Width Parameterizations for Knot-Free Vertex Deletion on Digraphs

Stéphane Bessy, Marin Bougeret, Alan D.A. Carneiro, Fábio Protti, Uéverton dos Santos Souza

Width Parameterizations for Knot-Free Vertex Deletion on Digraphs

Stéphane Bessy
Université de Montpellier - CNRS, LIRMM, Montpellier, France
stephane.bessy@lirmm.fr

Marin Bougeret
Université de Montpellier - CNRS, LIRMM, Montpellier, France
marin.bougeret@lirmm.fr

Alan D. A. Carneiro
Universidade Federal Fluminense - Instituto de Computação, Niterói, Brazil
aaurelio@ic.uff.br

Fábio Protti
Universidade Federal Fluminense - Instituto de Computação, Niterói, Brazil
fabio@ic.uff.br

Uéverton S. Souza[1]
Universidade Federal Fluminense - Instituto de Computação, Niterói, Brazil
ueverton@ic.uff.br

Abstract

A knot in a directed graph G is a strongly connected subgraph Q of G with at least two vertices, such that no vertex in $V(Q)$ is an in-neighbor of a vertex in $V(G) \setminus V(Q)$. Knots are important graph structures, because they characterize the existence of deadlocks in a classical distributed computation model, the so-called OR-model. Deadlock detection is correlated with the recognition of knot-free graphs as well as deadlock resolution is closely related to the Knot-Free Vertex Deletion (KFVD) problem, which consists of determining whether an input graph G has a subset $S \subseteq V(G)$ of size at most k such that $G[V \setminus S]$ contains no knot. Because of natural applications in deadlock resolution, KFVD is closely related to Directed Feedback Vertex Set. In this paper we focus on graph width measure parameterizations for KFVD. First, we show that: (i) KFVD parameterized by the size of the solution k is W[1]-hard even when p, the length of a longest directed path of the input graph, as well as κ, its Kenny-width, are bounded by constants, and we remark that KFVD is para-NP-hard even considering many directed width measures as parameters, but in FPT when parameterized by clique-width; (ii) KFVD can be solved in time $2^{O(tw)} \times n$, but assuming ETH it cannot be solved in $2^{o(tw)} \times n^{O(1)}$, where tw is the treewidth of the underlying undirected graph. Finally, since the size of a minimum directed feedback vertex set (dfv) is an upper bound for the size of a minimum knot-free vertex deletion set, we investigate parameterization by dfv and we show that (iii) KFVD can be solved in FPT-time parameterized by either $dfv + \kappa$ or $dfv + p$. Results of (iii) cannot be improved when replacing dfv by k due to (i).

2012 ACM Subject Classification Mathematics of computing → Graph theory; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Knot, deadlock, width measure, FPT, W[1]-hard, directed feedback vertex set

Digital Object Identifier 10.4230/LIPIcs.IPEC.2019.2

Related Version A full version of the paper is available at `http://arxiv.org/abs/1910.01783`.

[1] corresponding author

14th International Symposium on Parameterized and Exact Computation (IPEC 2019).
Editors: Bart M. P. Jansen and Jan Arne Telle; Article No. 2; pp. 2:1–2:16

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding Supported by Grant E-26/203.272/2017, Rio de Janeiro Research Foundation (FAPERJ) and by Grant 303726/2017-2, National Council for Scientific and Technological Development (CNPq).

Acknowledgements We thank Ignasi Sau for introducing Alan Carneiro to Stéphane Bessy and Marin Bougeret.

1 Introduction

The study of the Knot-Free Vertex Deletion problem emerges from its application in resolution of deadlocks, where a deadlock is detected in a distributed system and then a minimum cost deadlock-breaking set must be found and removed from the system. More precisely, distributed computations are usually represented by directed graphs called *wait-for graphs*. In a wait-for graph $G = (V, E)$, the vertex set V represents processes, and the set E of directed arcs represents wait conditions [4]. An arc exists in E directed away from $v_i \in V$ towards $v_j \in V$ if v_i is blocked waiting for a signal from v_j. The graph G changes dynamically according to a set of prescribed rules (the *deadlock model*), as the computation progresses. In essence, the deadlock model governs how processes should behave throughout computation, i.e., the deadlock model specifies rules for vertices that are not *sinks* (vertices with at least one out-neighbor) in G to become sinks [3] (vertices without out-neighbors). The two main classic deadlock models are the AND model, in which a process v_i can only become a sink when it receives a signal from *all* the processes in $N^+(v_i)$, where $N^+(v_i)$ stands for the set of out-neighbors of v_i (a conjunction of resources is needed); and the OR model, in which it suffices for a process v_i to become a sink to receive a signal from *at least one* of the processes in $N^+(v_i)$ (a disjunction of resources is sufficient). Distributed computations are dynamic, however deadlock is a stable property, in the sense that once it occurs in a consistent global state of a distributed computation, it still holds for all the subsequent states. Therefore, as it is typical in deadlock studies, G represents a static wait-for graph that corresponds to a *snapshot* of the distributed computation in the usual sense of a consistent global state [13]. Thus, the motivation of our work comes from *deadlock resolution*, where deadlocks are detected into a consistent global state G, and must be solved through some external intervention such as aborting one or more processes to break the circular wait condition causing the deadlock.

Deadlock resolution problems differ according to the considered deadlock model, i.e., according to the graph structure that characterizes the deadlock situation. In the AND-model, the occurrence of deadlocks is characterized by the existence of cycles [3, 5]. Therefore, deadlock resolution by vertex deletion in the AND-model corresponds precisely to the well-known Directed Feedback Vertex Set (DFVS) problem, proved to be NP-hard in the seminal paper of Karp [24], and proved to be FPT in [14]. On the other hand, the occurrence of deadlocks in wait-for graphs G working according to the OR-model are characterized by the existence of *knots* in G [5, 21]. A knot in a directed graph G is a strongly connected subgraph Q of G (with at least two vertices) such that there is no arc uv of G with $u \in V(Q)$ and $v \notin V(Q)$. Thus, deadlock resolution by vertex deletion in the OR-model can be viewed as the following problem.

Knot-Free Vertex Deletion (KFVD)
Instance: A directed graph $G = (V, E)$; a positive integer k.
Question: Determine if G has a set $S \subset V(G)$ such that $\|S\| \leq k$ and $G[V \setminus S]$ is knot-free.

Notice that a digraph G is knot-free if and only if for any vertex v of G, v has a path to a sink.

In [12], Carneiro, Souza, and Protti proved that KFVD is NP-complete; and, in [11], it was shown that KFVD is W[1]-hard when parameterized by k.

KFVD is closely related to DFVS not only because of their relation to deadlocks, but also some structural similarities between them: the goal of DFVS is to obtain a direct acyclic graph (DAG) via vertex deletion (in such graphs *all* maximal directed paths end at a sink); the goal of KFVD is to obtain a knot-free graph, and in such graphs for every vertex v there *exists* at least one maximal path containing v that ends at a sink. Finally, every directed feedback vertex set is a knot-free vertex deletion set; thus an optimum for DFVS provides an upper bound for KFVD. Although Directed Feedback Vertex Set is a well-known problem, this is not the case of Knot-Free Vertex Deletion, which we propose to analyze more deeply in this work.

Let S be a solution for KFVD, and let Z be the set of sinks in $G[V \setminus S]$. One can see that any $v \in V \setminus S$ has a path (that does not use any vertex in S) to a vertex in Z. Thus, KFVD can be seen as the problem of creating a set Z of sinks (doing at most k vertex removals) such that every remaining vertex has a path (in $G[V \setminus S]$) to a vertex in Z. In this paper, we denote the set of deleted vertices by S, and the set of sinks in $G[V \setminus S]$ by Z.

To get intuition on KFVD, note that the choice of the vertices to be removed must be carefully done, since the removal of a subset of vertices can turn some strongly connected components into new knots that will need to be broken by the removal of some internal vertices. Ideally, it is desirable to solve the current knots by removing as few vertices as possible for each knot, without creating new ones. Unfortunately, the generation of other knots can not always be avoided.

In [10, 12], Carneiro, Souza, and Protti present a polynomial-time algorithm for KFVD in graphs with maximum degree three. They also show that the problem is NP-complete even restricted to planar bipartite graphs G with maximum degree four. Later, in [11], a parameterized analysis of KFVD is presented, where it was shown that: KFVD is W[1]-hard when parameterized by the size of the solution; and it can be solved in $2^{k \log \varphi} n^{O(1)}$ time, but assuming SETH it cannot be solved in $(2-\epsilon)^{k \log \varphi} n^{O(1)}$ time, where φ is the size of the largest strongly connected subgraph.

Since the introduction of directed treewidth, much effort has been devoted to identify algorithmically useful digraph width measures [26]. Useful width measures imply polynomial time tractability for many combinatorial problems on digraphs of constant width. Since KFVD is W[1]-hard when parameterized by k, in this paper we investigate the ecology of width measures in order to find useful parameters to solve KFVD in FPT time. First, taking k as parameter, we show that KFVD remains W[1]-hard even on instances with both longest directed path and K-width bounded by constants. From the same reduction, it follows that KFVD is para-NP-hard even considering many width measures as parameters, such as directed treewidth and DAG-width. Contrasting with the hardness of KFVD on several directed width measure parameterizations, we show that KFVD is FPT when parameterized by the clique-width of the underlying undirected graph; and it can be solved in $2^{O(tw)} \times n$ time, but assuming ETH it cannot be solved in $2^{o(tw)} \times n^{O(1)}$ time, where tw is the treewidth of the underlying undirected graph. After that, we consider the most natural width parameter related to KFVD, the size of a minimum directed feedback vertex set (dfv). Such a parameter is at the same time a measure of the distance from the input graph to a DAG as well as an upper bound for the size of a minimum knot-free vertex deletion set. Finally, we show that KFVD can be solved in FPT time either parameterized by dfv and K-width, or dfv and the length of a longest directed path. The complexity of KFVD parameterized only by dfv remains open.

In the rest of this section we give necessary definitions and concepts used in this work. In Section 2 we present some useful observations and preliminary results. In Section 3 we discuss digraph width measures and show the W[1]-hardness. In Section 4 we discuss the consequences of treewidth parameterization. Finally, Section 5 we explore the directed feedback vertex set number as a parameter.

Due to space constraints, some proofs are omitted.

Additional notation. We use standard graph-theoretic and parameterized complexity notations and concepts, and any undefined notation can be found in [9, 17]. We consider here directed graphs. Given a vertex v and a subset of vertices Z, we say that there is a path from v to Z iff there exists $z \in Z$ such that there is a vz-(directed) path. For $v \in V(G)$, let $D(v)$ denote the set of descendants of v in G , i.e. nodes that are reachable from v by a non-empty directed path. Given a set of vertices $C = \{v_1, v_2, \ldots, v_p\}$ of G, we define $D(C) = \bigcup_{i=1}^{p} D(v_i)$. Let $A(v_i)$ denote the set of ancestors of v_i in G, i.e., nodes that reach v_i through a non-empty directed path. We also define $A[v_i] = A(v_i) \cup \{v_i\}$, and given a set of vertices $C = \{v_1, v_2, \ldots, v_p\}$ of G, we define $A(C) = \bigcup_{i=1}^{p} A(v_i)$. For a vertex v of G, the out-neighborhood of v is denoted by $N^+(v) = \{u | vu \in E\}$, and given a set of vertices $C = \{v_1, v_2, \ldots, v_p\}$, we define $N^+(C) = \bigcup_{i=1}^{p} N^+(v_i) \setminus C$. We refer to a Strongly Connected Component as an SCC. A knot in a directed graph G is an SCC Q of G with at least two vertices such that there is no arc uv of G with $u \in V(Q)$ and $v \notin V(Q)$. Finally, a sink (resp. a source) of G is a vertex with out-degree 0 (resp. in-degree 0). Given a subset of vertices S, we denote $G_S = G[S]$ and $\bar{S} = V \setminus S$. Thus, $G_{\bar{S}}$ denote the graph obtained by removing S.

We denote by $dfv(G)$ the size of a minimum directed feedback vertex set of G. We generally use F to denote a directed feedback vertex set and by R the remaining subset, i.e., $R = V \setminus F$. The length of a longest directed path of G is denoted by $p(G)$. The Kenny-width [18] or K-width of G is denoted by $\kappa(G)$ and is the maximum number of distinct directed st-paths in G over all pairs of distinct vertices $s, t \in V(G)$, where two st-paths are distinct iff they do not use the exact same set of arcs. For any function g (like dfv, κ, p), $g(G)$ will be denoted simply by g when the considered graph G can be deduced from the context. In what follows we denote by g-KFVD the KFVD problem parameterized by g ($g = k$ denotes the parameterization by the solution size).

2 Preliminaries

In this section we present some useful remarks and reduction rules. Remind that in the decision version of the problem we are given G and a positive integer k.

The first observation is immediate, as if we can make the graph acyclic, then it will be knot-free.

▶ **Observation 1.** *If $k \geq dfv(G)$ then G is a* yes*-instance.*

The two others observations are less obvious but rather natural.

▶ **Observation 2.** *Let S be a solution with set of sinks Z in $G_{\bar{S}}$, and $s \in S$. Let $S' = S \setminus \{s\}$ and Z' be the set of sinks of $G_{\bar{S'}}$. If there is a path from s to Z' in $G_{\bar{S'}}$ then S' is also a solution.*

Informally, after deleting a vertex s, we can add s back to the graph when it is certain that s has a path to a sink in the current graph. This is detailed by the following lemma and its corollary.

▶ **Lemma 1.** *Let S be a solution with set of sinks Z in $G_{\bar{S}}$. If there exists $s \in S$ with $s \notin N^+(Z)$, then $S' = S \setminus \{s\}$ is also a solution.*

▶ **Corollary 2.** *In any optimal solution S with set of sinks Z in $G_{\bar{S}}$, we have $N^+(Z) = S$.*

▶ **Observation 3.** *Let S be a knot-free vertex deletion with set of sinks Z in $G_{\bar{S}}$. If $|S| \leq k$ then for any vertex v with $d^+(v) > k$ it holds that $v \notin Z$.*

To complete the previous observations, we can design two general reduction rules.

▶ **Reduction Rule 1.** *If $v \in V(G)$ is an SCC of size one then remove $A[v]$.*

Proof. Let G' be the graph obtained by removing $A[v]$. Let of first show that (G, k) is a *yes*-instance implies that (G', k) is also a *yes*-instance. Let S be a solution of G of size at most k with set of sinks Z in $G_{\bar{S}}$. Let $S' = S \setminus A[v]$, and Z' the set of sinks in $G'_{\bar{S}'}$. Let us prove that every $u \in V(G'_{\bar{S}'})$ has a path ot Z' in $G'_{\bar{S}'}$. Let $u \in V(G'_{\bar{S}'})$. As u is also in $V(G_{\bar{S}})$, there is a uz-path P in $G_{\bar{S}}$ where $z \in Z$. As $u \notin A[v]$, $V(P) \cap A[v] = \emptyset$ and thus, the path P still exists in $G'_{\bar{S}'}$. Moreover, $u \notin A[v]$ implies that $N^+(z) \cap A[v] = \emptyset$, and thus that $N^+(v) \subseteq S'$, implying that $z \in Z'$.

Let us now consider the reverse implication, and let S' be a solution of G' of size at most k with set of sinks Z' in $G'_{\bar{S}'}$ and prove that S' is a solution of G. Let us start with $u \in V(G_{\bar{S}'}) \setminus A[v]$. As S' is a solution of G' and $u \in V(G'_{\bar{S}'})$, there is uz'-path P' in $G'_{\bar{S}'}$ where $z' \in Z'$, and this path still exists in $G_{\bar{S}'}$. As $N^+(z') \cap A[v] = \emptyset$, z' is still a sink in $G_{\bar{S}'}$ and we are done. Consider now a vertex $u \in V(G_{\bar{S}'}) \cap A[v]$. As $S' \cap A[v] = \emptyset$, there is uv-path P in $G_{\bar{S}'}$. If $N^+(v) \subseteq S'$ then v is a sink in $G_{\bar{S}'}$ and we are done. Otherwise, let $w \in N^+(v) \setminus S'$. As v is a SCC of size 1, $N^+(v) \cap A[v] = \emptyset$, implying that $w \in V(G_{\bar{S}'}) \setminus A[v]$, and thus according to the previous case w has a path to a sink in $G_{\bar{S}'}$. ◀

The previous reduction rule removes in particular sources and sinks, as they are SCC's of size one.

▶ **Reduction Rule 2.** *Let U_i be a strongly connected component of G with strictly more than k out-neighbors in $G[V \setminus V(U_i)]$. Then we can safely remove $A[U_i]$.*

Proof. Let G' be the graph obtained by removing $A[U_i]$. Let us first show that (G, k) is a *yes*-instance implies that (G', k) is also a *yes*-instance. Let S be a solution of G of size at most k and Z the set of sinks in $G_{\bar{S}}$. Let $S' = S \setminus A[U_i]$, and Z' the set of sinks in $G'_{\bar{S}'}$. Using the same argument (replacing $A[v]$ by $A[U_i]$) as in the first part of proof of Reduction 1, we get that every $u \in V(G'_{\bar{S}'})$ has a path ot Z' in $G'_{\bar{S}'}$.

Let us now consider the reverse implication, and let S' be a solution of G' of size at most k with set of sinks Z' in $G'_{\bar{S}'}$ and prove that S' is a solution of G. Let us start with $u \in V(G_{\bar{S}'}) \setminus A[v]$. As S' is a solution of G' there is uz'-path P' in $G'_{\bar{S}'}$ where $z' \in Z'$, and this path still exists in $G_{\bar{S}'}$. As $N^+(z') \cap A[U_i] = \emptyset$, z' is still a sink in $G_{\bar{S}'}$ and we are done. Consider now a vertex $u \in V(G_{\bar{S}'}) \cap A[U_i]$. As $S' \cap A[U_i] = \emptyset$, there is uU_i-path P in $G_{\bar{S}'}$. As U_i has strictly more than k out-neighbors in $G[V \setminus V(U_i)]$, there is arc from U_i to $w \in V(G_{\bar{S}'})$ and thus according to the previous case w has a path to a sink in $G_{\bar{S}'}$. ◀

3 W[1]-hardness and directed width measures

k-KFVD was shown to be W[1]-hard using a reduction from k-Multicolored Independent Set (k-MIS) [11]. However, the gadget used in this reduction to encode each color class has a longest directed path of unbounded length. First, we remark that it is possible to modify the reduction in order to prove that k-KFVD is W[1]-hard even if the input graph G has longest path length and K-width bounded by constants.

▶ **Theorem 3.** *There is a polynomial-time reduction, preserving the size of the parameter, from* k-MIS *to* k-KFVD *such that the resulting graph has longest directed path of length at most* 5 *and K-width equal to* 2.

Proof. Let (G', k) be an instance of Multicolored Independent Set, and let $V^1, V^2, \ldots, V^k$ be the color classes of G'. We construct an instance (G, k) of Knot-Free Vertex Deletion with bounded longest path length and K-width as follows.

1. for each $v_i \in V(G')$, create a directed cycle of size two with the vertices w_i and z_i in G;
2. for a color class V^j in G', create one vertex u_j;
3. for each vertex z_i in G corresponding to a vertex v_i of the color class V^j in G', create an arc from z_i to u_j and from u_j to z_i.
4. for each vertex w_i in G corresponding to a vertex v_i of the color class V^j in G', create an arc from u_j to w_i
5. for each edge $e_p = (v_i, v_l)$ in G' create a set X_p with two artificial vertices x_p^i and x_p^l and the arcs $x_p^i x_p^l$ and $x_p^l x_p^i$;
6. for each artificial vertex x_p^i, create an edge from x_p^i towards z_i in G.

Finally, set $Y_j = \{w_i, z_i \ : \ v_i \in V^j\} \cup \{u_j\}$, Y_j is the set of vertices of G corresponding to the vertices of G' in the same color class V^j. Notice that, the longest path of G has at most 5 vertices, and for any pair s, t in $V(G)$ there are at most 2 distinct directed st-paths in G.

Now, suppose that now S' is a k-independent set with exactly one vertex of each set V^j of G'. By construction, G has k knots which are $G[Y_1], \ldots, G[Y_k]$. Thus, at least k vertex removals are necessary to make G free of knots. We set $S = \{z_i \mid v_i \in S'\}$ and show that $G[V \setminus S]$ is knot-free. For $j = 1, \ldots, k$ the vertex w_j is a sink in $G \setminus S$, and every vertex of $Y_j \setminus S$ still reaches w_j. Now, as S' is a k-independent set of G' each set X_p in G is adjacent to at least one vertex that is not in S. Hence, each X_p will still have at least one arc pointing outside X_p, i.e., no new knots are created, and $G \setminus S$ is knot-free.

Conversely, suppose that G has a set of vertices S of size k such that $G[V \setminus S]$ is knot-free. In particular S has to contain exactly one vertex of each of the knot Y_j, for $j = 1, \ldots, k$. Since at least one sink has to be created in order to untie the knot Y_j, and since the only vertices of Y_j with only one out-neighbor are the w's ones, S has to contain a vertex z_i of each set $Y_1, \ldots, Y_k$. Moreover by deleting one vertex z_i in a knot Y_j, the vertex w_j is turned into sink and every other vertex of the same knot still has a path to w_j. Since $G[V \setminus S]$ is knot-free, no new knots are created by the deletion of S; thus, every SCC X_p will still have at least one arc pointing outside it. So, we set $S' = \{v_i \mid z_i \in S\}$. Since each SCC X_p corresponds to an edge of G', and at least one vertex of each edge of G' is not in S', the set S' contains no pair of adjacent vertices. Moreover, S' is composed by one vertex of each knot, which corresponds to a color of G'. Therefore, S' is a multicolored independent set of G'. ◀

▶ **Corollary 4.** k-KFVD *is W[1]-hard even if the input graph has longest directed path of length at most* 5 *and K-width equal to* 2.

After the introduction of the notion of directed treewidth (dtw) [23], a large number of width measures in digraphs were developed, such as: cycle rank [20] (cr); directed pathwidth [2] (dpw); zig-zag number [25] (zn); Tree-Zig-Zag number [26] (Tzn); Kelly-width [22] (Kelw); DAG-width [6] (dagw); D-width [29] (Dw); weak separator number [26] (s); entanglement [7] (ent); DAG-depth [18] (ddp). However, if a graph problem is hard when both the longest directed path length and the K-width are bounded, then it is hard for all these measures (see Figure 1).

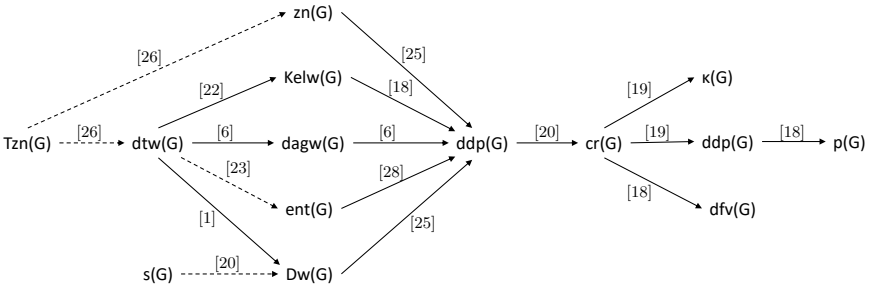

Figure 1 A hierarchy of digraph width measure parameters. $\alpha \to \beta$ indicates that $\alpha(G) \leq f(\beta(G))$ for any digraph G and some function f. More details about the relationships between these parameters can be found in the references corresponding to each arrow.

Therefore, from the reduction presented in Theorem 3 we can observe that KFVD is para-NP-hard with respect to all these width measures, and k-KFVD is W[1]-hard even on inputs where such width measures are bounded. Thus, it seems to be extremely hard to identify nice width parameters for which KFVD can be solved in FPT-time or even in XP-time. Fortunately, there remain some parameters for which, at least, XP-time solvability is achieved. One of them is the *directed feedback vertex set number* (dfv). This invariant is an upper bound on the size of a minimum knot-free vertex deletion set, so XP-time algorithms are trivial. This parameter is discussed in more detail in Section 5.

Another interesting width parameter for directed graphs G that is not bounded by a function of the K-width and the length of a longest directed path is the clique-width of G. Courcelle et al. [16] showed that every graph problem definable in LinEMSOL can be solved in time $f(w) \times n^{O(1)}$ on graphs with clique-width at most w, when a w-expression is given as input. Using a result of Oum [27], the same follows even if no w-expression is given.

▶ **Proposition 5.** *[15] There is a monadic second-order formula expressing the following property of vertices x, y and of a set of vertices X of a directed graph G: "$x, y \in X$ and there is a directed path from x to y in the subgraph induced by X".*

From Proposition 5 one can show that KFVD is LinEMSOL-definable. Thus Theorem 6 holds.

▶ **Theorem 6.** KFVD *is FPT when parameterized by clique-width of the underlying undirected graph.*

The fixed-parameter tractability for clique-width parameterization implies fixed-parameter tractability of KFVD for many other popular parameters. For example, it is well-known that the clique-width of a directed graph G is at most $2^{2tw(G)+2} + 1$, where $tw(G)$ is the treewidth of the underlying undirected graph (see [15, Proposition 2.114]). However, although Theorem 6 implies the FPT-membership of the problem parameterized by the treewidth of the underlying undirected graph, the dependence on $tw(G)$ provided by the model checking framework is huge. So, it is still a pertinent question whether such a parameterized problem admits a single exponential algorithm, which is discussed in Section 4.

4 The treewidth of the underlying undirected graph as parameter

Given a tree decomposition $\mathcal{T}$, we denote by t one node of $\mathcal{T}$ and by X_t the vertices contained in the *bag* of t. We assume w.l.o.g that $\mathcal{T}$ is a *nice* tree decomposition (see [17]), that is, we assume that there is a special root node r such that $X_t = \emptyset$ and all edges of the tree are directed towards r and each node t has one of the following four types: *Leaf*, *Introduce vertex*, *Forget vertex*, and *Join*.

Based on the following results we can assume that we are given a nice tree decomposition of G.

▶ **Theorem 7.** *[8] There exists an algorithm that, given an n-vertex graph G and an integer k, runs in time $2^{O(k)} \times n$ and either outputs that the treewidth of G is larger than k, or constructs a tree decomposition of G of width at most $5k + 4$.*

▶ **Lemma 8.** *[17] Given a tree decomposition $(T, \{X_t\}_{t \in V(T)})$ of G of width at most k, one can in time $O(k^2 \cdot \max(|V(T)|, |V(G)|))$ compute a nice tree decomposition of G of width at most k that has at most $O(k|V(G)|)$ nodes.*

Now we are ready to use a nice tree decomposition in order to obtain an FPT-time algorithm with single exponential dependency on $tw(G)$ and linear with respect to n.

▶ **Theorem 9.** Knot-Free Vertex Deletion *can be solved in $2^{O(tw)} \times n$ time, but assuming ETH there is no $2^{o(tw)} n^{O(1)}$ time algorithm for* KFVD*, where tw is the treewith of the underlying undirected graph of the input G.*

Proof. Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a nice tree decomposition of the input digraph G, with width equal to tw. First, we consider the following additional notation and definitions: t is the index of a bag of T; G_t is the graph induced by all vertices $v \in X_{t'}$ such that either $t' = t$ or $X_{t'}$ is a descendant of X_t in T; Given a knot-free vertex deletion set S, for any bag X_t there is a partition of X_t into S_t, Z_t, F_t, B_t where

- S_t (removed) is the set of vertices of X_t that are going to be removed ($S_t = S \cap X_t$);
- Z_t (sinks) is the set of vertices of X_t that are going to be turned into sinks after the removal of S;
- F_t (free/released) is the set of vertices of X_t that, after the removal of S, are going to reach a sink that belongs to $V(G_t)$;
- B_t (blocked) is the set of vertices of X_t that, after the removal of S, are going to reach *no* sink that belongs to $V(G_t)$;

Let $Y \subseteq X_t$. We denote by $A_t(Y)$ the set of vertices in F_t that reach some vertex of Y in the graph induced by $V(G_t) \setminus S_t$.

The recurrence relation of our dynamic programming has the signature $C[t, S_t, Z_t, F_t, B_t]$, representing the minimum number of vertices in G_t that must be removed in order to produce a graph such that for every remaining vertex v either v reaches a vertex in B_t (meaning that it may still be released in the future) or v reaches a vertex that became a sink (possibly the vertex itself), where every vertex in S_t is removed, every vertex in Z_t becomes a sink, every vertex in F_t will have a path to a sink in G_t, and S_t, Z_t, F_t, B_t form a partition of X_t. Notice that the generated table has size $4^{tw} \times tw \times n$, and when $t = r$, $X_t = \emptyset$ and therefore $C[r, \emptyset, \emptyset, \emptyset, \emptyset]$ contains the size of a minimum knot-free vertex deletion set of $G_r = G$.

The recurrence relation for each type of node is described as follows.

First, notice that if $v \in Z_t$ and there is an out-neighbor w of v that is not in S_t, there is an inconsistency, i.e. w must be deleted (must belong to S_t). In addition, if $v \in B_t$ but has an out-neighbor in $Z_t \cup F_t$, there is another inconsistency (v is not blocked), and if $v \in F_t$ but the removal of $S_t \cup B_t$ turns v into an isolated vertex, v is not released, and it must belong to B_t. For the inconsistent cases, $C[t, S_t, Z_t, F_t, B_t] = +\infty$. Such cases can be recognized and treated by simple preprocessing in linear time on the size of the table. Therefore, we consider next only consistent cases.

Leaf Node: If X_t is a leaf node then $X_t = \emptyset$. Therefore

$$C[t, \emptyset, \emptyset, \emptyset, \emptyset] = 0.$$

Insertion Node: Let X_t be a node of T with a child $X_{t'}$ such that $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$. We have the following:

$$C[t, S_t, Z_t, F_t, B_t] = \begin{cases} 1)\ case\ v \in S_t: \\ - \quad C[t', S_t \setminus \{v\}, Z_t, F_t, B_t] + 1, \\ 2)\ case\ v \in Z_t: \\ - \quad \min_{A' \subseteq A_t(v)} \{C[t', S_t, Z_t \setminus \{v\}, F_t \setminus A', B_t \cup A']\}, \\ 3)\ case\ v \in F_t: \\ - \quad \min_{A' \subseteq A_t(v)} \{C[t, S_t, Z_t, F_t \setminus \{A' \cup \{v\}\}, B_t \cup A']\}, \\ 4)\ case\ v \in B_t: \\ - \quad C[t', S_t, Z_t, F_t, B_t \setminus \{v\}] \end{cases}.$$

Recall that $A_t(v)$ is the set of vertices in F_t that reach v in the graph induced by $V(G_t) \setminus S_t$, i.e., the set of vertices that can be released by v if it was blocked in $G_{t'}$. Also note that, for simplicity, we consider only consistent cases, thus in case 2 it holds that $N^+(v) \cap X_t \subseteq S_t$, in case 3 it holds that $N^+(v) \cap (Z_t \cup F_t) \neq \emptyset$, and in case 4 it holds that $N^+(v) \cap \{Z_t \cup F_t\} = \emptyset$.

Forget Node: Let X_t be a forget node with a child $X_{t'}$ such that $X_t = X_{t'} \setminus \{v\}$, for some $v \in X_{t'}$. The forget node selects the best scenario considering all the possibilities for the forgotten vertex, discarding cases that lead to non-feasible solutions. In this problem, unfeasible cases are identified when the forgotten vertex v of $X_{t'}$ was blocked and reached no other node in B_t. Hence:

- If $N^+(v) \cap B_{t'} \neq \emptyset$ then

$$C[t, S_t, Z_t, F_t, B_t] = \min \begin{cases} C[t', S_t \cup \{v\}, Z_t, F_t, B_t], \\ C[t', S_t, Z_t \cup \{v\}, F_t, B_t], \\ C[t', S_t, Z_t, F_t \cup \{v\}, B_t], \\ C[t', S_t, Z_t, F_t, B_t \cup \{v\}] \end{cases}.$$

- If $N^+(v) \cap B_{t'} = \emptyset$ then

$$C[t, S_t, Z_t, F_t, B_t] = \min \begin{cases} C[t', S_t \cup \{v\}, Z_t, F_t, B_t], \\ C[t', S_t, Z_t \cup \{v\}, F_t, B_t], \\ C[t', S_t, Z_t, F_t \cup \{v\}, B_t], \end{cases}.$$

Join Node: Let X_t be a join node with children X_{t_1} and X_{t_2}, such that $X_t = X_{t_1} = X_{t_2}$. For any optimal knot-free vertex deletion set S of G it holds that $V(G_t) \cap S = \{V(G_{t_1}) \cap S\} \cup \{V(G_{t_2}) \cap S\}$. Clearly, if $S_t \subseteq S$ then we can assume that $S_t = S_{t_1} = S_{t_2}$. In addition, $Z_t = Z_{t_1} = Z_{t_2}$ otherwise we will have an inconsistency. Also note that a vertex is released in G_t if it reaches a vertex (possibly the vertex itself) that is released either in G_{t_1} or G_{t_2}. Thus:

$$C[t, S_t, Z_t, F_t, B_t] = \min_{\forall F', F''} \{C[t_1, S_t, Z_t, F', B'] + C[t_2, S_t, Z_t, F'', B'']\} - |S_t|,$$
$$\text{where } A_t(F' \cup F'') = F_t.$$

Note that $A_t(F' \cup F'')$ is the set of vertices that either are released in G_{t_i} $(i \in \{1, 2\})$ or can be released in G_t by vertices of $F' \cup F''$, even if they are blocked in both G_{t_1} and

G_{t_2}; this can occur, for example, if a blocked vertex v reaches another blocked node w in G_{t_1}, and in G_{t_2} vertex w is released.

Now, in order to run the algorithm, one can visit the bags of $\mathcal{T}$ in a bottom-up fashion, performing the queries described for each type of node. Since the reachability between the vertices of a bag can be stored in a bottom-up manner on $\mathcal{T}$, one can fill each entry of the table in $2^{O(tw)}$ time, and as the table has size $2^{O(tw)} \times n$, the dynamic programming can be performed in time $2^{O(tw)} \times n$.

Regarding correctness, let S^* be a minimum knot-free vertex deletion set of a digraph G with a tree decomposition $\mathcal{T}$. Let $S_t^*, Z_t^*, F_t^*, B_t^*$ be a partition of the vertices of X_t into removed, sinks, released and blocked, with respect to G_t after the removal of S^*. Note that $S_t^* = X_t \cap S^*$.

Fact 1. *There is no vertex $w \in V(G_t) \setminus X_t$ such that w reaches a vertex $v \in B_t^*$ in $G[V(G_t) \setminus S_t]$ and $w \in S^*$. Otherwise, since every vertex in B_t^* will reach a sink that is not in G_t, by Observation 2 one can remove from S^* every vertex that reaches B_t^* in $G[V(G_t) \setminus S_t]$, obtaining a subset of S^* which is also a knot-free vertex deletion set, contradicting the fact that S is minimum.*

This fact implies that the paths considered to compute $A_t(v)/A_t(F' \cup F')$' can in fact be used to release blocked vertices. Similarly, Fact 2 also holds.

Fact 2. *Let $\widehat{S}$ be a set for which the minimum is attained in the definition of $C[t, S_t^*, Z_t^*, F_t^*, B_t^*]$. Then $\widehat{S} \cup (S^* \setminus V(G_t))$ is also a solution (which is minimum) for* KFVD. *Otherwise, from $\widehat{S} \cup (S^* \setminus V(G_t))$ we can also obtain a knot-free vertex deletion set smaller than S^*, which is a contradiction.*

Fact 2 implies that we have stored enough information. At this point, the correctness of the recursive formulas is straightforward.

Finally, to show a lower bound based on ETH, we can transform an instance F of 3-SAT into an instance G_F of KFVD using the polynomial reduction presented in [11, Theorem 4], obtaining in polynomial time a graph with $|V| = 2n + 2m$, and so $tw = O(n + m)$. Therefore, if KFVD can be solved in $2^{o(tw)}|V|^{O(1)}$ time, then we can solve 3-SAT in $2^{o(n+m)}(n + m)^{O(1)}$ time, i.e., ETH fails. ◀

5 The size of a minimum directed feedback vertex set as parameter

Recall that k-KFVD is $W[1]$-hard (for fixed K-width and longest directed path) and that, as noticed in Observation 1, we can assume $k < dfv(G)$. This motivates us to determining the status of dfv-KFVD. In this section, we present two FPT-algorithms. Both with the size of a minimum directed feedback vertex set as parameter but with an aggregate parameter, the K-width, $\kappa(G)$, for the first one and the length of a longest directed path, $p(G)$, for the second one. Since finding a minimum directed feedback vertex set F in G can be solved in FPT-time (with respect to dfv) [14], we consider that F, a minimum DFVS, is given. Namely, we show that both (dfv, κ)-KFVD and (dfv, p)-KFVD are FPT.

At this point, we need to define the following variant of KFVD.

Disjoint Knot-Free Vertex Deletion (Disjoint-KFVD)
Instance: A directed graph $G = (V, E)$; a subset $X \subseteq V$; and a positive integer k.
Question: Determine if G has a set $S \subset V(G)$ such that $\|S\| \leq k$, $S \cap X = \emptyset$ and $G[V \setminus S]$ is knot-free.

We call *forbidden vertices* the vertices of the set X. It is clear that Disjoint-KFVD generalizes KFVD by taking $X = \emptyset$.

Let us now define two more steps that are FPT parameterized by dfv and that will be used for both (dfv, κ)-KFVD and (dfv, p)-KFVD. The next step will allow us to consider that the vertices of F are forbidden. We need the following straightforward observation.

▶ **Observation 4.** *Let (G, k) be an instance of* KFVD *and $v \in V(G)$.*

- *if (G, k) is a* yes*-instance and there exists a solution S with $v \in S$, then $(G \setminus \{v\}, k-1)$ is a* yes*-instance*
- *if $(G \setminus \{v\}, k-1)$ is a* yes*-instance then (G, k) is a* yes*-instance*

▶ **Branching 1** (On the directed feedback vertex set F)**.** *Let (G, F, k) be an instance of dfv-*KFVD*. In time $3^{dfv} \times O(n)$ we can build 3^{dfv} instances (G^i, F_1^i, X^i, k^i) of dfv-*Disjoint-KFVD *as follows. We consider all possible partitions of F into three parts: F_1, the set of vertices of F that will not be removed (i.e., they become forbidden); F_2, the set of vertices in F that will be removed; and F_3, the set of vertices in F that will be turned into sinks. For each such a partition (indicated by the index i), we remove the set $Y^i = F_2^i \cup N^+(F_3^i)$ of vertices and we apply exhaustively Reduction Rules 1 and 2 (see Section 2). We denote by G^i the obtained graph, $X^i = F_1^i$, and $k^i = k - |Y|$.*

According to Observation 4, it is clear that (G, F, k) is a *yes*-instance of dfv-KFVD if and only if one of the instances (G^i, F_1^i, X^i, k^i), $1 \leq i \leq 3^{dfv}$, of dfv-Disjoint-KFVD is a *yes* instance. Since there are at most 3^{dfv} partitions of F, the branching reduction can be performed in FPT time. Although at this point $X^i = F_1^i$, in the next steps some vertices of $V(G) \setminus F_1$ may become forbidden and therefore should be added to X^i. Also, from this point forward, we assume that we are given an instance (G, F_1, X, k) of dfv-Disjoint-KFVD.

Notice that after applying Reduction Rule 1 (Section 2), each strongly connected component of G is at least of size two. Thus, each of them must contain at least one cycle; therefore, the number of strongly connected components of G is bounded by dfv. Moreover, for any strongly connected component U of G, Reduction Rule 2 gives an upper bound for the number of vertices in $N^+(V(U))$ (i.e., vertices that are not in U but it is out-neighbour of some vertex in U). This implies that G has at most $dfv \times k \leq dfv^2$ such vertices between its strongly connected components. This observation leads to a branching rule.

▶ **Branching 2** (On strongly connected components)**.** *Let S_H be the set of vertices that are extremities of arcs between the strongly connected components of G. We have $|S_H| \leq 2 \times dfv \times k \leq 2 \times dfv^2$ and we can branch in FPT-time trying all possible partitions of S_H into two sets: S_1, the set of vertices to be deleted in G such that $|S_1| \leq k$; and $S_2 = S_H \setminus S_1$, the set of vertices marked as forbidden, and then added into X.*

Notice that this step involves a $2^{|S_H|}$ branching. At this point, we may consider that we have an instance (G, F, X, k) where $F \subseteq X$ and such that for any arc uv between two SCC's U_i and U_j, $\{u, v\} \subseteq X$. We call such an instance as a *nice* instance.

▶ **Lemma 10** (After cleaning of Branching 2)**.** *If there is an algorithm running in time $g(dfv) \times poly(n)$ for dfv-*Disjoint-KFVD *restricted to nice instances that are strongly connected, then there is an FPT algorithm running in time $g(dfv) \times poly(n) \times c.n.log(dfv)$ (where c is a constant) to solve dfv-*Disjoint-KFVD *for any nice instance.*

Proof. Let (G, F, X, k) be a nice instance and S be a solution. Let $\mathcal{U} = \{U_1, \ldots, U_s\}$ be the partition of $V(G)$ where each U_i is an SCC, and let $\mathcal{K} = \{U_i : U_i \text{ is a knot}\}$. Without loss of generality we can assume that $\mathcal{K} = \{U_1, \ldots, U_t\}$ for some $t \leq s$. Let $S_i = S \cap U_i$. Notice that if S is a solution then for any $i \in [t]$, S_i is a solution of $(G[U_i], F \cap U_i, X \cap U_i, |S_i|)$. Moreover, for any solutions S_i' to $(G[U_i], F \cap U_i, X \cap U_i, |S_i'|)$ where $\sum_{i=1}^{t} |S_i'| \leq k$, $S' = \bigcup_{i=1}^{t} S_i'$ will be

a solution to (G, F, X, k) because vertices of some $U_j \notin \mathcal{K}$ will still have a path to a set $U_i \in \mathcal{K}$ in $G_{\bar{S}'}$ since any arc between two SCC's has forbidden endpoints. Thus, given a nice instance (G, F, X, k) and an algorithm A for a nice instance restricted to one SCC, for any $U_i \in \mathcal{K}$ we perform a binary search to find the smallest k_i such that $A(G[U_i], F \cap U_i, X \cap U_i, k_i)$ answers *yes*, and we answer *yes* iff $\sum_{i=1}^{t} k_i \leq k$. ◀

From Lemma 10, we may assume that we have an instance (G, F, X, k) such that $F \subseteq X$ and G is strongly connected (there is only one SCC). We call such an instance as a *super nice* instance.

5.1 Combining DFVS-number and K-width

In this section, we prove that (dfv, κ)-Disjoint-KFVD restricted to super nice instances is FPT.

Let $F = \{v_1, \ldots, v_{dfv}\}$. For every pair of integers i, j with $1 \leq i, j \leq dfv$ we define $H_{i,j}$ as the (i,j)-*connectivity set*, that is, the set of vertices which are contained in a directed path from v_i to v_j in the induced subgraph $G[V \setminus (F \setminus \{v_i, v_j\})]$ (if $i = j$ then $H_{i,i}$ is the set of vertices contained in a cycle in $G[V \setminus (F \setminus \{v_i\})]$). Let us define a set B on which we will later branch in a way to ensure connectivity among different connectivity sets. We start with $B = \{\emptyset\}$, and then, for each possible pair of connectivity sets $H_{i,j}$, $H_{i',j'}$ we increase B as follows:

- **(i)** add $N^+(H_{i,j} \setminus H_{i',j'}) \cap H_{i',j'}$ to B.
- **(ii)** add $N^+(H_{i,j} \cap H_{i',j'}) \cap (H_{i',j'} \setminus H_{i,j})$ to B.
- **(iii)** add $N^+(H_{i',j'} \setminus H_{i,j}) \cap H_{i,j}$ to B.
- **(iv)** add $N^+(H_{i',j'} \cap H_{i,j}) \cap (H_{i,j} \setminus H_{i',j'})$ to B.

For a given pair of connectivity sets, in each of the items *i*), *ii*), *iii*) and *iv*) the number of added vertices to B is at most κ. For instance,let $y_1, \ldots, y_l$ be the vertices added by item *i*), where each $y_s \in N^+(H_{i,j} \setminus H_{i',j'}) \cap H_{i',j'}$. By definition, there exist vertices $x_1, \ldots, x_l$ of $H_{i,j} \setminus H_{i',j'}$ such that $x_s y_s$ are arcs of G for $s = 1, \ldots, l$. Notice that while the y_s's are distinct, the x_s's are not forced to be so. For any $s \in \{1, \ldots, l\}$, there exists a path P_s in $H_{i',j'}$ from y_s to $v_{j'}$, and such a path does not intersect $H_{i,j} \setminus H_{i',j'}$. In the same way, by finding a path Q_s from v_i to x_s for every $s \in \{1, \ldots, l\}$, we form l distinct paths $Q_s P_s$ from v_i to $v_{j'}$, implying $l \leq \kappa$, the K-width of G. So, as there are dfv^2 different connectivity sets, at the end of the process B contains at most $\kappa \times dfv^4$ vertices. Figure 2 shows examples of vertices to be added in B regarding the interaction of two different connectivity sets.

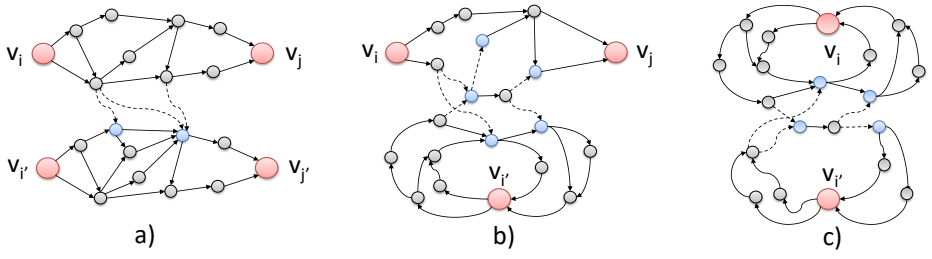

Figure 2 a) two connectivity sets with no intersection. b) an intersection with two vertices belonging to both connectivity sets. c) two connectivity sets $H_{i,j}$ with $i = j$. Vertices to be added in B are marked in blue.

Next we establish our last branching rule.

▶ **Branching 3** (On the connectivity sets). *We branch by partitioning B into three parts: B_1, the set of vertices that will not be removed (ie. they become forbidden); B_2, the set of vertices that will be removed; and B_3, the set of vertices that will become sinks. Since $|B| \leq \kappa \times dfv^4$, we branch at most $3^{\kappa.dfv^4}$ times.*

At this point, without loss of generality, one can assume that none of the above branching and reductions rules are applicable. Hence, the analysis boils down to the case where $F \cup B \subseteq X$, meaning that all the vertices of $F \cup B$ are forbidden to be deleted or become sinks, and G is strongly connected.

▶ **Observation 5** (The consequences of Branching 3). *Let G be a graph for which no Reduction Rules 1 and 2 or Branching Rules 1 to 3 can be applied. Let $H_{i,j}$ and $H_{i',j'}$ be two different connectivity arc sets in G. If there is an arc from $H_{i,j} \setminus H_{i',j'}$ to $H_{i',j'} \setminus H_{i,j}$ or $H_{i,j} \cap H_{i',j'}$ to $H_{i',j'} \setminus H_{i,j}$ in $G[H_{i,j} \cup H_{i',j'}]$, then the head vertex of such an arc is a forbidden vertex.*

We now aim to show that, for any vertex v^* such that v^* can be turned into a sink, that is, $N^+(v^*) \cap X = \emptyset$ and $d^+(v^*) \leq k$, the deletion of $N^+(v^*)$ is sufficient for G to become knot-free.

▶ **Lemma 11.** *Let (G, F, X, k) be an instance of (dfv, κ)-Disjoint-KFVD such that G is strongly connected and none of the branching and reduction rules can be applied. If there is a vertex v^* with no forbidden out-neighbors, then $G[V \setminus N^+(v^*)]$ is knot-free.*

Proof. Let (G, F, k, X) and v^* as stated. Denote by G' the resulting graph, i.e, $G' = G[V \setminus N^+(v^*)]$. For contradiction, assume that G' contains a knot K. As G is strongly connected, K was not a knot in G, implying that there exists an arc xy of G such that $x \in V(K)$ and $y \in N^+(v^*)$. Notice that $v^* \notin F$ since vertices from F cannot become sinks and $y \notin X$, since y has to be deleted in order to v^* to become a sink. Let us now define a connectivity set containing both y and v^*. Let s be any source of the DAG $G[V \setminus F]$ such that there is a sv^* path in $G[V \setminus F]$, and let z be any sink of $G[V \setminus F]$ such that there is a yz path in $G[V \setminus F]$. As G is strongly connected, there exist arcs $v_i s$ and zv_j where $\{v_i, v_j\} \subseteq F$ and we get that $\{v^*, y\} \subseteq H_{i,j}$. Notice that $i = j$ is possible. Similarly, as $G[V(K)]$ is strongly connected, it contains a cycle C' containing x and thus there exists a connectivity set $H_{k,l}$ containing a path P from v_k to v_l which is a subpath of $G[V(K)]$ containing x, and with $\{v_k, v_l\} \subseteq V(K)$. Notice first that $v^*, y \notin F$. In addition, v^* is not a vertex of $H_{k,l}$, otherwise there would exist a path P' from v_k to v^* containing no vertex of $F \setminus \{v_k\}$, which is not possible. Indeed, either $V(P') \cap N^+(v^*) = \emptyset$ and we would get that K is not a knot, or $V(P') \cap N^+(v^*) \neq \emptyset$, implying that there is a cycle with no vertex of F. Thus, as y was not a forbidden vertex, it means that $y \notin H_{k,l}$ otherwise the arc v^*y would go from $H_{i,j} \setminus H_{k,l}$ to $H_{i,j} \cap H_{k,l}$ and y should be forbidden by Branching 3 item i). Then we have $y \in H_{i,j} \setminus H_{k,l}$. Similarly, we have $x \notin H_{i,j} \cap H_{k,l}$, otherwise by item ii) of Branching 3, vertex y would be forbidden. Finally $x \in H_{k,l} \setminus H_{i,j}$ and $y \in H_{i,j} \setminus H_{k,l}$, since $(H_{i,j} \setminus H_{k,l}) \subseteq H_{i,j}$, and by item iii) of Branching 3, vertex y would again be a forbidden vertex, a contradiction. ◀

In conclusion, by Lemma 11, we can find in polynomial time the optimum solution for G: we choose a vertex v^* with minimum out-degree.

▶ **Theorem 12.** Knot-free Vertex Deletion *can be solved in $2^{O(\kappa dfv^5)} \times n^{O(1)}$.*

Proof. Let us now compute the running time of the overall algorithm. First notice that applying Branchings 1 and 2 results in $3^{dfv} \times 2^{2dfv^2}$ branches. Branching 3 can be done

in time $3^{\kappa.dfv^4}$, but may re-create several SCC's, forcing us to apply again Branching 2 and reduction rules again, but decreasing k. This implies that the total running time is $3^{dfv} \times (2^{2dfv^2}3^{\kappa.dfv^4})^k \times n^{O(1)}$, thus the result holds. ◀

5.2 Combining DFVS-number and length of a longest directed path

In this subsection we investigate the length of a longest path and $dfv(G)$ as aggregate parameters.

▶ **Lemma 13.** *(dfv, p)-Disjoint-KFVD on super nice instances can be solved in $2^{O(dfv^3)}p^{O(dfv)} \times n^{O(1)}$.*

Proof. Let (G, F, X, k) be a super nice instance. Recall that the directed feedback vertex set F is a set of forbidden vertices ($F \subseteq X$) and G is strongly connected. The proof is by induction on $|F|$. If $|F| = 1$, then, for any vertex v of $V(G) \setminus F$ that can be turned into a sink, $N^+(v)$ will be a solution set for G. Therefore, the optimum solution can be found in polynomial time. Assume now that $|F| \geq 2$ and denote F by $\{v_1, \ldots, v_{dfv}\}$. As G is strongly connected, there exists a path P_1 of length at most p from v_1 to v_2 and a path P_2 of length at most p from v_2 to v_1. Denote by C the digraph $G[V(P_1) \cup V(P_2)]$; it is strongly connected, contains v_1 and v_2 and at most $2p$ vertices. Since the number of vertices in C is bounded, we may branch $2p + 1$ times by trying to guess a vertex that will be deleted in C. Each time a vertex of C will be guessed as deleted, the parameter k will decrease by one. So, k will decrease in all branches, except in the one where we guess that no vertex is deleted, and then where all the vertices of C are forbidden. In this case, C is a strongly connected component whose vertices are all forbidden and containing at least two vertices of F. So, we contract C to obtain a new instance G'. Formally, we remove $V(C)$ from G, add a new vertex v_C, and for all vertices of $G \setminus C$ having at least one in-neighbor (resp. out-neighbor) in C, we add an arc from v_C (resp. to v_C) to this vertex. Let F' be the set $\{v_C\} \cup F \setminus V(C)$ and notice that F' is a directed feedback vertex set of G' and that $|F'| < |F|$. Similarly, let X' be the set $(X \setminus V(C)) \cup \{v_C\}$. We claim that both instances (G, F, k, X) and (G', F', k, X') are equivalent. Indeed, it suffices to notice that as $V(C)$ contains only forbidden vertices in G and that v_C is forbidden in G', then any solution to the KFVD problem for G is a solution of G', and conversely. Then, we apply Branchings 1 and 2 to obtain a super nice instance equivalent to (G', F', k, X'), and we can apply the induction hypothesis.
So at each branching, either the parameter k decreases by at least one or the size of F decreases by at least one. As both values are bounded above by dfv, we branch consecutively at most $2dfv$ times. And since Branching rules 1 and 2 create at most $3^{dfv} \times 2^{2dfv^2}$ branches, and branching on cycle C creates $2p + 1$ branches, the total number of branches is $(3^{dfv} \times 2^{2dfv^2} \times (2p + 1))^{2dfv} = 2^{O(dfv^3)}p^{O(dfv)}$, and we get the desired running time. ◀

Given that we can obtain a super nice instance in $2^{O(dfv^3)} \times n^{O(1)}$, it holds that Knot-free Vertex Deletion can be solved in time $2^{O(dfv^3)}p^{O(dfv)} \times n^{O(1)}$.

References

1 Saeed Akhoondian Amiri, Lukasz Kaiser, Stephan Kreutzer, Roman Rabinovich, and Sebastian Siebertz. Graph searching games and width measures for directed graphs. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

2 János Barát. Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22(2):161–172, 2006.

3 Valmir C. Barbosa. The Combinatorics of Resource Sharing. In *Models for Parallel and Distributed Computation*, pages 27–52. Springer, 2002.

4 Valmir C. Barbosa and Mario R. F. Benevides. A graph-theoretic characterization of AND-OR deadlocks. Technical Report COPPE-ES-472/98, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, 1998.

5 Valmir C. Barbosa, Alan D. A. Carneiro, Fábio Protti, and Uéverton S. Souza. Deadlock Models in Distributed Computation: Foundations, Design, and Computational Complexity. In *Proceedings of the 31st ACM/SIGAPP Symposium on Applied Computing*, pages 538–541, 2016.

6 Dietmar Berwanger, Anuj Dawar, Paul Hunter, Stephan Kreutzer, and Jan Obdržálek. The dag-width of directed graphs. *Journal of Combinatorial Theory, Series B*, 102(4):900–923, 2012.

7 Dietmar Berwanger and Erich Grädel. Entanglement – A Measure for the Complexity of Directed Graphs with Applications to Logic and Games. In Franz Baader and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 209–223, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

8 Hans L Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-Approximation Algorithm for Treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.

9 John A. Bondy and Uppaluri S. R. Murty. *Graph theory with applications*, volume 290. Macmilan, 1976.

10 Alan D. A. Carneiro, Fábio Protti, and Uéverton S. Souza. Deletion Graph Problems Based on Deadlock Resolution. In *The 23rd International Computing and Combinatorics Conference, COCOON 2017, Hong Kong, China, August 3-5, 2017. Lecture Notes in Computer Science*, volume 10392, pages 75–86. Springer, 2017.

11 Alan D. A. Carneiro, Fábio Protti, and Uéverton S. Souza. Fine-Grained Parameterized Complexity Analysis of Knot-Free Vertex Deletion – A Deadlock Resolution Graph Problem. In *The 24th International Computing and Combinatorics Conference, COCOON 2018, Qingdao, China , July 2-4, 2018. Lecture Notes in Computer Science*, volume 10976, pages 84–95. Springer, 2018.

12 Alan D. A. Carneiro, Fábio Protti, and Uéverton S Souza. Deadlock resolution in wait-for graphs by vertex/arc deletion. *Journal of Combinatorial Optimization*, 37(2):546–562, 2019.

13 K Mani Chandy and Leslie Lamport. Distributed snapshots: determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3:63–75, 1985.

14 Jianer Chen, Yang Liu, Songjian Lu, Barry O'sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM (JACM)*, 55(5):21, 2008.

15 Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.

16 Bruno Courcelle, Johann A Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

17 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

18 Robert Ganian, Petr Hliněný, Joachim Kneis, Alexander Langer, Jan Obdržálek, and Peter Rossmanith. Digraph width measures in parameterized algorithmics. *Discrete applied mathematics*, 168:88–107, 2014.

19 Robert Ganian, Petr Hliněný, Joachim Kneis, Daniel Meister, Jan Obdržálek, Peter Rossmanith, and Somnath Sikdar. Are there any good digraph width measures? In *International Symposium on Parameterized and Exact Computation*, pages 135–146. Springer, 2010.

20 Hermann Gruber. Digraph Complexity Measures and Applications in Formal Language Theory. *Discrete Mathematics & Theoretical Computer Science*, 14(2):189–204, 2012.

21 Richard C Holt. Some deadlock properties of computer systems. *ACM Computing Surveys (CSUR)*, 4(3):179–196, 1972.

22 Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. *Theoretical Computer Science*, 399(3):206–219, 2008. Graph Searching.

23 Thor Johnson, Neil Robertson, P.D. Seymour, and Robin Thomas. Directed Tree-Width. *Journal of Combinatorial Theory, Series B*, 82(1):138–154, 2001.

24 RichardM. Karp. Reducibility among Combinatorial Problems. In RaymondE. Miller, JamesW. Thatcher, and JeanD. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972.

25 Mateus de O. Oliveira. Subgraphs satisfying MSO properties on z-topologically orderable digraphs. In *International Symposium on Parameterized and Exact Computation*, pages 123–136. Springer, 2013.

26 Mateus de O. Oliveira. An algorithmic metatheorem for directed treewidth. *Discrete Applied Mathematics*, 204:49–76, 2016.

27 Sang-Il Oum. Approximating Rank-width and Clique-width Quickly. *ACM Transactions on Algorithms*, 5(1):10:1–10:20, 2008.

28 Roman Rabinovich and Lehr-und Forschungsgebiet. *Complexity measures of directed graphs*. PhD thesis, RWTH Aachen University, 2008.

29 Mohammad Ali Safari. D-Width: A More Natural Measure for Directed Tree Width. In Joanna Jędrzejowicz and Andrzej Szepietowski, editors, *Mathematical Foundations of Computer Science 2005*, pages 745–756, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.