A Quartic Kernel for Pathwidth-One Vertex Deletion

Geevarghese Philip[1], Venkatesh Raman[1], and Yngve Villanger[2]

[1] The Institute of Mathematical Sciences, Chennai, India
{gphilip,vraman}@imsc.res.in
[2] University of Bergen, N-5020 Bergen, Norway
yngve.villanger@uib.no

Abstract. The pathwidth of a graph is a measure of how path-like the graph is. Given a graph G and an integer k, the problem of finding whether there exist at most k vertices in G whose deletion results in a graph of pathwidth at most one is NP-complete. We initiate the study of the parameterized complexity of this problem, parameterized by k. We show that the problem has a quartic vertex-kernel: We show that, given an input instance $(G = (V, E), k); |V| = n$, we can construct, in polynomial time, an instance (G', k') such that (i) (G, k) is a YES instance if and only if (G', k') is a YES instance, (ii) G' has $\mathcal{O}(k^4)$ vertices, and (iii) $k' \leq k$. We also give a fixed parameter tractable (FPT) algorithm for the problem that runs in $\mathcal{O}(7^k k \cdot n^2)$ time.

1 Introduction

The treewidth of a graph is a measure of how "tree-like" the graph is. The notion of treewidth was introduced by Robertson and Seymour in their seminal Graph Minors series [35]. It has turned out to be very important and useful, both in the theoretical study of the properties of graphs [6,27] and in designing graph algorithms [7,9]. A graph has treewidth at most one if and only if it is a forest (a collection of trees), and a set of vertices in a graph G whose removal from G results in a forest is called a *feedback vertex set* (FVS) of the graph.

Given a graph G and an integer k as input, the Feedback Vertex Set problem asks whether G has an FVS of size at most k. This is one of the first problems that Karp showed to be NP-complete [25]. The problem and its variants have extensively been investigated from the point of view of various algorithmic paradigms, including approximation and parameterized algorithms. The problem is known to have a 2-factor approximation algorithm [4], and the problem parameterized by the solution size k is fixed parameter tractable (FPT) and has a polynomial kernel[1].

The quest for fast FPT algorithms and small kernels for the parameterized Feedback Vertex Set problem presents an illuminative case study of the evolution of the field of fixed parameter tractability, and stands out among the

[1] See Section 2 for the terminology and notation used in this paper.

D.M. Thilikos (Ed.): WG 2010, LNCS 6410, pp. 196–207, 2010.
© Springer-Verlag Berlin Heidelberg 2010

many success stories of this algorithmic approach towards solving hard problems. The first FPT algorithm for the problem, with a running time of $\mathcal{O}^*(k^4!)$, was developed by Bodlaender [5] and by Downey and Fellows [18]. After a series of improvements [19,24,33], a running time of the form $\mathcal{O}^*(c^k)$ was first obtained by Guo et.al [22], whose algorithm ran in $\mathcal{O}^*(37.7^k)$ time. This was improved by Dehne et.al [15] to $\mathcal{O}^*(10.6^k)$ in 2007, and to the current best $\mathcal{O}^*(3.83^k)$ by Cao et.al [13] in 2010. For classes of graphs that exclude a fixed minor H (for example, planar graphs), Dorn et.al [17] have recently obtained an FPT algorithm for the problem with a running time of the form $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k})})$.

Proving polynomial bounds on the size of the kernel for different parameterized problems has been a significant practical aspect in the study of the parameterized complexity of NP-hard problems, and many positive results are known. See [23] for a survey of kernelization results. The existence of a polynomial kernel for the Feedback Vertex Set problem was open for a long time. It was settled in the affirmative by Burrage et. al [12] as recently as 2006, when they exhibited a kernel with $\mathcal{O}(k^{11})$ vertices. This was soon improved to a cubic vertex-kernel ($\mathcal{O}(k^3)$ vertices) by Bodlaender [8,10]. The current smallest kernel, on $\mathcal{O}(k^2)$ vertices, is due to Thomassé [36].

The *pathwidth* of a graph is a notion closely related to treewidth, and was also introduced by Robertson and Seymour in the Graph Minors series [34]. The pathwidth of a graph denotes how "path-like" it is. A graph has pathwidth at most one if and only if it is a collection of *caterpillars*, where a caterpillar is a special kind of tree: it is a tree that becomes a path (called the *spine* of the caterpillar) when all its pendant vertices are removed. Graphs of pathwidth at most one are thus a very special kind of forests, and have even less structure than forests (which are themselves very "simple" graphs). As a consequence, some problems that are NP-hard even on forests can be solved in polynomial time on graphs of pathwidth at most one. Examples include (Weighted) Bandwidth [3,31,28], the Proper Interval Colored Graph problem, and the Proper Colored Layout problem [1].

In contrast to the case of forests, the corresponding vertex deletion problem for obtaining a collection of caterpillars (equivalently, a graph of pathwidth at most one) has not received much attention in the literature. In fact, to the best of our knowledge, the following problem has not yet been investigated at all: Given a graph G and an integer k as input, find whether G contains a set of at most k vertices whose removal from G results in a graph of pathwidth at most one. We call such a set of vertices a pathwidth-one deletion set (PODS), and the problem the Pathwidth-One Vertex Deletion problem. It follows from a general NP-hardness result of Lewis and Yannakakis that this problem is NP-complete.

Our results. We study the parameterized complexity of the Pathwidth-One Vertex Deletion problem parameterized by the solution size k, and show that (i) the problem has a vertex-kernel of size $\mathcal{O}(k^4)$, and (ii) the problem can be solved in $\mathcal{O}^*(7^k)$ time (Compare with the values $\mathcal{O}(k^2)$ and $\mathcal{O}^*(3.83^k)$ for FVS, respectively).

Note that, in general, a PODS "does more" than an FVS: It "kills" all cycles in the graph, like an FVS, and, in addition, it kills all non-caterpillar trees in the graph. In fact, the difference in the sizes of a smallest FVS and a smallest PODS of a graph can be arbitrarily large. For example, the treewidth of a binary tree is one, while for any integer c there exists a binary tree T_c of pathwidth at least $c+1$. Removing a single vertex from a graph will reduce the pathwidth by at most one, and so for T_c, the difference between the two numbers is at least c. Partly as a consequence of such differences, many of the techniques and reduction rules that have been developed for obtaining FPT algorithms and kernels for the Feedback Vertex Set problem do *not* carry over to the Pathwidth-One Vertex Deletion problem. Instead, we use a characterization of graphs of pathwidth at most one to obtain the FPT algorithm and the polynomial kernel.

Update. After this paper was presented at WG 2010, Cygan et. al [14] improved both the results in the paper. Using the same general idea of our FPT algorithm and a clever branching strategy, they obtained an $\mathcal{O}^*(4.65^k)$ FPT algorithm for the problem. Using some of our reduction rules and a different approach based on the α-expansion Lemma of Thomassé [36], they obtained a quadratic ($\mathcal{O}(k^2)$) kernel as well.

Organization of the rest of the paper. In Section 2 we give an overview of the notation and terminology used in the rest of the paper. In Section 3 we formally define the Pathwidth-One Vertex Deletion problem, show that the problem is NP-complete, and sketch an FPT algorithm for the problem that runs in $\mathcal{O}^*(7^k)$ time. We show in Section 4 that the problem has a vertex-kernel of size $\mathcal{O}(k^4)$. We conclude in Section 5. Due to space constraints, many proofs have been deferred to a full version [32] of the paper.

2 Preliminaries

In this section we state some definitions related to graph theory and parameterized complexity, and give an overview of the notation used in this paper; we also formally define the Pathwidth-One Vertex Deletion problem and show that it is NP-hard. In general we follow the graph terminology of [16]. For a vertex $v \in V$ in a graph $G = (V, E)$, we call the set $N(v) = \{u \in V | (u, v) \in E\}$ the *open neighborhood* of v. The elements of $N(v)$ are said to be the *neighbors* of v, and $N[v] = N(v) \cup \{v\}$ is called the *closed neighborhood* of v. For a set of vertices $X \subseteq V$, the open and closed neighborhoods of X are defined, respectively, as $N(X) = \bigcup_{u \in X} N(u) \setminus X$ and $N[X] = N(X) \cup X$. For vertices u, v in G, u is said to be a *pendant* vertex of v if $N(u) = \{v\}$. A *caterpillar* is a tree that becomes a path (called the *spine* of the caterpillar) when all its pendant vertices are removed. A nontrivial caterpillar is one that contains at least two vertices. A T_2 is the graph on seven vertices shown in Figure 1. The *center* of a T_2 is the one vertex of degree 3, and its *leaves* are the three vertices of degree 1.

The operation of *contracting* an edge (u, v) consists of deleting vertex u, renaming vertex v to uv, and adding a new edge (x, uv) for each edge $(x, u); x \neq v$.

Multiple edges that may possibly result from this operation are preserved. Note that the operation is symmetric with respect to u and v. A graph H is said to be a *minor* of a graph G if a graph isomorphic to H can be obtained by contracting zero or more edges of some subgraph of G.

A *graph property* is a subset of the set of all graphs. Graph property Π is said to hold for graph G if $G \in \Pi$. Π is said to be nontrivial if Π and its complement are both infinite. Π is said to be *hereditary* if Π holds for every induced subgraph of graph G whenever it holds for G. The *membership testing* problem for Π is to test whether Π holds for a given input graph.

A *tree decomposition* of a graph $G = (V, E)$ is a pair (T, χ) in which $T = (V_T, E_T)$ is a tree and $\chi = \{\chi_i \mid i \in V_T\}$ is a family of subsets of V, called *bags*, such that

(i) $\bigcup_{i \in V_T} \chi_i = V$;
(ii) for each edge $(u, v) \in E$ there exists an $i \in V_T$ such that both u and v belong to χ_i; and
(iii) for all $v \in V$, the set of nodes $\{i \in V_T \mid v \in \chi_i\}$ induces a connected subgraph of T.

The maximum of $|\chi_i| - 1$, over all $i \in V_T$, is called the *width* of the tree decomposition. The *treewidth* of a graph G is the minimum width taken over all tree decompositions of G. A *path decomposition* of a graph $G = (V, E)$ is a tree decomposition of G where the underlying tree T is a path. The *pathwidth* of G is the minimum width over all possible path decompositions of G.

To describe the running times of algorithms we sometimes use the $\mathcal{O}^*$ notation. Given $f : \mathbb{N} \to \mathbb{N}$, we define $\mathcal{O}^*(f(n))$ to be $\mathcal{O}(f(n) \cdot p(n))$, where $p(\cdot)$ is some polynomial function. That is, the $\mathcal{O}^*$ notation suppresses polynomial factors in the expression for the running time.

A parameterized problem Π is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. An instance of a parameterized problem is a tuple (x, k), where k is called the parameter. A central notion in parameterized complexity is *fixed-parameter tractability (FPT)* which means, for a given instance (x, k), decidability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial. The notion of *kernelization* is formally defined as follows.

Definition 1. [Kernelization, Kernel] [21,30] *A kernelization algorithm for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs, in time polynomial in $|x| + k$, a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$ and (b) $|x'|, k' \leq g(k)$, where g is some computable function. The output instance x' is called the kernel, and the function g is referred to as the size of the kernel. If $g(k) = k^{\mathcal{O}(1)}$ then we say that Π admits a polynomial kernel.*

When a kernelization algorithm outputs a graph on $h(k)$ vertices, we sometimes say that the output is an $h(k)$ vertex-kernel.

3 The Pathwidth-One Vertex Deletion Problem

In this section we formally define the Pathwidth-One Vertex Deletion problem, show that it is NP-complete, and briefly sketch an $\mathcal{O}^*(7^k)$ FPT algorithm for the problem. We begin with the observation that caterpillars are the quintessential graphs of pathwidth at most one:

Fact 1. [2] *A graph G has pathwidth at most one if and only if it is a collection of vertex-disjoint caterpillars.*

A vertex set $S \subseteq V$ of a graph G is said to be a *pathwidth-one deletion set* (PODS) if $G[V \setminus S]$ has pathwidth at most one. In this paper we investigate the parameterized complexity of the following problem:

Pathwidth-One Vertex Deletion (POVD)

Input:	An undirected graph $G = (V, E)$, and a positive integer k.
Parameter:	k
Question:	Does there exist a set $S \subseteq V$ of at most k vertices of G such that $G[V \setminus S]$ has pathwidth at most one (i.e., S is a PODS of G)?

The following general NP-completeness result is due to Lewis and Yannakakis:

Fact 2. [29] *The following problem is* NP*-complete for any nontrivial hereditary graph property Π for which the membership testing problem can be solved in polynomial time:*

Input:	*Graph $G = (V, E)$, positive integer k.*
Question:	*Is there a subset $S \subseteq V, \lvert S\rvert \leq k$ such that $G[V \setminus S] \in \Pi$?*

The NP-completeness of the Pathwidth-One Vertex Deletion problem follows directly from this result:

Theorem 1. [$\star$][2] *The* Pathwidth-One Vertex Deletion *problem is* NP*-complete.*

In the rest of the paper we focus on the parameterized complexity of the Pathwidth-One Vertex Deletion problem. We now sketch an $\mathcal{O}^*(7^k)$ time FPT algorithm, and in the next section we describe an $\mathcal{O}(k^4)$ vertex-kernel for the problem. Let $(G = (V, E), k)$ be the input instance, where $|V| = n$. Let $S \subseteq V$ be a PODS of G of size at most k. Observe that if (G, k) is a YES instance, then the number of edges in G is at most $k(n-1)+(n-1) = (k+1)(n-1)$. The first term on the left is an upper bound on the number of edges that are incident the vertices in S; the second term is a loose upper bound on the number of edges in $G \setminus S$. So, if G has more than $(k+1)(n-1)$ edges, then we can immediately

[2] Due to space constraints, proofs of results labeled with a [$\star$] have been moved to a full version [32] of the paper.

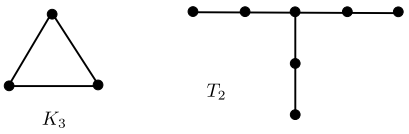

Fig. 1. The set of excluded minors for graphs of pathwidth at most one

reject the input. Since each reduction rule in the sequel is sound, and no rule increases the number of vertices or edges, from now on we assume, without loss of generality, that the graph has at most $(k+1)(n-1)$ edges.

The kernel arguments are based on Fact 1, while our starting point for the FPT algorithm is the following characterization, in terms of excluded minors, of graphs of pathwidth at most one:

Fact 3. [11,20] *A graph G has pathwidth at most one if and only if it does not contain K_3 or T_2 as a minor, where K_3 and T_2 are as in Figure 1.*

Fact 3 is not very helpful in the given form in checking for a small PODS. Instead, we derive and use the following alternate characterization and the two succeeding lemmas:

Lemma 1. [$\star$] *A graph G has pathwidth at most one if and only if it does not contain a cycle or a T_2 as a subgraph.*

Lemma 2. [$\star$] *Let $\mathcal{S} = \{T_2, K_3, C_4\}$, where C_4 is a cycle of length 4. Given a graph $G = (V, E); |V| = n$, we can find whether G contains a subgraph H that is isomorphic to one of the graphs in $\mathcal{S}$, and also locate such an H if it exists, in $\mathcal{O}(kn^2)$ time.*

Lemma 3. [$\star$] *Let $\mathcal{S} = \{T_2, K_3, C_4\}$, where C_4 is a cycle of length 4. If G is a graph that does not contain any element of $\mathcal{S}$ as a subgraph, then each connected component of G is either a tree, or a cycle with zero or more pendant vertices ("hairs") attached to it.*

3.1 An FPT Algorithm for POVD

Let $(G = (V, E), k)$ be the input instance, where $|V| = n$. We use a branching strategy inspired by Lemmas 1 and 3. First we locate a (not necessarily induced) subgraph T of G that is isomorphic to one of $\mathcal{S} = \{T_2, K_3, C_4\}$. From Lemma 2, this can be done in $\mathcal{O}(kn^2)$ time. At least one of the (at most seven) vertices of T must be in any PODS of G. So we branch on the vertices of T: We pick each one, in turn, into the minimal PODS that we are constructing, delete the picked vertex and all its adjacent edges, and recurse on the remaining graph after decrementing the parameter by one.

The leaves of this recursion tree correspond to graphs which do not have a subgraph isomorphic to any graph in $\mathcal{S}$. By Lemma 3, each connected component of

such a graph is a tree, or a cycle with zero or more pendant vertices ("hairs") attached to it. The trees can be ignored — they do not have a T_2 as a subgraph — and each cycle (with or without hairs) forces exactly one vertex into any minimal solution. Thus the base case of the recursion can be solved in linear time.

This is a 7-way branching, where the depth of the recursion is at most k, and where the algorithm spends $\mathcal{O}(kn^2)$ time at each node. Hence we have

Theorem 2. *The* Pathwidth-One Vertex Deletion *problem parameterized by the solution size k has an FPT algorithm that runs in $\mathcal{O}(n^2 \cdot 7^k k)$ time.*

By a folklore result of parameterized complexity, it follows immediately from Theorem 2 that the Pathwidth-One Vertex Deletion problem parameterized by the solution size k has a kernel of size $\mathcal{O}(7^k)$ (See, for example, [21]). We now show that the kernel size can be brought down significantly from this trivial bound.

4 A Polynomial Kernel for POVD

We turn to the main result of this paper. We describe a polynomial-time algorithm (the *kernelization algorithm*) that, given an instance (G, k) of POVD, returns an instance (G', k') (the *kernel*) of POVD such that (i) (G, k) is a YES instance if and only if (G', k') is a YES instance, (ii) G' has $\mathcal{O}(k^4)$ vertices, and (iii) $k' \leq k$. The kernelization algorithm (Algorithm 1) exhaustively applies the reduction rules of Section 4.1 to the input instance. The resulting instance, to which no rule applies, is said to be *reduced* with respect to the reduction rules. To demonstrate a quartic vertex-kernel for the problem, it suffices to show that

1. The rules can be exhaustively applied in polynomial time;
2. Each rule is *sound*: the output of a rule is a YES instance if and only if its input is a YES instance; and
3. If the input instance (G, k) is a YES instance, then the reduced instance (G', k') has $\mathcal{O}(k^4)$ vertices.

The reduction rules are based on the following idea: Suppose $(G = (V, E), k)$ is a YES instance of the problem that is reduced with respect to the reduction rules. Then there is a set $S \subseteq V, |S| \leq k$ such that $G[V \setminus S]$ is a collection of caterpillars, and it suffices to show that $|V \setminus S| = \mathcal{O}(k^4)$. We express $V \setminus S$ as the union of different kinds of vertices, and devise reduction rules that help us bound the total number of vertices of each kind. To be more specific, we set $V \setminus S = V_1 \cup V_2 \cup V_3 \cup V_4 \cup V_5$ where

1. $V_1 = \{v \in (V \setminus S); N(v) \cap (V \setminus S) = \emptyset$ and $|N(v) \cap S| \leq 1\}$
2. $V_2 = \{v \in (V \setminus S); N(v) \cap (V \setminus S) = \emptyset$ and $|N(v) \cap S| \geq 2\}$
3. $V_3 = \{v \in ((V \setminus S) \setminus V_1); v$ is on the spine of a nontrivial caterpillar in $G[V \setminus S]\}$
4. $V_4 = \{v \in (V \setminus S); |N(v) \cap S| = 0$ and v is a pendant vertex in $G[V \setminus S]\}$
5. $V_5 = \{v \in (V \setminus S); |N(v) \cap S| \geq 1$ and v is a pendant vertex in $G[V \setminus S]\}$

Algorithm 1. The kernelization algorithm

```
1: procedure KERNELIZE(G, k)
2:     CurrentInstance ← (G, k)
3:     repeat
4:         Apply Rules 1 to 6 and set CurrentInstance to be the output.
5:     until None of the rules cause any change to CurrentInstance.
6: end procedure
```

It is easy to verify that these sets together exhaust $V \setminus S$. We state the reduction rules and describe their consequences in the next section; the claims of soundness of the rules and a more formal bound on the running time and kernel size are deferred to Section 4.2.

4.1 Reduction Rules

For each rule below, let $(H = (V_H, E_H), k)$ be the instance on which the rule is applied, and (H', k') the resulting instance. Let $G = (V, E)$ be a YES instance of the problem that is reduced with respect to all the reduction rules, and let $S, V_1, \ldots, V_5$ be as described above. To bound the sizes of various subsets of $V \setminus S$, we use the fact that no reduction rule applies to G.

Rule 1. *If a connected component $H[X]; X \subseteq V_H$ of H has pathwidth at most 1, then remove X from H. The resulting instance is $(H' = H[V_H \setminus X], k' = k)$.*

Rule 2. *If a vertex u in H has two or more pendant neighbors, then delete all but one of these pendant neighbors to obtain H'. The resulting instance is $(H', k' = k)$.*

Rules 1 and 2 together ensure that every caterpillar in $G[V \setminus S]$ has at least one neighbor in S, and that $|V_1| \leq k$: See Lemma 5.

Rule 3. *Let u be a vertex of H with at least two neighbors. If for every two vertices $\{v, w\} \subseteq N(u)$ there exist $k + 2$ vertices excluding u that are adjacent to both v and w, then delete u from H. The resulting instance is $(H' = H[V_H \setminus \{u\}], k' = k)$.*

Rule 3 ensures that $|V_2| \leq \binom{k}{2}(k + 2)$: Set $A = V_2$ and $X = S$ in Lemma 6.

Rule 4. *For a vertex u of H, if there is a matching M of size $k + 3$ in H where (i) each edge in M has at least one end vertex in $N(u)$, and, (ii) u is not incident with any edge in M, then delete u and decrement k by one. The resulting instance is $(H' = H[V \setminus \{u\}], k' = k - 1)$.*

Rule 5. *Let x, y be the end vertices of the spine $x, v_1, v_2, v_3 \ldots, v_p, y$ of an induced caterpillar C in H such that (1) no $v_i; 1 \leq i \leq p$ is adjacent in H to any vertex outside C, and (2) every pendant vertex of C is a pendant vertex in H. If $p \geq 5$, then contract the edge (v_2, v_3) in H to obtain the graph H'. The resulting instance is $(H', k = k')$.*

From Rules 1 to 5 it follows that $|V_3| \leq 17k(k+2)$ (Lemma 7), and that $|V_5| \leq 17(k+2)^2k(2k-1)$ (Lemma 8). Each vertex in G can have at most one pendant neighbor, or else Rule 2 would apply. From this we get $|V_4| \leq |V_3| = 17k(k+2)$. Putting all the bounds together, $|V| \leq 34k^4+120k^3+103k^2+k$, and so we have:

Rule 6. *If none of the Rules 1 to 5 can be applied to the instance* (H, k)*, and* $|V_H| > 34k^4+120k^3+103k^2+k$ *, then set the resulting instance to be the trivial NO instance* (H', k') *where* H' *is a cycle of length* 3 *and* $k' = 0$*.*

In the next section we prove that these rules are sound, and that they can all be applied exhaustively in polynomial time. Hence we get

Theorem 3. *The* Pathwidth-One Vertex Deletion *problem parameterized by solution size* k *has a polynomial vertex-kernel on* $\mathcal{O}(k^4)$ *vertices.*

4.2 Correctness and Running time

We now show that the reduction rules are sound. That is, we show that for each rule, (using the notation of the previous section) (H, k) is a YES instance if and only if (H', k') is a YES instance. We also show that each rule can be implemented in polynomial time. For discussing the rules, we reuse the notation from the respective rule statement in Section 4.1. In each case, n is the number of vertices in the input to the kernelization algorithm.

Claim. [$\star$] Rule 1 is sound, and can be applied in $\mathcal{O}(kn)$ time.

Claim. [$\star$] Rule 2 is sound, and can be applied in $\mathcal{O}(kn)$ time.

Claim. [$\star$] Rule 3 is sound, and can be applied in $\mathcal{O}(n^3)$ time.

Claim. [$\star$] Rule 4 is sound, and can be applied in $\mathcal{O}(kn^{1.5})$ time.

Claim. [$\star$] Rule 5 is sound, and can be applied in $\mathcal{O}(kn)$ time.

From these claims, we get

Lemma 4. *On an input instance* $(G = (V, E), k); |V| = n$ *of* Pathwidth-One Vertex Deletion*, the kernelization algorithm (Algorithm 1) runs in* $\mathcal{O}(n^4)$ *time and outputs a kernel on* $\mathcal{O}(k^4)$ *vertices.*

Proof. From the above claims it follows that Rules 1 to 5 are sound, and that each can be applied in $\mathcal{O}(n^3)$ time. From the discussion in Section 4.1 (using Lemmas 5 to 8 below) it follows that Rule 6 is sound, and it is easy to see that this rule can be applied in $\mathcal{O}(n)$ time. Each time a rule is applied, the number of vertices in the graph reduces by at least one (contracting an edge also reduces the vertex count by one). Hence the loop in lines 3 to 5 of Algorithm 1 will run at most $|V| + 1 = n + 1$ times. The algorithm produces its output either at a step where Rule 6 applies, or when none of the rules applies and the remaining instance has $\mathcal{O}(k^4)$ vertices. Thus the algorithm runs in $\mathcal{O}(n^4)$ time and outputs a kernel on $\mathcal{O}(k^4)$ vertices. □

We now list the lemmas used in Section 4.1 to bound the sizes of $V_1, \ldots, V_5$.

Lemma 5. [$\star$] *Let $(G = (V, E), k)$ be a YES instance of the problem that is reduced with respect to Rules 1 and 2, and let S be a PODS of G of size at most k. Let $V_1 = \{v \in (V \setminus S); (N(v) \cap (V \setminus S)) = \emptyset$ and $|N(v) \cap S| \leq 1\}$. Then every caterpillar in $G[V \setminus S]$ has at least one neighbor in S, and $|V_1| \leq k$.*

Lemma 6. [$\star$] *Let (G, k) be a YES instance of the problem that is reduced with respect to Rule 3. For a set $X \subseteq V$, if $A \subseteq V \setminus X$ is such that every $v \in A$ has (i) at least two neighbors in X, and (ii) no neighbors outside X, then $|A| \leq \binom{|X|}{2}(k+2)$.*

Lemma 7. [$\star$] *Let $(G = (V, E), k)$ be an instance of the problem that is reduced with respect to Rules 1 to 5, and let $S \subseteq V$ be such that $G[V \setminus S]$ has pathwidth at most one. Let $X \subseteq (V \setminus S)$ be the set of vertices in $(V \setminus S)$ that lie on the spines of nontrivial caterpillars in $G[V \setminus S]$. Then $|X| \leq 17k(k+2)$.*

Lemma 8. [$\star$] *Let $(G = (V, E), k)$ be a YES instance of the problem that is reduced with respect to Rules 1 to 5, and let $S \subseteq V; |S| \leq k$ be such that $G[V \setminus S]$ has pathwidth at most one. Let $P \subseteq (V \setminus S)$ be the set of pendant vertices in $G[V \setminus S]$ that have at least one neighbor in S. Then $|P| \leq 17(k+2)^2 k(2k-1)$.*

5 Conclusion

We defined the Pathwidth-One Vertex Deletion problem as a natural variant of the iconic Feedback Vertex Set problem, and initiated the study of its algorithmic complexity. We established that the problem is NP-complete, and showed that the problem parameterized by the solution size k is fixed-parameter tractable. We gave an FPT algorithm for the problem that runs in $\mathcal{O}^*(7^k)$ time, and showed that the problem has a polynomial kernel on $\mathcal{O}(k^4)$ vertices.

An immediate question is whether these bounds can be improved upon[3]. A more challenging problem is to try to solve the analogous problem for larger values of pathwidth. That is, we know that for any positive integer c, the Pathwidth c Vertex Deletion problem, defined analogously to Pathwidth-One Vertex Deletion, is FPT parameterized by the solution size. This follows from the Graph Minor Theorem of Robertson and Seymour because, for each fixed c, the set of YES instances for this problem form a minor-closed class. However, for $c = 2$, the number of graphs in the obstruction set is already a hundred and ten [26], and so our approach would probably be of limited use for $c \geq 2$. Thus the interesting open problems for $c \geq 2$ are: (i) Can we get an $\mathcal{O}^*(d^k)$ FPT algorithm for the problem for some small constant d, and (ii) Does the problem have a polynomial kernel?

Acknowledgements. We thank our anonymous reviewers for a number of useful comments for improving the paper.

[3] After we presented our work at WG 2010, Cygan et al. [14] improved both these bounds, to $\mathcal{O}^*(4.65^k)$ and $\mathcal{O}(k^2)$, respectively. See Introduction for more details.

References

1. Álvarez, C., Serna, M.: The Proper Interval Colored Graph problem for caterpillar trees. Electronic Notes in Discrete Mathematics 17, 23–28 (2004)
2. Arnborg, S., Proskurowski, A., Seese, D.: Monadic Second Order Logic, Tree Automata and Forbidden Minors. In: Schönfeld, W., Börger, E., Kleine Büning, H., Richter, M.M. (eds.) CSL 1990. LNCS, vol. 533, pp. 1–16. Springer, Heidelberg (1991)
3. Assmann, S.F., Peck, G.W., Sysło, M.M., Zak, J.: The bandwidth of caterpillars with hairs of length 1 and 2. SIAM Journal on Algebraic and Discrete Methods 2(4), 387–393 (1981)
4. Bafna, V., Berman, P., Fujito, T.: A 2-Approximation Algorithm for the Undirected Feedback Vertex Set problem. SIAM Journal of Discrete Mathematics 12(3), 289–297 (1999)
5. Bodlaender, H.L.: On disjoint cycles. International Journal of Foundations of Computer Science 5(1), 59–68 (1994)
6. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theoretical Computer Science 209(1–2), 1–45 (1998)
7. Bodlaender, H.L.: Treewidth: Characterizations, Applications, and Computations. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 1–14. Springer, Heidelberg (2006)
8. Bodlaender, H.L.: A Cubic Kernel for Feedback Vertex Set. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 320–331. Springer, Heidelberg (2007)
9. Bodlaender, H.L., Koster, A.M.C.A.: Combinatorial Optimization on Graphs of Bounded Treewidth. The Computer Journal 51(3), 255–269 (2008)
10. Bodlaender, H.L., van Dijk, T.C.: A cubic kernel for feedback vertex set and loop cutset. Theory of Computing Systems 46(3), 566–597 (2010)
11. Bryant, R.L., Kinnersley, N.G., Fellows, M.R., Langston, M.A.: On Finding Obstruction Sets and Polynomial-Time Algorithms for Gate Matrix Layout. In: Proceedings of the 25th Allerton Conference on Communication, Control and Computing, pp. 397–398 (1987)
12. Burrage, K., Estivill Castro, V., Fellows, M.R., Langston, M.A., Mac, S., Rosamond, F.A.: The Undirected Feedback Vertex Set Problem Has a Poly(k) Kernel. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 192–202. Springer, Heidelberg (2006)
13. Cao, Y., Chen, J., Liu, Y.: On Feedback Vertex Set: New Measure and New Structures. In: Kaplan, H. (ed.) Algorithm Theory - SWAT 2010. LNCS, vol. 6139, pp. 93–104. Springer, Heidelberg (2010)
14. Cygan, M., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: An improved fpt algorithm and quadratic kernel for pathwidth one vertex deletion. Accepted at IPEC 2010 (2010)
15. Dehne, F., Fellows, M.R., Langston, M.A., Rosamond, F.A., Stevens, K.: An $O(2^{O(k)}n^3)$ FPT-Algorithm for the Undirected Feedback Vertex Set problem. Theory of Computing Systems 41(3), 479–492 (2007)
16. Diestel, R.: Graph Theory, 3rd edn. Springer, Heidelberg (2005)
17. Dorn, F., Fomin, F.V., Thilikos, D.M.: Subexponential parameterized algorithms. Computer Science Review 2(1), 29–39 (2008)
18. Downey, R.G., Fellows, M.R.: Fixed Parameter Tractability and Completeness. In: Complexity Theory: Current Research, pp. 191–225. Cambridge University Press, Cambridge (1992)

19. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
20. Fellows, M.R., Langston, M.A.: On Search, Decision and the Efficiency of Polynomial-time Algorithms. In: Proceedings of STOC 1989, pp. 501–512. ACM Press, New York (1989)
21. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006)
22. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. Journal of Computer and System Sciences 72(8), 1386–1396 (2006)
23. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. SIGACT News 38(1), 31–45 (2007)
24. Kanj, I.A., Pelsmajer, M.J., Schaefer, M.: Parameterized Algorithms for Feedback Vertex Set. In: Downey, R.G., Fellows, M.R., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 235–247. Springer, Heidelberg (2004)
25. Karp, R.M.: Reducibility among combinatorial problems. Complexity of Computer Communications, 85–103 (1972)
26. Kinnersley, N.G., Langston, M.A.: Obstruction Set Isolation for the Gate Matrix Layout problem. Discrete Applied Mathematics 54(2-3), 169–213 (1994)
27. Kloks, T.: Treewidth – computations and approximations. LNCS, vol. 842. Springer, Heidelberg (1994)
28. Lin, M., Lin, Z., Xu, J.: Graph bandwidth of weighted caterpillars. Theoretical Computer Science 363(3), 266–277 (2006)
29. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. Journal of Computer and System Sciences 20(2), 219–230 (1980)
30. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, Oxford (2006)
31. Papadimitriou, C.H.: The NP-Completeness of the bandwidth minimization problem. Computing 16(3), 263–270 (1976)
32. Philip, G., Raman, V., Villanger, Y.: A quartic kernel for pathwidth-one vertex deletion. A full version of the current paper, `http://www.imsc.res.in/~gphilip/publications/pwone.pdf`
33. Raman, V., Saurabh, S., Subramanian, C.: Faster fixed parameter tractable algorithms for finding feedback vertex sets. ACM Transactions on Algorithms 2(3), 403–415 (2006)
34. Robertson, N., Seymour, P.D.: Graph minors I. Excluding a forest. Journal of Combinatorial Theory, Series B 35(1), 39–61 (1983)
35. Robertson, N., Seymour, P.D.: Graph Minors. II. Algorithmic Aspects of Tree-Width. Journal of Algorithms 7(3), 309–322 (1986)
36. Thomassé, S.: A quadratic kernel for feedback vertex set. In: Proceedings of SODA 2009, Society for Industrial and Applied Mathematics, pp. 115–119 (2009)