PUSHING THE FRONTIERS OF SUBEXPONENTIAL FPT TIME FOR FEEDBACK VERTEX SET

GAÉTAN BERTHE, MARIN BOUGERET, DANIEL GONÇALVES, AND JEAN-FLORENT RAYMOND

Abstract. The paper deals with the Feedback Vertex Set problem parameterized by the solution size. Given a graph G and a parameter k, one has to decide if there is a set S of at most k vertices such that $G - S$ is acyclic. Assuming the Exponential Time Hypothesis, it is known that FVS cannot be solved in time $2^{o(k)}n^{\mathcal{O}(1)}$ in general graphs. To overcome this, many recent results considered FVS restricted to particular intersection graph classes and provided such $2^{o(k)}n^{\mathcal{O}(1)}$ algorithms.

In this paper we provide generic conditions on a graph class for the existence of an algorithm solving FVS in subexponential FPT time, i.e. time $2^{k^{\varepsilon}}$ poly(n), for some $\varepsilon < 1$, where n denotes the number of vertices of the instance and k the parameter. On the one hand this result unifies algorithms that have been proposed over the years for several graph classes such as planar graphs, map graphs, unit-disk graphs, pseudo-disk graphs, and string graphs of bounded edge-degree. On the other hand it extends the tractability horizon of FVS to new classes that are not amenable to previously used techniques, in particular intersection graphs of "thin" objects like segment graphs or more generally s-string graphs.

1. Introduction

1.1. **Context.** Given an n-vertex graph G and a parameter $k \in \mathbb{N}$, the Feedback Vertex Set problem (FVS for short) asks whether there exists a set S of at most k vertices such that $G - S$ has no cycle. This is a fundamental decision problem in graph theory and one of Karp's 21 NP-complete problems. Because of its hardness in a classical setting, the problem has been widely studied within the realm of parameterized complexity. This line of research aims to investigate the existence of *FPT algorithms*, i.e., algorithms that run in time $f(k) \cdot n^{\mathcal{O}(1)}$, for some computable function f. Such algorithms provide a fine-grained understanding on the time complexity of a problem and describe regions of the input space where the problem can be solved in polynomial time. Note that it is crucial here to obtain good bounds on the function f since the (potentially) super-polynomial part of the running time is confined in the $f(k)$ term. In this direction it was proved that under the Exponential Time Hypothesis of Impagliazzo, Paturi and Zane, FVS does not admit an algorithm with running time $2^{o(n)}n^{\mathcal{O}(1)}$ (see [CFK+15]). Nevertheless certain classes of graphs (typically planar graphs) have been shown to admit algorithms with running times of the form $2^{\mathcal{O}(k^{\varepsilon})}n^{\mathcal{O}(1)}$ (for some $\varepsilon < 1$), i.e., where the contribution of the parameter k is subexponential. Such algorithms are called *subexponential parameterized algorithms* and they are the topic of this paper. Given the numerous existing results on this theme, there are two main directions of research: to improve the running times in the classes where an algorithm is already known, or to extend the tractability horizon of FVS by providing more general settings where subexponential FPT algorithms exist.

We are here interested by the second direction, that can be summarized by the following question.

Question 1.1. What are the most general graph classes where FVS admits a subexponential parameterized algorithm ?

Historically, a primary source of graph classes studied to make progress on the above question was geometric intersection graphs. In an *intersection graph*, each vertex corresponds to a subset of some ambient space, and two vertices are adjacent if and only if the subsets intersect. Taking the Euclidean plane as the ambient space, many graph classes can been defined by setting restrictions on the subsets used to represent the vertices. One can for instance consider intersection graphs of disks in the plane, or segments, or Jordan arcs[1]. With such subsets, one defines the class of *disk graphs*, *segment graphs*, and *string graphs* respectively. It is also often the case that there are conditions dealing with all the n subsets representing the vertices of a given graph. For example, if we consider disks (resp. segments) one can ask those to have the same diameter (resp. to use at most d different slopes), and this defines the class of *unit disk graphs* (resp. *d-DIR graphs*). When considering strings, one possible property is that any string has at most d points shared with the other considered strings. This defines string graphs with *edge*

Date: April 2025.

[1]In the following, those are called strings.

degree at most d. A weaker condition is to ask every pair of the considered strings to intersect on at most s points, this defines *s-string graphs.* This class generalizes several natural classes such as planar graphs, map graphs, unit-disk graphs, segment graphs, string graphs of bounded edge-degree, and intersection graphs of α-convex bodies that exclude a fixed subgraph (see [Mat14b, KM94, BT22]).

For all these graph classes, FVS is NP-complete, and actually under ETH none of them admits a $2^{o(\sqrt{n})}$-time algorithm. Indeed, this lower bound was given in [dBBKB$^+$20] for induced grid graphs, which form a subclass of unit disk graphs and 2-DIR graphs. On the other hand for each of the aforementioned classes there is an algorithm solving FVS in subexponential time. More precisely, this algorithm applies to any string graph and runs in time $2^{\tilde{\mathcal{O}}(n^{2/3})}$ [BR19].[2]

Regarding subexponential parameterized algorithms, the case of unit disk graphs was settled with an algorithm whose running time matches the ETH lower bound [AO21]. This result uses the fact that these graphs admit vertex partitions into cliques such that each of these cliques is adjacent to only a constant number of the other cliques. Such a property does not hold for the other graph classes mentioned above. However, other techniques have been developed to deal with the other aforementioned classes such as the classes of bounded edge degree string graphs [BT22], contact-segment and square graphs [BBGR24a], disk graphs [LPS$^+$22, ACO23], or the pseudo-disk graphs [BBGR25]. Note that when dealing with classes of intersection graphs, the representation of the input (if known) could be used by the algorithm. Some of these algorithms are *robust*, meaning that the input graph G is provided using one of the classical graph data structures, where there is no indication of the intersection model of G. Because the recognition problem is difficult for most of the classes discussed above, robustness is a substantial advantage.

1.2. **Our contribution.**

Toward answering Question 1.1, we identify sufficient conditions for a graph class (then said to be *nice*) to admit a subexponential parameterized algorithm for FVS. As we will see later these conditions are satisfied by several natural graph classes, some of which were not known to admit a subexponential parameterized algorithm prior to this work.

Let us now provide some intuition behind the conditions we require for a *nice* graph class. We discuss here the similarities between these conditions and classical studied properties, while the reasons why these conditions help to get a subexponential parameterized algorithm for FVS are examined in Section 2. A starting point is to review known results about string graphs, which constitute a good candidate to answer the previous question. In particular, the following results are known for string graphs. For a graph H, let us say that a graph is *H-free* if it does not contain H as subgraph.

Theorem 1.2 ([Lee17],[DN19])**.** *$K_{r,r}$-free string graphs on n vertices have treewidth $\mathcal{O}(\sqrt{nr\log r})$.*

Theorem 1.3 ([Lee17])**.** *There exists a constant c such that for $r > 0$ it holds that every $K_{r,r}$-free string graph on n vertices has at most $cr(\log r)n$ edges.*

These results[3] are interesting in our case, as a simple folklore branching allows us to reduce the problem to the case where the instance (G, k) of FVS is $K_{r,r}$-free for $r = \lceil k^{\varepsilon}\rceil$. Thus, among the conditions required for a graph class to be nice, two of them correspond to a relaxed version of the above theorems.

Our last main condition is related to neighborhood complexity. A graph class $\mathcal{G}$ has *linear neighborhood complexity (with ratio c)* if for any graph $G \in \mathcal{G}$ and any $X \subseteq V(G)$, $|\{N(v) \cap X, v \in V(G)\}| \le c|X|$. It is known that bounded-expansion graph classes have linear neighborhood complexity [RVS19] as well as bounded twin-width graphs [BFLP24]. In previous work on parameterized subexponential algorithms [LPS$^+$23, ACO23, BBGR25, BBGR24a], it appeared useful that the considered graphs have the property that, if G is $K_{r,r}$-free (or even K_r-free), then for any $X \subseteq V(G)$, $|\{N(v) \cap X, v \in V(G)\}| \le r^{\mathcal{O}(1)}|X|$. Notice that this is slightly stronger than requiring that a class that is $K_{r,r}$-free (or K_r-free) for a fixed r has linear neighborhood complexity, as it is important for our purpose that the dependency in r is polynomial. We point out that $K_{r,r}$-free string graphs have bounded-expansion, hence linear neighborhood complexity, however this does not imply that the dependency in r is polynomial. Thus, our last main condition (called *bounded tree neighborhood complexity*) can be seen as a slightly stronger version of this "polynomially dependent" neighborhood complexity. Let us now proceed to the formal definitions.

Definition 1.4. *We say that a graph class $\mathcal{G}$ has* bounded tree neighborhood complexity *(with parameters α, f_1, f_2) if there exist an integer α and two polynomial functions f_1, f_2 such that the following conditions*

[2]The notation $\tilde{\mathcal{O}}$ ignores polylogarithmic factors, i.e. we write $g(x) = \tilde{\mathcal{O}}(f(x))$ if for some c we have $g(x) = \mathcal{O}(f(x)\cdot\log^c x)$.

[3]Note that an error was found in the proof of the above results in [Lee17], but a claim by the author was made that the proof can be corrected in the case of string graphs (see [BR24]). Moreover the earlier bound in [Mat14a] yields similar results, up to logarithmic factors.

hold. For every r, every $K_{r,r}$-free graph $G \in \mathcal{G}$, every set $A \subseteq V(G)$ and every family $\mathcal{T}$ of disjoint non-adjacent[4] *vertex subsets of $G - A$, each inducing a tree:*

(1) $|\{N_A(T),\ T \in \mathcal{T}\}| \le f_1(r)|A|^\alpha$, where $N_A(T)$ denotes the neighbors of the vertices of T in A, and

(2) $|\{N_A(T),\ T \in \mathcal{T}\}| \le f_2(r,p,m)|A|$, where p and m denote the maximum over all $T \in \mathcal{T}$ of $|N_A(T)|$ and $|T|$ respectively.

Definition 1.5. *We say that an hereditary graph class $\mathcal{G}$ is* nice *(for parameters $\alpha, f_1, f_2, \delta, f, d$) if all the following conditions hold:*

(1) $\mathcal{G}$ is stable by contraction of an edge between degree-two vertices that do not belong to a triangle.

(2) $\mathcal{G}$ has bounded tree neighborhood complexity (for some parameters α, f_1, f_2).

(3) There exist $\delta < 1$ and a constant f_r that depends polynomially in r such that for any $K_{r,r}$-free graph $G \in \mathcal{G}$, $\mathsf{tw}(G) = \mathcal{O}(f_r \cdot n^\delta)$.

(4) There is a constant d_r that depends polynomially in r such that for any $K_{r,r}$-free graph $G \in \mathcal{G}$, $|E(G)| \le d_r \cdot |V(G)|$. Without loss of generality we will assume $d_r \ge r$.

Our main result is the following.

Main Theorem. *For every nice hereditary graph class $\mathcal{G}$ there is a constant $\eta < 1$ such that FVS can be solved in $\mathcal{G}$ in time $2^{k^\eta} \cdot n^{\mathcal{O}(1)}$.*

Actually we provide a single generic algorithm for all nice classes and the parameters of the class (in the definition of nice) appear in the complexity analysis and are used to define η. The techniques used to prove the above result are discussed in Section 2. For the time being, let us focus on consequences. As hinted above, being nice is a natural property shared by several well-studied classes of graphs. In particular we show that it is the case for s-string graphs and pseudo-disk graphs, hence we have the following applications.

Corollary 1.6. *There exists $\eta < 1$, such that for all s there is a robust parameterized subexponential algorithm solving FVS in time $2^{\tilde{\mathcal{O}}\left(s^{\mathcal{O}(1)} k^\eta\right)} n^{\mathcal{O}(1)}$ for n-vertex s-string graphs.*

Corollary 1.7. *There exists $\eta < 1$, such that there is a robust parameterized subexponential algorithm solving FVS in time $2^{k^\eta} n^{\mathcal{O}(1)}$ for n-vertex pseudo-disk graphs.*

Observe that the two corollaries above encompass a wide range of classes of geometric intersection graphs for which subexponential parameterized algorithms have been given in previous work such as planar graphs, map graphs, unit-disk graphs, disk graphs, or more generally pseudo-disk graphs, and string graphs of bounded edge-degree. In this sense our main result unifies the previous algorithms.

Also, it captures new natural classes such as segment graphs, or more generally s-string graphs, where previous tools were unsuitable (as discussed in Section 2). We point out that before this work, the existence of subexponential parameterized algorithm for FVS was open even for the very restricted class of 2-DIR graphs.

Generality has a cost and the running time bound we obtain for pseudo-disk graphs is worst than the one obtained in [BBGR25], which heavily relied on the input pseudo-disk representation. On the other hand our algorithm has the extra property of being robust (i.e., it does not require a geometric representation) which is a relevant advantage for those classes of intersection graphs where computing a representation is difficult.

1.3. Basic notations and organisation of the paper.

In this paper logarithms are binary and all graphs are non-oriented and simple. Unless otherwise specified we use standard graph theory terminology, as in [Die05] for instance. For a graph G, and $v \in V(G)$, we denote $N_G(v)$ the neighbors of v. We omit the subscript when it is clear from the context. For $A \subseteq V(G)$, we use the notation $N(A) = (\cup_{v \in A} N(v)) \setminus A$ and denote $G[A]$ the subgraph induced by G on A. For $v \in V(G)$ and $B \subseteq V(G)$, we denote $N_B(v) = N(v) \cap B$ and for $A \subseteq V(G)$ we denote $N_B(A) = N(A) \cap B$, with the additional notation $d_B(A) = |N_B(A)|$. For a graph H we say that G is H-free if H is not a subgraph of G. Two disjoint vertex subsets or subgraphs Z, Z' of a graph G are said to be *non-adjacent* (in G) if there is no edge in G with one endpoint in Z and the other in Z'.

[4] Two vertex subsets are *non-adjacent* in a graph if there is no edge from one to the other, see Subsection 1.3.

Organisation. In Section 2 we explain why the approaches developed for other intersection graph classes in the papers [ACO23, BT22, BBGR25, LPS+22] do not apply here and present the main ideas behind our algorithm. Section 3 and Section 4 are devoted to the description and analysis of the algorithm. In Section 5 we provide applications of our main theorem by showing in particular that s-string graphs are nice. Finally, in Section 6 we discuss open problems and possible extensions of the approach developed here.

2. Our techniques

2.1. **Why bidimensionality fails and differences with classes of "fat" objects.**

Even if our goal is to abstract from a specific graph class, let us consider in this section the class of 2-DIR graphs, corresponding to the intersection graphs of vertical or horizontal segments in the plane. As these objects are non "fat"[5] and can cross (unlike pseudo-disks), this class constitute a good candidate to exemplify the difficulty.

A common approach is as follows. Given an instance (G, k) of FVS, we compute first in polynomial time a 2-approximation, implying that we either detect a no-instance, or define a set M with $|M| \leq 2k$ and such that $G - M$ is a forest. The goal is now to reduce, using kernelization or subexponential branching rules, to an equivalent instance (G', k') with small treewidth $\mathsf{tw}(G') = k^{1-\varepsilon}$. As FVS can be solved in $\mathsf{tw}(G')^{\mathsf{tw}(G')} n^{\mathcal{O}(1)}$ using a classical dynamic programming approach, we get a subexponential parameterized algorithm. Thus, one has to find a way to destroy in G the obstructions preventing a small treewidth. A first type of obstructions is K_r and $K_{r,r}$, which are easy to handle as there are folklore subexponential branchings when $r = k^{\varepsilon}$. Now, one can see the hard part is destroying $K_{r,r}$ hidden (as a minor for example) (see Figure 1).

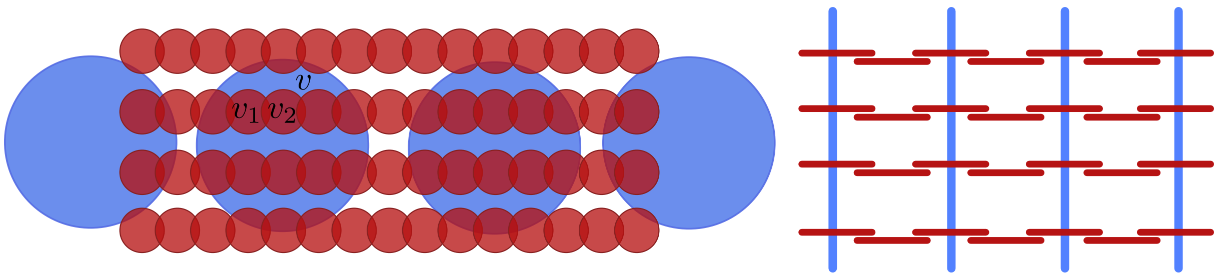

Figure 1. Example of a $K_{r,r}$ contained as a minor for $r = 4$ in a disk graph (left) and a 2-DIR graph (right). In the case of disk graph, v has a matching of size $r - 2$ in its neighborhood, forming a triangle bundle, which can be exploited to branch. The set M are depicted in blue. For the 2-DIR graph, the vertices of the long paths are represented by segments with small variation in their height and not intersecting for better clarity, but are in fact on the same level and intersecting.

A point that seems crucial to us is the following. In intersection graphs of "fat objects" (like disks, squares, or pseudo-disks more generally), the "locally non planar structure" when an object (vertex v in Figure 1) is "traversed" (by v_1, v_2 in Figure 1) comes to the price of an edge ($\{v_1, v_2\}$) in the neighborhood of v. Thus, the presence of a large $K_{r,r}$ as a minor implies that a large matching E_v (of size $\Omega(r)$) will appear in the neighborhood of a vertex v. However, as $G[\{v\} \cup E_v]$ (called a triangle bundle in [LPS+23]) contains r triangles identified on vertex v, the set $\{v\} \cup E_v$ is a good structure to perform a subexponential branching for FVS. Indeed, [LPS+22] proposed a "virtual branching" to handle this structure by either taking v in the solution, or absorbing E_v in M, implying then that the parameter virtually decreases by $|E_v|$ as a solution which does not contain v has to hit all these edges, even if we cannot branch to determine which are exactly the vertices in the solution.

Once no more virtual branching is possible on large triangle bundles, they obtain by some additional specialized techniques that any vertex in M is such that $N_{V(G) \setminus M}(v)$ is an independent set. Then, it is proved ([LPS+22], Corollary 1.1) that in a disk graph where for any $v \in M$, $N_{V(G) \setminus M}(v)$ is an independent set, and where there does not exist a vertex in $V(G) \setminus M$ whose neighborhood is contained in M, then $\mathsf{tw}(G) = \mathcal{O}\left(\sqrt{|M|}\omega(G)^{2.5}\right)$, where $\omega(G)$ denotes the maximum size of a clique in G. This no

[5]A regions R of the plane is said to be α-*fat* if the radius of smallest disk enclosing R is at most α times larger than the radius of the largest disk enclosed in S. A family of regions of the plane is then said to be *fat* if there exists α such that all the elements of the family are α-fat.

longer holds for 2-DIR: the family of pairs (G, M) depicted on the right of Figure 1 is indeed a counter example as they respect the conditions, have $\omega(G) = 2$, but $\mathsf{tw}(G) = \Omega(|M|)$.

More generally, the role of the size of a matching in the neighborhood was studied in [BBGR24a] which shows how subexponential parameterized algorithms can be obtained for graph classes having the "almost square grid minor property" (ASQGM), corresponding informally[6] to $\mathsf{tw}(G) = \mathcal{O}(\omega(G)\mu_N(G)\boxplus(G))$ where $\mu_N(G)$ is the maximum size of a matching in a neighborhood of a vertex, and $\boxplus(G)$ is the largest size of a grid contained as a minor in G. The previous counter example shows that 2-DIR does not have the ASQGM property, implying that we need another approach to handle them.

2.2. A simpler case study: when trees are only paths.

To simplify the arguments, but still understand why properties in Definition 1.5 of a nice graph class are needed, let us assume that the forest $G - M$ only contains paths $(P_i)_i$. This case remains challenging as a large $K_{r,r}$ can still be hidden as a minor (as in Figure 1), and we need to destroy it. To keep notations simple, we use the notation $\mathsf{poly}(.)$ to denote a polynomial dependency on the parameter, and thus we do not try to compute tight formulas depending on the polynomial f_1, f_2, f, d given in the definition of a nice class. We assume that we performed folklore branching and that we are left with a $K_{r,r}$-free graph G for $r = k^\varepsilon$. It is known for string graphs (and thus 2-DIR) that in this case $\mathsf{tw}(G) = \mathsf{poly}(k^\varepsilon)n^{1/2}$ (corresponding to item 3 of the definition of nice). Thus, our goal is to reduce $|V(G) \setminus M|$ to $\mathcal{O}(k^{2-\varepsilon'})$ for some ε' while keeping $|M| = \mathsf{poly}(k^\varepsilon)k$ as it implies $\mathsf{tw}(G) = o(k)$. A first obvious rule is to iteratively contract edges of the P_i's whose endpoints have no neighbors in M. This explains the property of item 1 of the definition of nice. (Actually, we could only require that paths with internal vertices of degree 2 can be replaced with bounded size path without leaving the class, which could be useful for dealing with parity-constrained problems such as OCT.)

Let us now explain how property of item 2 (bounded tree neighborhood complexity, abbreviated bounded T-NC) allows to obtain the following "degree-related size property": for any subpath P of a P_i, $|P| \le \mathsf{poly}(r)(d_M(P)^{\alpha+1})$. Define an independent set $\mathcal{T}$ of size $|P|/2$ by picking every second vertex in P. According to the definition of bounded T-NC (item 1) with $A = N_M(P)$, we get $|\{N_M(v),\ v \in \mathcal{T}\}| \le f_1(r)(d_M(P))^\alpha$. Thus, if $|P| \ge x \cdot f_1(r)(d_M(P))^\alpha$, we found x vertices in P having the same neighborhood M' in M, and thus a vertex $u \in M'$ adjacent to x vertices in P. If x is large enough (about $d_M(P)$), it is always better to take any arbitrary vertex $u \in M'$. This leads to a kernelization rule (corresponding to (KR$_5$)): if there is a vertex $u \in M$ adjacent to approximately $d_M(P)$ vertices in a subpath P, take u and decrease k by one. To sum it up, after applying this rule, for any subpath P of a P_i, if we have $d_M(P) = \mathsf{poly}(r)$, then $|P| = \mathsf{poly}(r)$.

Before the next step we need to apply the following "large degree rule" (corresponding to (KR$_3$)): if there is a vertex v in a P_i such that $d_M(v) > t$, then add v to M. One can prove that by taking $t = 2d_r$, with $d_r = \mathsf{poly}(r)$ the constant defined in item 4 of the definition of a nice class, M does not grow too much after applying this rule exhaustively: by denoting $A \subseteq M$ the set of vertices already in M previously added by this rule, we always have $|A| = \mathsf{poly}(r)|M \setminus A|$. This claim will be discussed in Subsection 2.3.

Observe that at this stage we may still have large $K_{r,r}$ as minor, with for example graphs as in Figure 1 where no rule applies. It remains to define a crucial rule to destroy these $K_{r,r}$. Let us now present the analogue of the `Partitionning-Algorithm` of Subsection 3.7 that partitions the $P_i's$ as follows. For any connected component P_i in $G - M$, we start (see Figure 2) from an endpoint of P_i and collect greedily vertices until we find a subpath P_i^1 such that $d_M(P_i^1) \ge t$, or that there is no more vertices in P_i. If $d_M(P_i^1) \ge t$, then restart a new path starting from the next vertex to create P_i^2, and so on. This defines a partition $P_i = \bigcup_{\ell \in [x(P_i)]}(P_i^\ell)$, where $d_M(P_i^\ell) \ge t$ for any $\ell \in [1, x(P_i) - 1]$ and no lower bound for $d_M\left(P_i^{x(P_i)}\right)$. As we applied the large degree rule, we also know that $d_M(P_i^\ell) \le 2t$ for any $\ell \in [1, x(P_i)]$, because collecting at each step a new vertex in the path P_i^ℓ can increase $d_M(P_i^\ell)$ by at most t. This implies, using the degree-related size property introduced above, that $|P_i^\ell| \le \mathsf{poly}(r)$ for any $\ell \in [1, x(P_i)]$. Let us denote $\mathcal{T}^+$ the set of P_i^ℓ such that $d_M(P_i^\ell) \ge t$ the "large-degree subpaths". Observe that the last considered subpath $P_i^{x(P_i)}$ of each connected component P_i (on the right of each path of $G - M$ in Figure 2) may have $d_M\left(P_i^{x(P_i)}\right) < t$ as there was no more vertices to complete it, hence it is not contained in $\mathcal{T}^+$. We denote $\mathcal{T}^-$ those remaining "small-degree subpaths".

[6]In the correct definition $\mu_N(G)$ is replaced by a slightly more technical parameter.

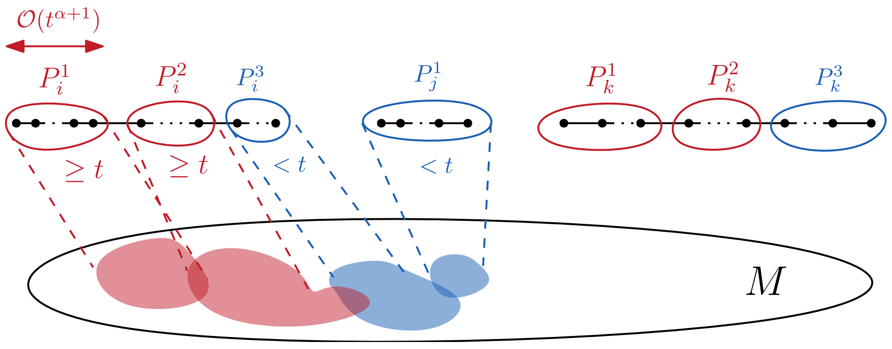

FIGURE 2. Example of partition where P_i is partitioned into $x(P_i) = 3$ subpaths, with P_i^1 and P_i^2 in $\mathcal{T}^+$ and $P_i^3 \in \mathcal{T}^-$.

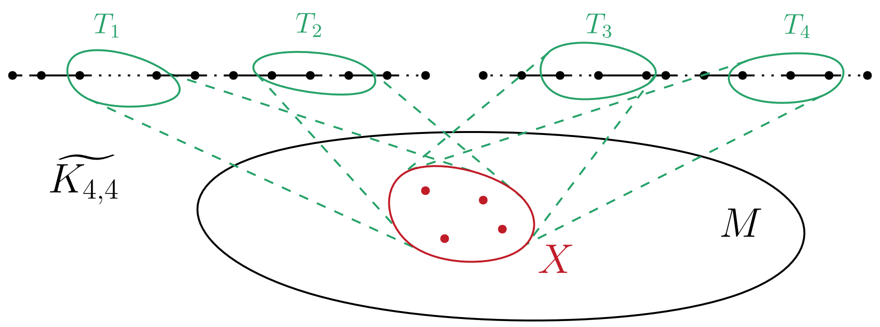

FIGURE 3. Example of $\widetilde{K_{t,t}}$ for $t = 4$. Here we have $X \subseteq N_M(T_i)$ for $1 \leq i \leq 4$.

Let us now explain how the bounded T-NC property (item 2) allows to obtain the following "small number of large-degree subpaths" property. By removing half of the $\mathcal{T}^+$'s, we can get a set $\mathcal{T}^{+'}$ of non-adjacent trees (meaning with no edge between the $T_i's$) such that $|\mathcal{T}^{+'}| \geq \frac{1}{2}|\mathcal{T}^+|$. We can then apply the T-NC property with $A = M$, $\mathcal{T} = \mathcal{T}^{+'}$ and $p = m = \mathsf{poly}(r)$ to obtain $|\{N_M(T),\ T \in \mathcal{T}^{+'}\}| \leq \mathsf{poly}(r)|M|$. Thus, if $|\mathcal{T}^{+'}| \geq x \cdot \mathsf{poly}(r)|M|$, we found x large-degree subpaths (denoted T_i') having the same neighborhood X' in M with $|X'| \geq t$. By choosing $x = t$ and considering $X \subseteq X'$ with $|X| = t$, we found a structure that we call a $\widetilde{K_{t,t}}$ (see Figure 3), formally defined as a pair $(X, (T_i)_{1 \leq i \leq t})$ where

- $X \subseteq M$ has size t,
- $(T_i)_i$ a family of t vertex-disjoint non-adjacent subtrees (paths here) of $G - M$ such that for all $1 \leq i \leq t$, $X \subseteq N_M(T_i)$, and
- for any T_i, $|T_i| \leq \mathsf{poly}(r)$.

Now, inspired by the "virtual branching rule" for a triangle bundle, we introduce a branching rule (corresponding to (BR$_2$)) that either takes almost all vertices in X, or absorbs by adding to M a subset of $t-1$ of the $T_i's$. The complexity behind this rule is fine, as in the second branch, the parameter virtually decreases by $t-1$, and M grows by $(t-1)\max_i\{|T_i|\} = \mathsf{poly}(r)$. This explains how we deal with $K_{t,t}$ hidden as a minor.

Finally, if this $\widetilde{K_{t,t}}$ rule cannot be applied, it remains to bound $|V(G)\backslash M|$. Recall that any $P \in \mathcal{T}^+ \cup \mathcal{T}^-$ is such that $|P| \leq \mathsf{poly}(r)$, and thus we only need to bound $|\mathcal{T}^+ \cup \mathcal{T}^-|$. As we cannot apply the previous rule, we know that the number of big paths is small: $|\mathcal{T}^+| \leq \mathsf{poly}(r)|M|$. Now, to bound $|\mathcal{T}^-|$, observe that we can partition $\mathcal{T}^- = \mathcal{T}_1^- \cup \mathcal{T}_2^-$, where

- $\mathcal{T}_1^-$ is the set of small-degree paths P_i^ℓ for some $\ell > 1$ (belonging to the same path P_i than a large-degree path $P_i^{\ell-1}$).
- $\mathcal{T}_2^-$ is the set of small-degree path which is an entire connected component of $G - M$.

As $|\mathcal{T}_1^-| \leq |\mathcal{T}^+|$, it only remains to bound $|\mathcal{T}_2^-|$. Now, we can exploit once again the bounded T-NC property (item 2) to obtain that, if $|\mathcal{T}_2^-| \geq x \cdot \mathsf{poly}(r)|M|$, then we can find x disjoint non-adjacent paths in $\mathcal{T}_2^-$ having the same neighborhood X in M. This case is different from the $\widetilde{K_{t,t}}$ case as X may be arbitrarily small (we only know that $|X| < t$), and thus unlike $\widetilde{K_{t,t}}$ this does not allow to decrease the parameter by a large amount. However, in this case, paths of $\mathcal{T}_2^-$ are just connected components, and this help us to add a last rule ((KR$_4$)) that identifies a "redundant" path that can be safely removed. It can be shown that such a redundant path can be found just by taking $x = t + 2$. Hence after applying the rule exhaustively we can assume $|\mathcal{T}_2^-| < x \cdot \mathsf{poly}(r)|M| = \mathsf{poly}(r)|M|$.

This concludes the proof for this restricted setting where the connected components of $G - M$ are paths, as we obtain by taking ε small enough $|V(G) \setminus M| \leq \mathsf{poly}(r)|M| \leq \mathsf{poly}(k^\varepsilon)k = \mathcal{O}(k^{2-\varepsilon'})$ as required.

2.3. **Challenges to lift the result from paths to trees.**

We now consider the real setting where given (G, k) where G is $K_{r,r}$-free for $r = k^\varepsilon$, and given M a feedback vertex set of size at most $2k$, we want to reduce the graph to obtain $|V(G) \setminus M| = \mathcal{O}(k^{2-\varepsilon'})$. The approach still consists in partitioning $G - M$ in an appropriate way (called a t-uniform partition).

A first problem when trying to adapt the approach of Subsection 2.2 is the degree-related size property. Indeed, after the first two sections Subsection 3.4 and Subsection 3.5, we are now only able to obtain that for any subtree T of $G-M$, $|T| \leq \mathsf{poly}(r)\mu(T)^{\mathcal{O}(\alpha)}$ where $\mu(T) = \max(d_M(T), b_{\overline{M}}(T))$ and $b_{\overline{M}}(T) = |\{v \in T,\ N(v) \not\subseteq M \cup T\}|$ is the size of the "border of T". Observe that $b_{\overline{M}}(P)$ is at most 2 for any subpath P of path P_i, whereas $b_{\overline{M}}(T)$ can only be bounded by $|T|$ for a subtree T. Informally, in the path case $|P|$ was only polynomially dependent on $d_M(P)$, and now $|T|$ is also polynomially depends on $b_{\overline{M}}(T)$.

A second problem is the large degree rule. Suppose that this rule no longer applies (meaning that for every $u \in V(G - M)$, we have $d_M(u) \leq t$), and suppose now that because of another rule a vertex $v \in V(G) \setminus M$ is added to M, denoting $M' = M \cup \{v\}$. Then this can create a new large degree vertex v' with $d_{M'}(v') > t$ (and so $d_M(v') = t$). Then v' would need to be added and the problem may arise again for another vertex v''. This "cascading" can easily be prevented if $G - M$ is a forest of paths: it suffices to apply the rule a first time at the start of the algorithm, but with $t' = t - 2$. We then have for each $v \in V(G - M)$ the bound $d_M(v) \leq t - 2$, and we do not need to applies the rule again after as adding vertices to M may increase $d_M(v)$ by at most 2 ensuring the wanted bound $d_M(v) \leq t$ for $v \in V(G-M)$. However, in the case of a tree, we can have in $G - M$ a vertices of arbitrarily large degree, so we cannot apply the same solution. The problem is treated with the help of a technical lemma (that we prove at the end of the proof, see Lemma 3.32) which ensures that throughout the execution of the algorithm we keep $|M| = \mathsf{poly}(r)k$.

Finally, a third problem is the definition of the partition. As in the case of paths we want to partition $G - M$ into a "t-uniform partition" $\mathcal{T}$, where in particular we have $\mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$, and for any $T \in \mathcal{T}$, $d_M(T) \leq 2t$ and $|T| \leq \mathsf{poly}(r)\mu(T)^{\mathcal{O}(\alpha)}$ (see Definition 3.23 for the complete definition). The greedy approach presented for the case of paths is now more involved, as we have to cut each tree of $G-M$ into subtrees that have small border $b_{\overline{M}}(T)$, as otherwise the previous bound $|T| \leq \mathsf{poly}(r)\mu(T)^{\mathcal{O}(\alpha)}$ becomes useless when $\mu(T)$ is too large.

This partitioning procedure is defined in Subsection 3.7. It can either:

- Fail and find a subtree T with $|T| > \mathsf{poly}(r)\mu(T)^{C\alpha}$ for some constant C, implying that our degree-related size rule can be applied.
- Fail and find too many subtrees $T_i \in \mathcal{T}^+$ with large degree, implying that we found a $\widetilde{K_{t,t}}$, and that our $\widetilde{K_{t,t}}$ rule can be applied.
- Produce a t-uniform partition with $|\mathcal{T}^+| \leq \mathsf{poly}(r)|M|$.

The third case is treated in Subsection 3.8, where we either find another way to apply one more time a reduction rule, or prove that $|V(G) \setminus M| = \mathcal{O}(k^{2-\varepsilon'})$.

3. FVS in subexponential FPT time in nice graph classes

3.1. **Preliminary branching to remove $K_{r,r}$.**

To avoid confusion, we refer to the initial instance with (G_0, k_0). In this section we use a folklore branching for FVS to remove the large bicliques $K_{r,r}$, where $r = k_0^\varepsilon$ with ε to be set later depending on the considered graph class $\mathcal{G}$. Before performing any branching, we compute a 2-approximation of a minimum feedback vertex set of G_0 using the following result, and denote it by M_0.

Theorem 3.1 ([BBF99, BG96]). *A 2-approximation of a minimum feedback vertex set can be constructed in polynomial time.*

If $|M_0| \leq k_0$ or if $|M_0| > 2k_0$ we can conclude that the instance is positive or negative. We thus are left with the case where $k_0 < |M_0| \leq 2k_0$.

Let us now describe a branching algorithm starting with (G_0, k_0, M_0) (with M_0 is a feedback vertex set of G_0 with $|M_0| \leq 2k_0$) and leading to a set of instances $\mathcal{I}$, whose properties are discussed below.

The algorithm initializes $\mathcal{I}$ to $\{(G_0, k_0, M_0)\}$ and applies the following branching rule to the elements of $\mathcal{I}$ as long as possible.

(BR$_1$) Given an instance $(G, k, M) \in \mathcal{I}$, if G contains a $K_{r,r}$-subgraph with parts A and B, we replace this instance by $2r$ instances $(G - X, k - (r - 1), M \setminus X)$, for any set X of size $r - 1$ that is either contained in A, or contained in B.

Remark 3.2. If the parameter $k - (r - 1)$ is negative, then this instance is negative and we remove it from $\mathcal{I}$.

Lemma 3.3. *There is a $2^{\mathcal{O}(r \log |M|)}|V(G)|^{\mathcal{O}(1)}$-time algorithm that, given an instance $(G, k, M) \in \mathcal{I}$, applies rule (BR$_1$) or correctly concludes that G is $K_{r,r}$-free.*

Proof. TOPROVE 0 □

We summary the properties obtained after this series of branchings with the following lemma:

Lemma 3.4. *At the end of the preliminary branching, the set $\mathcal{I}$ satisfies the properties:*

(1) The instance (G_0, k_0) is a YES-instance if and only if $\mathcal{I}$ contains a YES-instance.
(2) For any $(G, k, M) \in \mathcal{I}$ the graph G is $K_{r,r}$-free induced subgraph of G_0.
(3) For any $(G, k, M) \in \mathcal{I}$, M is a feedback vertex set of G with $|M| \leq 2k_0$.
(4) The total time to generate $\mathcal{I}$ is in $2^{\mathcal{O}(r \log k_0)}|V(G_0)|^{\mathcal{O}(1)}(2r)^{\frac{k_0}{r-1}}$ and $|\mathcal{I}| = \mathcal{O}\left((2r)^{\frac{k_0}{r-1}}\right)$.

Proof. TOPROVE 1 □

3.2. **The main recursive algorithm.** We now consider each element (G_i, k_i, M_i) of $\mathcal{I}$ as an instance $(G_i, k_i, M_i, \emptyset)$ of the following problem $(r, \mathcal{G})$-Ann-FVS, and our goal now is to solve these instances of $(r, \mathcal{G})$-Ann-FVS using our main recursive Algorithm 1.

Definition 3.5. *Given a nice graph class $\mathcal{G}$ and $r > 3$, the $(r, \mathcal{G})$-Annotated Feedback Vertex Set problem ($(r, \mathcal{G})$-Ann-FVS for short) is the decision problem where given $(G, k, M, \mathcal{H})$ where G is a $K_{r,r}$-free graph of $\mathcal{G}$, k an integer, $M \subseteq V(G)$ a feedback vertex set of G, and $\mathcal{H}$ a family of connected disjoint subsets of M (meaning that for any $H \in \mathcal{H}$, $G[H]$ is connected) such that $|\mathcal{H}| \leq k$, and where the question is whether there exists a feedback vertex set S of G of size at most k that additionally intersects every set of $\mathcal{H}$.*

For the sake of completeness we provide here the complete pseudo-code of Algorithm 1, even if it uses rules and subroutine which will be defined later. At this stage, we recommend the reader to only read the following sketch, as the following sections will cover in detail the properties we obtain after each step. The sketch of Algorithm 1 is as follows. We first try to apply (line 1) rules (KR$_1$), (KR$_2$), (KR$_3$), and (KR$_4$), which are like kernelization rules: given the instance $(G, k, M, \mathcal{H})$ we perform a single recursive call on a slightly simpler instance $(G', k', M', \mathcal{H}')$. If none of these first rules apply, the algorithm tries to build a special partition of $G - M$ using the `Partitionning-Algorithm` . If `Partitionning-Algorithm` fails (line 5 or 8) and fall into what we call Case 1 or Case 2, then we apply a kernelization or branching rule. Otherwise we either apply (KR$_4$) or (KR$_5$) (line 15 or line 18), or reach our final point (line 22) where we can prove that $|V(G)|$ is small, implying that $\mathsf{tw}(G) = o(k)$, and solve the instance using a classical DP algorithm.

Algorithm 1. $A(G,k,M,\mathcal{H})$

```
Input: $(G,k,M,\mathcal{H})$ an instance of $(r,\mathcal{G})$-Ann-FVS.
1: if (one of Rule (KR$_1$), Rule (KR$_2$), Rule (KR$_3$), or Rule (KR$_4$) applies on $(G,k,M,\mathcal{H})$) then
2:     Apply the first possible Rule to obtain $(G',k',M',\mathcal{H}')$ and return $A(G',k',M',\mathcal{H}')$
3: end if
4: Apply Partitionning-Algorithm (with $t = 2d_r$) of Lemma 3.26 that tries to build $\mathcal{T}$: a $t$-uniform
   partition of $G - M$ with $|\mathcal{T}^+| \leq p_3(r,t)|M|$ (where $p_3$ is defined in Lemma 3.25)
5: if (procedure fails and falls into Case 1 (output a large subtree $T$)) then
6:     Apply Rule (KR$_5$) on $T$ to obtain $(G',k',M',\mathcal{H}')$ and return $A(G',k',M',\mathcal{H}')$
7: end if
8: if (procedure fails and falls into Case 2 (output a $\widetilde{K_{t,t}}$)) then
9:     Apply the branching Rule (BR$_2$) on this $\widetilde{K_{t,t}}$, generating a set $\mathcal{C}$ of instances
10:    return $\bigvee_{(G',k',M',\mathcal{H}')\in\mathcal{C}} A(G',k',M',\mathcal{H}')$
11: end if
12: // $\mathcal{T}$ is as required
13: Let $Z_1(\mathcal{T})$ and $Z_2(\mathcal{T})$ as defined in Definition 3.28, and $\tilde{M} = M \cup Z_1(\mathcal{T}) \cup Z_2(\mathcal{T})$
14: // By Lemma 3.29, Rule (KR$_1$) and Rule (KR$_2$) do not apply on $(G,k,\tilde{M},\mathcal{H})$
15: if (Rule (KR$_4$) applies on $(G,k,\tilde{M},\mathcal{H})$, and finds a subtree $T$ that can be removed) then
16:     return $A(G-T,k,M,\mathcal{H})$
17: end if
18: if (Rule (KR$_5$) applies on $(G,k,\tilde{M},\mathcal{H})$ and a connected component $T$ of $G-\tilde{M}$, and finds a vertex
    $u \in \tilde{M}$ that can be taken) then
19:     return $A(G-T,k-1,M\setminus\{u\},\mathcal{H}-\{u\})$
20: end if
21: // $|V(G)| = p_4(r,t)|M|$ by Lemma 3.30 implying $\mathsf{tw}(G) = o(|M|)$ by Theorem 1.2
22: return $DP(G,k,M,\mathcal{H})$ // Solves the instance using Theorem 4.1
```

3.3. **Kernelization rules.** Here we provide the four kernelization rules (KR$_1$), (KR$_2$), (KR$_3$), and (KR$_4$) that Algorithm 1 tries to apply Line 1. Each of these rules takes as input an instance $(G,k,M,\mathcal{H})$ of $(r,\mathcal{G})$-Ann-FVS and outputs a single instance $(G',k',M',\mathcal{H}')$. Such a rule is said to be *safe* if:

- On input an instance of $(r,\mathcal{G})$-Ann-FVS it returns an instance of $(r,\mathcal{G})$-Ann-FVS (in particular the graph of the output instance is a $K_{r,r}$-free graph in $\mathcal{G}$), and
- the input instance is a YES-instance if and only if the output instance is a YES-instance.

Notation. Given any instance $(G,k,M,\mathcal{H})$, recall that every connected component of $G-M$ is a tree. We root each of them at an arbitrary vertex. We define a *subforest* of $G-M$ as a subset $T \subseteq V(G)\setminus M$ and say the set is a *subtree* of $G-M$ if $G[T]$ is a tree. Given a vertex v of a connected component T in $G-M$, we define the subtree T_v of $G-M$ as the connected component of v in $G-M-u$, where u is the parent of v, if any. If v is the root of T, then $T_v = T$. In any case T_v is rooted at v. Given $X \subseteq V(G)$, we denote $\mathcal{H}-X = \{H \in \mathcal{H} \mid H \cap X = \emptyset\}$. Given a subtree T of $G-M$, $\partial_{\overline{M}}(T)$ denotes the set $\{v \in T,\ N(v) \not\subseteq M \cup T\}$, and $b_{\overline{M}}(T)$ denotes the size of this set. We also denote $\mu(T) = \max(d_M(T), b_{\overline{M}}(T))$.

We start with two basic reduction rules often used to deal with FVS and that allow to get rid of vertices of degree 1 and arbitrarily long paths of vertices of degree 2. Notice that we only apply here the reduction to vertices in $G-M$.

(KR$_1$) Given an instance $(G,k,M,\mathcal{H})$, if there exists a vertex $v \in V(G)\setminus M$ of degree $d(v) \leq 1$, output $(G-\{v\},k,M,\mathcal{H})$.

(KR$_2$) Given an instance $(G,k,M,\mathcal{H})$, if there exists a path $quvw$ in $V(G)\setminus M$ such that the four vertices have degree 2 in G and $d_M(u) = d_M(v) = 0$, output $(G',k,M,\mathcal{H})$, where G' is the graph obtained from G by contracting the edge uv.

Lemma 3.6. *The rules (KR$_1$) and (KR$_2$) are safe and can be applied in polynomial time.*

Proof. TOPROVE 2 □

Remark 3.7. Observe that we did not use the condition in (KR$_2$) that the endpoints of the considered path have degree at most 2 in G. However it will later be used in Lemma 3.32 to bound the size of M during the execution of the algorithm.

The next rule ensures that the vertices outside M have a small neighborhood in M. It increases the size of M, but in a controlled manner as we will see later in Lemma 3.32.

(KR$_3$) Given an instance $(G, k, M, \mathcal{H})$, if there is a vertex $v \in V(G) \setminus M$ such that $d_M(v) \geq t = 2d_r$, with d_r the value defined in Definition 1.5, output $(G, k, M \cup \{v\}, \mathcal{H})$.

It is immediate that rule (KR$_3$) is safe and can be applied in polynomial time.

The fourth kernelization rule will ensure that the number of neighbors outside M of a vertex $v \in V(G - M)$ is strongly correlated with the size of the neighborhood of its descendants in M. Consider an instance $(G, k, M, \mathcal{H})$ where none of the previous rules applies. Given a family $\mathcal{T}$ of disjoint subtrees of $G - M$, a tree $T \in \mathcal{T}$ is *redundant* (for $\mathcal{T}$) if for all $v \in M$ such that $d_T(v) \geq 2$, there exists $T' \in \mathcal{T}$ with $T' \neq T$ such that $d_{T'}(v) \geq 2$.

Lemma 3.8. *Consider a set $X \subseteq M$ and a set $\mathcal{T}$ of subtrees of $G - M$ with $|\mathcal{T}| \geq |X| + 1$, and such that $N_M(T) = X$ for every $T \in \mathcal{T}$. Then, there exists a redundant tree for $\mathcal{T}$, and it can be found in polynomial time.*

Proof. TOPROVE 3 □

Recall that we consider a fixed child-parent orientation in the forest $G - M$. In what follows, we say that a subset $F \subseteq V(G - M)$ is a *downward-closed subtree* (or *subforest* when the set is not necessarily connected) of $G - M$ when for any $v \in F$ and any children u of v, $u \in F$. The fourth kernelization rule is as follow:

(KR$_4$) Given an instance $(G, k, M, \mathcal{H})$, a set $X \subseteq M$ with $|X| \geq 1$ and a set $\mathcal{T}$ of at least $|X| + 2$ disjoint downward-closed subtrees of $G - M$ such that:
- for all $T \in \mathcal{T}$, we have $N_M(T) = X$, and
- either all the roots of the trees in $\mathcal{T}$ have a common parent r, or $\mathcal{T}$ consists only in connected components of $G - M$,

arbitrarily pick one redundant $T \in \mathcal{T}$ (which exists as shown in Lemma 3.8) and output $(G - V(T), k, M, \mathcal{H})$.

Lemma 3.9. *Rule (KR$_4$) is safe and can be applied in polynomial time.*

Proof. TOPROVE 4 □

Definition 3.10. *Given an instance $(G, k, M, \mathcal{H})$, we say that (KR$_4$)* applies on *$(G, k, M, \mathcal{H})$ if there exists X and $\mathcal{T}$ such that Rule (KR$_4$) can be applied on $(G, k, M, \mathcal{H})$, X and $\mathcal{T}$.*

Observe that in (KR$_4$), we consider that the sets X and $\mathcal{T}$ are given, whereas in Algorithm 1 line 1, we have to find these sets. Thus, we need the following lemma.

Lemma 3.11. *Given an instance $(G, k, M, \mathcal{H})$, deciding if Rule (KR$_4$) can applies on $(G, k, M, \mathcal{H})$ (and finding X and $\mathcal{T}$ if it is the case) can be done in polynomial time.*

Proof. TOPROVE 5 □

3.4. **Properties of the kernelized instances.** The goal of this section is to prove that for an instance $(G, k, M, \mathcal{H})$ for which the kernelization rules do not apply anymore (meaning that we reach Line 4 in Algorithm 1), the size of a subtree T of $G - M$ is strongly related to $\mu(T)$. (Recall that $\mu(T) = \max(d_M(T), b_{\overline{M}}(T))$.)

Remember that because the considered graph class $\mathcal{G}$ is nice, it has bounded tree neighborhood complexity for some parameters α, f_1, f_2. This implies the easy following lemma.

Lemma 3.12. *For every $K_{r,r}$-free $G \in \mathcal{G}$, every set $A \subseteq V(G)$, and every family $\mathcal{T}$ of disjoint non-adjacent subtrees of $G - A$, and every $x \in \mathbb{R}$, if $|\mathcal{T}| \geq x f_1(r)|A|^\alpha$ then there exists $X \subseteq A$ such that at least x subtrees $T \in \mathcal{T}$ satisfy $N_A(T) = X$. Moreover suppose that for every $T \in \mathcal{T}$ we have $d_A(T) \leq p$ and $|T| \leq m$, then if $|\mathcal{T}| \geq x f_2(r, p, m)|A|$, there exists $X \subseteq A$ such that at least x subtrees $T \in \mathcal{T}$ satisfy $N_A(T) = X$.*

Proof. TOPROVE 6 □

Recall that given a subtree T of $G - M$, $\partial_{\overline{M}}(T)$ denotes the set $\{v \in V(T),\ N(v) \not\subseteq M \cup V(T)\}$, and $b_{\overline{M}}(T)$ denotes the size of this set.

We are now ready to bound the degree of the subtrees of $G - M$:

Lemma 3.13. *Consider an instance $(G, k, M, \mathcal{H})$ of $(r, \mathcal{G})$-ANN-FVS such that neither the rule (KR$_1$) nor (KR$_4$) applies. For any subtree T of $G - M$ and any vertex v of T, we have*

$$d_T(v) \leq \mathcal{O}(\max(b_{\overline{M}}(T), f_1(r) d_M(T)^{\alpha+1})).$$

Proof. TOPROVE 7 □

We now show that the previous rules allow to bound the size of certain types of trees that we define now.

Definition 3.14. *A subtree T of $G - M$ is* weakly connected *to M if $G[T \cup \{u\}]$ is acyclic for every u in M (i.e. $d_T(u) \leq 1$ for all $u \in M$). A subtree T of $G - M$, rooted at a vertex v, is* sharp w.r.t. M *if T is not weakly connected to M but for every children u of v, T_u is weakly connected to M.*

Lemma 3.15. *Consider an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$) and (KR$_2$) applies. For any subtree T in $G - M$ that is weakly connected to M, we have that $|T| \leq \mu(T)$.*

Proof. TOPROVE 8 □

We now consider sharp subtrees. Notice that in such a tree, v has bounded degree (by Lemma 3.13), and the subtrees below v have bounded size (by Lemma 3.15). This leads to the following corollary.

Corollary 3.16. *Consider an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$), (KR$_2$) and (KR$_4$) applies. Given a sharp subtree T of $G - M$, we have $|T| = O\left(f_1(r) d_M(T)^\alpha \mu(T)^2\right)$.*

Proof. TOPROVE 9 □

3.5. **Kernelizing when a big tree is found.** When reaching line 4 of Algorithm 1, we call the method `Partitionning-Algorithm` which tries to build a special partition of $G - M$. As we will see later, one output of this procedure is a failure (called case 1) where a "big" (whose size is too large with respect to $d_M(T)$ and $b_{\overline{M}}(T)$) tree T is found in $G - M$. In this section, we explain how we can get rid of such a big tree.

(KR$_5$) Consider an instance $(G, k, M, \mathcal{H})$, with a subtree T of $G - M$ which contains $d_M(T) + b_{\overline{M}}(T)$ vertex disjoint paths of length at least 1 and whose endpoints are all adjacent to some vertex $u \in N_M(T)$. Then output $(G - u, k - 1, M \setminus \{u\}, \mathcal{H} - \{u\})$ if $k \geq 1$ (or a trivial no instance otherwise).

Lemma 3.17. *The rule (KR$_5$) is safe.*

Proof. TOPROVE 10 □

Lemma 3.18. *There is a multivariate polynomial p_1 with $p_1(x, y) = \mathcal{O}(f_1(x) y^{6+\alpha})$ such that for every instance $(G, k, M, \mathcal{H})$ of $(r, \mathcal{G})$-*Ann-FVS *where none of the rules (KR$_1$), (KR$_2$), and (KR$_4$) applies, and for every subtree T of $G - M$ such that the Rule (KR$_5$) does not apply, we have $|T| \leq p_1(r, \mu(T))$.*

Proof. TOPROVE 11 □

Corollary 3.19. *Given an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$), (KR$_2$), and (KR$_4$) applies, and a subtree T of $G - M$ with $|T| > p_1(r, \mu(T))$, Rule (KR$_5$) can be applied in polynomial time on this tree.*

Proof. TOPROVE 12 □

3.6. **Branching when there is a $\widetilde{K_{t,t}}$.** When reaching line 4 of Algorithm 1, we call the method `Partitionning-Algorithm` which tries to build a special partition of $G - M$. As we will see later, one output of this procedure is a failure (called case 2) where a certain dense structure (denoted $\widetilde{K_{t,t}}$, with $t = 2d_r$) is found. In this section, we explain how we branch on such a $\widetilde{K_{t,t}}$.

This new branching rule is a variation of Rule (BR$_1$) that was presented in Subsection 3.1: instead of dealing with $K_{r,r}$ subgraphs, we will now consider a set A of t vertices and a set B of small trees such that contracting those trees would result in a $K_{t,t}$-subgraph. More formally:

Definition 3.20. *Given an instance $(G, k, M, \mathcal{H})$ of $(r, \mathcal{G})$-*Ann-FVS *and an integer t, a $\widetilde{K_{t,t}}$ (of $(G, k, M, \mathcal{H})$) is a pair $(X, (T_i)_{1 \leq i \leq t})$ where*

- *$X \subseteq M$ has size t,*
- *$(T_i)_i$ a family of t disjoint non-adjacent subtrees of $G - M$ such that for all $1 \leq i \leq t$, $X \subseteq N_M(T_i)$, and*
- *for any T_i, $|T_i| \leq p_2(r, t)$ with $p_2(r, t) = p_1(r, 2t)$.*

Remark 3.21. Similarly to a $K_{t,t}$ subgraph, given a $\widetilde{K_{t,t}}$ with parts $(X, (T_i)_{1 \leq i \leq t})$, the subgraph of G induced by two vertices of X and two trees of $(T_i)_i$ always contains a cycle. So a feedback vertex set of G must either contain at least $t - 1$ vertices of X, or intersects each tree of $(T_i)_i$ except at most one.

The difference between the treatment of $K_{t,t}$ and $\widetilde{K_{t,t}}$ is that, when we are in the branch where a solution must intersect each tree of $(T_i)_i$, we cannot branch in subexponential time to guess which vertex of each T_i is picked, and rather perform the following "virtual branching" trick (introduced in [LPS+22] for the special case where T_i are edges). In this branch, we add all these T_i in M, and maintain a packing $\mathcal{H}$ of such sets T_i. The size of the packing (and thus the growth of M) will be bounded by observing that $|\mathcal{H}|$ is a lower bound on the solution size.

We are now ready to define the last branching rule:

(BR$_2$) Given an instance $(G, k, M, \mathcal{H})$ and a $\widetilde{K_{t,t}}$ with parts $(X, (T_i)_i)$, we generate the following instances:
- if $k \geq t-1$, then for each $v \in X$, denoting $X_{\overline{v}} = X \setminus \{v\}$, output the instance $((G - X_{\overline{v}}), k - (t-1), M \setminus X_{\overline{v}}, \mathcal{H} - X_{\overline{v}})$.
- if $|\mathcal{H}| + (t-1) \leq k$, then for each $1 \leq i \leq t$, denoting $R_i = \bigcup_{j \neq i} T_j$ and $\mathcal{T}_{R_i} = \{T_j,\ j \neq i\}$, output the instance $(G, k, M \cup R_i, \mathcal{H} \cup \mathcal{T}_{R_i})$.

Lemma 3.22. *The Rule (BR$_2$) is safe, and can be applied in polynomial time (assuming a $\widetilde{K_{t,t}}$ is provided).*

Proof. TOPROVE 13 □

3.7. **Attempting to build a t-uniform partition of $G - M$.** In this section we define the method `Partitionning-Algorithm` that is called line 4 when (KR$_1$), (KR$_2$), (KR$_3$), and (KR$_4$) do not apply. The algorithm will either fail (case 1) and find a "big" subtree T that allows to apply (KR$_3$) of Subsection 3.5, fail (case 2) and find a $\widetilde{K_{t,t}}$ that allows to apply (BR$_2$) of Subsection 3.6, or find a special partition of $G - M$ that we define now.

Definition 3.23. *Let $(G, k, M, \mathcal{T})$ be an instance of $(r, \mathcal{G})$-Ann-FVS and t be a positive integer. Let $F \subseteq V(G) \setminus M$ be a downward-closed subforest of $G - M$. A family of trees $\mathcal{T}$ of $G - M$ with vertices in F is a* t-uniform partition of F *if:*

(1) Each vertex of F is in exactly one tree of $\mathcal{T}$.
(2) Every $T \in \mathcal{T}$ have $d_M(T) \leq 2t$, $|T| \leq p_1(r, \mu(T))$ (where p_1 is the polynomial function defined in Lemma 3.18) and $\mathcal{T}$ is partitioned into two subsets, $\mathcal{T}^-$ and $\mathcal{T}^+$ such that a tree $T \in \mathcal{T}$ belongs to $\mathcal{T}^-$ if $d_M(T) < t$, and to $\mathcal{T}^+$ otherwise.
(3) If a root r of a tree $\mathcal{T}^-$ has a parent v in $G - M$, then v is the root of a tree in $\mathcal{T}^+$.

Remark 3.24. The definition implies that there is no edges between trees of $\mathcal{T}^-$.

Our goal is to apply the branching rule each time the procedure of Lemma 3.26 end up in the item 2, giving a large number of saturated trees. We prove that it can always be done, and even in polynomial time:

When `Partitionning-Algorithm` tries to build a t-uniform partition, one case (case 2) of failure will be when $\mathcal{T}^+$ becomes to large, and thus we show in the next lemma that in this case we can find a $\widetilde{K_{t,t}}$.

Lemma 3.25. *Consider an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$), (KR$_2$), and (KR$_4$) applies, a downward-closed subforest $F \subseteq V(G - M)$ and a t-uniform partition $\mathcal{T}$ of F with $|\mathcal{T}^+| = p_3(r, t)|M|$ for $p_3(r, t) = 4tf_2(r, 2t, p_2(r, t))$. Then, we can find a $\widetilde{K_{t,t}}$ in polynomial time.*

Proof. TOPROVE 14 □

Lemma 3.26. *Let t be an integer. Consider an instance $(G, k, M, \mathcal{H})$ where none of Rules (KR$_1$), (KR$_2$), (KR$_4$) applies, and such that $d_M(v) < t$ for every $v \in V(G - M)$. There exists a polynomial time algorithm that given $(G, k, M, \mathcal{H})$ either returns:*

(1) a subtree T of the forest $G - M$ such that Rule (KR$_5$) applies,
(2) a $\widetilde{K_{t,t}}$ such that Rule (BR$_2$) applies, or
(3) a t-uniform partition $\mathcal{T}$ of the whole forest $G - M$ with $|\mathcal{T}^+| < p_3(r, s, t)|M|$ (p_3 being defined in Lemma 3.25).

Proof. TOPROVE 15 □

Remark 3.27. Observe that in Line 4, all required conditions listed in Lemma 3.26 to use the method `Partitionning-Algorithm` are fulfilled, as in particular as (KR$_3$) does not apply, we get $d_M(v) \leq 2d_r \leq t$ for any $v \in V(G) - M$.

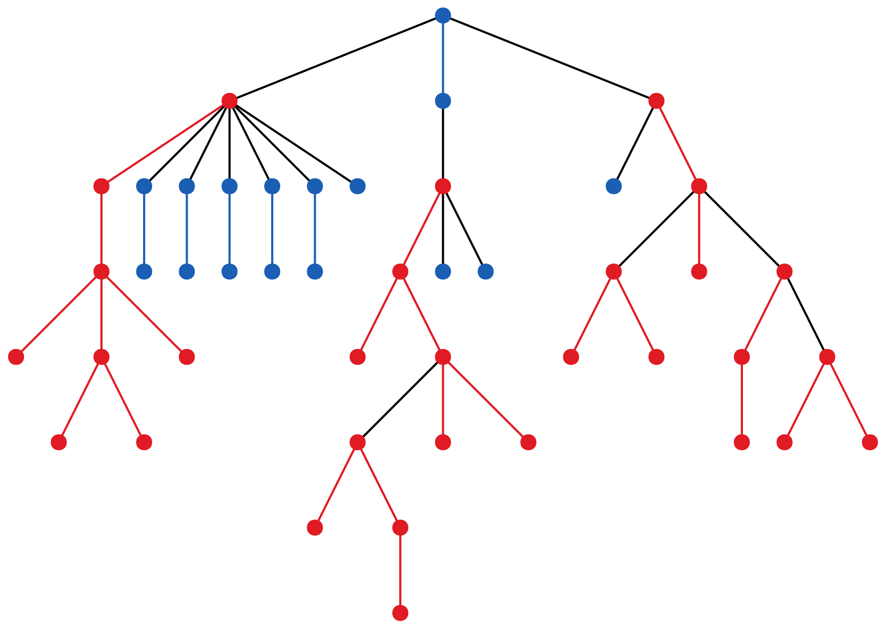

FIGURE 4. Representation of a t-uniform partition of a subtree T of $G - M$. Here are represented only vertices of F (and not vertices of M). Vertices of the trees in $\mathcal{T}^+$ (respectively $\mathcal{T}^-$) are represented in red (respectively blue), so as the edges between two vertices in the same tree of $\mathcal{T}^+$ (respectively $\mathcal{T}^-$). Edges between distinct trees of $\mathcal{T}$ are represented in black.

3.8. **Final step when $G - M$ admits a t-uniform partition with a small number of large-degree parts.** In this section, we consider Line 12 of Algorithm 1 where we found a t-uniform partition $\mathcal{T}$ as required. To bound $|V(G)|$ the algorithm moves some sets $Z_1(\mathcal{T})$ and $Z_2(\mathcal{T})$ (as defined below) to M (to get a slightly larger set $\tilde{M}$ and tries to apply Rule (KR$_4$) or Rule (KR$_5$) on $(G, k, \tilde{M}, \mathcal{H})$. If it not possible, we can prove that $|V(G) - M|$ is small.

Definition 3.28. *Given a t-uniform partition $\mathcal{T}$ of $G - M$, we define the set $Z_1(\mathcal{T})$ as the roots of the trees in $\mathcal{T}^+$, and $Z_2(\mathcal{T})$ as the set of vertices v of $G - M$ having three edge disjoint paths P_1, P_2, P_3 in $G - M$ linking them to vertices of $Z_1(\mathcal{T})$.*

Lemma 3.29. *Given an instance $(G, k, M, \mathcal{H})$, an integer t, and $\mathcal{T}$ a t-uniform partition of $G - M$, we have $|Z_1(\mathcal{T})| + |Z_2(\mathcal{T})| \leq 2|\mathcal{T}^+|$, and by denoting $\tilde{M} = M \cup Z_1(\mathcal{T}) \cup Z_2(\mathcal{T})$, we have the following properties:*

- *Rule (KR$_1$) and Rule (KR$_2$) do not apply on $(G, k, \tilde{M}, \mathcal{H})$.*
- *A connected component T of $G - \tilde{M}$ satisfies $d_{\tilde{M}}(T) \leq 2t + 2$.*

Proof. TOPROVE 16 □

Lemma 3.30. *Given an instance $(G, k, \tilde{M}, \mathcal{H})$ such that Rules (KR$_1$), (KR$_2$), and (KR$_4$) do not apply, and such that for any T connected component of $G - \tilde{M}$, Rule (KR$_5$) does not apply on T and $d_{\tilde{M}}(T) \leq 2t + 2$. Then, $|G - \tilde{M}| \leq p_4(r, t)|\tilde{M}|$ with $p_4(r, t) = (2t + 4)f_2(r, 2t + 2, p_1(r, 2t + 2))$.*

Proof. TOPROVE 17 □

Lemma 3.31. *Let $(G, k, M, \mathcal{H})$ be an input of $(r, \mathcal{G})$-ANN-FVS. If $A(G, k, M, \mathcal{H})$ reaches Line 22, then $|G - M| = \mathcal{O}(p_3(r, t)p_4(r, t)|M|)$.*

Proof. TOPROVE 18 □

Let us now bound the size of M in any call to $A(G, k, M, \mathcal{H})$ by providing invariants on the input of Algorithm 1. Informally, $|M|$ can be bounded as follows. Recall that we perform our first call with $A(G_i, k_i, M_i, \emptyset)$ where $(G_i, k_i, M_i) \in \mathcal{I}$ and $M_i \subseteq M_0$, with $|M_0| \leq 2k_0$. Then, if we follow any path in the tree of calls whose root is $A(G_i, k_i, M_i, \emptyset)$, we may add some vertices to M because of (KR$_3$), let denote A_1 those vertices, or because of (BR$_2$), we denote A_2 those vertices. On a given branch the rule (BR$_2$) add at most k_0 trees of size at most $p_2(r, t)$ so $|A_2| \leq k_0 p_2(r, t)$. Remains to bound the vertices of A_1. Observe that when a vertex v is added to the set M because of (KR$_3$) (and so to A_2) we have

$d_M(v) \geq 2d_r$, and so the number of edges in the graph $G[M]$ increases by at least $2d_r$. We may think that as we have at most $d_r|M|$ edges in $G[M]$, we obtain $2d_r|A_2| \leq d_r|M| \leq d_r(|M_0|+|A_1|+|A_2|)$ which gives a bound on the size of A_1. However this reasoning is flawed as we needed to consider all the vertices added to M, even the one deleted because of (KR$_5$). But then in the supergraph of G' containing those vertices they may be some vertices obtained after contraction of edges by (KR$_2$), which were not of degree 2 in their current supergraph considering deleted vertices. So this supergraph may not be in $\mathcal{G}$, and in fact may even contains a $K_{r,r}$ as subgraph. So a more technical analysis is needed to bound the size of M.

Lemma 3.32. *For any input $(G,k,M,\mathcal{H})$ of Algorithm 1, we have $|M| = \mathcal{O}(k_0 p_5(r,t))$ with $p_5(r,t) = 4+2p_2(r,t)$.*

Proof. TOPROVE 19 □

Using Lemma 3.31 and Lemma 3.32, the following corollary is now immediate.

Corollary 3.33. *Let $(G,k,M,\mathcal{H})$ be an input of $(r,\mathcal{G})$-Ann-FVS. If $A(G,k,M,\mathcal{H})$ reaches Line 22, then $|V(G)| = \mathcal{O}(k_0 p_6(r,t))$ where $p_6(r,t) = p_3(r,t)p_4(r,t)p_5(r,t)$.*

4. Complexity analysis

4.1. **Complexity of the DP in the base case of Algorithm 1.** Let us first describe how Algorithm 1 solves an instance $(G,k,M,\mathcal{H})$ using dynamic programming when no rule applies (DP in the Line 22). Recall that by definition of $(r,\mathcal{G})$-Ann-FVS, G is a $K_{r,r}$-free graph from a nice graph class $\mathcal{G}$, and $\mathcal{H}$ a family of disjoint subsets of M, with each $H \in \mathcal{H}$ inducing a connected graph in G.

Theorem 4.1 ([CFK$^+$15])**.** *An $(r,\mathcal{G})$-Ann-FVS instance $(G,k,M,\mathcal{H})$ with G an n-vertex graph can be solved in time $\mathsf{tw}(G)^{\mathcal{O}(\mathsf{tw}(G))}n^{\mathcal{O}(1)}$.*

Proof. TOPROVE 20 □

Corollary 4.2. *Let $(G,k,M,\mathcal{H})$ be an input of $(r,\mathcal{G})$-Ann-FVS. If $A(G,k,M,\mathcal{H})$ reaches Line 22 and calls $DP(G,k,M,\mathcal{H})$, then the worst case running time of $DP(G,k,M,\mathcal{H})$ is $T_{DP}(k_0,r,t) = 2^{\mathcal{O}\left(k_0^\delta p_7(r,t)\right)}$ with $p_7(r,t) = f_r \log(f_r k_0 p_6(r,t))(p_6(r,t))^\delta$.*

Proof. TOPROVE 21 □

4.2. **Complexity of Algorithm 1.** Informally, the complexity of Algorithm 1 is dominated by the only branching rule ((BR$_2$)). The appropriate parameter α associated to an instance $(G,k,M,\mathcal{H})$ is $\alpha = k+(k-|\mathcal{H}|)$, as informally $\mathcal{H}$ is a packing of hyperedges that we have to hit, and thus $|\mathcal{H}|$ is a lower bound of the cost of any solution. Thus, if $p(n)$ is the (polynomial) complexity of all operations performed in one call of Algorithm 1, and $f(n,\alpha)$ is the worst complexity of Algorithm 1 when $|V(G)| = n$ and $\alpha = k+(k-|\mathcal{H}|)$, (BR$_2$) leads to $f(n,\alpha) \leq p(n)+(2t)f(n,\alpha-(t-1))$, leading to $f(n,\alpha) \leq p(n)(2t)^{\frac{2k-h}{t-1}}$.

Lemma 4.3. *Given an input $(G,k,M,\mathcal{H}) \in \mathcal{I}$ with G a n-vertex graph, $A(G,k,M,\mathcal{H})$ runs in time*

$$T_{DP}\left(k_0,r,t\right)(2t)^{\frac{2k}{t-1}}n^{\mathcal{O}(1)}$$

where T_{DP} is defined in Corollary 4.2.

Proof. TOPROVE 22 □

Let us now bound the running time of the main algorithm (that branches to create the family of instances $\mathcal{I}$ and runs Algorithm 1 on each instance).

Main Theorem. *For every nice hereditary graph class $\mathcal{G}$ there is a constant $\eta < 1$ such that FVS can be solved in $\mathcal{G}$ in time $2^{k^\eta} \cdot n^{\mathcal{O}(1)}$.*

Proof. TOPROVE 23 □

5. Applications

In this section, we prove that s-string graphs and pseudo-disk graphs are nice graph classes for some parameters. We recall that a system of pseudo-disks is a collection of regions in the plane homeomorphic to a disk such that any two of them share at most 2 points of their boundary. Similar arguments are used for both considered classes, but in the case of pseudo-disks the arguments are a bit simpler, we then consider this class in first.

In order to give bounds on tree neighborhood complexity we will use the following theorem.

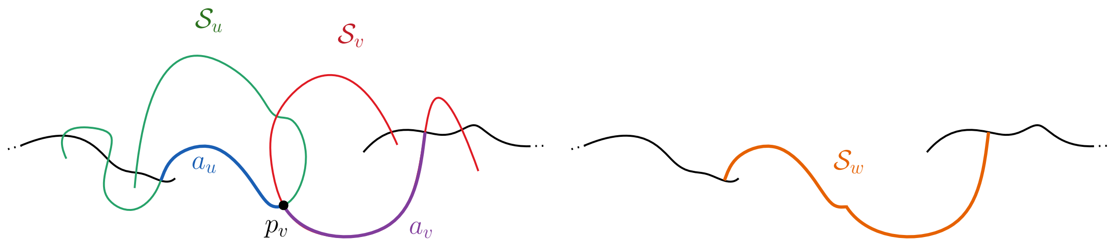

FIGURE 5. Illustration of the construction used in the proof of Lemma 5.5 for contracting an edge between adjacent degree-two vertices without a common neighbor.

Theorem 5.1 ([Kes20]). *Given $\mathcal{F}$ a family of pseudo-disks and $\mathcal{B}$ a finite family of pseudo-disks, consider the hypergraph $H(\mathcal{B},\mathcal{F})$ whose vertices are the pseudo-disks in $\mathcal{B}$ and the edges are all subsets of $\mathcal{B}$ of the form $\{D \in \mathcal{B},\ D \cap S \neq \emptyset\}$, with $S \in \mathcal{F}$. Then the number of edges of cardinality at most $k \geq 1$ in $H(\mathcal{B},\mathcal{F})$ is $\mathcal{O}(|\mathcal{B}|k^3)$.*

A very important aspect in this result is that it is not required that $\mathcal{F} \cup \mathcal{B}$ is a family of pseudo-disks, and thus pseudo-disks of $\mathcal{B}$ may "cross" pseudo disks of $\mathcal{F}$. This is indeed the case in our applications where in particular we associate to each tree in $G - M$ a pseudo-disk in $\mathcal{B}$, and this pseudo disk may cross pseudo-disks associated to vertices of $\mathcal{M}$ (see proof of Lemma 5.4).

5.1. **Application to pseudo-disk graphs.** In this section, we prove that the class of pseudo-disk graphs is nice, and so by our main theorem it admits a robust subexponential FPT algorithm for FVS. Note that for this graph class, the existence of such algorithm was revy recently given in [BBGR25].

Lemma 5.2. *There exists a constant c such that the class of pseudo-disk graphs has bounded tree neighborhood complexity with parameters $\alpha = 4,\ f_1(r) = c$ and $f_2(r,p,m) = cp^3$.*

Proof. TOPROVE 24 □

Lemma 5.3. *There exist constants c_1, c_2, c_3, c_4 such that the class of pseudo-disk graphs is a nice class with parameters $\alpha = 4,\ f_1(r) = c_1,\ f_2(r,p,m) = c_2p^3,\ \delta = \frac{1}{2},\ f_r = c_3\sqrt{r \log r}$ and $d_r = c_4 r \log r$.*

Proof. TOPROVE 25 □

And so applying our main theorem we obtain:

Corollary 1.7. *There exists $\eta < 1$, such that there is a robust parameterized subexponential algorithm solving FVS in time $2^{k^\eta} n^{\mathcal{O}(1)}$ for n-vertex pseudo-disk graphs.*

Note that our main theorem gives that it suffices to take $\eta > \frac{44}{45}$ for the result to hold. Sharper results already exist in the literature for pseudo-disk graphs, by generalizing the robust algorithm of [LPS+22] dealing with disk graphs to pseudo-disk graphs, or by using the representation as pseudo-disks for obtaining an even better running time. See [BBGR24b], the full version of [BBGR25], for both methods.

5.2. **Application to s-string graphs.** We now show how to adapt the arguments from the previous section to the case of s-strings.

Lemma 5.4. *There exist constants c_1, c_2 such that the class of s-string graphs has bounded tree neighborhood complexity with parameter $\alpha = 4, f_1(r) = c_1 s^4 r \log r$ and $f_2(r,p,m) = c_2(sr\log r)^4(p+m)^3$.*

Proof. TOPROVE 26 □

Lemma 5.5. *There exist constants c_1, c_2, c_3, c_4 such that for every $s \geq 1$ the class of s-string graphs is a nice class with parameters $\alpha = 4,\ f_1(r) = c_1 s^4 r \log r,\ f_2(r,p,m) = c_2(sr\log r)^4(p+m)^3,\ \delta = \frac{1}{2}$, $f_r = c_3\sqrt{r \log r}$ and $d_r = c_4 r \log r$.*

Proof. TOPROVE 27 □

And so applying our main theorem we obtain:

Corollary 1.6. *There exists $\eta < 1$, such that for all s there is a robust parameterized subexponential algorithm solving FVS in time $2^{\tilde{\mathcal{O}}\left(s^{\mathcal{O}(1)} k^\eta\right)} n^{\mathcal{O}(1)}$ for n-vertex s-string graphs.*

More precisely, we can take $\eta = \frac{52}{53}$.

6. Conclusion

In this paper we gave general sufficient conditions for the existence of subexponential parameterized algorithms for the Feedback Vertex Set problem. Our main theorem unifies the previously known results on several classes of graphs such as planar graphs, map graphs, unit-disk graphs, disk graphs, or more generally pseudo-disk graphs, and string graphs of bounded edge-degree. It also applies to classes where such algorithms were not known to exist, notably intersection graphs of thin objects such as segment graphs and more generally s-string graphs.

However, we have so far no evidence that our concept of nice class could be an answer to Question 1.1. So a natural direction for future research would be to investigate more general graph classes than those listed above or to discover new classes that fit in our framework. There are two natural candidates:

- Intersection graphs of objects in higher dimensions. However, as proved in [FLS18], intersection graphs of unit balls in $\mathbb{R}^3$ do not admit a subexponential parameterized algorithm for FVS, unless $P = NP$.
- Graph classes excluding an induced minor.[7] This is more general than string graphs (which exclude a subdivided K_5 as an induced minor). In such classes, Korhonen and Lokshtanov [KL24] recently provided an algorithm solving FVS in time $2^{\mathcal{O}_H(n^{2/3}\log n)}$, that is, subexponential in the input size n.

Less general than the previous item is the class of string graphs, which is nevertheless the most general class of intersection graphs of (arc-connected) objects in the Euclidean plane. At the time of the writing, it is not excluded that it could be a nice class. So far, we are still missing the property about the bounded tree neighborhood complexity. Note that in our proof for s-strings, the bound on the number s of crossing is only used to prove that the class satisfies this property of bounded tree neighborhood complexity. Obtaining a similar result without using the bound on the number of crossings would thus yield the desired generalization. But an even simpler way could be to prove that the general case of string graphs can be reduced to the case of s-strings graphs for some "small" s function of the problem parameter k. A first idea would be to prove that $K_{t,t}$-free string graphs are t^c-string graphs for some constant c, however this fails as there are string graphs with n vertices, and so which are $K_{n,n}$-free, that require 2^{cn} crossings for some constant c [KM91].

References

[ACO23] Shinwoo An, Kyungjin Cho, and Eunjin Oh. Faster algorithms for cycle hitting problems on disk graphs. In *Algorithms and Data Structures: 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 – August 2, 2023, Proceedings*, page 29–42, Berlin, Heidelberg, 2023. Springer-Verlag.

[AO21] Shinwoo An and Eunjin Oh. Feedback Vertex Set on Geometric Intersection Graphs. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, volume 212 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:12, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[BBF99] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[BBGR24a] Gaétan Berthe, Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond. Subexponential Algorithms in Geometric Graphs via the Subquadratic Grid Minor Property: The Role of Local Radius. In Hans L. Bodlaender, editor, *19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024)*, volume 294 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[BBGR24b] Gaétan Berthe, Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond. Feedback vertex set for pseudo-disk graphs in subexponential fpt time, 2024.

[BBGR25] Gaétan Berthe, Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond. Feedback Vertex Set for pseudo-disk graphs in subexponential FPT time. In Daniel Kráľ and Martin Milanič, editors, *Graph-Theoretic Concepts in Computer Science*, pages 65–78, Cham, 2025. Springer Nature Switzerland.

[BFLP24] Édouard Bonnet, Florent Foucaud, Tuomo Lehtilä, and Aline Parreau. Neighbourhood complexity of graphs of bounded twin-width. *European Journal of Combinatorics*, 115:103772, 2024.

[BG96] Ann Becker and Dan Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83(1):167–188, 1996.

[BR19] Édouard Bonnet and Paweł Rzążewski. Optimality program in segment and string graphs. *Algorithmica*, 81:3047–3073, 2019.

[BR24] Edouard Bonnet and Paweł Rzążewski. An 11/6-approximation algorithm for vertex cover on string graphs, 2024. https://arxiv.org/abs/2409.18820, to appear in SOCG 2025.

[BT22] Julien Baste and Dimitrios M Thilikos. Contraction bidimensionality of geometric intersection graphs. *Algorithmica*, 84(2):510–531, 2022.

[7]A graph H is said to be an *induced minor* of a graph G if it can be obtained from G by deleting vertices and contracting edges. Otherwise, G is said to *exclude* H as induced minor or to be *H-induced minor free*.

[CFK$^+$15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer Publishing Company, Incorporated, 1st edition, 2015.

[dBBKB$^+$20] Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for exponential-time-hypothesis–tight algorithms and lower bounds in geometric intersection graphs. *SIAM Journal on Computing*, 49(6):1291–1331, 2020.

[Die05] Reinhard Diestel. Graph theory 3rd ed. *Graduate texts in mathematics*, 173(33):12, 2005.

[DN19] Zdeněk Dvořák and Sergey Norin. Treewidth of graphs with balanced separations. *Journal of Combinatorial Theory, Series B*, 137:137–144, 2019.

[FLS18] Fedor V Fomin, Daniel Lokshtanov, and Saket Saurabh. Excluded grid minors and efficient polynomial-time approximation schemes. *Journal of the ACM (JACM)*, 65(2):1–44, 2018.

[Kes20] Balázs Keszegh. Coloring intersection hypergraphs of pseudo-disks. *Discret. Comput. Geom.*, 64(3):942–964, 2020.

[KL24] Tuukka Korhonen and Daniel Lokshtanov. Induced-minor-free graphs: Separator theorem, subexponential algorithms, and improved hardness of recognition. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2024.

[KM91] Jan Kratochvíl and Jiří Matoušek. String graphs requiring exponential representations. *Journal of Combinatorial Theory, Series B*, 53(1):1–4, 1991.

[KM94] Jan Kratochvíl and Jirí Matousek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994.

[Kor22] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–192. IEEE, 2022.

[Lee17] James R. Lee. Separators in Region Intersection Graphs. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:8, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[LPS$^+$22] Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. Subexponential parameterized algorithms on disk graphs (extended abstract). In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2031. SIAM, 2022.

[LPS$^+$23] Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. A framework for approximation schemes on disk graphs. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2228–2241. SIAM, 2023.

[Mat14a] Jirí Matousek. Near-optimal separators in string graphs. *Comb. Probab. Comput.*, 23(1):135–139, 2014.

[Mat14b] Jiří Matoušek. String graphs and separators. In *Geometry, structure and randomness in combinatorics*, pages 61–97. Springer, 2014.

[RVS19] Felix Reidl, Fernando Sánchez Villaamil, and Konstantinos Stavropoulos. Characterising bounded expansion by neighbourhood complexity. *European Journal of Combinatorics*, 75:152–168, 2019.

(G. Berthe, M. Bougeret, D. Gonçalves) LIRMM, Université de Montpellier, CNRS, Montpellier, France.
Email address: `firstname.lastname@lirmm.fr`

(J.-F. Raymond) CNRS, LIP, ENS de Lyon, France.
Email address: `jean-florent.raymond@cnrs.fr`