Parameterized Algorithms for Generalizations of Directed Feedback Vertex Set*

Alexander Göke † Dániel Marx‡ Matthias Mnich§

Abstract

The Directed Feedback Vertex Set (DFVS) problem takes as input a directed graph G and seeks a smallest vertex set S that hits all cycles in G. This is one of Karp's 21 NP-complete problems. Resolving the parameterized complexity status of DFVS was a long-standing open problem until Chen et al. [STOC 2008, J. ACM 2008] showed its fixed-parameter tractability via a $4^k k! n^{\mathcal{O}(1)}$-time algorithm, where $k = |S|$.

Here we show fixed-parameter tractability of two generalizations of DFVS:

- Find a smallest vertex set S such that every strong component of $G - S$ has size at most s: we give an algorithm solving this problem in time $4^k(ks + k + s)! \cdot n^{\mathcal{O}(1)}$. This generalizes an algorithm by Xiao [JCSS 88, pp. 260-270, 2017] for the undirected version of the problem.
- Find a smallest vertex set S such that every non-trivial strong component of $G - S$ is 1-out-regular: we give an algorithm solving this problem in time $2^{\mathcal{O}(k^3)} \cdot n^{\mathcal{O}(1)}$.

We also solve the corresponding arc versions of these problems by fixed-parameter algorithms.

Keywords. Fixed-parameter algorithms; directed feedback vertex set

1 Introduction

The Directed Feedback Vertex Set (DFVS) problem is that of finding a smallest vertex set S in a given directed graph G such that $G - S$ is a directed acyclic graph. This problem is among the most classical problems in algorithmic graph theory. It is one of the 21 NP-complete problems on Karp's famous list [17].

Consequently, the DFVS problem has long attracted researchers in approximation algorithms. The current best known approximation factor that can be achieved in polynomial time for n-vertex directed graphs with optimal fractional solution value τ^* is $\mathcal{O}(\min\{\log \tau^* \log\log \tau^*, \log n \log\log n\})$ due to Seymour [23], Even et al. [9] and Even et al. [10]. On the negative side, Karp's NP-hardness reduction shows the problem to be APX-hard, which rules out the existence of a polynomial-time approximation scheme (PTAS) assuming P $\neq$ NP. Assuming the Unique Games Conjecture, the DFVS problem does not admit a polynomial-time $\mathcal{O}(1)$-approximation [15, 16, 24].

*The full version of this paper appears in *Discrete Optimization* [14] at https://doi.org/10.1016/j.disopt.2022.100740. A preliminary version appeared in the Proceedings of the 11th International Conference on Algorithms and Complexity [12].

†Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany. alexander.goeke@tuhh.de. Supported by DFG grant MN 59/1-1.

‡CISPA Helmholtz Center for Information Security, Saarbrücken, Germany. dmarx@mpi-inf.mpg.de. Supported by ERC Consolidator Grant SYSTEMATICGRAPH (755978).

§Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany. matthias.mnich@tuhh.de. Supported by DFG grant MN 59/4-1.

The DFVS problem has also received a significant amount of attention from the perspective of parameterized complexity. The main parameter of interest there is the optimal solution size $k = |S|$. The problem can easily be solved in time $n^{\mathcal{O}(k)}$ by enumerating all k-sized vertex subsets $S \subseteq V(G)$ and then seeking a topological order of $G - S$. The interesting question is thus whether the DFVS problem is *fixed-parameter tractable* with respect to k, which is to devise an algorithm with run time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f depending only on k. It was a long-standing open problem whether DFVS admits such an algorithm. The question was finally resolved by Chen et al. who gave a $4^k k! k^4 \cdot \mathcal{O}(nm)$-time algorithm for directed graphs with n vertices and m arcs. Recently, an algorithm for DFVS with run time $4^k k! k^5 \cdot \mathcal{O}(n+m)$ was given by Lokshtanov et al. [19]. It is well-known that the *arc* deletion variant is parameter-equivalent to the *vertex* deletion variant and hence Directed Feedback Arc Set (DFAS) can also be solved in time $4^k k! k^5 \cdot \mathcal{O}(n+m)$.

Once the breakthrough result for DFVS was obtained, the natural question arose how much further one can push the boundary of (fixed-parameter) tractability. On the one hand, Chitnis et al. [6] showed that the generalization of DFVS where one only wishes to hit cycles going through a specified subset of nodes of a given directed graph is still fixed-parameter tractable when parameterized by solution size. On the other hand, Lokshtanov et al. [20] showed that finding a smallest set of vertices of hitting only the *odd* directed cycles of a given directed graph is $\mathsf{W}[1]$-hard parameterized by solution size; hence, the problems not fixed-parameter tractable unless $\mathsf{FPT} = \mathsf{W}[1]$. Similarly, Göke et al. [13] showed that hitting all cycles of length at least ℓ in a given digraph G is NP-hard for constant parameter solution size, but is fixed-parameter tractable if ℓ is taken as additional parameter. In conclusion, it is a valuable research direction to understand which abstractions of DFVS are fixed-parameter tractable parameterized by solution size and which ones are not (assuming $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{FPT} \neq \mathsf{W}[1]$).

Our contributions. One such abstraction of DFVS whose parameterized complexity (parameterized by solution size) is still open is the following: In the Eulerian Strong Component Arc (Vertex) Deletion problem, one is given a directed multigraph G, and asks for a set S of at most k vertices such that every strong component of $G - S$ is Eulerian, that is, every vertex has the same in-degree and out-degree within its strong component. The arc version of this problem was suggested by Cechlárová and Schlotter [3] in the context of housing markets. Marx [21] explicitly posed determining the parameterized complexity of Eulerian Strong Component Vertex Deletion as an open problem. Notice that these problems generalize the DFAS/DFVS problems, where each strong component of $G - S$ has size one and thus is Eulerian.

Theorem 1.1. Eulerian Strong Component Vertex Deletion *is* NP*-hard and* $\mathsf{W}[1]$*-hard parameterized by solution size* k*, even for* $(k+1)$*-strong directed graphs.*

Alas, we are unable to determine the parameterized complexity of Eulerian Strong Component Arc Deletion, which appears to be more challenging. Hence, we consider two natural generalizations of DFAS which may help to gain better insight into the parameterized complexity of that problem.

First, we consider the problem of deleting a set of k vertices or arcs from a given directed graph such that every strong component has size at most s. Thus, the DFVS/DFAS problems corresponds to the special case when $s = 1$. Formally, the problem Bounded Size Strong Component Vertex (Arc) Deletion takes as input a directed multigraph G and integers k, s, and seeks a set S of at most k arcs or vertices such that every strong component of $G - S$ has size at most s.

The *undirected* case of Bounded Size Strong Component Vertex (Arc) Deletion was studied recently. There, one wishes to delete at most k vertices of an undirected n-vertex

graph such that each connected component of the remaining graph has size at most s. For s being constant, Kumar and Lokshtanov [18] obtained a kernel of size $2sk$ that can be computed in $n^{\mathcal{O}(s)}$ time; note that the degree of the run time in the input size n depends on s and is thus not a fixed-parameter algorithm. For general s, there is a $9sk$-sized kernel computable in time $\mathcal{O}(n^4 m)$ by Xiao [25]. The directed case—which we consider here—generalizes the undirected case by replacing each edge by arcs in both directions.

Our main result here is to solve the directed case of the problem by a fixed-parameter algorithm:

Theorem 1.2. *There is an algorithm that solves* Bounded Size Strong Component Vertex (Arc) Deletion *in time* $4^k(ks + k + s)! \cdot n^{\mathcal{O}(1)}$ *for* n*-vertex directed multigraphs* G *and parameters* $k, s \in \mathbb{N}$.

In particular, our algorithm exhibits the same asymptotic dependence on k as does the algorithm by Chen et al. [4] for the DFVS/DFAS problem, which corresponds to the special case $s = 1$.

Another motivation for this problem comes from the k-linkage problem, which asks for k pairs of terminal vertices in a directed graph whether they can be connected by k mutually arc-disjoint paths. The k-linkage problem is NP-complete already for $k = 2$ [11]. Recently, Bang-Jensen and Larsen [1] solved the k-linkage problem in directed graphs where strong components have size at most s. Thus, finding induced subgraphs with strong components of size at most s can be of interest in computing k-linkages.

Our second problem is that of deleting a set of k arcs or vertices from a given directed graph such that each remaining strong component is r_C*-out-regular*, meaning that every vertex has out-degree exactly r_C in its strong component C, for $r_C \leq 1$. So in particular, every strong component is Eulerian, as in the Eulerian Strong Component Arc Deletion problem. Observe that in the DFVS/DFAS problem we delete k vertices or arcs from a given directed graph such that each remaining strong component is 0-out-regular (trivial). Formally, we consider the 1-Out-Regular Vertex (Arc) Deletion problem in which for a given directed multigraph G and integer k, we seek a set S of at most k vertices (arcs) such that every component C of $G - S$ is r_C-out-regular with $r_C \in \{0, 1\}$. Note that this problem is equivalent to deleting a set S of at most k vertices (arcs) such that every non-trivial (consisting of more than one vertex) strong component of $G - S$ is an induced directed cycle. In contrast to Eulerian Strong Component Vertex Deletion, the 1-Out-Regular Vertex (Arc) Deletion problem *is* monotone, in that every superset of a solution is again a solution: if we delete an additional arc or vertex that breaks a strong component that is an induced cycle into several strong components, then each of these newly created strong components is trivial.

Our result for this problem reads as follows:

Theorem 1.3. *There is an algorithm solving* 1-Out-Regular Vertex (Arc) Deletion *in time* $2^{\mathcal{O}(k^3)} \cdot \mathcal{O}(n^4)$ *for* n*-vertex directed graphs* G *and parameter* $k \in \mathbb{N}$.

Notice that for Bounded Size Strong Component Vertex (Arc) Deletion and 1-Out-Regular Vertex (Arc) Deletion, there are infinitely many instances for which solutions are arbitrarily smaller than those for DFVS (DFAS), and for any instance they are never larger. Moreover, the DFVS (DFAS) problem is the special case of Bounded Size Strong Component Vertex (Arc) Deletion with $s = 1$. Therefore, our algorithms strictly generalize the one by Chen et al. [4] for DFVS (DFAS). As a possible next step towards resolving the parameterized complexity of Eulerian Strong Component Arc Deletion, one may generalize our algorithm for 1-Out-Regular Arc Deletion to r-Out-Regular Arc Deletion for arbitrary r.

We continue the paper by collecting notions and notations, in Section 2. What follows is a description of three main tools shared by both fixed-parameter algorithms, in Section 3. Next, in Section 4 we describe the parameterized hardness reduction for the Eulerian strong component vertex deletion problem. Thereafter, in Section 5 and Section 6, we give fixed-parameter algorithms for vertex deletion variants only. In Section 7, we reduce the arc deletion variants to the vertex deletion variants. We conclude in Section 8.

2 Notions and Notations

We consider finite directed graphs (or digraphs) G with vertex set $V(G)$ and arc set $A(G)$. We allow multiple arcs and arcs in both directions between the same pairs of vertices. For each vertex $v \in V(G)$, its *out-degree* in G is the number $d^+_G(v)$ of arcs of the form (v, w) for some $w \in V(G)$, and its *in-degree* in G is the number $d^-_G(v)$ of arcs of the form (w, v) for some $w \in V(G)$. A vertex v is *balanced* if $d^+_G(v) = d^-_G(v)$. A digraph G is *balanced* if every vertex $v \in V(G)$ is balanced. A *walk* (of length ℓ) in G is a subgraph P of G which consists of vertices $v_0, \ldots, v_\ell$ and arcs (v_i, v_{i+1}) for $i = 0, \ldots, \ell - 1$; we also call P a *v_0-v_ℓ-walk* and refer to v_0, v_ℓ as the *endpoints* of P, whereas its other vertices are the *internal vertices*. If $v_0 = v_\ell$, the walk is *closed*. In a walk, it is allowed that some vertices or arc appear multiple times; if all vertices (and hence all arcs) of a walk are distinct, the walk is a *path*.

For each subset $V' \subseteq V(G)$, the subgraph induced by V' is the graph $G[V']$ with vertex set V' and arc set $\{(u, v) \in A(G) \mid u, v \in V'\}$. For any set X of arcs or vertices of G, let $G - X$ denote the subgraph of G obtained by deleting the elements of X from G. For subgraphs G' of G and vertex sets $X \subseteq V(G)$ let $R^+_{G'}(X)$ denote the set of vertices that are *reachable* from X in G', i.e. vertices to which there is a path from some vertex in X. For an s-t-walk P and a t-q-walk R we denote by $P \circ R$ the *concatenation* of these paths, i.e. the s-q-walk resulting from first traversing P and then R.

Let G be a digraph. Then G is 1-*out-regular* if every vertex has out-degree exactly 1. Further, G is called *strong* if either G consists of a single vertex (then G is called *trivial*), or for any distinct $u, v \in V(G)$ there is a directed path from u to v. A *strong component* of G is an inclusion-maximal strong induced subgraph of G. Also, G is *t-strong* for some $t \in \mathbb{N}$ if for any $X \subseteq V(G)$ with $|X| < t$, $G - X$ is strong. We say that G is *weakly connected* if its underlying undirected graph $\langle G \rangle$ is connected. Further, G is *Eulerian* if there is a closed walk in G using each arc exactly once. Finally, G is *complete* if there any two of its vertices are connected by an arc in both directions.

Definition 2.1. *For disjoint non-empty vertex sets X, Y of a digraph G, an arc or vertex set S is an* $X \to Y$-separator *if S is disjoint from $X \cup Y$ and there is no path from X to Y in $G - S$.*

An $X \to Y$-separator S is minimal *if no proper subset of S is an $X \to Y$-separator. An $X \to Y$-separator S is* important *if there is no $X \to Y$-separator S' with $|S'| \leq |S|$ and $R^+_{G-S}(X) \subset R^+_{G-S'}(X)$.*

Proposition 2.2 ([5])**.** *Let G be a digraph and let $X, Y \subseteq V(G)$ be disjoint non-empty vertex sets. For every $p \geq 0$ there are at most 4^p important $X \to Y$-separators of size at most p, all of which can be enumerated in time $4^p \cdot n^{\mathcal{O}(1)}$.*

3 Tools for Abstractions of DFVS/DFAS Problems

We collect tools used by our algorithms for solving abstractions of DFVS and DFAS. These abstractions take as input a digraph G and an integer k, and seek a set S of at most k vertices (resp. arcs) such that every strong component of $G - S$ satisfies some hereditary property Π; we call S a *solution* of (G, k). If (G, k) admits a solution, we call it a "yes"-instance; else, we call it a "no"-instance.

Iterative Compression. We use the standard technique of iterative compression. For this, we label the vertices of the input digraph G arbitrarily by $v_1, \ldots, v_n$, and set $G_i = G[\{v_1, \ldots, v_i\}]$. We start with G_1 and the solution $S_1 = \{v_1\}$. As long as $|S_i| < k$, we can set $S_{i+1} = S_i \cup \{v_{i+1}\}$ and continue. As soon as $|S_i| = k$, the set $T_{i+1} = S_i \cup \{v_{i+1}\}$ is a solution of $(G_{i+1}, k+1)$. The *compression variant* of our abstraction (for fixed Π) then takes as input a digraph G and a solution T of $(G, k+1)$, and seeks a solution S of (G, k) or decides that none exists.

We call an algorithm for the compression variant on (G_{i+1}, T_{i+1}) to obtain a solution S_{i+1} of (G_{i+1}, k), or find out that (G_{i+1}, k) does not have a solution, but then neither has G (as Π is hereditary). By at most n calls to this algorithm we can deduce the (non-)existence of a solution of the original input instance $(G_n = G, k)$.

Disjoint solution. Given an input (G, T) to the compression variant of the abstraction, the next step is to ask for a solution S of (G, k) that is disjoint from the given solution T of $(G, k+1)$. This assumption can be made by guessing the intersection $T' = S \cap T$, and deleting those vertices from G. Since T has $k+1$ elements, this step creates 2^{k+1} candidates T'. The *disjoint compression variant* of our abstraction then takes as input a digraph $G - T'$, a solution $T \setminus T'$ of $(G, k+1-|T'|)$, and seeks a solution S' of $(G, k-|T'|)$ which is disjoint from $T \setminus T'$.

Covering the shadow of a solution. Given a digraph G with solution T of $(G, k+1)$, the "shadow" of a hypothetical solution S of (G, k) is the set of those vertices that are disconnected from T (in either direction) after the removal of S from G. A common idea of several fixed-parameter algorithms on digraphs is to first ensure that there is a solution whose shadow is empty, as finding such a *shadowless* solution can be a significantly easier task. A generic framework by Chitnis et al. [6] shows that for special types of problems as defined below, one can invoke the random sampling of important separators technique and obtain a set Z which is disjoint from a minimum solution and covers its shadow, i.e. the shadow is contained in Z. What one does with this set, however, is problem-specific. Typically, given such a set, one can use (some problem-specific variant of) the "torso operation" to find an equivalent instance that has a shadowless solution. Therefore, one can focus on the simpler task of finding a shadowless solution, or more precisely, finding any solution under the guarantee that a shadowless solution exists.

Definition 3.1 (shadow)**.** *Let G be a digraph and let $T, S \subseteq V(G)$. A vertex $v \in V(G)$ is* in the forward shadow $f_{G,T}(S)$ of S (with respect to T) *if S is a $T \to \{v\}$-separator in G, and v is* in the reverse shadow $r_{G,T}(S)$ of S (with respect to T) *if S is a $\{v\} \to T$-separator in G.*

A vertex is in the shadow of S *if it is in the forward or reverse shadow of S.*

Note that S itself is not in the shadow of S by definition of separators.

Definition 3.2 (T-connected and $\mathcal{F}$-transversal)**.** *Let G be a digraph, let $T \subseteq V(G)$ and let $\mathcal{F}$ be a set of subgraphs of G. We say that $\mathcal{F}$ is* T-connected *if for every $F \in \mathcal{F}$, each vertex of F can reach some and is reachable by some (maybe different) vertex of T by a walk completely contained in F. For a set $\mathcal{F}$ of subgraphs of G, an* $\mathcal{F}$-transversal *is a set of vertices that intersects the vertex set of every subgraph in $\mathcal{F}$.*

Chitnis et al. [6] show how to deterministically cover the shadow of $\mathcal{F}$-transversals:

Proposition 3.3 (deterministic covering of the shadow, [6]). *Let $T \subseteq V(G)$. In time $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$ one can construct $t \leq 2^{\mathcal{O}(k^2)} \log^2 n$ sets $Z_1, \ldots, Z_t$ such that for any set of subgraphs $\mathcal{F}$ which is T-connected, if there exists an $\mathcal{F}$-transversal of size at most k then there is an $\mathcal{F}$-transversal S of size at most k that is disjoint from Z_i and such that Z_i covers the shadow of S, for some $i \leq t$.*

4 Parameterized Hardness of Vertex Deletion to Eulerian Strong Components

In this section we prove Theorem 1.1, by showing NP-hardness and W[1]-hardness of the Eulerian Strong Components Vertex Deletion problem. Before the hardness proof, we recall an equivalent characterization of Eulerian digraphs:

Lemma 4.1 (folklore). *Let G be a weakly connected digraph. Then G is Eulerian if and only if G is balanced.*

We can now state the hardness reduction, which relies on the hardness of the following problem introduced by Cygan et al. [7]. In Directed Balanced Vertex Deletion, one is given a directed multigraph G and an integer $k \in \mathbb{N}$, and seeks a set S of at most k vertices such that $G - S$ is balanced.

Proposition 4.2 ([7]). Directed Balanced Vertex Deletion *is* NP*-hard and* W[1]*-hard with parameter k.*

We will prove the hardness of Eulerian Strong Component Vertex Deletion for $(k+1)$-strong digraphs, by adding vertices ensuring this level of strong connectivity for the input digraph.

Theorem 1.1 (restated). Eulerian Strong Component Vertex Deletion *is* NP*-hard and* W[1]*-hard parameterized by solution size k, even for $(k+1)$-strong directed graphs.*

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

Let S' be a solution of (G', k) as instance of Eulerian Strong Components Vertex Deletion. As G' is $(k+1)$-strong, $G' - S'$ is strong. Moreover, S' is a solution of (G', k), so $G' - S'$ is Eulerian (because it is the only strong component). Therefore, by Lemma 4.1 every vertex of $G' - S'$ is balanced. Deleting the remaining vertices of $\{z_1, \ldots, z_{k+1}\}$ does not harm the balance of the remaining vertices, as for each $v \in V(G)$ and z_i we delete one outgoing and one incoming arc of v. Thus $G' - (S' \cup \{z_1, \ldots, z_{k+1}\}) = G - (S' \setminus \{z_1, \ldots, z_{k+1}\})$ is balanced. Hence, $S' \setminus \{z_1, \ldots, z_{k+1}\}$ is a solution of (G, k) as instance of Directed Balanced Vertex Deletion.

Let S be a solution of (G, k) as instance of Directed Balanced Vertex Deletion. Then $G - S$ is balanced, and (by construction) $G' - S$ is balanced as well. Furthermore, $G' - S$ is strong, and thus by Lemma 4.1 also Eulerian. Hence, the only strong component of $G' - S$ is Eulerian and therefore S is a solution of (G', k) as instance of Eulerian Strong Component Vertex Deletion. □

5 Fixed-Parameter Algorithm for Vertex Deletion to Strong Components of Bounded Size

In this section we show a fixed-parameter algorithm for the vertex deletion variant of Bounded Size Strong Component Vertex Deletion.

We give an algorithm that, given an n-vertex digraph G and integers k, s, decides in time $4^k(ks+k+s)! \cdot n^{\mathcal{O}(1)}$ if G has a set S of at most k vertices such that every strong component of $G - S$ has size at most s. Such a set S will be called a *solution* of the instance (G, k, s).

The algorithm first executes the general steps "Iterative Compression" and "Disjoint Solution"; it continues with a reduction to a skew separator problem.

Reduction to Skew Separator Problem Now the goal is, given a digraph G, integers $k, s \in \mathbb{N}$, and a solution T of $(G, k+1, s)$, to decide if (G, k, s) has a solution S that is disjoint from T. We solve this problem—which we call Disjoint Bounded Size Strong Component Vertex Deletion Reduction—by reducing it to finding a small "skew separator" in one of a bounded number of reduced instances.

Definition 5.1. *Let G be a digraph, and let $\mathcal{X} = (X_1, \ldots, X_t), \mathcal{Y} = (Y_1, \ldots, Y_t)$ be two ordered collections of $t \geq 1$ vertex subsets of G. A* skew separator *S for $(G, \mathcal{X}, \mathcal{Y})$ is a vertex subset of $V(G) \setminus \bigcup_{i=1}^{t}(X_i \cup Y_i)$ such that for any index pair (i, j) with $t \geq i \geq j \geq 1$, there is no path from X_i to Y_j in the graph $G - S$.*

This definition gives rise to the Skew Separator problem, which for a digraph G, ordered collections $\mathcal{X}, \mathcal{Y}$ of vertex subsets of G, and an integer $k \in \mathbb{N}$, asks for a skew separator for $(G, \mathcal{X}, \mathcal{Y})$ of size at most k. Chen et al. [4] showed:

Proposition 5.2 ([4, Thm. 3.5])**.** *There is an algorithm solving* Skew Separator *in time $4^k k \cdot \mathcal{O}(n^3)$ for n-vertex digraphs G.*

The reduction from Disjoint Bounded Size Strong Component Vertex Deletion Reduction to Skew Separator is as follows. As T is a solution of $(G, k+1, s)$, it holds that every strong component of $G - T$ has size at most s. Similarly, we can assume that every strong component of $G[T]$ has size at most s, as otherwise there is no solution S of (G, k, s) that is disjoint from T. Let $\{t_1, \ldots, t_{k+1}\}$ be a labeling of the vertices in T.

Lemma 5.3. *There is an algorithm that, given an n-vertex digraph G, integers $k, s \in \mathbb{N}$, and a solution T of $(G, k+1, s)$, in time $\mathcal{O}((ks+s-1)!) \cdot n^{\mathcal{O}(1)}$ computes a collection $\mathcal{C}$ of at most $(ks+s-1)!$ vectors $C = (C_1, \ldots, C_{k+1})$ of length $k+1$, where $t_h \in C_h \subseteq V(G)$ for $h = 1, \ldots, k+1$, such that for any solution S of (G, k, s) disjoint from T, there is a vector $C \in \mathcal{C}$ such that the strong component of $G - S$ containing t_h is exactly $G[C_h]$ for $h = 1, \ldots, k+1$.*

Proof. Fix a hypothetical solution S of (G, k, s) that is disjoint from T. The algorithm computes, for each vertex $t_h \in T$, a set $C_h \ni t_h$ of at most s vertices such that C_h induces a strong component of $G - S$. (These sets C_h must exist as S is required to be disjoint from T.) Notice that the definition of C_h must only depend on t_h but not on S. Vertices in C_h (other than t_h) may or may not belong to T, and in particular, it can be that $t_{h'} \in C_h$ for some $h' \in \{1, \ldots, k+1\} \setminus \{h\}$. Thus, for distinct $t_h, t_{h'} \in T$, sets C_h and $C_{h'}$ possibly overlap.

Intuitively, to compute C_h define a "candidate vertex for t_h" as a vertex $u \in V(G) \setminus \{t_h\}$ that can potentially belong to the same strong component of $G - S$ as t_h, for a hypothetical solution S of size at most k that is disjoint from T. We want to bound the number of candidate vertices for each $t_h \in T$, or, more precisely, the number of "candidate sets" C_h for which

$C_h \ni t_h$ can be exactly the vertex set of the strong component that contains t_h, after deleting a set $S \subseteq V(G) \setminus T$ of at most k vertices from G.

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

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

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

Let us remark that the branching algorithm produces many vectors that are inconsistent: consequently, for different h, h' the sets C_h and C'_h may overlap without being equal, since you can add a vertex in one strong component and then later add it to S; such vectors do not impact the correctness of the algorithm, and avoiding them will not reduce its asymptotic run time.

We analyze the run time of the algorithm. To construct all possible vectors $C = (C_1, \ldots, C_{k+1}) \in \mathcal{C}$, the search tree that arises from the branching has a number of leaves that is bounded by a function f of k and q only, where $q = \sum_{h=1}^{k+1}(s - |C_h^i|)$ is the sum of the remaining capacities of the C_h's. By the above branching, this function satisfies the recursion $f(k,q) \leq (s - |C_h^i|)f(k-1,q) + f(k,q-1)$, as in the first branch there are at most $s - |C_h^i|$ choices for vertex u each of which reduces the budget of $k - |S|$ by 1, and one branch which reduces the budget of q by the number of internal vertices of P which is at least 1.

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

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

where in the last inequality we used that $q \geq s - |C_i^h|$ and $k \geq 1$. Hence, the search tree has at most $(q+k)!$ leaves, each leaf corresponding to some vector $C \in \mathcal{C}$. The initial capacity q satisfies $q = (k+1)(s-1)$, and thus $|\mathcal{C}| \leq (ks + s - 1)!$. Since each branching step can be executed in polynomial time, the search tree (and hence the set $\mathcal{C}$) can be constructed in time $(q+k)! \cdot n^{\mathcal{O}(1)}$. Thus, the overall run time is $(q+k)! \cdot n^{\mathcal{O}(1)} = (ks + s - 1)! \cdot n^{\mathcal{O}(1)}$. □

Observe that for each vertex $t_h \in T$ each set C_h contains at most s vertices, and together with the run time of the algorithm in Lemma 5.3 directly implies that $\mathcal{C}$ contains at most $(ks+s-1)!$ vectors.

Armed with Lemma 5.3, we can hence restrict our search for a solution S of (G,k,s) disjoint from T to those S that additionally are "compatible" with a vector in $\mathcal{C}$. Formally, a solution S of (G,k,s) is *compatible* with a vector $C=(C_1,\ldots,C_{k+1}) \in \mathcal{C}$ if the strong component of $G-S$ containing t_h is exactly C_h for $h=1,\ldots,k+1$. For a given vector $C=(C_1,\ldots,C_{k+1})$, to determine whether a solution S of (G,k,s) disjoint from T and compatible with C exists, we create several instances of the Skew Separator problem. To this end, note that if two sets $C_h, C_{h'}$ for distinct $t_h, t'_h \in T$ overlap, then actually $C_h = C_{h'}$ (and $t_h, t'_h \in C_h$). So for each set C_h we choose exactly one (arbitrary) *representative* T*-vertex* among all T-vertices in C_h with consistent choice over overlapping (and thus equal) C_h's. Let $T' \subseteq T$ be the set of these representative vertices. For convenience, let us assume, without loss of generality, that $T' = \{t_1,\ldots,t_{|T'|}\}$. Now we generate precisely one instance $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'}, k)$ of Skew Separator for each permutation σ' of T'. The digraph G' is the same in all these instances, and is obtained from G by replacing each unique set C_h by two vertices t_h^+, t_h^- (where t_h is the representative of C_h), and connecting all vertices incoming to C_h in G by an in-arc to t_h^+ and all vertices outgoing from C_h in G by an arc outgoing from t_h^-. This way also arcs of the type (t_j^-, t_h^+) are added but none of type (t_j^-, t_h^-), (t_j^+, t_h^-) or (t_j^+, t_h^+). Notice that this operation is well-defined and yields a simple digraph G', even if $t_{h'} \in C_h$ for some distinct h, h'. The sets $\mathcal{X}_{\sigma'}$ and $\mathcal{Y}_{\sigma'}$ of "sources" and "sinks" depend on the permutation σ' with elements $\sigma'(1),\ldots,\sigma'(|T'|)$: let $\mathcal{X}_{\sigma'} = (t^-_{\sigma'(1)},\ldots,t^-_{\sigma'(|T'|)})$ and let $\mathcal{Y}_{\sigma'} = (t^+_{\sigma'(1)},\ldots,t^+_{\sigma'(|T'|)})$.

Thus, per triple $((G,k,s),T,C)$ we generate at most $|T'|! \leq |T|! = (k+1)!$ instances $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'}, k)$, the number of permutations of T'.

We establish the correctness of this reduction, in the next two lemmas:

Lemma 5.4. *If an instance (G,k,s) admits a solution S disjoint from T, compatible with C and for which $(t_{\sigma'(1)},\ldots,t_{\sigma'(|T'|)})$ is a topological order of the connected components of $G'-S$, then S forms a skew separator of size k for $(G, \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'})$.*

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

- For two vertices $t_h, t_{h'} \in T'$ with $\sigma'(h) < \sigma'(h')$, there would be a path P_1 from $t_{h'}^-$ to t_h^+ in $G'-S$. This corresponds to a $C_{h'} \to C_h$ path in $G-S$. Either there is a $C_h \to C_{h'}$ path in G, meaning that $C_h = C_{h'}$, in contradiction to our choice of T'; or the topological order σ' of strong components was incorrect (as $C_{h'}$ must be before C_h).
- The in-vertex t_h^- of the strong component containing v_h would be reachable from the out-vertex t_h^+ of this strong component in the graph $G'-S$, because then the component would contain all the vertices on this path P_2 from t_h^+ to t_h^-, and by the way we constructed C_h, the size of $|C_h \cup V(P_2)|$ in G would be at least $s+1$, contradicting that the strong component of $G'-S$ containing v_h has at most s vertices. □

Lemma 5.5. *Conversely, if S is a skew separator of $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'})$ of size at most k, then S is a solution of (G,k,s) disjoint from T and compatible with C.*

Proof. Suppose, for sake of contradiction, that S is not a solution of (G,k,s). Then there is some strong component Q in $G-S$ of size more than s. By abuse of notation, let $C = \cup_{h=1}^{k+1} C_h$. Since neither $G[C]$ (by choice of C) nor $V(G)-C$ (as subdigraph of $G[V(G) \setminus T]$) contain strong components of size more than s, this component Q must contain vertices from both C

and $V(G) \setminus C$. Let K be a closed walk of Q that intersects both $G[C]$ and $G[V(G) \setminus C]$. Such a closed walk K must exist by Q being strong.

We consider two cases:

- The closed walk K intersects a single unique component C_h. Then all other vertices of K are in $V(G) \setminus C_h$. Let t_h be the representative of C_h. As K intersects $G[V(G) \setminus C]$, K leaves and enters C_h at least once. This means that there is a walk P_1 in $G' - S$ that starts with the vertex t_h^- and ends with the vertex t_h^+, and all internal vertices of P_1 (of which there is at least one) are outside $\cup_{i=1}^{|T'|}(X_i \cup Y_i)$. But this contradicts the assumption that S is a skew separator for the tuple $(G', (t^-_{\sigma'(1)}, \ldots, t^-_{\sigma'(|T'|)}), (t^+_{\sigma'(1)}, \ldots, t^+_{\sigma'(|T'|)}))$ that should cut all walks from t_h^- to t_h^+.
- The closed walk K intersects several different components C_h. Let $(C_{h_1}, \ldots, C_{h_d}, C_{h_1})$ be the order of components that we encounter when traversing along the walk K, starting from an arbitrary component C_{h_1}, where $d > 1$. Let $(t_{h_1}, \ldots, t_{h_d}, t_{h_1})$ be the corresponding representative vertices. Then there must be an index j such that h_j occurs after $h_{j+1 \pmod{d+1}}$ in $(\sigma'(1), \sigma'(2), \ldots \sigma'(|T'|)$. Hence in $G' - S$ is no path from $t^-_{h_j}$ to $t^+_{h_{j+1}}$. Now consider the subpath P_2 of K that starts from the component C_{h_j} and ends at component $C_{h_{j+1}}$ and has its interior disjoint from both. Since all internal vertices on P_2 (by definition of P_2) are not in any C_h, all such internal vertices of P_2 must be from $G - S - C$, and the path P_2 corresponds to a path P_2' in the graph $G' - S$ that starts from vertex $t^-_{h_j}$ and ends at vertex $t^+_{h_{j+1}}$. Again, this contradicts the assumption that S is a skew separator for $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'})$.

Thus, the skew separator S for $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'})$ is a solution of (G, k, s). □

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

Lemma 5.6. *An input (G, k, s, T) to the* Disjoint Bounded Size Strong Component Vertex Deletion Reduction *problem is a "yes"-instance if and only if at least one of the instances $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'}, k)$ is a "yes"-instance for the* Skew Separator *problem.*

So we invoke the algorithm of Proposition 5.2 for each of the instances $(G', \mathcal{X}_{\sigma'}, \mathcal{Y}_{\sigma'}, k)$. If at least one of them is a "yes"-instance, then (G, k, s, T) is a "yes"-instance, otherwise (G, k, s, T) is a "no"-instance. Hence, we conclude that Disjoint Bounded Size Strong Component Vertex Deletion Reduction is fixed-parameter tractable with respect to the joint parameter (k, s), and so is Bounded Size Strong Component Vertex Deletion. The overall run time of the algorithm is thus bounded by $|\mathcal{C}| \cdot |T|! \cdot n^{\mathcal{O}(1)} \cdot 4^k kn^3 = (ks + s - 1)! \cdot (k + 1)! \cdot 4^k \cdot n^{\mathcal{O}(1)} = 4^k(ks + k + s)! \cdot n^{\mathcal{O}(1)}$.

This completes the proof of the vertex deletion variant of Theorem 1.2. For the arc deletion variant, we refer to Section 7.1.

6 Fixed-Parameter Algorithm for Vertex Deletion to 1-Out-Regular Strong Components

In this section we give a fixed-parameter algorithm for the vertex deletion variant of Theorem 1.3. Let G be a digraph and let $k \in \mathbb{N}$. A *solution* of (G, k) is a set S of at most k vertices of G such that every non-trivial strong component of $G - S$ is 1-out-regular.

We first apply the steps "Iterative Compression" and "Disjoint Solution" from Section 3. This yields the Disjoint 1-Out-Regular Vertex Deletion Reduction problem, where we seek a solution S of (G, k) that is disjoint from and smaller than a solution T of $(G, k+1)$.

Then we continue with the technique of covering of shadows, as described in Section 3. In our setting, let $\mathcal{F}$ be the collection of vertex sets of G (with size at least 2) that induce a strong digraph different from a simple directed cycle. Then clearly $\mathcal{F}$ is T-connected, and any solution S of (G, k) must intersect every such induced subgraph in $\mathcal{F}$.

So we can use Proposition 3.3 on (G, k, T) to construct sets $Z_1, \ldots, Z_t$ with $t \leq 2^{\mathcal{O}(k^2)} \log^2 n$ such that one of these sets covers the shadow of our hypothetical solution S with respect to T. For each Z_i we construct an instance (G, k, T, Z_i), where we assume that $Z = Z_i \setminus T$ covers the shadow. Note that all vertices of T are outside of the shadow. As we assume that $Z \cup T$ is disjoint from a solution, we reject an instance (G, k, T, Z) as "no"-instance if $G[Z \cup T]$ contains some member of $\mathcal{F}$ as a subgraph.

Observation 6.1. *$G[Z \cup T]$ has no subgraph in $\mathcal{F}$.*

Normally, one would give a "torso" operation which transforms (G, k) with the use of Z into an instance (G', k') of the *same* problem, which has a shadowless solution if and only if the original instance has any solution. Instead, our torso operation reduces the original instance to an instance of a *similar* problem, while maintaining the (non-)existence of solutions.

6.1 Reducing the Instance by the Torso Operation

Our torso operation works directly on the input digraph G. It reduces the original instance (G, k, T) of Disjoint 1-Out-Regular Vertex Deletion Reduction to an instance of a new "similar" problem called Disjoint Shadow-less Good 1-Out-Regular Vertex Deletion Reduction; afterwards we show how to map solutions for the two instances to each other.

Definition 6.2. *Let G be a digraph and let $Z \subseteq V(G)$. Then $\mathsf{torso}(G, Z)$ defines the digraph with vertex set $V(G) \setminus Z$ and* good *and* bad *arcs. An arc (u, v) for $u, v \notin Z$ is introduced whenever there is a $u \to v$-path in G (of length at least 1) whose internal vertices are all in Z. We mark (u, v) as* good *if this path P is unique and all cycles in $G[Z]$ are disjoint from P. Otherwise, we mark it as a* bad *arc.*

Note that every arc between vertices not in Z also forms a path as above. Therefore, $G[V(G) \setminus Z]$ is a subdigraph of $\mathsf{torso}(G, Z)$. Also, $\mathsf{torso}(G, Z)$ may contain self-loops at vertices v from cycles with only the vertex v outside of Z. In $\mathsf{torso}(G, Z)$, we call a cycle *good* if it consists of only good arcs. (A non-good cycle in $\mathsf{torso}(G, Z)$ can contain both good arcs and bad arcs.)

Now we introduce the aforementioned problem Disjoint Shadow-less Good 1-Out-Regular Vertex Deletion Reduction, which seeks a vertex set S of size at most k whose deletion from $G' = \mathsf{torso}(G, Z)$ yields a digraph whose every non-trivial strong component is a cycle of good arcs; we call S a *solution* of (G, k, Z). Notice that this problem is defined only in terms of G and $Z \subseteq V(G)$, and hence does not depend on T. To simplify notation, we construct a set $\mathcal{F}_{\mathsf{bad}}$ which contains all strong subdigraphs of G that are neither trivial nor good cycles. Then S is a solution of (G', k, Z) if and only if $G' - S$ contains no subdigraph in $\mathcal{F}_{\mathsf{bad}}$.

In the next lemma we verify that *any* solution S of (G', k, Z) for our new problem Disjoint Shadow-less Good 1-Out-Regular Vertex Deletion Reduction can be mapped to a solution of (G, k, T) for our original problem, and vice-versa, assuming that the latter admits a solution of (G', k) which is disjoint from $T \cup Z$. In the statement of the lemma, we can think of B as a "bad" set, in the sense that B is not a solution of (G, k) as $G - B$ contains some "forbidden" subdigraph from $\mathcal{F}$.

Lemma 6.3 (torso preserves obstructions)**.** *Let G be a digraph, $T, Z \subseteq V(G)$ as above and $G' = \mathsf{torso}(G, Z)$. For any $B \subseteq V(G) \setminus (Z \cup T)$, it holds that $G - B$ contains a subdigraph in $\mathcal{F}$ if and only if $G' - B$ contains a subdigraph in $\mathcal{F}_{\mathsf{bad}}$.*

Proof. Fix $B \subseteq V(G) \setminus (Z \cup T)$.

In the forward direction, suppose that $G' - B$ contains a subdigraph $F' \in \mathcal{F}_{\mathsf{bad}}$. We replace the arcs of F' as follows:

- All good arcs are replaced by their unique path in the torso operation.
- For a bad arc (x, y), we insert all $x \to y$-paths whose internal vertices completely belong to Z. If there is only a single such path P, then by definition there is a cycle O in $G[Z]$ that intersects P. We also insert all cycles O of this type.

Call the resulting digraph F. This digraph F is a subdigraph of $G - B$ and is strong, as F' was strong and all added vertices have a path from and to $V(F')$. Now, either F' was not a cycle, then F is also not a cycle, or it contained a bad arc and we have inserted at least two parallel paths or a cycle. In any case, we have $F \in \mathcal{F}$.

In the backward direction, let $G - B$ have a subdigraph $F \in \mathcal{F}$. We show that $F' = \mathsf{torso}(F, Z) \in \mathcal{F}_{\mathsf{bad}}$. Note that the torso operation preserves subdigraph relations and the level of strong connectivity.

By Observation 6.1, we know that there is a $v \in V(F) \setminus (Z \cup T)$. Furthermore, we know that there is also a $t \in V(F) \cap T$, as T is a solution of $(G, k+1)$. As $Z \cap T = \emptyset$, by definition, we know that $v, t \notin Z$ and hence $v, t \in V(F')$. As F is strong, there is a closed walk O through v and t in F.

First, suppose that O is *not* a cycle. Then, by definition of a closed walk, there exists some vertex $w \in V(O)$ that is visited at least twice when traversing O. Let $x_1 \rightsquigarrow w \rightsquigarrow y_1$ be the first traversal of w, and $x_2 \rightsquigarrow w \rightsquigarrow y_2$ be the second one. Without loss of generality, we can assume that $x_1, x_2, y_1, y_2 \notin Z$ by replacing them by the next vertex (with respect to the traversal) outside of Z; such a vertex does exist, since $v, t \in V(O) \setminus Z$. This replacement operation could result in $x_1 = y_1$. In that case, though, if $w \notin Z$ then $w \in V(F')$ and w has out-degree at least 2 in F', which means that F' is not a good cycle and thus belongs to $\mathcal{F}_{\mathsf{bad}}$. Whereas if $w \in Z$, then there is an $x_1 \to x_1$-path of length at least 1 in G whose only internal vertex w belongs to Z, and thus there is a loop arc at $x_1 = y_1$ in F'. If moreover $x_1 = y_1 = x_2$, then this loop arc is a bad arc in F', which is a non-good cycle in F' and thus $F' \in \mathcal{F}_{\mathsf{bad}}$.

Else, if $x_1 = y_1 \neq x_2$, there is also some other arc leaving $x_1 = y_1$ in F, because the traversal is from $x_1 = y_1$ to x_2. This arc also exists in F', as it is between two vertices from $V(F) \setminus Z$; hence, x_1 has degree at least two in F', and therefore F' belongs to $\mathcal{F}_{\mathsf{bad}}$. A symmetric argument applies if $x_2 = y_2$.

At this point we can assume that x_1, y_1, x_2, y_2 are all distinct. We distinguish two cases. In case $w \notin Z$ the arcs $(x_1, w), (x_2, w), (w, y_1), (w, y_2)$ exist, so w has degree at least 2 in F', showing that a subdigraph in $\mathcal{F}_{\mathsf{bad}}$ exists.

In the second case, $w \in Z$; then the arcs $(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)$ all exist in the strong subdigraph $\mathsf{torso}(O, Z)$. Thus, $\mathsf{torso}(O, Z) \in \mathcal{F}_{\mathsf{bad}}$.

Second, suppose that O is a cycle. Now F is strong and not a cycle (as $F \in \mathcal{F}$), and therefore has to contain an $x \to y$-path R with the following properties:

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

Then there are paths O_x and O_y in O whose endpoints are not in Z, but all their interior vertices are, and furthermore $x \in V(O_x)$ respectively $y \in V(O_y)$. If $x \notin Z$ (resp. $y \notin Z$), set $x_1 = x_2 = x$ (resp. $y_1 = y_2 = y$), where x_1, x_2, y_1, y_2 have the same meaning as above.

If R contains some interior vertex $u \notin Z$, the path $O_x[x_1, x] \circ R[x, u]$ is in F and shrinks to a $x_1 \to u$-path in F'. As $u \notin V(O)$ we get that x_1 has at least two out-arcs $(x_1, x_2), (x_1, u)$ in F' and therefore $F' = \mathsf{torso}(F, Z) \in \mathcal{F}_{\mathsf{bad}}$, as desired. Thus, the interior of R lies in Z. Furthermore, if $(x_1, x_2) \neq (y_1, y_2)$ then $O_x[x_1, x] \circ R \circ O_y[y, y_2]$, is a $x_1 \to y_2$ path in F. Note that $x_2 \neq y_2$, as O is a cycle and $(x_1, x_2) \neq (y_1, y_2)$. Therefore, the path is shrunk by the torso operation to the arc (x_1, y_2). But then x_1 has two outgoing arcs in F', and as F' is strong, it holds $F' = \mathsf{torso}(F, Z) \in \mathcal{F}_{\mathsf{bad}}$. Therefore, we have $(x_1, x_2) = (y_1, y_2)$ and also $O_x = O_y$, as otherwise the arc would be bad (because there are two different $x_1 \to x_2$-paths). If x lies before y on O_x the path $P = O_x[x_1, x] \circ R \circ O_x[y, x_2]$ is a $x_1 \to x_2$-path in F. As the interior of O_x and R is in Z this would give a second $x_1 \to x_2$-path, making (x_1, x_2) bad. The last case is if y lies before x on O_x. Then $R \circ O_x[y, x]$ forms a cycle in Z which intersects O_x at least in the vertex x, again proving that (x_1, x_2) should be bad. So again we have shown that $F' \in \mathcal{F}_{\mathsf{bad}}$. □

The above lemma shows that some set S is a solution of an instance (G, k, T) of Disjoint 1-Out-Regular Vertex Deletion Reduction disjoint from Z if and only if it is a solution of $(\mathsf{torso}(G, Z), k, T, Z)$ for Disjoint Shadow-less Good 1-Out-Regular Vertex Deletion Reduction. As connections between vertices are preserved by the torso operation, and the torso graph contains no vertices in Z, we can reduce our search for $(\mathsf{torso}(G, Z), T, k)$ to shadow-less solutions (justifying the name).

6.2 Finding a Shadowless Solution

Consider an instance (G, k, Z) of Disjoint Shadow-less Good 1-Out-Regular Vertex Deletion Reduction. Normally, after the torso operation a pushing argument is applied. However, we give an algorithm that recovers the last connected component of G. As T is already a solution, but disjoint of the new (hypothetical) solution S that we are seeking, we take it as a starting point of our recovery. Observe that, without loss of generality, each vertex t in T has out-degree at least one in $G - T \setminus \{t\}$, for otherwise already $T - t$ is a solution.

Consider a topological order of the strong components of $G - S$, say $C_1, \ldots, C_\ell$, i.e., there can be an arc from C_i to C_j only if $i < j$. We claim that the last strong component C_ℓ in the topological ordering of $G - S$ contains a non-empty subset T_0 of T. For if C_ℓ did not contain any vertex from T, then the vertices of C_ℓ cannot reach any vertex of T, contradicting that S is a shadowless solution of (G, k).

Since T_0 is the subset of T present in C_ℓ and arcs between strong components can only be from earlier to later components, we have that there are no outgoing arcs from C_ℓ in $G - S$.

We guess a vertex t inside T_0. This gives $|T| \leq k + 1$ choices for t. For each guess of t we try to find the component C_ℓ, similarly to the bounded-size case. The component C_ℓ will either be trivial or not.

If C_ℓ is a trivial component, then $V(C_\ell) = \{t\}$, and so we delete all out-neighbors of t in $G - T \cup \{t\}$ and place them into the new set S. Hence, we must decrease the parameter k by the number of out-neighbors of t in $G - T \cup \{t\}$, which by assumption is at least one.

Else, if the component C_ℓ is non-trivial, define $v_0 = t$ and notice that exactly one out-neighbor v_1 of v_0 belongs to C_ℓ. Set $i = 0$ and notice that every out-neighbor of v_i other than v_{i+1} must be removed from the graph G as C_ℓ is the last component in the topological ordering of $G - S$, there is no later component where those out-neighbors could go. This observation gives rise to a natural branching procedure: we guess the out-neighbor v_{i+1} of v_i that belongs to C_ℓ and remove all other out-neighbors of v_i from the graph. We then repeat

this branching step with $i \mapsto i+1$ until we get back to the vertex t of T_0 we started with. This way, we obtain exactly the last component C_ℓ, forming a cycle. This branching results in at least one deletion as long as v_i has out-degree at least two. If the out-degree of v_i is exactly one, then we simple proceed by setting $v_i := v_{i+1}$ (and increment i). In any case we stop early if (v_i, v_{i+1}) is a bad arc, as this arc may not be contained in a strong component.

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

Let us shortly analyze the run time of the branching steps. As for each vertex v_i, we have to remove all its out-neighbors from G except one and include them into the hypothetical solution S of size at most k, we immediately know that the degree of v_i in G can be at most $k+1$. Otherwise, we have to include v_i into S. Therefore, there are at most $k+1$ branches to consider to identify the unique out-neighbor v_{i+1} of v_i in C_ℓ. So for each vertex v_i with out-degree at least two we branch into at most $k+1$ ways, and do so for at most k vertices, yielding a run time of $O((k+1)^k)$ for the entire branching.

Once we recovered the last strong component C_ℓ of $G - S$, we remove the set $V(C_\ell)$ from G and repeat: we then recover $C_{\ell-1}$ as the last strong component, and so on until C_1.

6.3 Algorithm for Disjoint 1-Out-Regular Vertex Deletion Reduction

Lemma 6.3 and the branching steps combined give a bounded search tree approach for Disjoint 1-Out-Regular Vertex Deletion Reduction:

Step 1. For a given instance $I = (G, T, k)$, use Proposition 3.3 to obtain a set of instances $\{Z_1, \ldots, Z_p\}$ where $p \leq 2^{\mathcal{O}(k^2)} \log^2 n$. Then Lemma 6.3 implies:

- If I is a "no"-instance then all reduced instances $(\mathsf{torso}(G, Z_j), T, k)$ are "no"-instances, for $j = 1, \ldots, p$.
- If I is a "yes"-instance then there is at least one $i \in \{1, \ldots, p\}$ such that there is a solution $S^\star$ for I which is a shadowless solution of the reduced instance $(\mathsf{torso}(G, Z_i), T, k)$.

So at this step we branch into $p \leq 2^{\mathcal{O}(k^2)} \log^2 n$ directions.

Step 2. For each of the instances obtained from Step 1, recover the component C_ℓ by guessing the vertex $t = v_0$. Afterwards, recover $C_{\ell-1}, \ldots, C_1$ in this order.

So at this step we branch into at most $\mathcal{O}(k \cdot (k+1)^k)$ directions.

We then repeatedly perform Step 1 and Step 2, once per instance of Disjoint 1-Out-Regular Vertex Deletion Reduction. Note that for every instance, one execution of Step 1 and Step 2 gives rise to $2^{\mathcal{O}(k^2)} \log^2 n$ instances such that for each instance, we either know that the answer is "no" or the budget k has decreased, because each important separator is non-empty. Therefore, considering a level as an execution of Step 1 followed by Step 2, the height of the search tree is at most k. Each time we branch into at most $2^{\mathcal{O}(k^2)} \log^2 n \cdot \mathcal{O}(k \cdot (k+1)^k)$ directions. Hence the total number of nodes in the search tree

is

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

We then check the leaf nodes of the search tree and see if there are any strong components other than cycles left after the budget k has become zero. If for at least one of the leaf nodes the corresponding graph only has strong components that are cycles then the given instance is a "yes"-instance. Otherwise, it is a "no"-instance. This gives an $2^{\mathcal{O}(k^3)}\cdot n^{\mathcal{O}(1)}$-time algorithm for Disjoint 1-Out-Regular Vertex Deletion Reduction. So overall, we have an $2^{\mathcal{O}(k^3)}\cdot n^{\mathcal{O}(1)}$-time algorithm for the 1-Out-Regular Vertex Deletion problem.

7 Polynomial Parameter Transformations Between Arc Deletion and Vertex Deletion

In this section we prove the existence of polynomial parameter transformations between Bounded Size Strong Component Arc Deletion and Bounded Size Strong Component Vertex Deletion in both directions, as well as a polynomial parameter transformation from 1-Out-Regular Arc Deletion to 1-Out-Regular Vertex Deletion. These complete the proofs of Theorem 1.2 and Theorem 1.3.

7.1 Bounded Size Strong Component Deletion: Polynomial Parameter Transformation from Arc to Vertex Version

Our first transformation reduces an instance of Bounded Size Strong Component Arc Deletion to an instance of Bounded Size Strong Component Vertex Deletion. This completes the proof of Theorem 1.2. While our transformation keeps the parameter k constant, the parameter s increases to $(k+1)s^3$. This is due to a replacement of all vertices by complete graphs of size roughly ks^2, therefore increasing the size of eligible components.

Lemma 7.1. *Given an instance (G,k,s) of* Bounded Size Strong Component Arc Deletion *we can compute in polynomial time a solution-wise equivalent instance (G',k',s') of* Bounded Size Strong Component Vertex Deletion *with $k'=k$ and $s'=(k+1)s^3$.*

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

follows:

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

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

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

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

7.2 Bounded Size Strong Component Deletion: Polynomial Parameter Transformation from Vertex to Arc Version

Here we state a transformation from Bounded Size Strong Component Vertex Deletion to Bounded Size Strong Component Arc Deletion. This transformation is not needed for any theorem, but we state it here nonetheless for completeness. Note that, unlike the reduction in backwards direction, the parameter increase of s is only linear and does not depend on k.

Lemma 7.2. *Given an instance (G, k, s) of* Bounded Size Strong Component Vertex Deletion *we can compute in polynomial time a solution-wise equivalent instance (G', k', s') of* Bounded Size Strong Component Arc Deletion *with $k' = k$ and $s' = 2s$.*

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

from u^+ to v^- in G', and all out-neighbors u of v in G by $k+1$ parallel arcs from v^+ to u^- in G'. In other words, we set

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

We further set $k' = k$ and $s' = 2s$. Then (G', k', s') is an instance of Bounded Size Strong Component Arc Deletion. It remains to check solution-wise equivalence.

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

In the backward direction, let S' be a set of at most k arcs such that in $G' - S'$ every strong component has at most $s' = 2s$ vertices. We first argue that we can change S' in such a way that it will only consist of arcs of the form (v^-, v^+) for some vertex $v \in V(G)$. This is clear if we use the trick with $k+1$ parallel arcs. Else, we have to argue as follows: For suppose there is an arc $(v^+, u^-) \in S'$ for some vertices $v^+, u^- \in V(G')$ corresponding to distinct vertices $v, u \in V(G)$. Then for $S'' = S' \setminus \{(v^+, u^-)\} \cup \{(v^-, v^+)\}$, vertices v^+, u^- do not belong to the same strong component of $G' - S''$ since v^+ is a source of $G' - S''$. Therefore, for each vertex $v^\pm \in V(G')$ the size of the (uniquely determined) strong component in $G - S''$ containing $v^\pm$ is at most the size of the strong component in $G' - S'$ containing $v^\pm$. This justifies the assumption that $S' = \{(v^-, v^+) \mid v \in V(G)\}$.) Now let $S = \{v \in V(G) \mid (v^-, v^+) \in S'\}$ be the set of at most k vertices in G corresponding to the arcs in S'. The number of vertices in each strong component of $G - S$ is now exactly half the number of vertices in its corresponding component in $G' - S'$. Therefore, every strong component of $G - S$ consists of at most $s = s'/2$ vertices. □

7.3 1-Out Regular Deletion: Polynomial Parameter Transformation from Arc to Vertex Version

Last but not least, we show a transformation from 1-Out-Regular Arc Deletion to 1-Out-Regular Vertex Deletion. Thus, by the fixed-parameter tractability of the vertex deletion version as shown in Section 6, we obtain fixed-parameter tractability of the arc deletion version. Note that this reduction, unlike the others is parameter preserving.

Lemma 7.3. *Given an instance (G, k) of* 1-Out-Regular Arc Deletion *we can compute in polynomial time a solution-wise equivalent instance (G', k') of* 1-Out-Regular Vertex Deletion *with $k' = k$.*

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

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

For the forward direction, let S be a solution to (G, k). Let $S' = \{v_a \mid a \in S\}$. As $|S'| = |S| \leq k$, our candidate fulfills the size bound. Let now X' be a strong component

of $G' - S'$. Assume for contradiction that X' is neither trivial nor 1-out-regular. By above correspondence there is a non-trivial strong component X in $G - S$ that has the arcs which X' possesses as vertices. As S is a solution to (G, k), X is 1-out-regular (as it is not trivial). Therefore, $G[X]$ forms a cycle O. This cycle has a corresponding cycle O' in $G'[X']$. Since O visits all arcs of $G[X]$, O' is a Hamiltonian cycle for $G'[X']$. As $G'[X']$ is not 1-out-regular, there must be an arc $(v_a, v_b) \in E(G'[X'])$ which is not part of O'. This arc means that the arcs a and b share a vertex v in $G[X]$ albeit being not adjacent in O. Thus, v has out-degree at least two in $G[X]$, a contradiction. Therefore, S' is a solution to (G', k).

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

8 Discussion

In this work we considered two natural abstractions of the fundamental DFVS problem, namely the problem of finding a smallest vertex set S in a given digraph G such that every strong component of $G - S$ has size at most s, and the problem of finding a smallest vertex set S in a given digraph G such that every non-trivial strong component of $G - S$ is 1-out-regular. For both problems we devised fixed-parameter algorithms, which solve the first problem in time $2^{\mathcal{O}(ks \log(ks))} \cdot n^{\mathcal{O}(1)}$ and the second problem in time $2^{\mathcal{O}(k^3)} \cdot n^{O(1)}$.

The relevance of these problems is substantiated by recent follow-up work to the extended abstract of this work. Namely, Bang-Jensen et al. [2] improved the run time of the algorithm for the Bounded Size Strong Component Vertex Deletion problem from $2^{\mathcal{O}(ks \log(ks))} \cdot n^{\mathcal{O}(1)}$ to $2^{\mathcal{O}(k \log(ks))} \cdot n^{\mathcal{O}(1)}$. It seems a very challenging question to reduce the asymptotic run time much further, since even for the case of $s = 1$ no algorithm of run time $2^{o(k \log(k))} \cdot n^{\mathcal{O}(1)}$ is known. In fact, Drange et al. [8] established that no algorithm with run time $2^{o(k \log(s))} \cdot n^{\mathcal{O}(1)}$ exists for this problem, even if G is an undirected split graph. A natural problem arising from these results is therefore to close the complexity gap between the lower and upper bounds on the asymptotic run times.

Regarding the 1-Out-Regular Vertex Deletion problem, Neogi et al. [22] extend this line of research by initiating the study of a wide generalization, which they call the $\mathcal{H}$-Strong Connected Component Deletion, for a fixed family $\mathcal{H}$ of digraphs, where one seeks a set S of at most k vertices in a given digraph G such that no strong component of $G - S$ contains a graph in $\mathcal{H}$ as a subgraph. When $\mathcal{H}$ comprises the star with two leaves and both arcs oriented away from the center, then one obtains the 1-Out-Regular Vertex Deletion problem. Neogi et al. [22] give a fixed-parameter algorithm for this general problem, but its run time is somewhat slower than that of our algorithm here. A natural question is therefore whether a fast fixed-parameter algorithm, say with run time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, exists for the general problem or even for the case of 1-Out-Regular Vertex Deletion as we have introduced it here.

Acknowledgements. We are grateful for the reviewers whose comments and suggestions led to cleaner proofs and an improved presentation.

References

[1] J. Bang-Jensen and T. M. Larsen. DAG-width and circumference of digraphs. *J. Graph Theory*, 82(2):194–206, 2016.

[2] J. Bang-Jensen, E. Eiben, G. Gutin, M. Wahlström, and A. Yeo. Component order connectivity in directed graphs. In *Proc. IPEC 2020*, volume 180 of *Leibniz Int. Proc. Informatics*, pages 2:1–2:16, 2020.

[3] K. Cechlárová and I. Schlotter. Computing the deficiency of housing markets with duplicate houses. In *Proc. IPEC 2010*, volume 6478 of *Lecture Notes Comput. Sci.*, pages 72–83. 2010.

[4] 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):Art. 21, 19, 2008.

[5] R. Chitnis, M. Hajiaghayi, and D. Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM J. Comput.*, 42(4):1674–1696, 2013.

[6] R. Chitnis, M. Cygan, M. Hajiaghayi, and D. Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(4):Art. 28, 28, 2015.

[7] M. Cygan, D. Marx, M. Pilipczuk, M. Pilipczuk, and I. Schlotter. Parameterized complexity of Eulerian deletion problems. *Algorithmica*, 68(1):41–61, 2014.

[8] P. G. Drange, M. Dregi, and P. van't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016.

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

[10] G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM*, 47(4):585–616, 2000.

[11] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoret. Comput. Sci.*, 10(2):111–121, 1980.

[12] A. Göke, D. Marx, and M. Mnich. Parameterized algorithms for generalizations of directed feedback vertex set. In *Proc. CIAC 2019*, volume 11458 of *Lecture Notes Comput. Sci.*, pages 249–261. 2019.

[13] A. Göke, D. Marx, and M. Mnich. Hitting long directed cycles is fixed-parameter tractable. In *Proc. ICALP 2020*, volume 168 of *Leibniz Int. Proc. Informatics*, pages 59:1–59:18, 2020.

[14] A. Göke, D. Marx, and M. Mnich. Parameterized algorithms for generalizations of directed feedback vertex set. *Discrete Optimization*, 46:100740, 2022.

[15] V. Guruswami and E. Lee. Simple proof of hardness of feedback vertex set. *Theory Comput.*, 12:Paper No. 6, 11, 2016.

[16] V. Guruswami, J. Håstad, R. Manokaran, P. Raghavendra, and M. Charikar. Beating the random ordering is hard: every ordering CSP is approximation resistant. *SIAM J. Comput.*, 40(3):878–914, 2011.

[17] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM)*, pages 85–103. 1972.

[18] M. Kumar and D. Lokshtanov. A $2\ell k$ kernel for ℓ-component order connectivity. In *Proc. IPEC 2016*, volume 63 of *Leibniz Int. Proc. Informatics*, pages 20:1–20:14, 2017.

[19] D. Lokshtanov, M. S. Ramanujan, and S. Saurabh. When recursion is better than iteration: A linear-time algorithm for acyclicity with few error vertices. In *Proc. SODA 2018*, pages 1916–1933, 2018.

[20] D. Lokshtanov, M. Ramanujan, and S. Saurabh. Parameterized complexity and approximability of directed odd cycle transversal. In *Proc. SODA 2020*, pages 2181–2200, 2020.

[21] D. Marx. What's next? Future directions in parameterized complexity. In H. L. Bodlaender, R. Downey, F. V. Fomin, and D. Marx, editors, *The Multivariate Algorithmic Revolution and Beyond*, volume 7370 of *Lecture Notes Comput. Sci.*, pages 469–496. 2012.

[22] R. Neogi, M. S. Ramanujan, S. Saurabh, and R. Sharma. On the parameterized complexity of deletion to $\mathcal{H}$-free strong components. In *Proc. MFCS 2020*, volume 170 of *Leibniz Inte. Proc. Informatics*, pages 75:1–75:13, 2020.

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

[24] O. Svensson. Hardness of vertex deletion and project scheduling. *Theory Comput.*, 9: 759–781, 2013.

[25] M. Xiao. Linear kernels for separating a graph into components of bounded size. *J. Comput. Syst. Sci.*, 88:260–270, 2017.