup C$ has out-degree (in D) at most $k+1$.*

▶ **Lemma 12 (*).** *For any vertex $v \in V(D)$, $|N^-(v) \cap C|, |N^+(v) \cap C| \leq 2(k+1)$.*

The set of in-neighbors (or out-neighbors) of any vertex $v \in V(D)$ is contained in $A \cup B \cup C \cup S \cup \{s, t\}$. Since $|A| \leq d^+(s)$, $|B| \leq d^-(t)$ and $|S| \leq k$, Lemma 12 gives us the following corollary.

▶ **Corollary 13.** *For any vertex $v \in V(D)$, $d^-(v), d^+(v) \leq 3k + d^+(s) + d^-(t) + 4$.*

We further strengthen this corollary to obtain the following result.

▶ **Lemma 14 (*).** *For any vertex $v \in V(D)$, $d^-(v), d^+(v) \leq \min\{4k + 2d^+(s) + 4, 4k + 2d^-(t) + 4\}$.*

Let $M = \max_{v \in V(D)}\{d^+(v), d^-(v)\}$. The next corollary (derived from Lemma 14) and rule will bring us to the main goal of this subsection, summarized in Assumption 17 below.

▶ **Corollary 15 (*).** *If $M > 6k+6$, then $s \in \mathsf{HO}$ and $t \in \mathsf{HI}$.*

▶ **Reduction Rule 4.5.** *If $|V(D)| > 2k^2M + 4kM + k + 2$, return (D, k) is a NO-instance.*

▶ **Lemma 16 (*).** *Reduction Rule 4.5 is safe.*

By Rule 4.5, if $M \leq 6k+6$, we obtain the desired kernel. Thus, by Corollary 15, we have the following observation.

▶ **Assumption 17.** *From now on, we can assume that if a solution exists, in the resulting pumpkin the source belongs to HO and the target belongs to HI.*

Next, it will be convenient to assume that HI and HO are disjoint sets. To this end, we apply the following rule exhaustively, where safeness follows directly from Lemma 11.

▶ **Reduction Rule 4.6.** *Remove all vertices in $HI \cap HO$ and decrease k by $|\mathsf{HI} \cap \mathsf{HO}|$.*

We will also assume that the following rule has been applied exhaustively. This assumption will be used at the end of the following subsection (in the proof of Lemma 25).

▶ **Reduction Rule 4.7.** *If there exists a vertex $v \notin \mathsf{HI} \cup \mathsf{HO}$ such that $N^-(v) \cap (V(D) \setminus \mathsf{HI}) = \emptyset$ or $N^+(v) \cap (V(D) \setminus \mathsf{HO}) = \emptyset$, delete v from D and decrease k by 1.*

▶ **Lemma 18 (*).** *Reduction Rule 4.7 is safe.*

4.2 Marking Approach

We are now ready to present our marking approach, handling instances to which Assumption 17 applies and none of the rules in Section 4.1 is applicable. Let $\mathcal{P}^*$ is the set of connected components in $D \setminus (\mathsf{HO} \cup \mathsf{HI})$ that are directed paths whose internal vertices have in-degree 1 and out-degree 1 in D, and let V^* be the union of the vertex-sets of the paths in $\mathcal{P}^*$. It turns out that by relying on Lemma 12 and Rule 4.4, one can directly bound the number of vertices in $V(D) \setminus V^*$ by $\mathcal{O}(k^3)$, assuming that the input instance is a YES-instance (see the proof of Lemma 23). However, bounding the size of V^* is more tricky, and our marking process is devoted to this cause. In this process, we will mark $\mathcal{O}(k^3)$ vertices from V^*, and prove that because we are handling instances to which Assumption 17 applies, all of the vertices that are not marked are essentially irrelevant. We will perform two "rounds" of marking. Roughly speaking, for each pair of vertices in HO (or HI) the first round aims to capture enough vertices of paths that describe the relation between the vertices in this pair, or, more precisely, why one of the vertices of the pair is a "better choice" than the other when one should decide which vertex (from HO) is the source of the pumpkin. However, this round is not sufficient, since some vertices in HO (or HI) have conflicts (independent of the other vertices in $\mathsf{HO} \cup \mathsf{HI}$) relating to the endpoints of the paths in $\mathcal{P}^*$. In the second round of marking, for each vertex in $\mathsf{HO} \cup \mathsf{HI}$, we mark enough vertices from these problematic paths.

First Round of Marking. Towards the performance of the first round, we need the following notations. For each vertex $v \in V(D) \setminus (\mathsf{HI} \cup \mathsf{HO})$, let $\widehat{P}(v)$ denote the connected component in $D \setminus (\mathsf{HI} \cup \mathsf{HO})$ containing v. For each $s \in \mathsf{HO}$, let $\widehat{N}(s)$ denote the set of each out-neighbor $v \in V(D) \setminus (\mathsf{HI} \cup \mathsf{HO})$ of s such that $\widehat{P}(v) \in \mathcal{P}^*$ and the first vertex of (the directed path) $\widehat{P}(v)$ is v. Symmetrically, for each $t \in \mathsf{HI}$, let $\widehat{N}(t)$ denote the set of each in-neighbor $v \in V(D) \setminus (\mathsf{HI} \cup \mathsf{HO})$ of t such that $\widehat{P}(v) \in \mathcal{P}^*$ and the last vertex of $\widehat{P}(v)$ is v. By Rule 4.6, $\mathsf{HI} \cap \mathsf{HO} = \emptyset$, and therefore these notations are well defined (i.e., we have not defined $\widehat{N}$ twice for the same vertex). Given $u \in (\mathsf{HI} \cup \mathsf{HO})$, we also denote $\widehat{\mathcal{P}}(u) = \{\widehat{P}(v) \mid v \in \widehat{N}(u)\}$. Observe that the paths in $\widehat{\mathcal{P}}(u)$ are pairwise vertex-disjoint.

Next, we identify enough vertices from paths that capture the relation between each pair of vertices in HO (or HI). For each pair $(s, s') \in \mathsf{HO} \times \mathsf{HO}$, let $\widehat{MK}_P(s, s')$ be an arbitrarily chosen set of minimal size of paths from $\widehat{\mathcal{P}}(s) \setminus \widehat{\mathcal{P}}(s')$ that together contain at least $k + 1$ vertices not having s' as an in-neighbor. In this context, observe that only the last vertex on a path in $\widehat{\mathcal{P}}(s) \setminus \widehat{\mathcal{P}}(s')$ can have s' as an in-neighbor. In this case, the path must contain at least two vertices (since its first vertex cannot have s' as an in-neighbor), and while we insert the entire path into $\widehat{MK}_P(s, s')$, its last vertex is not "counted" when we aim to obtain at least $k + 1$ vertices not having s' as an in-neighbor. If there are not enough paths to obtain at least $k + 1$ such vertices, let $\widehat{MK}_P(s, s') = \widehat{\mathcal{P}}(s) \setminus \widehat{\mathcal{P}}(s')$. Symmetrically, for each pair $(t, t') \in \mathsf{HI} \times \mathsf{HI}$, let $\widehat{MK}_P(t, t')$ be an arbitrarily chosen set of minimal size of paths from

$\widehat{\mathcal{P}}(t) \setminus \widehat{\mathcal{P}}(t')$ that together contain at least $k+1$ vertices not having t' as an out-neighbor. If there are not enough paths, let $\widehat{MK}_P(t,t') = \widehat{\mathcal{P}}(t) \setminus \widehat{\mathcal{P}}(t')$.

Finally, given a pair $(v,v') \in (\mathsf{HO} \times \mathsf{HO}) \cup (\mathsf{HI} \times \mathsf{HI})$, let $\widehat{MK}(v,v')$ denote the union of the vertex-sets of the paths in $\widehat{MK}_P(v,v')$. We have the following claim.

▶ **Lemma 19 (*).** *For each pair $(v,v') \in (HO \times HO) \cup (HI \times HI)$, $|\widehat{MK}(v,v')| \leq 3(k+1)$.*

Second Round of Marking. We proceed to the second round of marking. For this purpose, we need the following notation. For each $u \in \mathsf{HI} \cup \mathsf{HO}$, let $\widetilde{MK}_P(u)$ denote the set of each directed path in $\mathcal{P}^*$ whose first and last vertices are both neighbors of u.

▶ **Reduction Rule 4.8.** *If there exists $u \in HI \cup HO$ such that $|\widetilde{MK}_P(u)| > k+1$, delete u from D and decrease k by 1.*

▶ **Lemma 20 (*).** *Reduction Rule 4.8 is safe.*

For each $u \in \mathsf{HI} \cup \mathsf{HO}$, let $\widetilde{MK}(u)$ be the union of the vertex-sets of the paths in $\widetilde{MK}_P(u)$. Since at this point, Rules 4.4 and 4.8 are not applicable, we have the following lemma.

▶ **Lemma 21.** *For each $u \in HI \cup HO$, $|\widetilde{MK}(u)| \leq (k+1)(k+2)$.*

The Size of the Kernel. For the sake of abbreviation, we define the following sets.

- $MK_{\mathcal{P}} = (\bigcup_{(u,u') \in (\mathsf{HO}\times\mathsf{HO})\cup(\mathsf{HI}\cup\mathsf{HI})} \widehat{MK}_P(u,u')) \cup (\bigcup_{u \in \mathsf{HO}\cup\mathsf{HI}} \widetilde{MK}_P(u))$, and
- $MK = (\bigcup_{(u,u') \in (\mathsf{HO}\times\mathsf{HO})\cup(\mathsf{HI}\cup\mathsf{HI})} \widehat{MK}(u,u')) \cup (\bigcup_{u \in \mathsf{HO}\cup\mathsf{HI}} \widetilde{MK}(u))$.

By Lemmas 19 and 21, and since Rule 4.1 is not applicable, we bound $|MK|$ as follows.

▶ **Lemma 22.** $|MK| \leq 2 \cdot (3(k+1)^3 + (k+1)^2(k+2)) \leq 8(k+2)^3$.

Let V^R denote the set of unmarked vertices in V^*, i.e., $V^* \setminus MK$. We construct the graph D' by removing from D all of the vertices in V^R, adding a set N_{k+2} of $k+2$ new vertices, and for each of the new vertices, adding an edge from each vertex in HO as well as an edge to each vertex in HI. If $V(D')$ contains at most $2k+4$ vertices, add to it one-by-one a vertex-set of a path in $\mathcal{P}^*$ until its size becomes at least $2k+5$ (by Lemma 4.4, the size will not exceed $3k+6$, and because $|V(D)| \geq k^3$, we will reach the desired size).

▶ **Lemma 23 (*).** *If $|V(D')| > 30(k+2)^3$, (D',k) is a NO-instance of* PVDS.

Correctness. Finally, Theorem 9 follows from Lemma 23 and the two lemmas below.

▶ **Lemma 24 (*).** *If (D,k) is a YES-instance then (D',k) is a YES-instance.*

▶ **Lemma 25.** *If (D',k) is a YES-instance then (D,k) is a YES-instance.*

Proof. Let S be a solution to (D',k). Let s and t be the source and target, respectively, of the pumpkin $D' \setminus S$. Because of the set N_{k+2} of $k+2$ vertices added to D' at its construction, and since $|S| \leq k$, $s \in \mathsf{HO}$ and $t \in \mathsf{HI}$. Moreover, by the definition of HO and HI, $(\mathsf{HO} \cup \mathsf{HI}) \setminus \{s,t\} \subseteq S$. We can also assume that S does not contain any vertex added to D' at its construction since by removing such a vertex from S, we still have a pumpkin. Our goal will be to show that S is also a solution to (D,k), which will imply that the lemma is correct. To this end, we will show that $D \setminus S$ is a pumpkin with source s and sink t.

First, note that we can assume that in $D \setminus S$ there exists a path from s to t. Indeed, if this is not true, then $D' \setminus S$ consists only of s, t and newly added vertices. That is, $V(D')$

contains at most $2k+4$ vertices, which contradicts its construction. By the definition of $\mathcal{P}^*$, each path in $\mathcal{P}^*$ has only internal vertices that have in-degree 1 and out-degree 1 in D , and its endpoints can only be adjacent to vertices in $\mathsf{HI} \cup \mathsf{HO}$ and in the path itself. Thus, to prove the lemma, it is sufficient to show that for each path in $\mathcal{P}^* \setminus MK_{\mathcal{P}}$, its first vertex has s as an ingoing neighbor, its last vertex has t as an out-neighbor, and if it contains at least two vertices, its first vertex is not a neighbor of t and its last vertex is not a neighbor of s.

Consider some path $P \in \mathcal{P}^* \setminus MK_{\mathcal{P}}$. First suppose, by way of contradiction, that the first vertex v of P does not have s as an in-neighbor. Because Rule 4.7 is not applicable, v has at least one in-neighbor $s' \in \mathsf{HO}$. Thus, since $v \notin MK$, $MK(s', s)$ contains at least $k+1$ vertices that are not out-neighbors of s and such that each of them belongs to a path in $\mathcal{P}^*$ whose first vertex is not an out-neighbor of s. The vertices in $MK(s', s)$ belong to D'. Since $|S| \leq k$, at least one of these vertices, say some u, should belong to the pumpkin $D' \setminus S$. However, in $D' \setminus ((\mathsf{HI} \cup \mathsf{HO}) \setminus \{s\})$, which is a supergraph of $D' \setminus S$, u cannot be reached from s, which contradicts the fact that $D' \setminus S$ is a pumpkin. Symmetrically, it is shown that the last vertex of P has t as an out-neighbor.

Now assume that P contains at least two vertices. Suppose, by way of contradiction, that the first vertex of P has t as a neighbor. We have already shown that the last vertex of P is also a neighbor of t, and therefore $P \in \widetilde{MK}_P(t)$. However, $\widetilde{MK}_P(t) \subseteq MK_{\mathcal{P}}$, which contradicts the fact that $P \in \mathcal{P}^* \setminus MK_{\mathcal{P}}$. Symmetrically, it is shown that the last vertex of P does not have s as a neighbor, concluding the proof of the lemma. ◀

References

1 F. N. Abu-Khzam. A kernelization algorithm for d-Hitting Set. *JCSS*, 76(7):524–531, 2010.

2 V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIDMA*, 12(3):289–297, 1999.

3 J. Bang-Jensen, A. Maddaloni, and S. Saurabh. Algorithms and kernels for feedback set problems in generalizations of tournaments. *Algorithmica*, pages 1–24, 2015.

4 R. Bar-Yehuda, D. Geiger, J. Naor, and R. M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SICOMP*, 27(4):942–959, 1998.

5 A. Becker and D. Geiger. Optimization of pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artif. Intell.*, 83(1):167–188, 1996.

6 Y. Cao, J. Chen, and Y. Liu. On feedback vertex set new measure and new structures. In *SWAT*, pages 93–104, 2010.

7 C. Chekuri and V. Madan. Constant factor approximation for subset feedback problems via a new LP relaxation. In *SODA*, pages 808–820, 2016.

8 J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved algorithms for feedback vertex set problems. *JCSS*, 74(7):1188–1198, 2008.

9 J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5), 2008.

10 R. H. Chitnis, M. Cygan, M. T. Hajiaghayi, and D. Marx. Directed subset feedback vertex set is fixed-parameter tractable. *TALG*, 11(4):28, 2015.

11 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

12 M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *FOCS*, pages 150–159, 2011. `doi:10.1109/FOCS.2011.23`.

13 M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIDMA*, 27(1):290–309, 2013. `doi:10.1137/110843071`.

14 R. Diestel. *Graph Theory, 4th Edition.* Springer, 2012.

15 M. Dom, J. Guo, F. Hüffner, R. Niedermeier, and A. Truß. Fixed-parameter tractability results for feedback set problems in tournaments. *JDA*, 8(1):76–86, 2010. `doi:10.1016/j.jda.2009.08.001`.

16 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity.* Springer, 2013.

17 P. Erdős and L. Pósa. On independent circuits contained in a graph. *Canad. J. Math*, 17:347–352, 1965.

18 G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.

19 V. Guruswami and E. Lee. Inapproximability of H-transversal/packing. In *APPROX/RANDOM*, pages 284–304, 2015.

20 N. Kakimura, K. Kawarabayashi, and Y. Kobayashi. Erdös-Pósa property and its algorithmic applications: parity constraints, subset feedback set, and subset packing. In *SODA*, pages 1726–1736, 2012.

21 N. Kakimura, K. Kawarabayashi, and D. Marx. Packing cycles through prescribed vertices. *J. Comb. Theory, Ser. B*, 101(5):378–381, 2011.

22 K. Kawarabayashi and Y. Kobayashi. Fixed-parameter tractability for the subset feedback set problem and the S-cycle packing problem. *J. Comb. Theory, Ser. B*, 102(4):1020–1034, 2012. `doi:10.1016/j.jctb.2011.12.001`.

23 K. Kawarabayashi, D. Král, M. Krcál, and S. Kreutzer. Packing directed cycles through a specified vertex set. In *SODA*, pages 365–377, 2013.

24 T. Kociumaka and M. Pilipczuk. Faster deterministic feedback vertex set. *IPL*, 114(10):556–560, 2014. `doi:10.1016/j.ipl.2014.05.001`.

25 M. Mnich and E. J. van Leeuwen. Polynomial kernels for deletion to classes of acyclic digraphs. In *STACS*, pages 1–13, 2016.

26 M. Pontecorvi and P. Wollan. Disjoint cycles intersecting a set of vertices. *J. Comb. Theory, Ser. B*, 102(5):1134–1141, 2012.

27 V. Raman, S. Saurabh, and C. R. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *TALG*, 2(3):403–415, 2006. `doi:10.1145/1159892.1159898`.

28 B. A. Reed, N. Robertson, P. D. Seymour, and R. Thomas. Packing directed circuits. *Combinatorica*, 16(4):535–554, 1996.

29 P. D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.

30 P. D. Seymour. Packing circuits in eulerian digraphs. *Combinatorica*, 16(2):223–231, 1996.

31 S. Thomassé. A $4k^2$ kernel for feedback vertex set. *TALG*, 6(2), 2010.

32 M. Wahlström. Half-integrality, LP-branching and FPT algorithms. In *SODA*, pages 1762–1781, 2014.