On the Parameterized Complexity of Deletion to $\mathcal{H}$-Free Strong Components

Rian Neogi
The Institute of Mathematical Sciences, HBNI, Chennai, India
rianneogi@imsc.res.in

M. S. Ramanujan
University of Warwick, Coventry, UK
r.maadapuzhi-sridharan@warwick.ac.uk

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
IRL 2000 ReLaX, Chennai, India
University of Bergen, Norway
saket@imsc.res.in

Roohani Sharma
The Institute of Mathematical Sciences, HBNI, Chennai, India
roohani@imsc.res.in

Abstract

Directed Feedback Vertex Set (DFVS) is a fundamental computational problem that has received extensive attention in parameterized complexity. In this paper, we initiate the study of a wide generalization, the $\mathcal{H}$-SCC Deletion problem. Here, one is given a digraph D, an integer k and the objective is to decide whether there is a vertex set of size at most k whose deletion leaves a digraph where every strong component excludes graphs in the fixed finite family $\mathcal{H}$ as (not necessarily induced) subgraphs. When $\mathcal{H}$ comprises only the digraph with a single arc, then this problem is precisely DFVS.

Our main result is a proof that this problem is fixed-parameter tractable parameterized by the size of the deletion set if $\mathcal{H}$ only contains rooted graphs or if $\mathcal{H}$ contains at least one directed path. Along with generalizing the fixed-parameter tractability result for DFVS, our result also generalizes the recent results of Göke et al. [CIAC 2019] for the 1-Out-Regular Vertex Deletion and Bounded Size Strong Component Vertex Deletion problems. Moreover, we design algorithms for the two above mentioned problems, whose running times are better and match with the best bounds for DFVS, without using the heavy machinery of shadow removal as is done by Göke et al. [CIAC 2019].

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Directed Cut Problems, Fixed-parameter Tractability, DFVS

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.75

Related Version A full version of the paper is available at `https://arxiv.org/abs/2005.01359`.

Funding *Saket Saurabh*: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

Acknowledgements The second author thanks Matthias Mnich for insightful discussions during Dagstuhl Seminar 19041 and for the pointer to [5].

1 Introduction

In the Directed Feedback Vertex Set (DFVS) problem, the input is a digraph D and an integer k and the objective is to decide whether there is a set $X \subseteq V(D)$ of size at most k such that $D - X$ is acyclic. DFVS is a fundamental computational problem that has received extensive attention in various subdomains of algorithmics. The parameterized complexity of this problem was a long standing open problem in the area until Chen et al. [2] gave a fixed-parameter tractable (FPT) algorithm with running time $O(k!4^k k^4 nm)$. Here, n and m denote the number of vertices and arcs in the digraph respectively. Although subsequent work [9] has improved the dependence on the input size to linear, it remains an open problem whether the $2^{O(k \log k)}$ dependence on k is asymptotically the best possible.

The result of Chen et al. and the techniques used therein also helped kick off a line of research in parameterized complexity where the goal is to understand how far the fixed-parameter tractability of DFVS can be extended to various generalizations of DFVS. Chitnis et al. [3] obtained an FPT algorithm for the Subset DFVS problem, where the goal is to delete at most k vertices that intersect all directed cycles passing through a specified subset of vertices. A general and abstract formulation of the powerful directed *shadow removal* technique first designed by Chitnis et al. [4], was developed in this work and it has found several applications in subsequent work [8, 1, 10, 5]. Lokshtanov et al. [10] studied the Directed Odd Cycle Transversal problem where the objective is to delete at most k vertices that intersect all directed *odd* cycles in the given digraph. They proved that this problem is W[1]-hard and so is unlikely to admit an FPT algorithm. Moreover, they used the shadow removal technique to obtain a fixed-parameter 2-approximation algorithm for this problem. More recently, Göke et al. [5] studied the problems of deleting at most k vertices to (i) obtain a digraph where every strong component is of bounded size and (ii) obtain a digraph where every strong component induces a graph where every vertex has out-degree exactly 1, i.e. is a 1-out-regular digraph.

In this paper, we extend this line of research by initiating the study of a wide generalization of the problems studied by Göke et al., which we call the $\mathcal{H}$-Strong Connected Component Deletion ($\mathcal{H}$-SCC Deletion) problem and define below. Here, $\mathcal{H}$ is a fixed finite family of digraphs.

$\mathcal{H}$-SCC Deletion

Input: A digraph D, an integer k.
Parameter: k
Problem: Does there exist a set S of at most k vertices such that no strong component of D-S contains a graph in $\mathcal{H}$ as a subgraph?

In all our results, n denote the number of vertices in the input graph and $h = \max_{H \in \mathcal{H}} |V(H)|$. Rooted $\mathcal{H}$-SCC Deletion (r-$\mathcal{H}$-SCC Deletion) denotes the special case of $\mathcal{H}$-SCC Deletion where every graph in $\mathcal{H}$ contains an arborescence. An arborescence is a rooted directed tree where every vertex except the root has in-degree exactly 1 and the root has in-degree 0. Notice that r-$\mathcal{H}$-SCC Deletion already generalizes several problems described above including DFVS ($\mathcal{H}$ comprises the graph with a single arc), obtaining strong components of size at most s ($\mathcal{H}$ comprises all arborescences of size $s + 1$) and obtaining strong components with out-degree at most 1 ($\mathcal{H}$ comprises the star with two leaves and both arcs oriented away from the centre). Our main result gives a unified proof of the fixed-parameter tractability of these problems.

▶ **Theorem 1.** *r-$\mathcal{H}$-SCC Deletion can be solved in time* $2^{O(k^3 \log k)} \cdot h^{O(k)} \cdot n^{O(h)}$.

Theorem 1 also holds for $\mathcal{H}$-SCC Deletion in the case where, for every graph in $\mathcal{H}$ there is a vertex that is reachable from every other vertex. One can infer this by simply reversing both the input graph and the forbidden graphs and applying the main theorem. We also remark that in general, the $n^{O(h)}$ dependence in the running time of the algorithm of Theorem 1 is very likely unavoidable. Indeed, consider the following reduction from the Clique problem where the input is an undirected graph G and $\ell \in \mathbb{N}$, and the objective is to decide whether G contains a clique of size ℓ. We orient all edges in G arbitrarily, add a universal sink vertex $v^\star$ and then a universal source vertex $u^\star$ and the arc $(v^\star, u^\star)$ to obtain a strongly connected digraph, set $k = 0$, and set $\mathcal{H}$ to be the set of all tournaments on exactly $\ell + 2$ vertices. Then, an FPT algorithm for r-$\mathcal{H}$-SCC Deletion parameterized by k and ℓ would imply an FPT algorithm for Clique parameterized by ℓ, which cannot exist unless FPT=W[1].

When $\mathcal{H}$ only comprises of the star with $d + 1$ leaves with all arcs oriented away from the centre, a closer inspection of the algorithm of Theorem 1 demonstrates that it can be implemented in a way that implies a fixed-parameter algorithm parameterized by both k and d for this problem. We call this problem, d-Out-Degree SCC Deletion. In this problem, the objective is to decide whether k vertices can be deleted from a given digraph to ensure that the graph induced by each strong component has out-degree at most d.

▶ **Theorem 2.** *d-Out-Degree SCC Deletion can be solved in time* $2^{O(k^3 \log k)} \cdot d^{O(k)} \cdot n^{O(1)}$.

Our next result concerns the Path $\mathcal{H}$-SCC Deletion (p-$\mathcal{H}$-SCC Deletion) problem, which is the special case of $\mathcal{H}$-SCC Deletion where $\mathcal{H}$ contains at least one directed path. We show that with an appropriate fixed-parameter preprocessing routine, this problem can be reduced to r-$\mathcal{H}$-SCC Deletion where $\mathcal{H}$ only comprises of the path of length $g(\mathcal{H})$ for some function g. Invoking Theorem 1 then leads us to the following result.

▶ **Theorem 3.** *p-$\mathcal{H}$-SCC Deletion can be solved in time* $2^{O(k^3 \log k)} \cdot h^{O(k)} \cdot 2^{h^6} \cdot n^{O(h^3)}$.

We then pay special attention to the r-$\mathcal{H}$-SCC Deletion problem when $\mathcal{H}$ contains only the out-directed 2-star, i.e., the 1-Out-Degree SCC Deletion problem. Notice that a strongly connected graph with at least two vertices that excludes this graph as a subgraph must be a simple cycle, and so is 1-out-regular. A 1-out-regular digraph is a digraph where every vertex has out-degree exactly 1. Therefore, this special case of r-$\mathcal{H}$-SCC Deletion is precisely the 1-Out-regular Deletion problem where one is given a digraph D and an integer k and the objective is to decide whether there is a set of vertices of size at most k whose deletion leaves a digraph where every strong component induces a 1-out-regular subgraph. Göke et al. [5] recently gave an algorithm for this problem with running time $2^{O(k^3)} \cdot n^{O(1)}$. We give an improved algorithm for this problem with an asymptotic dependence on k that matches that of the current best algorithm for DFVS [2], which is a special case of 1-Out-regular Deletion.

▶ **Theorem 4.** *1-Out-regular Vertex Deletion can be solved in time* $2^{O(k \log k)} \cdot n^{O(1)}$.

Finally, we also study the special case of r-$\mathcal{H}$-SCC Deletion when $\mathcal{H}$ is the set of all arborescences on exactly $s + 1$ vertices. Notice that the strongly connected graphs that exclude the graphs in $\mathcal{H}$ as subgraphs are precisely the strongly connected graphs of size at most s. This problem when $\mathcal{H}$ is the set of all arborescences on exactly $s + 1$ vertices is called the Bounded Size Strong Component Vertex Deletion (BSSCVD). We improve upon the result of Göke et al. [5] who gave an algorithm for BSSCVD with running time $4^k(ks + k + s)! \cdot n^{O(1)}$.

▶ **Theorem 5.** *BSSCVD can be solved in time* $2^{O(k(\log k+\log s))} \cdot n^{O(1)}$.

We now give an overview of the techniques used to prove our results.

Algorithm for r-$\mathcal{H}$-SCC Deletion. We begin by using the technique of iterative compression to obtain a tuple $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ such that W is a solution for the instance $(D, k+1)$ of r-$\mathcal{H}$-SCC Deletion, $S_1, \ldots, S_q$ partition W and moreover, if there is a solution for (D, k), then there is a solution that is disjoint from W and intersects S_i-S_j paths for $i < j$. We note that this step is standard when dealing with directed cut problems [12].

A commonly used technique subsequent to this step (albeit one that we *do not* employ) is the directed *shadow removal* technique introduced by Chitnis et al. [4] where one identifies a set of vertices Z such that for some hypothetical solution X, Z is disjoint from X and contains the set of vertices that are either unable to reach W or are unreachable from W in $D - X$. This set is then removed in a problem specific way while preserving all obstructions. While this can be easily achieved for certain simple obstructions, we are dealing with an arbitrary family of digraphs with the only assumption being that they are rooted. Consequently, it is not at all clear how one could implement the removal of vertices in Z and that makes our task significantly more challenging. To avoid this obstacle, we forgo the technique of shadow removal and directly design an intricate branching algorithm.

The crux of this algorithm is the observation that for a special type of solution X, for every forbidden subgraph F, either X intersects $V(F)$ or X contains an S_1-$\{r(F)\}$ separator ($r(F)$ denotes a fixed root of F) or X contains a $\{u\}$-W separator for some vertex $u \in V(F)$. We then show that there is always an efficiently computable forbidden subgraph upon which we can branch exhaustively using the above observation in such a way that we always make progress. The fact that such a subgraph can always be identified efficiently is far from obvious and proving it is one of our main technical challenges.

Algorithm for p-$\mathcal{H}$-SCC Deletion. We show that for every finite family of digraphs $\mathcal{H}$, there exists another (infinite) family $\mathcal{H}^*$, such that the $\mathcal{H}$-SCC Deletion problem is equivalent to the problem of deleting at most k vertices to exclude all graphs in $\mathcal{H}^*$ as subgraphs in the remaining graph (not necessarily in a single strong component). Moreover, we show that when $\mathcal{H}$ contains a directed path, then the family $\mathcal{H}^*$ can be partitioned into two, say $\mathcal{H}_1^*$ and $\mathcal{H}_2^*$ such that $\mathcal{H}_1^*$ is finite and the problem of deleting at most k vertices to exclude all graphs in $\mathcal{H}_2^*$ as subgraphs in the remaining graph is equivalent to the r-$\mathcal{X}$-SCC Deletion where $\mathcal{X}$ only contains a directed path whose length depends on $\mathcal{H}$. This allows us to branch on all subgraphs isomorphic to graphs in $\mathcal{H}_1^*$ and then invoke Theorem 1.

Improved algorithms for 1-Out-Regular Vertex Deletion and BSSCV. Here, we begin in the same way as for r-$\mathcal{H}$-SCC Deletion by obtaining a tuple $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ such that if there is a solution for (D, k), then there is a solution that is disjoint from W and intersects all S_i-S_j paths for $i < j$. In the case of 1-Out-Regular Vertex Deletion, the main new contribution that results in a speedup over Theorem 1 is a lemma that shows that if we consider an S_1-$W \setminus S_1$ separator C such that every vertex reachable from S_1 in $D - C$ has out-degree at most 1 in D, then there is a solution whose intersection with this set of reachable vertices is contained within an efficiently computable set of $O(k)$ vertices. This gives us a branching algorithm where we essentially compute a "furthest" S_1-$W \setminus S_1$ separator C of size at most k in time $2^{O(k)} \cdot n^{O(1)}$ and then branch on deleting a vertex of C or one of these $O(k)$ vertices in the reachable set. In the case of BSSCV, we show that if for some $x \in S_1$, D contains a subgraph of size $s+1$ with an arborescence rooted at x,

then at least one of these vertices must either be deleted or must have all its paths to W deleted when removing a solution. In the latter case, we will be able to branch on an $\{x\}$-W important separator [11] of size at most k.

Further Remarks. The results of Göke et al. [5] crucially use the technique of *covering the shadow* which adds a factor of $2^{O(k^2)} \cdot n^{O(1)}$ to the running time of any algorithm that uses it. Thus, our $2^{O(k \log k)} \cdot n^{O(1)}$ algorithm (Theorem 13) is an improvement over what is currently possible using the shadow covering technique. Moreover, although all our results are stated for the vertex deletion version of the problems, we would like to mention that these results will apply for the corresponding arc deletion versions of the problems as well, as all our proofs go through for the later case also. Due to space constraints, we only discuss the proof of Theorems 1 and 2 in this version. The proofs of the remaining theorems and the missing proofs of the lemmas in this version can be found in the appended full version of the paper.

2 Preliminaries

We use standard notion regarding digraphs. Given two digraphs D_1, D_2 we denote the digraph $D_1 \cup D_2$ as the digraph with vertex set $V(D_1) \cup V(D_2)$ and arc set $A(D_1) \cup A(D_2)$. By $D_1 \subseteq D_2$, we mean that D_1 is a *subdigraph* of D_2. We use $|D|$ as a shorthand for $|V(D)|$. A *strongly connected component (or strong component)* of a digraph D is a maximal set $S \subseteq V(D)$ such that for any pair u, v of vertices in S, there is a path from u to v and from v to u in D. For any $X \subseteq V(D)$, $D[X]$ denotes the subdigraph of D *induced* by X. For any $S, T \subseteq V(D)$, by an S-T *path* in D we mean a path from some vertex of S to some vertex of T in D. For a subset $S \subseteq V(G)$, by $N^+(S)$ we denote the set $\bigcup_{v \in S} N^+(v) \setminus S$. By $N^+[S]$ we denote $N^+(S) \cup S$. Similar definition hold for $N^-(S)$ and $N^-[S]$. Given a directed graph D and subsets $S, T, C \subseteq V(D)$ such that $S \cap T = \emptyset$, we say that C is an S-T *separator* if there is no directed S-T path in $D - C$ and $C \cap S = C \cap T = \emptyset$. For an S-T separator C, by $R_D(S, C)$ we denote the set of vertices v such that there exists an S-$\{v\}$ path in $D - C$. By $\overline{R}_D(S, C)$ we denote the set $V(D) \setminus R_D(S, C)$, that is the set of vertices that are unreachable from S after removing C. Note that for any set $R \subseteq V(D)$ such that $S \subseteq R$, $R \cap T = \emptyset$ and $N^+(R) \cap T = \emptyset$, if R is reachable from S in $D[R]$, then the set $C = N^+(R)$ is an S-T separator such that $R_D(S, C) = R$. We say that an S-T separator C *covers* an S-T separator C' if $R_D(S, C) \supseteq R_D(S, C')$. We say that C *tightly covers* C' if C covers C' and there does not exist a C'' that covers C' and is covered by C. Two S-T separators are *incomparable* if neither covers the other. $\lambda_D(S, T)$ denotes the size of a minimum S-T separator in D. It is well known [2] that there exists a unique minimum S-T separator $C_{\min}(S, T)$ and a unique minimum S-T separator $C_{\max}(S, T)$ such that for every minimum S-T separator C, $C_{\min}(S, T)$ is covered by C and $C_{\max}(S, T)$ covers C. We call $C_{\min}(S, T)$ the *closest minimum S-T separator* and $C_{\max}(S, T)$ the *furthest minimum S-T separator*. Moreover, we define $R_{\min}(S, T) = R_D(S, C_{\min}(S, T))$ and $R_{\max}(S, T) = R_D(S, C_{\max}(S, T))$. All four of these sets can be computed in polynomial time using max-flow computations (see [11]).

An *arborescence* is a rooted directed tree where every vertex except the root has in-degree exactly 1 and the root has in-degree 0. A digraph D is said to be *rooted* at $v \in V(D)$ if D contains as a subdigraph on $V(D)$ an arborescence rooted at v. That is, there is a directed v-w path in D for every $w \in V(D)$. A digraph D is simply said to be *rooted* if it is rooted at some vertex. By $r(D)$, we denote the vertex that is the root of D. If there are multiple roots for D, we canonically fix one vertex for $r(D)$. For a digraph D and a family of digraphs $\mathcal{H}$ (potentially containing rooted digraphs), we say that a subset $S \subseteq V(D)$ is $\mathcal{H}$*-free* if

$D[S]$ does not contain any graph in $\mathcal{H}$ as a subgraph. When $S = V(D)$, we say that D is $\mathcal{H}$-free. In the case when every graph in $\mathcal{H}$ is a rooted graph, we say that S is *root-$\mathcal{H}$-free* if the root of every subgraph of D that is isomorphic to a graph in $\mathcal{H}$ is not contained in S. Observe that if a set S is root-$\mathcal{H}$-free then it is also $\mathcal{H}$-free. We say that a set $X \subseteq V(D)$ is an *$\mathcal{H}$-deletion set* for D if there is no subgraph isomorphic to a graph in $\mathcal{H}$ that is contained in any strong component of $D - X$. Furthermore, we say that X is a *solution* for the tuple (D, k) if X is an $\mathcal{H}$-deletion set and $|X| \leq k$.

▶ **Lemma 6.** [9] *There is a polynomial-time algorithm that, given a digraph D and an S-T separator C in D, either correctly concludes that $C = R_{\max}(S, T)$ or outputs a minimum S-T separator that tightly covers C.*

▶ **Lemma 7.** [7] *Let D be a digraph and let $S, T \subseteq V(D)$ be disjoint. Let C be the closest (resp. furthest) S-T separator in D and let v be a vertex in $R_D(S, C)$ (resp. $\overline{R}_D(S, C)$). Then every S-$(T \cup \{v\})$ (resp. $(S \cup \{v\})$-T) separator is of size strictly greater than $\lambda_D(S, T)$.*

▶ **Lemma 8.** *Let D be a digraph and let S, T be disjoint subsets of $V(D)$. Let $C_1 = N^+(R_1)$ and $C_2 = N^+(R_2)$ be two minimum S-T separators such that C_2 covers C_1 and $C_1 \neq C_2$. Let $u \in C_1$ and $v \in R_2 \setminus N[R_1]$. Then every $(S \cup \{u\}) - (T \cup \{v\})$ separator is of size strictly greater than $\lambda_D(S, T)$.*

3 The FPT algorithm for rooted $\mathcal{H}$-SCC Deletion

We use the standard technique of Iterative Compression ([13]) to reduce the task of solving our instance of R-$\mathcal{H}$-SCC Deletion to that of solving at most $2^{k+1}n$ instances of the Disjoint R-$\mathcal{H}$-SCC Deletion Compression (R-$\mathcal{H}$-SCC DC) problem, where we are given a digraph D and a solution W of size at most $k + 1$ and the goal is to compute a solution of size at most k that is disjoint from W, if one exists. We further solve R-$\mathcal{H}$-SCC DC by making $2^{O(k \log k)}$ calls to a subroutine that is allowed to assume the existence of a specific type of solution for instances of the R-$\mathcal{H}$-SCC Partitioned Compression (R-$\mathcal{H}$-SCC PC) problem, which is described below. An instance of R-$\mathcal{H}$-SCC PC is a tuple $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$, where W is a solution for the instance $(D, k + 1)$ of $\mathcal{H}$-SCC and $\mathcal{S}$ is an ordered partition of W. A set $X \subseteq V(D)$ is said to be a solution for the instance $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ of R-$\mathcal{H}$-SCC PC if X is a solution for the instance (D, k) of $\mathcal{H}$-SCC, $X \cap W = \emptyset$, and X intersects all S_i-S_j paths in D, for every $j > i$. Observe that every solution of the instance $(D, \mathcal{S}, W, k)$ of R-$\mathcal{H}$-SCC PC, is also a solution of (D, W, k) of R-$\mathcal{H}$-SCC DC. We now define a special kind of solution for R-$\mathcal{H}$-SCC PC, which we call a *nice solution*, and as we will see soon, it turns out that it is enough to look for nice solutions for our purpose. Let $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ be an instance of R-$\mathcal{H}$-SCC PC. A solution X for this instance is said to be *nice* if for every subgraph $F \subseteq D$ such that F is isomorphic to a graph in $\mathcal{H}$, and each $i \in [q]$, one of the following holds: (1) X intersects $V(F)$, or (2) $r(F) \notin R(S_i, X)$, or (3) $\exists v \in V(F)$ such that there is no $\{v\}$-S_i path in $D - X$. Observe from the definition above, that if X is a solution of R-$\mathcal{H}$-SCC PC such that after its deletion each S_i is in exactly one strong component, then X is a nice solution. One can formalize the above discussion (refer to the full version for details) to conclude that (D, W, k) is a yes-instance of R-$\mathcal{H}$-SCC DC if and only if $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ has a nice solution for some ordered partition $\mathcal{S} = (S_1, \ldots, S_q)$ of W. Thus, to prove Theorem 1 it is enough to prove the following lemma. Recall that $h = \max_{H \in \mathcal{H}} |V(H)|$.

▶ **Lemma 9.** *There is an algorithm that, given an instance $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ of R-$\mathcal{H}$-SCC PC, runs in time $2^{O(k^3 \log k)} \cdot h^{O(k)} \cdot n^{O(h)}$ and either correctly concludes that there is no nice solution for $(D, \mathcal{S}, W, k)$ or outputs some solution (not necessarily nice) for $(D, \mathcal{S}, W, k)$.*

Recall that the most challenging aspect in our strategy (see overview of our algorithm for R-$\mathcal{H}$ SCC Deletion in Section 1) is the identification of appropriate "branchable" forbidden subgraphs. Specifically, we need to identify particular subgraphs F such that the natural exhaustive branchings reduce some measure depending on the parameter. The way we identify such subgraphs is the following: the algorithm will maintain a set Q such that Q is root-$\mathcal{H}$-free, with $Q = \emptyset$ initially. The algorithm will try to "grow" the set Q until the entire graph is covered by Q. Initially, when $Q = \emptyset$, we try to grow it to the set $R_{\min}(S, T)$ (where $S = S_1$ and $T = X \setminus S_1$), the closest minimum S-T separator. We prove that if we find a forbidden subgraph whose root lies in $R_{\min}(S, T)$, that subgraph is good for us in the sense that all branches will drop our measure. Once all roots have been removed from $R_{\min}(S, T)$, we set $Q = R_{\min}(S, T)$. Then we recurse and grow Q further towards T. To formalize the above strategy in Section 3.2, we next describe two crucial tools, in the form of pushing lemmas in the next section.

3.1 The Pushing Lemmas

Let $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ be an instance of R-$\mathcal{H}$-SCC PC. Let X be some solution for this instance. Suppose, for instance, one had a hold on the set of vertices, say R, that are in the strong components containing S_1 in $D - X$. Then, one could argue that there is some other solution that picks an important R-$(W \setminus S_1)$ separator (see Marx' survey [11] for definition) of size at most k. In this case, one can branch of these important sets. Unfortunately, the above mentioned set R is far from being found efficiently. However growing on this idea, our first pushing lemma, Lemma 10, says that even if one is able to find a "weaker" set, viz. some superset of the vertices that are reachable from S_1 after the deletion of the solution, do not contain a graph of $\mathcal{H}$ inside them and their out-neighbourhood forms a minimum S_1-$(W \setminus S_1)$, then one can always construct another solution that picks the out-neighbours of this set.

▶ **Lemma 10** (Pushing-Routine-1). *Let $\mathcal{I} = (D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ be an instance of R-$\mathcal{H}$-SCC PC. Consider a $\mathcal{H}$-free set $S_1 \subseteq Q \subseteq V(D) \setminus (W \setminus S_1)$ such that $N^+(Q)$ is a minimum S_1-$(W \setminus S_1)$ separator. Let X be a solution of $\mathcal{I}$ such that $R_D(S_1, X) \cap N^+(Q) = \emptyset$. Then, there is a solution X' for $\mathcal{I}$ that contains $N^+(Q)$.*

For our second pushing lemma, we first borrow definitions of shadows for directed graphs from [4]. Note that what we do with the concept of shadows in our article is very different from the way it has been used so far. More specifically, we do not resort to the shadow removal technique that has often been an effective technique to design FPT algorithms for directed cut problems. In fact, for the general problems that we consider, it is not at all clear how the shadow removal technique could be helpful. Let $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ be an instance of R-$\mathcal{H}$-SCC PC and let $X \subseteq V(D) \setminus W$. Then, $\mathsf{F\text{-}Shadow}(X)$ denotes the set of those vertices $u \notin X$ such that there is no $\{u\}$-W path in D-X. Similarly, $\mathsf{R\text{-}Shadow}(X)$ denotes the set of those vertices $u \notin X$ such that there is no W-$\{u\}$ path in D-X. $\mathsf{F\text{-}Shadow}(X)$ is called the *forward shadow of X with respect to W* and $\mathsf{R\text{-}Shadow}(X)$ is called the *reverse shadow of X with respect to W*. Notice that with these definitions, we have that if X is a nice solution for the instance $(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$, then for any subgraph $F \subseteq D$ that is isomorphic to a graph in $\mathcal{H}$ and any $i \in [q]$, either X intersects $V(F)$ or $r(F) \notin R(S_i, X)$ or $V(F) \cap \mathsf{F\text{-}Shadow}(X) \neq \emptyset$. Moreover, when $q = 1$, this implies that X intersects $V(F)$ or $r(F) \in \mathsf{R\text{-}Shadow}(X)$ or $V(F) \cap \mathsf{F\text{-}Shadow}(X) \neq \emptyset$. Our second pushing lemma, guarantees the existence of a set to branch on, provided we have identified some vertex in the forward or reverse shadow of X with respect to W.

▶ **Lemma 11** (Pushing-Routine-2). *There is an algorithm that, given an instance $\mathcal{I} = (D, \mathcal{S}, W, k)$ of r-$\mathcal{H}$-SCC PC and a vertex $u \in V(D)$ such that either there is a u-W path or a W-u path in D, runs in time $4^k \cdot n^{O(1)}$ and outputs a non-empty set $Z \subseteq V(D)$ of size at most $4^k \cdot 2k$ with the following property: if there is a solution X for $\mathcal{I}$ such that $u \in$ F-Shadow$(X) \cup$ R-Shadow(X), then there is a solution X' for $\mathcal{I}$ such that $X' \cap Z \neq \emptyset$.*

3.2 Solving instances of r-$\mathcal{H}$-SCC PC

Towards the proof of Lemma 9, we first find a set $\widehat{Z}$ such that $\widehat{Z}$ intersects some solution for $\mathcal{I} = (D, \mathcal{S}, W, k)$ (if one exists) and $|\widehat{Z}| = O(h) \cdot 2^{O(k^2 \log k)}$. Observe that, having such a set $\widehat{Z}$ at hand, one can proceed with a branching algorithm that branches on the vertices of $\widehat{Z}$, thereby producing an algorithm with running time $O(|Z|^k) \cdot n^{O(1)}$. We call the set $\widehat{Z}$, the *branch set* for the instance $\mathcal{I}$. The rest of the section is devoted to computing a branch set for $\mathcal{I}$ of the desired size. Henceforth, h denotes $\max_{H \in \mathcal{H}} |V(H)|$.

▶ **Lemma 12.** *Given an instance $\mathcal{I} = (D, \mathcal{S}, W, k)$ of r-$\mathcal{H}$-SCC PC, there is an algorithm, that runs in time $O(h) \cdot 2^{O(k^2 \log k)} \cdot n^{O(h)}$ and outputs a branch set for $\mathcal{I}$ of size $O(h) \cdot 2^{O(k^2 \log k)}$ if $\mathcal{I}$ has a nice solution.*

The algorithm of Lemma 12 has two parts: we first design a simple algorithm when $q = 1$ by exploiting the structure of a nice solution to identify a vertex that belongs to the shadow of the solution, thereby allowing the applicability of Lemma 11 to find a branch set. We defer the details of this part to the full version. The second part, when $q > 1$, is tricky. We design a recursive algorithm for the proof of Lemma 12 when $q > 1$, whose details we describe next. To maintain the recursive invariants, we enhance the instance of r-$\mathcal{H}$-SCC PC.

▶ **Definition 13** (Extended Instance of r-$\mathcal{H}$-SCC PC). *An instance $\mathcal{I}^{\text{ext}} = (D, \mathcal{S} = (S_1, \ldots, S_q), W, k, S, T, Q, N_Q)$ is called an* extended instance of r-$\mathcal{H}$-SCC PC *if the following holds:*

1. *$(D, \mathcal{S} = (S_1, \ldots, S_q), W, k)$ is an instance of r-$\mathcal{H}$-SCC PC,*
2. *$S_1 \subseteq S \subseteq V(D) \setminus (W \setminus S_1)$ and $W \setminus S_1 \subseteq T$,*
3. *either $Q = \emptyset$ or, $S \subseteq Q \subseteq V(D) \setminus T$ such that Q is root-$\mathcal{H}$-free and $N^+(Q)$ is a minimum S-T separator in D, and*
4. *$N_Q \subseteq N^+(Q)$.*

▶ **Definition 14** (Solution of an extended instance of r-$\mathcal{H}$-SCC PC). *For an extended instance $\mathcal{I}^{\text{ext}} = (D, \mathcal{S} = (S_1, \ldots, S_q), W, k, S, T, Q, N_Q)$ of r-$\mathcal{H}$-SCC PC, $X \subseteq V(D) \setminus W$ is said to be a solution for $\mathcal{I}^{\text{ext}}$ if the following holds.*

1. *X is a nice solution for $(D, \mathcal{S}, W, k)$,*
2. *X is an S-T separator,*
3. *$S \subseteq R_D(S_1, X)$,*
4. *$N_Q \cap R_D(S, X) = \emptyset$ and,*
5. *$X \cap (S \cup T) = \emptyset$.*

The idea behind extending an instance of r-$\mathcal{H}$-SCC PC in the way defined earlier is to get the situation closer to the applicability of Lemma 10. In fact, as we will see, this will form the base case for our arguments. To be more specific, the sets S, T defined in the definition correspond to the set of vertices that one has guessed to be reachable and unreachable, respectively from S_1 in $D - X$, where X is a solution for the original instance. Thus, any solution for the original instance is an S-T separator. Note that, since X is a solution for $(D, \mathcal{S}, W, k)$, the set S_1 itself is reachable from S_1 and $W \setminus S_1$ is unreachable

from S_1 in $D - X$. Therefore we could assume that $S_1 \subseteq S$ and $(W \setminus S_1) \subseteq T$. The set Q in the extended instance is such that $N^+(Q)$ is a minimum S-T separator and Q itself is root-$\mathcal{H}$-free (and hence $\mathcal{H}$-free). The set N_Q is meant to be the subset of $N^+(Q)$ that one has guessed to be unreachable from S in $D - X$. The algorithm aims to slowly "grow" Q using Lemma 6 until we find a subgraph F in D that is isomorphic to some graph in $\mathcal{H}$ and whose root lies in Q (i.e. Q is no longer root-$\mathcal{H}$-free). Then using the fact that a nice solution exists, one can construct the desired branch set by branching of instances with a smaller, appropriately defined, measure.

For the convenience of arguments, we further enhance an extended instance of R-$\mathcal{H}$-SCC PC. The idea behind this is to avoid asking that N_Q has to be unreachable from S in $D - X$ where X is a solution of the extended instance. As we see below, a slight modification to the digraph D and T help us achieve this, thereby easing the arguments used in the final proof.

▶ **Definition 15** (Auxiliary Instance of R-$\mathcal{H}$-SCC PC). *Given an extended instance $\mathcal{I}^{\text{ext}} = (D, \mathcal{S} = (S_1, \ldots, S_q), W, k, S, T, Q, N_Q)$ of R-$\mathcal{H}$-SCC PC, we define an auxiliary instance $\mathcal{I}^{\text{aux}} = (D^{\text{aux}}, \mathcal{S} = (S_1, \ldots, S_q), W, k, S, T^{\text{aux}}, Q, N_Q)$ of R-$\mathcal{H}$-SCC PC as follows: D^{aux} is a supergraph of D that is obtained from D by adding a new vertex t^{aux} in D and adding arcs (u, t^{aux}), for each $u \in N_Q$; $T^{\text{aux}} = T \cup \{t^{\text{aux}}\}$.*

▶ **Definition 16** (Solution of an auxiliary instance of R-$\mathcal{H}$-SCC PC). *Let $\mathcal{I}^{\text{aux}} = (D^{\text{aux}}, \mathcal{S} = (S_1, \ldots, S_q), W, k, S, T^{\text{aux}}, Q, N_Q)$ be an auxiliary instance of R-$\mathcal{H}$-SCC PC obtained from $\mathcal{I}^{\text{ext}} = (D, \mathcal{S} = (S_1, \ldots, S_q), W, k, S, T, Q, N_Q)$, then $X \subseteq V(D) \setminus W$ is said to be a solution for $\mathcal{I}^{\text{aux}}$ if the following holds:*

1. *X is a nice solution for $(D, \mathcal{S}, W, k)$,*
2. *X is an S-T^{aux} separator in D^{aux},*
3. *$S \subseteq R_{D^{\text{aux}}}(S_1, X)$ and,*
4. *$X \cap (S \cup T^{\text{aux}}) = \emptyset$.*

One can prove (details in the full version) that a solution to an extended instance of R-$\mathcal{H}$-SCC PC is also a solution for the corresponding auxiliary version. This together with the discussion above, concludes that it is enough to prove the following lemma.

▶ **Lemma 17** (Find-Branch-Set). *Given an auxiliary instance $\mathcal{I}^{\text{aux}} = (D^{\text{aux}}, \mathcal{S} = (S_1, \ldots, S_q), W, k, S, T^{\text{aux}}, Q, N_Q)$ of R-$\mathcal{H}$-SCC PC where $q > 1$, there is an algorithm that runs in time $O(h) \cdot 2^{O(k^2 \log k)} \cdot n^{O(h)}$ and returns a branch set of $(D, \mathcal{S}, W, k)$ of size $O(h) \cdot 2^{O(k^2 \log k)}$ if $\mathcal{I}^{\text{aux}}$ has a solution.*

Proof. Let $\mathcal{I} = (D, \mathcal{S}, W, k)$. We will design a recursive algorithm Find-Branch-Set. To analyse the depth of recursion, we associate a measure μ with an instance $\mathcal{I}^{\text{aux}} = (D^{\text{aux}}, \mathcal{S} = (S_1, \ldots, S_q), W, k, S, T^{\text{aux}}, Q, N_Q)$ as $\mu(\mathcal{I}^{\text{ext}}) = k^2 + k - \lambda_{D^{\text{aux}}}(S, T^{\text{aux}})^2 - |N_Q|$. For the sake of convenience, we will denote $\lambda_{D^{\text{aux}}}(S, T^{\text{aux}})$ by $\lambda(\mathcal{I}^{\text{aux}})$. In what follows, we give an exhaustive list of cases, and say what the algorithm outputs in each such case, give a proof of correctness for the same, point out the branching width of the recursive calls and argue that the measure μ drops for each of the instances called in each of the recursive calls.

Base Case: Observe that for any auxiliary instance $\mathcal{I}^{\text{aux}}$, if $\lambda(\mathcal{I}^{\text{aux}}) > k$, then $\mathcal{I}_{\text{aux}}$ has no solution. If $k \leq 0$, then check if any strong component of D has a graph isomorphic to some graph in $\mathcal{H}$. If it does, then $\mathcal{I}^{\text{aux}}$ has no solution, otherwise, return $\emptyset$. Another case that is handled as a base case is when either $\mu(\mathcal{I}^{\text{aux}}) \leq 0$ or $|N_Q| = \lambda(\mathcal{I}^{\text{aux}})$. If $\mu(\mathcal{I}^{\text{aux}}) \leq 0$, we first claim that $|N_Q| = |N^+(Q)| = \lambda(\mathcal{I}^{\text{aux}})$. Since $|N^+(Q)| = \lambda(\mathcal{I}^{\text{aux}})$, it is enough to prove that $|N_Q| = \lambda(\mathcal{I}^{\text{aux}})$. Since $N_Q \subseteq N^+(Q)$, we have that $|N_Q| \leq \lambda(\mathcal{I}^{\text{aux}})$. For the sake of contradiction, suppose that $|N_Q| < \lambda(\mathcal{I}^{\text{aux}})$. Since $\mu(\mathcal{I}^{\text{aux}}) \leq 0$, we have that

$k^2 + k \leq \lambda(\mathcal{I}^{\text{aux}})^2 + \lambda(\mathcal{I}^{\text{aux}})$. This implies that $|N_Q| \geq \lambda(\mathcal{I}^{\text{aux}})$, which is a contradiction. Thus, we have that $|N_Q| = \lambda(\mathcal{I}^{\text{aux}})$. In this case, Find-Branch-Set($\mathcal{I}^{\text{aux}}$) returns $N^+(Q)$. We now prove that $N^+(Q)$ is indeed a branch set for $\mathcal{I}^{\text{aux}}$. First observe that, in this case $N_Q = N^+(Q)$. From the construction of D^{aux}, there is an arc (u, t^{aux}) for each $u \in N_Q$. Thus, in any solution X of $\mathcal{I}^{\text{aux}}$, $N_Q \cap R_{D^{\text{aux}}}(S, X) = \emptyset$, that is, $N^+(Q) \cap R_{D^{\text{aux}}}(S, X) = \emptyset$. Since any solution X of $\mathcal{I}^{\text{aux}}$ is also a solution for $\mathcal{I}$, we have that there exists a solution X to $\mathcal{I}$, such that $N^+(Q) \cap R_{D^{\text{aux}}}(S, X) = \emptyset$. Then, observe that $\mathcal{I}, Q, N^+(Q), X$ satisfy the properties of Lemma 10, and hence there exists a solution to $\mathcal{I}$ that contains a vertex of $N^+(Q)$. That is, $N^+(Q)$ is a branch set for $\mathcal{I}$. Note that the size of the set outputted in this case is $\lambda(\mathcal{I}^{\text{aux}}) \leq k$. We now proceed to the recursive cases.

Case 1: $[Q = \emptyset]$ The algorithm first computes the unique minimum closest S-T^{aux} separator C. It then checks if there exists $F \subseteq D$ such that F is isomorphic to some graph in $\mathcal{H}$ and $r(F) \in R_D(S, C)$.

Case 1.1: [$\nexists$ a subgraph $F \subseteq D$ such that F is isomorphic to a graph in $\mathcal{H}$ and $r(F) \in R_D(S, C)$] In this case, the algorithm returns Find-Branch-Set($D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}}, R_D(S, C), \emptyset$).

Correctness: Let X be a solution of $\mathcal{I}^{\text{aux}}$. Note that in this case Q is root-$\mathcal{H}$-free and $N^+(Q)$ is a minimum S-T^{aux} separator in D^{aux}. Thus, X is a solution for the auxiliary instance $(D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}}, R_D(S, C), \emptyset)$.

Branching width: It is 1.

Measure drop: Let $\mathcal{I}'^{\text{aux}} = (D^{\text{aux}}, \mathcal{S}, W, k, S, T, R_D(S, C), \emptyset)$. Since the branching width is 1, in this case it is enough to prove that this case cannot occur more than n times and the measure does not increase whenever this case arises. Since in the new instance $\mathcal{I}'^{\text{aux}}$, the size of Q has strictly increased, as it was an empty set before, the resulting instance $\mathcal{I}'^{\text{aux}}$ does not fall into this case again (as we will see later that in all the cases the set Q only grows). Since k, N_Q remains the same in both the instances, and $\lambda(\mathcal{I}'^{\text{aux}}) \geq \lambda(\mathcal{I}^{\text{aux}})$ because $T^{\text{aux}} \cup \{r(F)\} \supseteq T^{\text{aux}}$, we conclude that $\mu(\mathcal{I}'^{\text{aux}}) \leq \mu(\mathcal{I}^{\text{aux}})$.

Case 1.2: [$\exists F \subseteq D$ that is isomorphic to a graph in $\mathcal{H}$, and $r(F) \in R_D(S, C)$] In this case, the algorithm returns $V(F) \cup$ Find-Branch-Set($D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}} \cup \{r(F)\}, \emptyset, \emptyset$) $\cup \bigcup_{u \in V(F)}$ Pushing-Routine-2($D, \mathcal{S}, W, k, u$).

Correctness: Let X be a solution for $\mathcal{I}^{\text{aux}}$. Since by definition, X is a nice solution of $\mathcal{I}$, either $V(F) \cap X \neq \emptyset$, or X contains an S_1-$\{r(F)\}$ separator or if it does not satisfy either of the above two conditions, then it must contain a $\{u\}$-S_1 separator, for some $u \in V(F)$. In the first case, since the returned set contains $V(F)$, we are done. In the second case, if X contains an S_1-$\{r(F)\}$ separator and $S \subseteq R_D(S_1, X)$ (from the definition of an auxiliary solution), then X must also contain an S-$\{r(F)\}$ separator. Thus, in this case X must also be a solution to Find-Branch-Set($D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}} \cup \{r(F)\}, \emptyset, \emptyset$). From induction hypothesis, $(D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}} \cup \{r(F)\}, \emptyset, \emptyset)$ returns a branch set for $(D, \mathcal{S}, W, k)$, and we are done. In the last case, if neither of the above two cases hold, then X contains a $\{u\}$-S_1 separator for some $u \in V(F)$. We will prove that u in in the forward shadow of X with respect to W. Suppose, for the sake of contradiction, that there is a path from u to a vertex $v \in W$ in $D - X$. Note that $v \notin S_1$ as X contains a $\{u\} - S_1$ separator. Since $r(F)$ is reachable from S_1 in $D - X$ (because the second case does not apply), X is disjoint from $V(F)$, and there is a path from $r(F)$ to u contained in $V(F)$ as F is rooted, it follows that u is reachable from S_1 in $D - X$. Furthermore, there is a path from u to v in $D - X$ and thus $v \in W \setminus S_1$ is reachable from S_1 in $D - X$. This is a contradiction to the fact that X is a solution for $\mathcal{I}$. Therefore, it follows that u

is in the forward shadow of X with respect to W. Moreover, since there is an S_1-$\{r(F)\}$ path in D, by rooted-ness there is an S_1-u path and since $S_1 \subseteq W$, there is a W-u path in D. By Lemma 11, PUSHING-ROUTINE-2$(D, \mathcal{S}, W, k, u)$ returns a branch set for $\mathcal{I}$.
Branching width: It is 1.
Measure drop: Let $\mathcal{I}'^{\text{aux}} = (D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}} \cup \{r(F)\}, \emptyset, \emptyset)$. Since the branching width is 1 and in the new instance $\mathcal{I}'^{\text{aux}}$ the size of T^{aux} grows, this case does not happen more than n number of times because, if $T^{\text{aux}} \setminus t^{\text{aux}}$ becomes equal to $V(D) \setminus S_1$, then there is a no solution to the problem and we stop. Again, since the branching width is 1, in this case, we only need to show that $\mu(\mathcal{I}'^{\text{aux}}) \leq \mu(\mathcal{I}^{\text{aux}})$. Since k, N_Q remains the same in both the instances, and $\lambda(\mathcal{I}'^{\text{aux}}) \geq \lambda(\mathcal{I}^{\text{aux}})$ because $T^{\text{aux}} \cup \{r(F)\} \supseteq T^{\text{aux}}$, we conclude that $\mu(\mathcal{I}'^{\text{aux}}) \leq \mu(\mathcal{I}^{\text{aux}})$.

Case 2: $[Q \neq \emptyset]$ The algorithm proceeds by invoking Lemma 6 and either finds a separator C' that tightly covers $N^+(Q)$ in D^{aux} or concludes that $N^+(Q)$ is the furthest minimum S-T^{aux} separator in D^{aux}.

Case 2.1: $[N^+(Q)$ is the furthest minimum S-T^{aux} separator in $D^{\text{aux}}]$ In this case, the algorithm returns $N^+(Q) \cup \bigcup_{v \in N^+(Q) \setminus N_Q}$ FIND-BRANCH-SET$(D^{\text{aux}}, \mathcal{S}, W, k, S \cup \{v\}, T^{\text{aux}}, \emptyset, \emptyset)$.
Correctness: Let $\mathcal{I}_v^{\text{aux}} = (D^{\text{aux}}, \mathcal{S}, W, k, S \cup \{v\}, T^{\text{aux}}, \emptyset, \emptyset)$. Let X be a solution for $\mathcal{I}^{\text{aux}}$. Suppose there exists $v \in N^+(Q) \setminus N_Q$ that is reachable from S in $D^{\text{aux}} - X$, then observe that X is also a solution for $\mathcal{I}_v^{\text{aux}}$. Thus, FIND-BRANCH-SET$(\mathcal{I}_v^{\text{aux}})$ returns a branch set for $\mathcal{I}$. Now we look at the case when no vertex in $N^+(Q) \setminus N_Q$ is reachable from S in $D^{\text{aux}} - X$. Recall that N_Q cannot be reachable from S in $D^{\text{aux}} - X$. Thus the entirety of $N^+(Q)$ is unreachable from S in $D^{\text{aux}} - X$. In this case, observe that $\mathcal{I}, Q, N^+(Q), X$ satisfy the conditions for Lemma 10, and thus, $N^+(Q)$ is a branch set for $\mathcal{I}$.
Branching width: It is at most $|N^+(Q)| = \lambda(\mathcal{I}^{\text{aux}}) \leq k$.
Measure drop: For each $v \in N^+(Q) \setminus N_Q$, we show that $\mu(\mathcal{I}_v^{\text{aux}}) < \mu(\mathcal{I}^{\text{aux}})$. From Lemma 7, the $\lambda(\mathcal{I}_v^{\text{aux}}) \geq \lambda(\mathcal{I}^{\text{aux}})+1$. However the size of N_Q in the new instance decreases to 0. Thus, the drop in μ is at least $(k^2+k-\lambda(\mathcal{I}^{\text{aux}})^2-|N_Q|)-(k^2+k-(\lambda(\mathcal{I}^{\text{aux}})+1)^2-0) = \lambda(\mathcal{I}^{\text{aux}})^2 - |N_Q| - (\lambda(\mathcal{I}^{\text{aux}})+1)^2 = 2\lambda(\mathcal{I}^{\text{aux}}) + 1 - |N_Q|$. Since $|N_Q| \leq \lambda(\mathcal{I}^{\text{aux}})$, the drop is $\geq \lambda(\mathcal{I}^{\text{aux}}) + 1$.

Case 2.2: $[N^+(Q)$ is not the furthest minimum S-T^{aux} separator in $D^{\text{aux}}]$ From Lemma 6, the algorithm first finds a separator C' that tightly covers $N^+(Q)$. Let $Q' = R_D(S, C')$ that is $C' = N(Q')$. It then checks if there exists $F \subseteq D$ such that F is isomorphic to some graph in $\mathcal{H}$ and $r(F) \in Q'$.

Case 2.2.1: $[\nexists$ a subgraph isomorphic to a graph in $\mathcal{H}$ whose root is in $Q']$ In this case, the algorithm returns FIND-BRANCH-SET$(D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}}, Q', N_Q)$.
Correctness: Let $\mathcal{I}'^{\text{aux}} = (D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}}, Q', N_Q)$. From construction, $N(Q')$ is a minimum $S - T^{\text{aux}}$ separator and Q' is root-$\mathcal{H}$-free. Also since $N^+(Q')$ covers $N^+(Q)$ and is an S-T^{aux} separator and for each $v \in N_Q$, (v, t^*) is an arc in D^{aux}, it follows that $N_Q \subseteq N^+(Q')$. Thus, if X is a solution to $\mathcal{I}^{\text{aux}}$, then it is also a solution to $\mathcal{I}'^{\text{aux}}$.
Branching width: It is 1.
Measure drop: Since the branching width is 1 and $Q' \supset Q$, it is enough to prove that the measure does not increase. This is indeed the case, as k, N_Q remain the same in both the instances. Also, since S and T^{aux} remain the same, $\lambda(\mathcal{I}^{\text{aux}}) = \lambda(\mathcal{I}'^{\text{aux}})$.

Case 2.2.2: $[\exists F \subseteq D$ that is isomorphic to a graph in $\mathcal{H}$ and $r(F) \in Q']$ In this case, the algorithm returns $V(F) \cup$ FIND-BRANCH-SET$(D^{\text{aux}}, \mathcal{S}, W, k, S \cup (N^+(Q) \setminus N_Q), T^{\text{aux}} \cup \{r(F)\}, \emptyset, \emptyset) \cup \bigcup_{v \in N^+(Q) \setminus N_Q}$FIND-BRANCH-SET$(D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}}, Q, N_Q \cup \{v\}) \cup \bigcup_{v \in V(F)}$ PUSHING-ROUTINE-2$(D, \mathcal{S}, W, k, u)$.

Correctness: Let X be a solution of $\mathcal{I}^{\text{aux}}$. Since X is a nice solution of $\mathcal{I}$, either $X \cap V(F) \neq \emptyset$ or, X contains an S_1-$\{r(F)\}$ separator or, X contains a $\{v\}$-S_1 separator, for some $v \in V(F)$. In the first case, since $V(F)$ is present in the set returned, we are done. In the second case, since S is reachable from S_1 in $D^{\text{aux}} - X$, it follows that X is an S-$\{T \cup r(F)\}$ separator. If there exists a vertex $v \in N^+(Q) \setminus N_Q$ that is unreachable from S in $D^{\text{aux}} - X$, then observe that X is also a solution for Find-Branch-Set$(D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}}, Q, N_Q \cup \{v\})$. Thus, we are done. Otherwise, $N^+(Q) \setminus N_Q$ is reachable from S in $D^{\text{aux}} - X$. In this case, X is also a solution for Find-Branch-Set$(D^{\text{aux}}, \mathcal{S}, W, k, S \cup (N^+(Q) \setminus N_Q), T^{\text{aux}} \cup \{r(F)\}, \emptyset, \emptyset)$, and hence, we are done. In the third case, X contains a $\{v\}$-S_1 separator, for some $v \in V(F)$. Using a similar argument as in **Case 1**, it follows that u is in the forward-shadow of X with respect to W and that there is a W-u path in D. Thus, from Lemma 11, Pushing-Routine-2$(D, \mathcal{S}, W, k, u)$ returns a branch set for $\mathcal{I}$.
Branching width: It is at most $|N^+(Q)| + 1 \leq \lambda(\mathcal{I}^{\text{aux}}) + 1 \leq k + 1$.
Measure drop: Consider the instance $\mathcal{I}'^{\text{aux}} = (D^{\text{aux}}, \mathcal{S}, W, k, S \cup (N^+(Q) \setminus N_Q), T^{\text{aux}} \cup \{r(F)\}, \emptyset, \emptyset)$. We show that $\mu(\mathcal{I}'^{\text{aux}}) < \mu(\mathcal{I}^{\text{aux}})$. From Lemma 8, the minimum $(S \cup (N^+(Q) \setminus N_Q))$-$(T \cup \{r(F), t^{\text{aux}}\})$ separator is of size greater than $\lambda(\mathcal{I}^{\text{aux}})$. Thus, $\lambda(\mathcal{I}'^{\text{aux}}) > \lambda(\mathcal{I}^{\text{aux}})$. However the N_Q (the last variable in the instance) for $\mathcal{I}'^{\text{aux}}$ is an empty set. Therefore μ drops by at least $(k^2 + k - \lambda(\mathcal{I}^{\text{aux}})^2 - |N_Q|) - (k^2 + k - (\lambda(\mathcal{I}^{\text{aux}}) + 1)^2 - 0) = \lambda(\mathcal{I}^{\text{aux}})^2 - |N_Q| - (\lambda(\mathcal{I}^{\text{aux}}) + 1)^2 = 2\lambda(\mathcal{I}^{\text{aux}}) + 1 - |N_Q|$. Since $|N_Q| \leq \lambda(\mathcal{I}^{\text{aux}})$, the drop is at least $\lambda(\mathcal{I}^{\text{aux}}) + 1$. For each $v \in V(F)$, consider the instance $\mathcal{I}_v^{\text{aux}} = (D^{\text{aux}}, \mathcal{S}, W, k, S, T^{\text{aux}}, Q, N_Q \cup \{v\})$. We now show that $\mu(\mathcal{I}_v^{\text{aux}}) < \mu(\mathcal{I}^{\text{aux}})$. Since $|N_Q \cup \{v\}| = |N_Q| + 1$ and $\lambda(\mathcal{I}_v^{\text{aux}}) = \lambda(\mathcal{I}^{\text{aux}})$, we conclude that the measure drops by one in this case.

This concludes the description of the recursive algorithm together with its correctness. To bound the number of nodes in the recursion tree, since the maximum branching width of the recursion tree is at most $k + 1$ and the depth of the recursion tree is at most $\mu(\mathcal{I}^{\text{aux}}) = k^2 + k - \lambda(\mathcal{I}^{\text{aux}})^2 - |N_Q| \leq k^2 + k$, we conclude that the number of nodes in the recursion tree is $(k+1)^{k^2+k} = 2^{O(k^2 \log k)}$. Since the size of the set returned at the leaf nodes of the recursion tree is at most k and at each level of the recursion tree a set of size at most $h + kh + 1$ is added to the set obtained from the recursive calls, we conclude that the size of the set that the algorithm outputs is at most $(h + kh + 1) \cdot 2^{O(k^2 \log k)} = O(h) \cdot 2^{O(k^2 \log k)}$. ◀

Given an instance $(D, \mathcal{S}, W, k)$ of r-$\mathcal{H}$-SCCPC, the algorithm of Lemma 9 creates an extended instance $\mathcal{I}^{\text{ext}} = (D, \mathcal{S}, W, k, S_1, W \setminus S_1, \emptyset, \emptyset)$ of r-$\mathcal{H}$-SCCPC and calls Find-Branch-Set on $\mathcal{I}^{\text{aux}}$, which is an auxiliary instance of $\mathcal{I}^{\text{ext}}$. The proof of Lemma 9 then follows from Lemma 17. As already argued, Lemma 9 implies Theorem 1. This completes the description and the proof of our FPT algorithm for r-$\mathcal{H}$-SCC Deletion. Observe that the only place where we incur a $n^{O(h)}$ factor in the running time of this algorithm is for checking whether there exists a subgraph of D isomorphic to a graph in $\mathcal{H}$. If this can be done in time $g(h) \cdot n^{O(1)}$, then our algorithm will run in time $g(k, h) \cdot n^{O(1)}$. In particular, if $\mathcal{H}$ comprises of only the $(d+1)$-out-star, the algorithm will run in time that is FPT in k and d (Theorem 2).

4 Conclusions

We have initiated the study of the parameterized complexity of $\mathcal{H}$-SCC Deletion problem, where the objective is to compute a maximum subdigraph where no strong component contains a forbidden subgraph from the family $\mathcal{H}$. We have obtained fixed-parameter algorithms for this problem when $\mathcal{H}$ either contains at least one path or only contains rooted graphs,

thus unifying several known fixed-parameter tractability results including the one for DFVS. Furthermore, we also have results that, for a pair of previously studied special cases of this problem, yield faster FPT algorithms by using our general strategy tailored to these special cases. Our algorithms are FPT parameterized by k for fixed families $\mathcal{H}$ (that contain a path or only rooted digraphs) while an FPT algorithm for this problem parameterized by both k and h (h being the size of the largest graph in $\mathcal{H}$) is unlikely to exist in general.

Our work identifies some natural directions for future research. In particular, can we completely characterize those finite families $\mathcal{H}$ for which this problem is fixed-parameter tractable? In a parallel line of research, Göke, Marx and Mnich [6] gave a fixed-parameter algorithm for the case where $\mathcal{H}$ is the set of all cycles of length *at least* a given positive integer s.

References

1 Karthekeyan Chandrasekaran and Sahand Mozaffari. Odd multiway cut in directed acyclic graphs. In *12th International Symposium on Parameterized and Exact Computation (IPEC)*, pages 12:1–12:12. Springer, 2017.

2 Jianer Chen, Yang Liu, Songjian Lu, Barry O'sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 177–186, 2008.

3 Rajesh Chitnis, Marek Cygan, Mohammataghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Transactions on Algorithms (TALG)*, 11(4):1–28, 2015.

4 Rajesh Chitnis, Mohammad Taghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM Journal on Computing*, 42(4):1674–1696, 2013.

5 Alexander Göke, Dániel Marx, and Matthias Mnich. Parameterized algorithms for generalizations of directed feedback vertex set. In *International Conference on Algorithms and Complexity*, pages 249–261. Springer, 2019.

6 Alexander Göke, Dániel Marx, and Matthias Mnich. Hitting long directed cycles is fixed-parameter tractable. In *47th International Colloquium on Automata, Languages and Programming (ICALP), to appear*, 2020.

7 Alexander Göke, Daniel Marx, and Matthias Mnich. Representative sets and irrelevant vertices: New tools for kernelization. In *47th International Colloquium on Automata, Languages and Programming (ICALP)*, page (to appear), 2020.

8 Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Magnus Wahlström. Fixed-parameter tractability of multicut in directed acyclic graphs. *SIAM Journal on Discrete Mathematics*, 29(1):122–144, 2015.

9 Daniel Lokshtanov, M. S Ramanujan, and Saket Saurabh. When recursion is better than iteration: a linear-time algorithm for acyclicity with few error vertices. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1916–1933. SIAM, 2018.

10 Daniel Lokshtanov, M. S Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2181–2200. SIAM, 2020.

11 Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

12 Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM Journal on Computing*, 43(2):355–388, 2014.

13 Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.