Parameterized Algorithms for Generalizations of Directed Feedback Vertex Set

Alexander Göke[1](✉), Dániel Marx[2], and Matthias Mnich[1]

[1] Universität Bonn, Bonn, Germany
{alexander.goeke,mmnich}@uni-bonn.de
[2] SZTAKI, Budapest, Hungary
dmarx@cs.bme.hu

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. in 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 $4^k(ks + k + s)! \cdot n^{\mathcal{O}(1)}$.
– 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.

1 Introduction

The Directed Feedback Vertex Set (DFVS) problem is that of finding a smallest vertex set S in a given digraph 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 [12].

Consequently, the DFVS problem has long attracted researchers in approximation algorithms. The current best known approximation factor achievable in polynomial time for n-vertex graphs with optimal fractional solution value[1] τ^* is $\mathcal{O}(\min\{\log\tau^* \log\log\tau^*, \log n \log\log n\})$ due to Seymour [17], Even et al. [8] and Even et al. [7]. On the negative side, Karp's NP-hardness reduction shows

[1] In unweighted digraphs, $\tau^* \leq n$; in weighted digraphs we assume all weights are at least 1.

© Springer Nature Switzerland AG 2019
P. Heggernes (Ed.): CIAC 2019, LNCS 11485, pp. 249–261, 2019.
https://doi.org/10.1007/978-3-030-17402-6_21

the problem to be APX-hard, which rules out the existence of a polynomial-time approximation scheme (PTAS) assuming $\mathsf{P} \neq \mathsf{NP}$. Assuming the Unique Games Conjecture, the DFVS problem does not admit a polynomial-time $\mathcal{O}(1)$-approximation [10,11,18].

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 running 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 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. [14]. 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. [4] showed that the generalization of DFVS where one only wishes to hit cycles going through a specified subset of nodes of a given digraph is still fixed-parameter tractable when parameterized by solution size. On the other hand, Lokshtanov et al. [15] show that finding a smallest set of vertices of hitting only the *odd* directed cycles of a given digraph is W[1]-hard, and hence not fixed-parameter tractable unless $\mathsf{FPT} = \mathsf{W[1]}$.

Our Contributions. For another generalization the parameterized complexity is still open: 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 [2] in the context of housing markets. Marx [16] 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. Eulerian Strong Component Vertex Deletion *is* W[1]-*hard parameterized by solution size k, even for $(k+1)$-strong digraphs.*

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 arcs or vertices from a given digraph such that every strong component has size at most s. Thus, the

DFAS/DFVS problems corresponds to the special case when $s = 1$. Formally, the problem Bounded Size Strong Component Arc (Vertex) Deletion takes as input a multi-digraph 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 Arc (Vertex) 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 [13] 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 [19]. 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 2. *There is an algorithm that solves* Bounded Size Strong Component Arc (Vertex) Deletion *in time* $4^k(ks + k + s)! \cdot n^{\mathcal{O}(1)}$ *for* n*-vertex multi-digraphs* 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. [3] 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 digraph if they can be connected by k mutually arc-disjoint paths. The k-linkage problem is NP-complete already for $k = 2$ [9]. Recently, Bang-Jensen and Larsen [1] solved the k-linkage problem in digraphs 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 digraph such that each remaining non-trivial strong component is *1-out-regular*, meaning that every vertex has out-degree exactly 1 in its strong component. (A strong component is *non-trivial* if it has at least two vertices.) So in particular, every strong component is Eulerian, as in the Eulerian Strong Component Arc Deletion problem. Observe that in the DFAS/DFVS problem we delete k arcs or vertices from a given directed graph such that each remaining strong component is 0-out-regular (trivial). Formally, we consider the 1-Out-Regular Arc (Vertex) Deletion problem in which for a given multi-digraph G and integer k, we seek a set S of at most k arcs (vertices) such that every non-trivial component of $G - S$ is 1-out-regular. Note that this problem is equivalent to deleting a set S of at most k arcs (vertices) such that every non-trivial strong component of $G - S$ is an induced directed cycle. In contrast to Eulerian Strong Component Vertex Deletion, the 1-Out-Regular Arc (Vertex) 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 3. *There is an algorithm solving* 1-Out-Regular Arc (Vertex) Deletion *in time* $2^{\mathcal{O}(k^3)} \cdot \mathcal{O}(n^4)$ *for* n*-vertex digraphs* G *and parameter* $k \in \mathbb{N}$.

Notice that for Bounded Size Strong Component Arc (Vertex) Deletion and 1-Out-Regular Arc (Vertex) Deletion, there are infinitely many instances for which solutions are arbitrarily smaller than those for DFAS (DFVS), and for any instance they are never larger. Therefore, our algorithms strictly generalize the one by Chen et al. [3] for DFAS (DFVS). 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 give algorithms for vertex deletion variants only, and defer algorithms for arc deletion variants and proofs marked by $\star$ to the full version of this paper.

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.

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. Finally, G is *Eulerian* if there is a closed walk in G using each arc exactly once.

Definition 4. *For disjoint non-empty vertex sets* X, Y *of a digraph* G*, a set* S *is an* $X - Y$ separator *if* S *is disjoint from* $X \cup Y$ *and there is no path from* X

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

Notice that S can be either a vertex set or an arc set.

Proposition 5 ([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 $X - Y$ separators of size at most p, all of which can be enumerated in time $4^p \cdot n^{\mathcal{O}(1)}$.*

3 Tools for Generalized DFVS/DFAS Problems

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 for G_{i+1} of size $k + 1$. The *compression variant* of our problem then takes as input a digraph G and a solution T of size $k+1$, and seeks a solution S of size at most k for G 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} or find out that G_{i+1} does not have a solution of size k, but then neither has G. By at most n calls to this algorithm we can deduce a solution for the original instance $(G_n = G, k)$.

Disjoint Solution. Given an input (G, T) to the compression variant, the next step is to ask for a solution S for G of size at most k that is disjoint from the given solution T of size $k + 1$. 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 problem then takes as input a graph $G - T'$, a solution $T \setminus T'$ of size $k + 1 - |T'|$, and seeks a solution S' of size at most $k - |T'|$ disjoint from $T \setminus T'$.

Covering the Shadow of a Solution. The "shadow" of a solution S is the set of those vertices that are disconnected from T (in either direction) after the removal of S. 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. [4] 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 6 (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 $T - \{v\}$-separator in G, and v is* in the reverse shadow $r_{G,T}(S)$ *of S (with respect to T) if S is a $\{v\} - T$-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 7 (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. [4] show how to deterministically cover the shadow of $\mathcal{F}$-transversals:

Proposition 8 (deterministic covering of the shadow, [4]). *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 Z_i covers the shadow of S, for some $i \leq t$.*

4 Hardness of Vertex Deletion

In this section we prove Theorem 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 9 (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. [6]. 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 10 ([6]). 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 connectivity. The proof of Theorem 1 is deferred to the full version of this paper.

5 Bounded Size Strong Component Arc (Vertex) Deletion

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 11. *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. [3] showed:

Proposition 12 ([3, Theorem 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 $(G, k+1, s)$, we can assume 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 13 ($\star$). *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 some 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 $h = 1, \ldots, k+1$.*

Armed with Lemma 13, 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 $C=(C_1,\dots,C_{k+1})\in\mathcal{C}$ if the strong component of $G-S$ containing t_h is exactly C_h for $h=1,\dots,k+1$. For a given vector $C=(C_1,\dots,C_{k+1})$, 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. Now we generate precisely one instance $(G',\mathcal{X}_{\sigma'},\mathcal{Y}_{\sigma'},k)$ of Skew Separator for each permutation σ' of T'. The graph G' is the same in all these instances, and is obtained from G by replacing each unique set C_h by two vertices t_h^+, t_h^- (where t_h is the representative of C_h), and connecting all 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 $\sigma'(1),\dots,\sigma'(|T'|)$: let $\mathcal{X}_{\sigma'}=(t^-_{\sigma'(1)},\dots,t^-_{\sigma'(|T'|)})$ and let $\mathcal{Y}_{\sigma'}=(t^+_{\sigma'(1)},\dots,t^+_{\sigma'(|T'|)})$.

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

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

Lemma 14 (⋆). *If an instance (G,k,s) admits a solution S disjoint from T, compatible with C and for which $(t_{\sigma'(1)},\dots,t_{\sigma'(|T'|)})$ is a topological order of the connected components of $G'-S$, then S forms a skew separator of size k for $(G,\mathcal{X}_{\sigma'},\mathcal{Y}_{\sigma'})$.*

Lemma 15 (⋆). *Conversely, if S is a skew separator of $(G',\mathcal{X}_{\sigma'},\mathcal{Y}_{\sigma'})$ with size at most k, then S is a solution of (G,k,s) disjoint from S and compatible with C.*

In summary, we have reduced a single instance to 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 16. *An input (G,k,s,T) to the* Disjoint Bounded Size Strong Component Vertex Deletion *problem is a "yes"-instance if and only if at least one of the instances $(G',\mathcal{X}_{\sigma'},\mathcal{Y}_{\sigma'},k)$ is a "yes"-instance for the* Skew Separator *problem.*

So we invoke the algorithm of Proposition 12 for each of the instances $(G',\mathcal{X}_{\sigma'},\mathcal{Y}_{\sigma'},k)$. If at least one of them is a "yes"-instance then so is (G,k,s,T), 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 Theorem 2.

6 1-Out-Regular Arc (Vertex) Deletion

In this section we give a fixed-parameter algorithm for the vertex deletion variant of Theorem 3. Let G be a digraph and let $k \in \mathbb{N}$. A *solution* for (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 Sect. 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 Sect. 3. In our setting, let $\mathcal{F}$ be the collection of vertex sets of G that induce a strongly connected graph different from a simple directed cycle. Then clearly $\mathcal{F}$ is T-connected and any solution S must intersect every such induced subgraph.

So we can use Proposition 8 to construct sets $Z_1, \dots, Z_t$ with $t \le 2^{\mathcal{O}(k^2)} \log^2 n$ 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, where we assume that $Z = Z_i \setminus T$ covers the shadow. Note that a vertex of T is never in the shadow. As we assume that $Z \cup T$ is disjoint of a solution we reject an instance if $G[Z \cup T]$ contains a member of $\mathcal{F}$ as a subgraph.

Observation 17. *$G[Z \cup T]$ 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 to a similar problem while maintaining solution equivalence.

Reducing the Instance by the Torso Operation. Our torso operation works directly on the graph. It reduces the original instance to one of a new problem called Disjoint Shadow-less Good 1-Out-Regular Vertex Deletion Reduction; afterwards we show the solution equivalence.

Definition 18. *Let (G, T, k) be an instance of* Disjoint 1-Out-Regular Vertex Deletion Reduction *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 an $u \to v$ 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 there is no cycle O in $G[Z]$ with $O \cap P \neq \emptyset$. 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 want to compute a vertex set of size 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 this problem Disjoint Shadow-less Good 1-Out-Regular Vertex Deletion Reduction. To simplify notation we construct a set $\mathcal{F}_{\mathsf{bad}}$ which contains all strong subdigraphs of G that are not trivial or good cycles. Then S is a solution to G' if and only if $G' - S$ contains no subdigraph in $\mathcal{F}_{\mathsf{bad}}$. In the next lemma we verify that our new problem is indeed equivalent to the original problem, assuming that there is a solution disjoint from Z.

Lemma 19 ($\star$, torso preserves obstructions). *Let G be a digraph, $T, Z \subseteq V(G)$ as above and $G' = \mathsf{torso}(G, Z)$. For any $S \subseteq V(G) \setminus (Z \cup T)$ it holds that $G - S$ contains a subdigraph in $\mathcal{F}$ if and only if $G' - S$ contains a subdigraph in $\mathcal{F}_{bad}$.*

The above lemma shows that S is a solution of an instance (G, T, k) for Disjoint 1-Out-Regular Vertex Deletion Reduction disjoint of Z if and only if it is a solution of $(\mathsf{torso}(G, Z), T, k)$ 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).

Finding a Shadowless Solution. Consider an instance (G, T, k) 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 solution S, 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 $G - T$ and place them into the new set S. Hence, we must decrease the parameter k by the number of out-neighbors of t in $G - T$, 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 $G - T'$, 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 step. As for each vertex v_i, we have to remove all its out-neighbors from G except one and include them into the hypothetical solution S of size at most k, we immediately know that the degree of v_i in G can be at most $k+1$. Otherwise, we have to include v_0 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.

Algorithm for Disjoint 1-Out-Regular Vertex Deletion Reduction. Lemma 19 and the branching procedure combined give a bounded search tree algorithm for Disjoint 1-Out-Regular Vertex Deletion Reduction:

Step1. For a given instance $I = (G, T, k)$, use Proposition 8 to obtain a set of instances $\{Z_1, \ldots, Z_t\}$ where $t \leq 2^{\mathcal{O}(k^2)} \log^2 n$, and Lemma 19 implies

- If I is a "no"-instance then all reduced instances I/Z_j are "no"-instances, for $j = 1, \ldots, t$.
- If I is a "yes"-instance then there is at least one $i \in \{1, \ldots, t\}$ such that there is a solution $T^\star$ for I which is a shadowless solution for the reduced instance I/Z_i.

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

Step2. For each of the instances obtained from Step 1, recover the component C_ℓ by guessing the vertex $t = v_0$. Afterwards, recover $C_{\ell-1}, \dots, C_1$ 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. 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

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

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.

References

1. Bang-Jensen, J., Larsen, T.M.: DAG-width and circumference of digraphs. J. Graph Theory **82**(2), 194–206 (2016)
2. Cechlárová, K., Schlotter, I.: Computing the deficiency of housing markets with duplicate houses. In: Raman, V., Saurabh, S. (eds.) IPEC 2010. LNCS, vol. 6478, pp. 72–83. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17493-3_9
3. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. J. ACM **55**(5), 19 (2008). Article No. 21
4. Chitnis, R., Cygan, M., Hajiaghayi, M., Marx, D.: Directed subset feedback vertex set is fixed-parameter tractable. ACM Trans. Algorithms **11**(4), 28 (2015). Article No. 28
5. Chitnis, R., Hajiaghayi, M., Marx, D.: Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. SIAM J. Comput. **42**, 1674–1696 (2013)
6. Cygan, M., Marx, D., Pilipczuk, M., Pilipczuk, M., Schlotter, I.: Parameterized complexity of Eulerian deletion problems. Algorithmica **68**(1), 41–61 (2014)
7. Even, G., Naor, J., Rao, S., Schieber, B.: Divide-and-conquer approximation algorithms via spreading metrics. J. ACM **47**(4), 585–616 (2000)
8. Even, G., Naor, J., Schieber, B., Sudan, M.: Approximating minimum feedback sets and multicuts in directed graphs. Algorithmica **20**(2), 151–174 (1998)

9. Fortune, S., Hopcroft, J., Wyllie, J.: The directed subgraph homeomorphism problem. Theoret. Comput. Sci. **10**(2), 111–121 (1980)
10. Guruswami, V., Håstad, J., Manokaran, R., Raghavendra, P., Charikar, M.: Beating the random ordering is hard: every ordering CSP is approximation resistant. SIAM J. Comput. **40**(3), 878–914 (2011)
11. Guruswami, V., Lee, E.: Simple proof of hardness of feedback vertex set. Theory Comput. **12**, 11 (2016). Article No. 6
12. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations, pp. 85–103. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
13. Kumar, M., Lokshtanov, D.: A $2\ell k$ kernel for ℓ-component order connectivity. In: Proceedings of the IPEC 2016. Leibniz International Proceedings in Informatics, vol. 63, pp. 20:1–20:14 (2017)
14. Lokshtanov, D., Ramanujan, M.S., Saurabh, S.: When recursion is better than iteration: a linear-time algorithm for acyclicity with few error vertices. In: Proceedings of the SODA 2018, pp. 1916–1933 (2018)
15. Lokshtanov, D., Ramanujan, M., Saurabh, S.: Parameterized complexity and approximability of directed odd cycle transversal (2017). https://arxiv.org/abs/1704.04249
16. Marx, D.: What's next? Future directions in parameterized complexity. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) The Multivariate Algorithmic Revolution and Beyond. LNCS, vol. 7370, pp. 469–496. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30891-8_20
17. Seymour, P.D.: Packing directed circuits fractionally. Combinatorica **15**(2), 281–288 (1995)
18. Svensson, O.: Hardness of vertex deletion and project scheduling. Theory Comput. **9**, 759–781 (2013)
19. Xiao, M.: Linear kernels for separating a graph into components of bounded size. J. Comput. Syst. Sci. **88**, 260–270 (2017)