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 $X \subseteq V(G)$, $|\{N(v) \cap X, v \in V(G)\}| \leq c|X|$. 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 $X \subseteq V(G)$, $|\{N(v) \cap X, v \in V(G)\}| \leq r^{\mathcal{O}(1)}|X|$. 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) $|\{N_A(T),\ T \in \mathcal{T}\}| \leq f_1(r)|A|^\alpha$, where $N_A(T)$ denotes the neighbors of the vertices of T in A, and
(2) $|\{N_A(T),\ T \in \mathcal{T}\}| \leq f_2(r,p,m)|A|$, 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 $G \in \mathcal{G}$, $|E(G)| \leq d_r \cdot |V(G)|$. Without loss of generality we will assume $d_r \geq r$.

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).

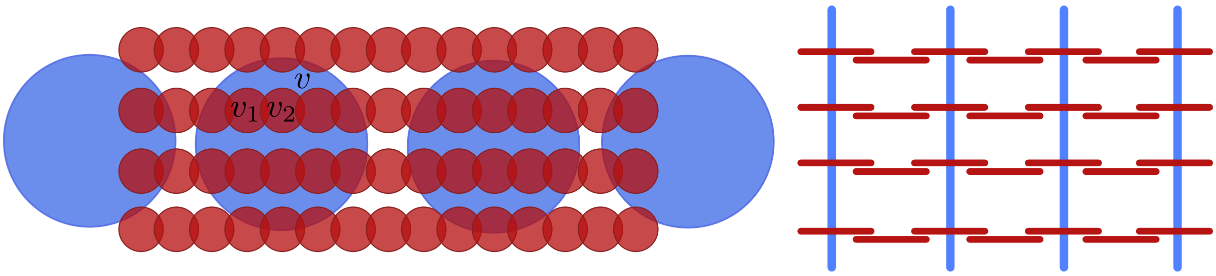

Figure 1. Example of a $K_{r,r}$ contained as a minor for $r = 4$ in a disk graph (left) and a 2-DIR graph (right). In the case of disk graph, v has a matching of size $r - 2$ in its neighborhood, forming a triangle bundle, which can be exploited to branch. The set M are depicted in blue. For the 2-DIR 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, $|P| \leq \mathsf{poly}(r)(d_M(P)^{\alpha+1})$. 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 $|\{N_M(v),\ v \in \mathcal{T}\}| \leq f_1(r)(d_M(P))^\alpha$. Thus, if $|P| \geq x \cdot f_1(r)(d_M(P))^\alpha$, 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 $d_M(P_i^1) \geq t$, or that there is no more vertices in P_i. If $d_M(P_i^1) \geq t$, 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 $d_M(P_i^\ell) \geq t$ 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 $d_M(P_i^\ell) \leq 2t$ 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 $|P_i^\ell| \leq \mathsf{poly}(r)$ for any $\ell \in [1, x(P_i)]$. Let us denote $\mathcal{T}^+$ the set of P_i^ℓ such that $d_M(P_i^\ell) \geq t$ 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.

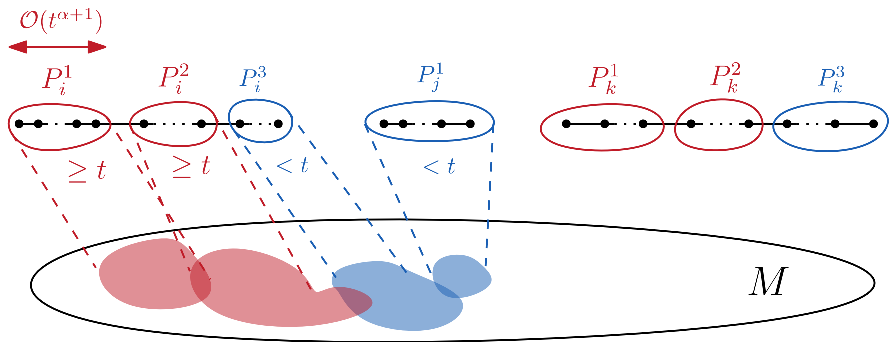

FIGURE 2. Example of partition where P_i is partitioned into $x(P_i) = 3$ subpaths, with P_i^1 and P_i^2 in $\mathcal{T}^+$ and $P_i^3 \in \mathcal{T}^-$.

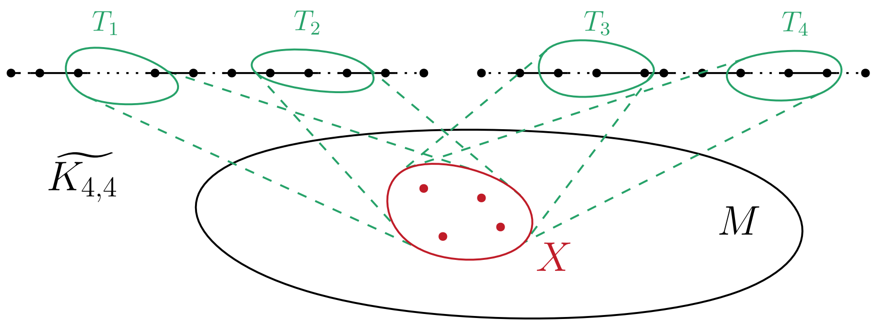

FIGURE 3. Example of $\widetilde{K_{t,t}}$ for $t = 4$. Here we have $X \subseteq N_M(T_i)$ for $1 \leq i \leq 4$.

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 $|V(G) \setminus M| \le \mathsf{poly}(r)|M| \le \mathsf{poly}(k^\varepsilon)k = \mathcal{O}(k^{2-\varepsilon'})$ 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 $G-M$, $|T| \le \mathsf{poly}(r)\mu(T)^{\mathcal{O}(\alpha)}$ 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 $d_M(u) \le t$), 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 $d_M(v) \le t-2$, 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 $d_M(v) \le t$ 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 3.34) 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 $T \in \mathcal{T}$, $d_M(T) \le 2t$ and $|T| \le \mathsf{poly}(r)\mu(T)^{\mathcal{O}(\alpha)}$ (see Definition 3.24 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 $|T| \le \mathsf{poly}(r)\mu(T)^{\mathcal{O}(\alpha)}$ 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 $|\mathcal{T}^+| \le \mathsf{poly}(r)|M|$.

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

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 $|M_0| \le 2k_0$) 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 ($\mathrm{BR}_1$) or correctly concludes that G is $K_{r,r}$-free.*

Proof. As $G - M$ is acyclic, for any copy of $K_{r,r}$ with parts A and B, one part of this copy is almost fully contained in M. More formally and without loss of generality we can assume that $|A \cap M| \geq r-1$. We can thus consider all the r vertices subsets of $V(G)$ consisting of $r-1$ vertices of M and a vertex of $V(G)$ to form the vertex set A. Then we can consider each vertex v of $V(G) \setminus A$ and check if $A \subseteq N(v)$. If there are r such vertices, then we have a copy of $K_{r,r}$. It is then trivial to produce the $2r$ instances of (BR_1). All this takes $\mathcal{O}(|M|^r)|V(G)|^{\mathcal{O}(1)} = 2^{\mathcal{O}(r \log |M|)}|V(G)|^{\mathcal{O}(1)}$ steps. □

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. The first item follows from the fact that if rule (BR_1) applies to a triplet (G, k, M), and replaces it with a set of $2r$ instances, we have the property that (G, k, M) is a YES-instance if and only if (at least) one of the $2r$ instances is. In order to show this let us call A and B the vertex sets of the parts of the considered $K_{r,r}$-subgraph.

If (G, k, M) is a YES-instance, let S be a feedback vertex set of G with $|S| \leq k$. Any 2 vertices of A induce a 4-cycle with any 2 vertices of B, so in order to intersect all cycles S intersects one of A and B on at least $r-1$ vertices. Suppose without loss of generality that S intersects A on (at least) $r-1$ vertices, let $a \in A$ be the r^{th} vertex of A (i.e. a is a vertex such that $A \setminus \{a\} \subseteq S$, but possibly $a \in S$), and let $X = A \setminus \{a\}$. Let (G', k', M') be the instance, among the $2r$ ones generated by rule (BR_1), such that $G' = G - X$, and $M' = M \setminus X$. Note that taking $S' = S \setminus X$, we have that $G - S = G' - S'$ and $|S'| = |S| - (r-1) \leq k - (r-1) = k'$. Hence, (G', k', M') is a YES-instance.

Conversely, if (G', k', M') is a YES-instance, with $G' = G - X$, and $M' = M \setminus X$, let S' be a feedback vertex set of G' with $|S'| \leq k' = k - (r-1)$. Note that taking $S = S' \cup X$, we have that $G' - S' = G - S$ and $|S| = |S'| + (r-1) \leq k' + (r-1) = k$. Hence, (G, k, M) is a YES-instance.

For the second item, if there was a copy of $K_{r,r}$ then the branching process would not be finished. Furthermore, each time rule (BR_1) is applied, the new instances are obtained by deleting vertices from the instance formerly in $\mathcal{I}$. Hence, all the generated instances are induced subgraph of the first one, G_0.

For the third item, this follows from the fact that each time rule (BR_1) is applied, the new sets M' are subsets of M, and thus subsets of M_0, which is of size at most $2k_0$.

For the last item, let us consider a recursive algorithm $\tilde{A}$ that given (G, k, M) as input, and using (BR_1), output all instances of $\mathcal{I}$ generated from (G, k, M) (implying that $\tilde{A}(G_0, k_0, M_0) = \mathcal{I}$). By Lemma 3.3, and by observing that for each call of the algorithm we have $|M| \leq 2k_0$, the worst case running time $f(n, k)$ of $\tilde{A}$ on a n-vertex graph and parameter k is such that $f(n, k) \leq 2^{\mathcal{O}(r \log k_0)} n^{\mathcal{O}(1)} + (2r) f(n, k - (r-1))$. This implies $f(n, k_0) \leq 2^{\mathcal{O}(r \log k_0)} n^{\mathcal{O}(1)} (2r)^{\frac{k_0}{r-1}}$. Moreover, as the number $g(k)$ of instances generated by a call to $\tilde{A}(G, k, M)$ is such that $g(k) \leq (2r) g(k - (r-1))$, we obtain the claimed bound on $|\mathcal{I}|$. □

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 3.27 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 3.26)
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 3.30, and $\tilde{M} = M \cup Z_1(\mathcal{T}) \cup Z_2(\mathcal{T})$
14: // By Lemma 3.31, 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 3.32 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. Notice that the vertices considered in the two rules belong to $G - M$, in particular they do not belong to any member of $\mathcal{H}$ (which are subsets of M). Contracting an edge with endpoints of degree two preserves being $K_{r,r}$-free and does not modify the size of the minimum feedback vertex sets. Moreover the obtained graph is still in $\mathcal{G}$ by definition of a nice class. The output is then an instance of $(r, \mathcal{G})$-ANN-FVS equivalent to the input one. The running time claim is immediate. □

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 3.34 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 3.34.

(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. For any $x \in X$, if there is at least one subtree $T \in \mathcal{T}$ such that $d_T(x) \geq 2$, we arbitrarily pick one of them and call it T_x. Now, let T' be one of the remaining trees in $\mathcal{T}$. Such a tree exists as $|\mathcal{T}| \geq |X| + 1$, and there are at most $|X|$ trees for the form T_x, for some $x \in X$. It is immediate from the definition that T' is redundant. □

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. The fact that the rule can be applied in polynomial time directly follows from Lemma 3.8. Also, as the graph G' obtained after applying the rule is an induced subgraph of $G \in \mathcal{G}$, we have $G' \in \mathcal{G}$.

Let us now show that the input and output instances are equivalent. As noted above G' is an induced subgraph of G so any feedback vertex set of G is also one for G'. Hence we only have to show that if G' has a feedback vertex set S', then G has a feedback vertex set of size $|S'|$ too. Consider such a set S' and let us transform it (if needed) into a feedback vertex set S of G according to the following cases.

First case:: $|X \setminus S'| \geq 2$. In this case we use the following claim.

Claim 3.10. *If $|X \setminus S'| \geq 2$, then S' intersects all the trees of $\mathcal{T}$ except T and at most one other.*

Proof. Let $x_1, x_2 \in X \setminus S'$ be distinct vertices, and suppose by contradiction that there are distinct $T_1, T_2 \in \mathcal{T} \setminus \{T\}$ such that S does not intersect T_1 nor T_2. There exist $a_1 \in N_{T_1}(x_1)$ and $b_1 \in N_{T_1}(x_2)$ as $N_M(T_1) = X$, and because T_1 is connected there exists a path from a_1 to b_1 in T_1 (observe that we may have $a_1 = b_1$). Similarly, we define a_2, b_2 and a path joining them in T_2. Then we have a cycle not hit by S', a contradiction. □

So if $|X \setminus S'| \geq 2$, there are at least $|X|$ vertices of S' in the trees of $\mathcal{T}$. Let denote Z this set. The set $S = S' \setminus Z \cup X$ is then a feedback vertex set of G with $|S| \leq |S'|$ as wanted.

Second case:: there is a unique vertex $x \in X \setminus S'$. Then either S' is a feedback vertex set of G (and so we can take $S = S'$) or there is a cycle in $G - S'$, in which case we consider the following two subcases.
- If $d_T(x) \geq 2$, let $T' \in \mathcal{T}$ with $T' \neq T$ and $d_{T'}(x) \geq 2$ (it exists as T is redundant). Then there is a cycle in $G[T' \cup \{x\}]$, which is hit by the feedback vertex set S' as it is a cycle in G' too. So there is a vertex $v \in T' \cap S'$, and taking $S = (S' \setminus \{v\}) \cup \{x\}$ is a feedback vertex set of G with $|S| \leq |S'|$.
- Otherwise, the graph $G[T \cup \{x\}]$ has no cycle. The cycle C of $G - S'$ necessarily uses some vertices of T and we have $N_M(T) \setminus S' = \{x\}$ and $G[T \cup \{x\}]$ without cycle. This configuration is not possible if T is a connected component of $G - M$, so we are in the case where there exists r a common parent for the roots of the trees in $\mathcal{T}$. The cycle C necessarily contains a path between x and r with inner vertices in T. Note that all the $|X|+1 \geq 2$ trees $T' \in \mathcal{T} \setminus \{T\}$ allow such path between x and r. So S' contains at least one vertex in some $T' \in \mathcal{T} \setminus \{T\}$ and by taking v one such vertex we can take $S = (S' \setminus \{v\}) \cup \{x\}$ and reach the same conclusion as above.

Third case:: in the remaining case where $X \subseteq S'$ we can take $S = S'$.

Hence the input and output instances are indeed equivalent. □

Definition 3.11. *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 3.12. *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. We first compute for each connected component T of $G - M$ their neighborhood in M (i.e., $N_M(T)$). If there is a set X such that the set $\mathcal{T}_X$ of connected components T in G_M such that $N_M(T) = X$ is large enough, that is $|\mathcal{T}_X| \geq |X| + 2$, we are done. Finding a family $\mathcal{T}$ for the second version of the rule, where we do not consider connected components of $G - M$ anymore but trees under a common parent r, can be done in a similar way that the first case, testing any $r \in V(G - M)$ as the potential common parent. These operations can be performed in polynomial time. □

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 3.13. *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. The results are obtained from the definition of Definition 1.4 and using the pigeonhole principle. □

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 3.14. *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. Remember that we chose an arbitrary root for T, and T_v denote the subtree of T with root v. For bounding the number of neighbors of v in T, it suffice to bound the number of children of v. For this, we first use that, by definition, the number of children u of v such that $V(T_u) \cap \partial_{\overline{M}}(T) \neq \emptyset$ is bounded by $b_{\overline{M}}(T)$. Let N be the set of remaining children, meaning the set of children u such that $T_u \cap \partial_{\overline{M}}(T) = \emptyset$. Applying Lemma 3.13, with the family $(T_u)_{u \in N}$, the vertex set $N_M(T)$ and $x = d_M(T) + 2$, if $|N| > (d_M(T) + 2) f_1(r) d_M(T)^\alpha$ then there would be a set $X \subseteq N_M(T)$ such that $d_M(T) + 2$ vertices $u \in N$ satisfy $d_M(T_u) = X$. Moreover $X \neq \emptyset$ as otherwise the rule (KR$_1$) would have applied. But then Rule (KR$_4$) would apply. Hence $|N| = \mathcal{O}(f_1(r) d_M(T)^{\alpha+1})$. □

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

Definition 3.15. *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 3.16. *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. Indeed, as Rules (KR$_1$) and (KR$_2$) do not apply, each vertex v of T belongs to (at least) one of the following types:

- $v \in \partial_{\overline{M}}(T)$,
- $d_M(v) \geq 1$,
- $d_T(v) \geq 3$, and
- $d_G(v) = 2$.

Observe that the leaves of T are either of the first type or the second. Let l be the number of leaves of T, there are $b_{\overline{M}}(T)$ vertices of the first type, and at most $d_M(T)$ vertices of the second type, so $l \leq b_{\overline{M}}(T) + d_M(T)$. Moreover the number of vertices of the third type is at most l. Let denote T_4 the vertices of the fourth type. Observe that the connected components of $G[T_4]$ are paths of size at most 3 (as otherwise (KR$_2$) would applies). Replacing such a connected component by an edge between the neighbors of the endpoints of the path would result in a tree with vertices $T \setminus T_4$ and whose number of edges (which is at most $|T \setminus T_4| - 1$) is an upper bound on the number of connected components of $G[T_4]$. So $|T_4| \leq 3|T \setminus T_4| \leq 6l$, and finally T has less than $8l = 8d_M(T) + 8b_{\overline{M}}(T)$ vertices in total. □

We now consider sharp subtrees. Notice that in such a tree, v has bounded degree (by Lemma 3.14), and the subtrees below v have bounded size (by Lemma 3.16). This leads to the following corollary.

Corollary 3.17. *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. The previous lemmas give

$$\begin{aligned}|T| &= O\left(\mu(T)\max(b_{\overline{M}}(T), f_1(r)d_M(T)^{\alpha+1})\right)\\ &= O\left(f_1(r)d_M(T)^{\alpha}\,\mu(T)^2\right).\end{aligned}$$

□

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 3.18. *The rule (KR$_5$) is safe.*

Proof. Again, as the graph G' obtained after applying the rule is an induced subgraph of G, the rule preserves the property of being a $K_{r,r}$-free s-string graph. For the safeness it is sufficient to prove that if G has a feedback vertex set of size at most k, then it has a solution S' of size at most k containing u. Let S be a feedback vertex set of size at most k, and suppose it does not contains u. Then S has at least a vertex in each of the $d_M(T) + b_{\overline{M}}(T)$ considered disjoints paths in T. So the set $S' = (S \setminus T) \cup N_M(T) \cup \partial_{\overline{M}}(T)$ satisfies $|S'| \leq |S| \leq k$ and contains u. Moreover this set is a feedback vertex set of G : a cycle in $G - S'$ would contains vertices of T as S is a feedback vertex set of G, but then, as T is a tree, such a cycle would need to intersect the set $N_G(T) = N_M(T) \cup \partial_{\overline{M}}(T) \subseteq S'$, which is a contradiction. □

Lemma 3.19. *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. Note that a sharp subtree of T contains a path of size at least 2 and whose endpoints are adjacent to a same vertex $u \in M$. So as rule (KR$_5$) does not apply, we have that T contains at most $d_M(T)(d_M(T) + b_{\overline{M}}(T))$ disjoint sharp subtrees.

Let $T^1 = T$ (and remember that the trees of $G - M$ are rooted). Then, iteratively let v_i be a lowest vertex of T^i such that $T^i_{v_i}$ is sharp, and let $T^{i+1} = T^i \setminus T^i_{v_i}$. This process stops when the remaining tree is weakly connected to M or empty. This leads to a partition of T into $c+1$ disjoint subtrees, where the first c are sharp subtrees, and the last one is weakly connected to M, with $c \leq d_M(T)(d_M(T)+b_{\overline{M}}(T)) = \mathcal{O}(\mu(T)^2)$. Using the bound on the size of the weakly connected and sharp trees from Lemma 3.16 and Corollary 3.17, we get:

$$\begin{aligned}|T| &= |T^{c+1}| + \sum_{1\leq i\leq c} |T^i_{v_i}|\\
&= \mathcal{O}\left(\mu(T) + \sum_{1\leq i\leq c} f_1(r)d_M(T^i_{v_i})^{\alpha}\,\mu(T^i_{v_i})^2\right) \quad \text{by Lemma 3.16 and Corollary 3.17}\\
&= \mathcal{O}\left(\mu(T) + f_1(r)d_M(T)^{\alpha} \sum_{1\leq i\leq c} \mu(T^i_{v_i})^2\right)\\
&= \mathcal{O}\left(\mu(T) + f_1(r)d_M(T)^{\alpha} \left(\sum_{1\leq i\leq c} \mu(T^i_{v_i})\right)^2\right).\end{aligned}$$

Note that each vertex of $\partial_{\overline{M}}(T)$ belongs to exactly one of these subtrees, and each subtree $T^i_{v_i}$ creates exactly one vertex in $\partial_{\overline{M}}(T^{i+1})$, and so will contribute to at most one of the $b_{\overline{M}}(T^j_{v_j})$ for $j > i$. Hence, $\sum_i b_{\overline{M}}(T^i_{v_i}) \leq b_{\overline{M}}(T) + c = \mathcal{O}(\mu(T)^2)$. Moreover observe that $\sum_i d_M(T^i_{v_i}) \leq c d_M(T) = \mathcal{O}(\mu(T)^3)$ so $\sum_i \mu(T^i_{v_i}) = \mathcal{O}(\mu(T)^3)$.

We then obtain $|T| = \mathcal{O}(f_1(r)\,\mu(T)^{6+\alpha})$. □

Corollary 3.20. *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. The fact that Rule (KR$_5$) applies is the contrapositive of Lemma 3.19, and the $d_M(T) + b_{\overline{M}}(T)$ paths needed to apply the rule can be found using the partition used in its proof, which can be computed in polynomial time. □

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 3.21. *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 3.22. 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 3.23. *The Rule (BR$_2$) is safe, and can be applied in polynomial time (assuming a $\widetilde{K_{t,t}}$ is provided).*

Proof. The obtained graph is an induced subgraph of the original graph, so the rule preserves the property of being $K_{r,r}$-free s-string graph. The running time bound is straightforward.

The fact that the input and output instances are equivalent comes from Remark 3.22. More formally, suppose that S is a solution to $(G, k, M, \mathcal{H})$. If $|S \cap X| \geq t - 1$, then there exists $v \in X$ such that $S \supseteq X_{\overline{v}}$, implying that $((G - X_{\overline{v}}), k - (t-1), M \setminus X_{\overline{v}}, \mathcal{H} - X_{\overline{v}})$ is a yes-instance. Otherwise ($|S \cap X| \leq t - 2$), there exists u, v two vertices in $X \setminus S$. As for any i, $G[T_i]$ is a tree and $X \subseteq N_M(T_i)$, there cannot be $i \neq j$ such that $S \cap T_i = S \cap T_j = \emptyset$, as otherwise $G[\{u, v\} \cup T_i \cup T_j]$ would contain a cycle not hit by S. This implies that there exists i such that all trees of $\mathcal{T}_{R_i}$ are hit by S. As S must also hit any tree in $\mathcal{H}$ (which are disjoint), it implies that $|\mathcal{H}| + t - 1 \leq k$, and thus the rule generates in particular $(G, k, M \cup R_i, \mathcal{H} \cup \mathcal{T}_{R_i})$, which is a YES-instance. The reverse direction of safeness is immediate.

□

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.

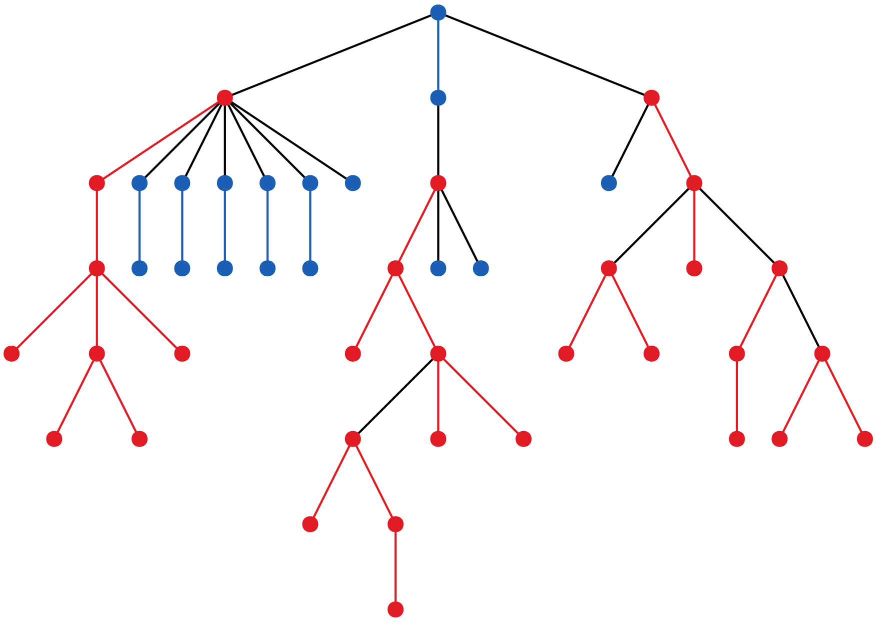

FIGURE 4. Representation of a t-uniform partition of a subtree T of $G - M$. Here are represented only vertices of F (and not vertices of M). Vertices of the trees in $\mathcal{T}^+$ (respectively $\mathcal{T}^-$) are represented in red (respectively blue), so as the edges between two vertices in the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

Definition 3.24. *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 3.19) 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 3.25. 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 3.27 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 3.26. *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 $p_3(r, t) = 4t f_2(r, 2t, p_2(r, t))$. Then, we can find a $\widetilde{K_{t,t}}$ in polynomial time.*

Proof. Consider $F_{\mathcal{T}^+}$ the forest obtained from $G - M$ by contracting each subtree in $\mathcal{T}$ to a single vertex, and removing the vertices of the trees in $\mathcal{T}^-$. Denoting V_s the set of vertices of $F_{\mathcal{T}^+}$ with degree at most 2, we have $|V_s| \geq |\mathcal{T}^+|/2$ as the number of leaves in a tree is an upper bound on the number of vertices of degree bigger than 3. Then using a 2-coloration of the forest we partition the vertices of $F_{\mathcal{T}^+}$ in two independent sets, V_1 and V_2. Without loss of generality we suppose that $|V_s \cap V_1| \geq |V_s \cap V_2|$. So denoting $\mathcal{T}'$ the set of trees in $\mathcal{T}^+$ associated to the vertices in $V_s \cap V_1$, we have $|\mathcal{T}'| \geq |V_s|/2 \geq |\mathcal{T}^+|/4$. Now for $T \in \mathcal{T}'$ we want to bound the size of its border. Firstly, for the trees in $\mathcal{T}^+$ adjacent to T, there are at most two of them by construction of $\mathcal{T}'$. Secondly, by item 3 of Definition 3.24 the only vertex in T which can be adjacent to a tree $T' \in \mathcal{T}^-$ is the root of T. So overall, the border of T contains at most 3 vertices, and so $b_{\overline{M}}(T) \leq 3$. Moreover, as V_1 is an independent set, there is no edge between two trees of $\mathcal{T}'$. By Definition 3.24, any tree of $T \in \mathcal{T}'$ has size at most $p_1(r, \mu(T))$, but we have $\mu(T) = \max(d_M(T), b_{\overline{M}}(T)) = d_M(T) \leq 2t$ so $|T| \leq p_1(r, 2t) = p_2(r, t)$. Let us now apply the second result of Lemma 3.13 with $x = t$, $A = M$, $p = 2t$ and $m = p_2(r, t)$. As

$|\mathcal{T}'| \geq |\mathcal{T}^+|/4 = tf_2(r, 2t, p_2(r,t))|M|$, there exists a subset $X \subseteq M$ and a subset $\mathcal{T}'_X \subseteq \mathcal{T}'$ such that $|\mathcal{T}'_X| = t$, and for all $T \in \mathcal{T}'_X$, we have $N_M(T) = X$. Observe that as $\mathcal{T}' \subseteq \mathcal{T}^+$, we have $|X| \geq t$, so we can take any subset X_0 of X such that $|X_0| = t$ and then $(X_0, \mathcal{T}'_X)$ is a $\widetilde{K_{t,t}}$, as wanted.

(Running time) The set $\mathcal{T}'$ can be computed in polynomial time, and finding the subset X and $\mathcal{T}'_X$ whose existence is proved by Lemma 3.13 can again be done in polynomial time by listing the neighborhoods in M of the trees in $\mathcal{T}'$. □

Lemma 3.27. *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 3.26).*

Proof. We try to construct a partition of $G - M$ fulfilling item 3, and if we fail we prove that we fall into case of item 1 or item 2. We denote $p = p_3(r, s, t)$ in this proof.

The construction proceeds from the deepest leaves to the roots of the trees of $G - M$. For this we define L_0 as the set of roots of the connected components of $G - M$, and more generally L_i the set of vertices of depth i in the trees of the forest $G - M$. Let h be the depth of the deepest leaves of the highest trees of $G - M$, and $F_i = \bigcup_{j=i}^{h} L_j$ the set of vertices with depth at least i. Observe that we have $F_0 = V(G - M)$.

In a first time we construct $\mathcal{T}_h$ a t-uniform partition of F_h.

Observe that two vertices of the same depth cannot share an edge, so we have $G[F_h]$ an edgeless graph. For constructing the partition we then have no choice but to take each connected component (containing only one vertex) as a tree in $\mathcal{T}$, and more precisely in $\mathcal{T}_h^-$ as for $v \in F_h$ we have $d_M(v) < t$ by hypothesis, and so $\mathcal{T}_h^+ = \emptyset$. $\mathcal{T}_h$ is then a t-uniform partition as there is no edge between two trees of $\mathcal{T}_h$, and each tree have degree in M at most $2t$ as wanted (even t as said above).

Now suppose we are given $\mathcal{T}_i$ a t-uniform partition of F_i with $i \geq 1$, such that $|\mathcal{T}_i^+| < p$. We will now see how to extend this partition to a new partition $\mathcal{T}_{i-1}$ of F_{i-1}, or stop the induction and fall into case of item 1 or item 2.

Let $\mathcal{T}_{i-1} = \mathcal{T}_i$, $\mathcal{T}_{i-1}^+ = \mathcal{T}_i^+$ and $\mathcal{T}_{i-1}^- = \mathcal{T}_i^-$. Let us now modify $\mathcal{T}_{i-1}$ by considering each $v \in L_{i-1}$ as follows.

For any $v \in L_{i-1}$, let $v_1, \dots, v_l$ be its children, which are in the level L_i. Observe that by definition of a t-uniform partition of F_i, each of these vertices v_j are the root of some trees $T(v_j)$ in $\mathcal{T}_i$. Informally, we will try to group some of the $T(v_j) \in \mathcal{T}_i^-$ to create a tree in $\mathcal{T}_{i-1}^+$. Let $Y^-(v) = \{j \mid T(v_j) \in \mathcal{T}_i^-\}$. If $d_M(v \cup \bigcup_{j \in Y^-(v)} T(v_j)) < t$, we set $X(v) = Y^-(v)$. Otherwise, we define $X(v)$ as an inclusion-wise minimal set $Y'(v) \subseteq Y^-(v)$ such that $d_M\left(\{v\} \cup \bigcup_{j \in Y'(v)} T(v_j)\right) \geq t$. Let

$$T(v) = \{v\} \cup \bigcup_{j \in X(v)} T(v_j).$$

If $|T(v)| > p_1(r, \mu(T))$ we stop and output $T(v)$ (we call this "Case 1"). From Corollary 3.20 we get that (KR$_5$) applies as stated in item 1. Let us now assume that $|T(v)| \leq p_1(r, \mu(T))$.

If $d_M(T(v)) < t$, we remove from $\mathcal{T}_{i-1}^-$ all the $T(v_j)$ for $j \in X(v)$, and add $T(v)$ to $\mathcal{T}_{i-1}^-$. We consider v as *treated*, and proceed with the next vertex $v \in L_{i-1}$.

Otherwise ($d_M(T(v)) \geq t$), we also remove from $\mathcal{T}_{i-1}^-$ all the $T(v_j)$ for $j \in X(v)$, and we add $T(v)$ to $\mathcal{T}_{i-1}^+$. If $|\mathcal{T}_{i-1}^+| < p$, then we consider v as *treated*, and proceed with the next vertex $v \in L_{i-1}$. Otherwise, $|\mathcal{T}_{i-1}^+| = p$, then we stop and output $F = \bigcup_{T \in \mathcal{T}_{i-1}} T$ and $\mathcal{T}_{=p} = \mathcal{T}_{i-1}$ (we call this "Case 2", and we will prove that it matches the requirements of item 2).

This concludes the description of the algorithm that, given $\mathcal{T}_i$ a t-uniform partition of F_i with $i \geq 1$ such that $|\mathcal{T}_i^+| < p$, either stops in Case 1, stops in Case 2, or compute a set $\mathcal{T}_{i-1}$.

Claim 3.28. *If the algorithm stops in Case 2, then the output matches the requirements of item 2. Otherwise, when the algorithm has processed all the vertices $v \in L_{i-1}$, then $\mathcal{T}_{i-1}$ is a t-uniform partition of F_{i-1} with $|\mathcal{T}_{i-1}^+| < p$.*

Proof. Let us consider as input $\mathcal{T}_i$ a t-uniform partition of F_i with $i \geq 1$ such that $|\mathcal{T}_i^+| < p$, and consider an execution of the algorithm that consider vertices $v \in L_{i-1}$ in an arbitrary order.

Observe first that for any $v \in L_{i-1}$, we have $d_M(T(v)) < 2t$. Indeed, in particular when $d_M(T(v)) \geq t$, we cannot have $X(v) = \emptyset$ as then $d_M(T(v)) = d_M(v) \leq t$. We can thus take any $j \in X(v)$, and we have $d_M(T(v)) \leq d_M\left(\{v\} \cup \bigcup_{k \in X(v)\setminus\{j\}} T(v_k)\right) + d_M(T(v_j)) < 2t$ by minimality of $X(v)$, and as $T(v_j) \in \mathcal{T}^-_{i-1}$. This directly implies that at any time $\mathcal{T}_{i-1}$ is a t-uniform partition of the treated vertices.

Now, let us prove the following invariant called Π_1. At any point when considering all the $v \in L_{i-1}$ and constructing $\mathcal{T}_{i-1}$, $\mathcal{T}^+_{i-1}$, and $\mathcal{T}^-_{i-1}$:

- If a root r of a tree in $\mathcal{T}^-_{i-1}$ has a parent u in $G - M$, then either $r \in L_{i-1}$ and $u \in L_{i-2}$ (u is not treated), or $r \in L_i$ and $u \in L_{i-1}$ with u being either not treated yet, or the root of a tree in $\mathcal{T}^+_{i-1}$.
- $\mathcal{T}^-_{i-1}$ is a set of disjoint non-adjacent subtrees.

For the first property, let r be the root of a tree $T \in \mathcal{T}^-_{i-1}$ which has a parent u in $G - M$. If $r \in L_j$ for $j > i$, by hypothesis $\mathcal{T}_i$ is a t-uniform partition of F_i, then u is the root of a tree $T \in \mathcal{T}^+_i$, and $T \in \mathcal{T}^+_{i-1}$ as we never remove trees from $\mathcal{T}^+_i$. If $r \in L_{i-1}$, then $u \in L_{i-2}$ and we are done. Finally, assume that $r \in L_i$. If u is not treated then we are done. Otherwise, as u is treated and r is still the root of a tree $T \in \mathcal{T}^-_{i-1}$, it implies that $X(u) \subsetneq Y^-(u)$, and thus that $T(u)$ was added to $\mathcal{T}^+_{i-1}$, satisfying the required condition.

For the second property, first note that the vertices that are in a tree of $\mathcal{T}^-_{i-1}$ are either vertices of L_{i-1}, or these vertices were in a tree of $\mathcal{T}^-_i$. By induction there is no edge between two distinct trees in $\mathcal{T}^-_i$, we are thus left with the case of an edge uv with $u \in L_i$ and $v \in L_{i-1}$, u being the child of v. If u and v are in different trees of $\mathcal{T}_{i-1}$, it is either because $u \notin Y^-(v)$ or because $X(v) \subset Y^-(v) \setminus \{u\}$. In the former case, u belongs to a tree of $\mathcal{T}^+_i \subseteq \mathcal{T}^+_{i-1}$, and in the latter case that is $T(v)$ which belongs to $\mathcal{T}^+_{i-1}$.

Let us now finish the proof of Claim 3.28. If the algorithm stops in case 2, then observe that by definition F is downward-closed. Moreover, the output $\mathcal{T}_{i-1}$ is indeed a t-uniform partition of F. Indeed, Property of item 2 of Definition 3.24 holds as, by item 1 of property Π_1, if a root r of a tree in $\mathcal{T}^-_{i-1}$ has a parent u in $G - M$, then $u \in L_{i-2}$ (and thus does not belong to F), or $u \in L_{i-1}$ and u is not treated (so u does not belong to F), or u is the root of a tree in $\mathcal{T}^+_{i-1}$. Thus, in Case 2 we obtain $\mathcal{T}_{i-1}$, a t-uniform partition of F with $\mathcal{T}^+_{i-1} = p$. By Lemma 3.26, we can find in polynomial time a $\widetilde{K_{t,t}}$ and output it.

Finally, if the algorithm does not stop, it remains to prove that $\mathcal{T}_{i-1}$ is a t-uniform partition of F_{i-1} with $|\mathcal{T}^+_{i-1}| < p$. The inequality $|\mathcal{T}^+_{i-1}| < p$ holds as we did not fall into Case 2. Property of item 2 of Definition 3.24 holds because we proved that for any $v \in L_{i-1}$, $d_M(T(v)) < 2t$, and we did not fall into Case 1. Property of item 3 of Definition 3.24 holds as by item 1 of property Π_1, if a root r of a tree in $\mathcal{T}^-_{i-1}$ has a parent u in $G - M$, then $u \in L_{i-2}$ (and thus does not belong to F_{i-1}), or $u \in L_{i-1}$ and is not treated yet (which is not possible as all vertices of L_{i-1} are treated in this case), or u is the root of a tree in $\mathcal{T}^+_{i-1}$. ☐

Now that Claim 3.28 is proved, Lemma 3.27 directly holds by induction on i. ☐

Remark 3.29. Observe that in Line 4, all required conditions listed in Lemma 3.27 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 $v \in V(G) - M$.

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 3.30. *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 3.31. *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. We denote $Z_1 = Z_1(\mathcal{T})$ and $Z_2 = Z_2(\mathcal{T})$. The bound $|Z_1| \leq |\mathcal{T}^+|$ is trivial. To bound $|Z_2|$, let us define the following graph G_Z. The graph G_Z is obtained from $G - M$ by iteratively deleting leaves ℓ if $\ell \notin Z_1 \cup Z_2$, or contracting every edge zx with $z \in Z_1 \cup Z_2$ and $x \notin Z_1 \cup Z_2$. Clearly G_Z is a tree with vertex set $Z_1 \cup Z_2$, and where $d_{G_Z}(v) \geq 3$ for every $v \in Z_2$. Indeed, the three edge disjoint paths leaving v do not completely vanish while constructing G_Z. As the number of vertices of degree at least three in a tree is bounded by the number of leaves, we get $|Z_2| \leq |Z_1| \leq |\mathcal{T}^+|$.

The non applicability of Rule (KR$_1$) and Rule (KR$_2$) on $(G, k, \tilde{M}, \mathcal{H})$ immediately follows from the non applicability of Rule (KR$_1$) and Rule (KR$_2$) on $(G, k, M, \mathcal{H})$.

It then remains to bound $d_{\tilde{M}}(T)$ for T a connected component of $G - \tilde{M}$. Let T be a connected component of $G - \tilde{M}$. We have $T \subseteq T'$ for some $T' \in \mathcal{T}$, as each edge between trees of $\mathcal{T}$ contains a vertex from $Z_1(\mathcal{T}) \subseteq \tilde{M}$, and as $\mathcal{T}$ is a t-uniform partition, we get $d_M(T) \leq d_M(T') \leq 2t$. Moreover, notice that $d_{\tilde{M}}(T) \leq d_M(T) + |\{vz \in E(G) \mid v \in T \text{ and } z \in Z_1 \cup Z_2\}|$. Thus, assume by contradiction that $d_{\tilde{M}}(T) \geq 2t + 3$, and thus there are three vertices v_1, v_2, v_3 in T such that $N(v_i) \cap (Z_1 \cup Z_2) \neq \emptyset$. Let v be the vertex of T such that the three edges v_1z_1, v_2z_2, and v_3z_3, where the vertices z_1, z_2, z_3 are distinct (but not necessarily the vertices v_1, v_2, v_3). The vertex v at the intersection of the three paths among v_1, v_2, v_3 should be in Z_2, a contradiction. □

Lemma 3.32. *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. Let $\mathcal{MC}$ be the set of connected components of $G - \tilde{M}$. Let $T \in \mathcal{MC}$. As Rule (KR$_1$), (KR$_2$), and (KR$_4$) do not apply, and Rule (KR$_5$) does not apply on T, by Corollary 3.20 we get $|T| \leq p_1(r, \mu(T)) \leq p_1(r, 2t + 2)$. Now, as Rule (KR$_4$) does not apply in particular with $\mathcal{MC}$, by the second result of Lemma 3.13 with $A = \tilde{M}$, $x = 2t + 4$, $p = 2t + 2$ and $m = p_1(r, 2t + 2)$, we get that if $|\mathcal{MC}| > xf_2(r, p, m)|\tilde{M}|$, then we would have x connected components of $\mathcal{MC}$ having the same neighborhood X in $\tilde{M}$. Then we would have $X \neq \emptyset$ as otherwise Rule (KR$_1$) would applied, and as $|X| \leq p = 2t + 2$ and $x \geq |X| + 2$, (KR$_4$) could be applied, a contradiction with the hypothesis. Thus, we get $|G - \tilde{M}| \leq (2t + 4)f_2(r, 2t + 2, p_1(r, 2t + 2))|\tilde{M}|$. □

Lemma 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 $|G - M| = \mathcal{O}(p_3(r, t)p_4(r, t)|M|)$.*

Proof. As $A(G, k, M, \mathcal{H})$ reaches Line 22, it implies that `Partitionning-Algorithm` output a t-uniform partition of $G - M$ (where $t = 2d_r$ and $|\mathcal{T}^+| \leq p_3(r, s, t)|M|$). Let $\tilde{M} = M \cup Z_1(\mathcal{T}) \cup T_2(\mathcal{T})$. By Lemma 3.31, Rule (KR$_1$) and Rule (KR$_1$) do not apply on $(G, k, \tilde{M}, \mathcal{H})$. As we did not apply Rule (KR$_4$) (Line 15) or Rule (KR$_5$) (Line 18), and as by Lemma 3.31, $d_{\tilde{M}}(T) \leq 2t + 2$ for any connected component T of $G - \tilde{M}$, we can apply Lemma 3.32 to obtain $|G - \tilde{M}| \leq p_4(r, t)|\tilde{M}|$. As $|G - M| = |G - \tilde{M}| + |Z_1(\mathcal{T})| + |Z_2(\mathcal{T})|$, we get $|G - M| = \mathcal{O}\left(p_4(r, t)(|M| + |Z_1(\mathcal{T})| + |Z_2(\mathcal{T})|)\right)$. By Lemma 3.31 we have $|Z_1(\mathcal{T})| + |Z_2(\mathcal{T})| \leq 2|\mathcal{T}^+|$ and $|\mathcal{T}^+| \leq p_3(r, s, t)|M|$, we get $|G - M| = \mathcal{O}(p_3(r, t)p_4(r, t)|M|)$. □

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 $|A_2| \leq k_0p_2(r, t)$. 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 3.34. *For any input $(G, k, M, \mathcal{H})$ of Algorithm 1, we have $|M| = \mathcal{O}(k_0p_5(r, t))$ with $p_5(r, t) = 4 + 2p_2(r, t)$.*

Proof. For proving the result, we need to define some invariant to our problem. For stating this invariant we need to consider additional annotations to our problem. We now consider instances $(G, k, M, \mathcal{H}, A_0, A_1, A_2, X, C, G')$. Informally A_0 will denote the vertices of $M \cap M_0$, A_1 the vertices of M added by (KR$_3$), A_2 for the ones added to M by rule (BR$_2$), C the set of vertices obtained by contracting at least one edge by (KR$_2$), X the vertices removed by application of (KR$_5$) (so X is a set of vertices outside of $V(G)$), and G' is a supergraph of G with $V(G') = (V(G) \cup X)$ that will be useful to "remember" the links between the vertices of $V(G)$ and X. More formally, initially for $(G, M, k, \emptyset) \in \mathcal{I}$ the instance is such that $A_1 = M$, $A_2 = A_3 = X = C = \emptyset$ and $G' = G$. The generated instance update the additional annotations in the following manner:

- When removing vertices from $V(G - M)$, so for (KR$_1$) and (KR$_4$), remove the vertices from G' as well and from the sets containing them.
- If we apply (KR$_2$), denoting uv the contracted edge in G and w the vertex added to G for replacing u and v, we do the same operation in G', with an edge between a vertex $q \in V(G) \cup X$ and w if and only if there was an edge between q and u, q and v, or both. Moreover we add w to the set C.
- If this is (KR$_3$) which is applied, the vertex v added to M is added to A_1 as well.
- For (KR$_5$), the vertex $v \in M$ removed from G is now added to X, and removed from the sets A_i in which it is. G' is not modified.
- For (BR$_2$), in the t branches where we add vertices to M, we add them to A_2 as well now. For the t branches where $t-1$ vertices are removed from M, again there are now added to X, and removed from the sets A_i which contain them.

We now state some invariants on those sets:

Claim 3.35.
(1) M is partitioned in A_0, A_1, A_2.
(2) $G'[(V(G) \cup X) \setminus C]$ is a $K_{r,r}$-free graph in $\mathcal{G}$.
(3) For $v \in V(G)$, if $v \in C$ or $N_{G'}(v) \cap C \neq \emptyset$ then $d_G(v) \leq 2$.
(4) $C \cap A_1 = \emptyset$.
(5) For $v \in A_1$, $|(N_{G'}(v) \cap (M \cup X)) \setminus C| \geq 2d_r$.

Proof. The first claim is trivial to verify. For the second, observe that the only modification to this graph is the deletion of vertices, so the result follows by hereditary of $\mathcal{G}$. For each call the degree of a vertex does not increases in G at any time (but it can in G'), so it suffice to prove the result when the vertex was just added. If a vertex $v \in V(G)$ is added to C or one of its neighbor it implies that $d_G(v) \leq 2$. Finally, for the last claim, this property is true when a vertex is added to A_2 and there is nor contraction of edges or deletion of vertices in $G'[M \cup X]$, so the inequality is kept. The fourth result can be proved by observing that an element of the intersection would have first been in C and then A_1. Adding an element in A_1 requires that it has $2d_r > 2$ neighbors in G, but this would contradict the previous bullet. Finally the last inequality is obtained by observing that the considered set stays the same once v is added to A_1. And when v is added to A_1 we have $d_M(v) \geq 2d_r > 2$, so $N_{G'}(v) \cap C = \emptyset$ and so $N_{G'}(v) \subseteq (M \cup X) \setminus C$ ☐

Now following the same reasoning than in the sketch but this time using the graph $G'[(V(G) \cup X) \setminus C]$, we obtain $|A_0| \leq 2k_0$, $|A_2| \leq k_0 p_2(r,t)$ and $2d_r|A_1| \leq d_r(|A_0|+|A_1|+|A_2|)$ so $|M| \leq 4k_0 + 2k_0 p_2(r,t)$. ☐

Using Lemma 3.33 and Lemma 3.34, the following corollary is now immediate.

Corollary 3.36. *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. For an instance $(G, M, k, \mathcal{H})$ with G a graph on n vertices, computing a tree decomposition with bags of size $\mathsf{tw}(G)$ up to a constant factor can be done in time $2^{\mathcal{O}(\mathsf{tw}(G))} n$ [Kor22]. Given such a tree decomposition observe that a bag of size b can intersect at most b distinct $H \in \mathcal{H}$. Moreover for $H \in \mathcal{H}$, having that H is connected ensures that the set of bags of the tree decomposition of G containing at least one vertex of H is a connected subset. Using those two observations it is easy to adapt the standard dynamic programming algorithm solving FVS for solving the more general $(r, \mathcal{G})$-ANN-FVS without worsening the running time, and obtaining the claimed complexity. ☐

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. Denoting $n = |V(G)|$, according to Corollary 3.36, we have $n = \mathcal{O}(k_0 p_6(r,t))$, and as $\mathcal{G}$ is a nice class (see Definition 1.4), we have $\mathsf{tw}(G) \leq f_r n^\delta$. Thus, Theorem 4.1 implies that $DP(G,k,M,\mathcal{H})$ runs in time $(f_r n)^{\mathcal{O}\left(f_r n^\delta\right)} n^{\mathcal{O}(1)}$. □

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. Let $f(n,k,m,h)$ be the worst case complexity of $A(G,k,M,\mathcal{H})$ when $|V(G)| = n$, $|M| = m$, and $\mathcal{H} = h$. We denote $C_{DP} = T_{DP}\left(k_0,r,t\right)$. Let $p(n)$ be the worst case complexity of all operations performed in one call of Algorithm 1 (which is bounded by the sum of the complexity of testing and applying all the kernelization and branching rules, and applying Subsection 3.7). Notice that, as all rules and Subsection 3.7 are polynomial, p is polynomial. Let us show by induction (on $2(n+k)-m-h$) that

$$f(n,k,m,h) \leq (p(n) + C_{DP})(2n-m)\left((2t)^{\frac{2k-h}{t-1}} - 1\right).$$

Without loss of generality, we assume that p is non-decreasing. Let us bound f according to which recursive call we perform. In particular, if we apply (KR$_5$), we delete one vertex v from M, and as we remove from the packing $\mathcal{H}$ the (at most one) hyperedge containing v, we decrease h by $x \in \{0,1\}$. If we apply (BR$_2$), then we make t recursive calls on instances $((G - X_{\overline{v}}), k-(t-1), M \setminus X_{\overline{v}}, \mathcal{H} - X_{\overline{v}})$ (re-using notations of (BR$_2$)), and thus in these calls we remove $x \in [t-1]$ hyperedges from $\mathcal{H}$, and in the worst case also t recursive calls on instances $(G,k,M \cup R_i, \mathcal{H} \cup \mathcal{T}_{R_i})$, where R_i is the union of $t-1$ parts of a $\widetilde{K_{t,t}}$, each part having size at most $p_2(r,t)$ by definition of a $\widetilde{K_{t,t}}$. This leads to the following bound:

$$f(n,k,m,h) \leq \begin{cases} p(n) + f(n-x,k,m,h), & \text{for } x \geq 1 \text{ when (KR}_1\text{), (KR}_2\text{) or (KR}_4\text{)} \\ p(n) + f(n,k,m+1,h), & \text{when (KR}_3\text{)} \\ p(n) + f(n-1,k-1,m-1,h-x), & \text{for } x \in \{0,1\} \text{ when (KR}_5\text{)} \\ p(n) + tf(n-(t-1),k-(t-1),m-(t-1),h-x) + & \\ tf(n,k,m+(t-1)p_2(r,t),h+(t-1)) & \text{for } 0 \leq x \leq t-1 \text{ when (BR}_2\text{)} \\ p(n) + C_{DP} & \text{if we don't apply any rule and call the DP} \end{cases}$$

It remains to check by induction that our bound hold in any of these cases. The first two and last cases are straightforward. For the third case, we get

$$\begin{aligned} f(n,k,m,h) &\leq p(n) + f(n-1,k-1,m-1,h-x) \\ &\leq p(n) + (p(n-1) + C_{DP})(2n-m-1)\left((2t)^{\frac{2k-h-1}{t-1}} - 1\right) \\ &\leq p(n) + (p(n) + C_{DP})(2n-m-1)\left((2t)^{\frac{2k-h-1}{t-1}} - 1\right) \\ &\leq (p(n) + C_{DP})(2n-m)\left((2t)^{\frac{2k-h}{t-1}} - 1\right) \end{aligned}$$

For the fourth case, let us first bound the terms $z_1 = f(n-(t-1), k-(t-1), m-(t-1), h-x)$ and $z_2 = f(n,k,m+(t-1)p_2(r,t), h+(t-1))$ by

$$\begin{aligned} z_1 &\leq (p(n-(t-1)) + C_{DP})(2n-m-(t-1))\left((2t)^{\frac{2k-h-2(t-1)+x}{t-1}} - 1\right) \\ &\leq (p(n) + C_{DP})(2n-m)\left((2t)^{\frac{2k-h}{t-1}-1} - 1\right) \\ z_2 &\leq (p(n) + C_{DP})(2n-m)\left((2t)^{\frac{2k-h}{t-1}-1} - 1\right) \end{aligned}$$

This leads to the following upper bound for the fourth case:

$$\begin{aligned} f(n,k,m,h) &\leq p(n) + t(z_1 + z_2) \\ &\leq p(n) + (p(n) + C_{DP})(2n-m)(2t)\left((2t)^{\frac{2k-h}{t-1}-1} - 1\right) \\ &\leq (p(n) + C_{DP})\,(2n-m)\left((2t)^{\frac{2k-h}{t-1}} - 1\right). \end{aligned}$$

This ends the induction. The wanted result is then easily obtained from the proven bound. □

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. Let denote $n_0 = |V(G_0)|$. According to Lemma 3.4, the complexity to generate the set $\mathcal{I}$ is in $2^{\mathcal{O}(r \log k_0)} n_0^{\mathcal{O}(1)} (2r)^{\frac{k_0}{r-1}}$ and $|\mathcal{I}| = \mathcal{O}\left((2r)^{\frac{k_0}{r-1}}\right)$. Deciding one instance $(G, k, M, \mathcal{H}) \in \mathcal{I}$ with G a n-vertex graph can be done in time $T_{DP}(k_0, r, t)(2t)^{\frac{2k_0}{t-1}} n_0^{\mathcal{O}(1)}$ where $T_{DP}(k_0, r, t) = 2^{\mathcal{O}\left(k_0^\delta p_7(r,t)\right)}$ according to Lemma 4.3, Corollary 4.2 and the observation that $k \leq k_0$ and $n \leq n_0$.

So the overall running time is bounded by:

$$\left(2^{\mathcal{O}\left(r \log k_0 + \frac{k_0 \log r}{r}\right)} + 2^{\mathcal{O}\left(\frac{k_0 \log r}{r} + \frac{k_0 \log t}{t} + k_0^\delta p_7(r,t)\right)}\right) n^{\mathcal{O}(1)}.$$

Now lets recall that t is defined in function of r as $t = 2d_r = r^{\mathcal{O}(1)}$ and so $p_7(r,t) = r^{\mathcal{O}(1)}$. More precisely if we have $f_1(r) = \tilde{\mathcal{O}}\left(r^{c_{f_1}}\right)$, $f_2(r,p,m) = \tilde{\mathcal{O}}(r^{c_{f_2}}(p+m)^{c'_{f_2}})$, $f_r = \tilde{\mathcal{O}}(r^{c_f})$ and $d_r = \tilde{\mathcal{O}}(r^{c_d})$ (remember that without loss of generality we supposed $c_d \geq 1$), then we have $p_7(r,t) = \tilde{\mathcal{O}}(r^{c_7})$ with

$$\begin{aligned} c_7 &= c_f + \delta(2(c_d + c_{f_2} + c_{f'_2}(c_{f_1} + (6+\alpha)c_d)) + c_{f_1} + (6+\alpha)c_d) \\ &= c_f + \delta(2(c_d + c_{f_2}) + (c'_{f_2} + 1)(c_{f_1} + (6+\alpha))). \end{aligned}$$

So by taking $r = k_0^\varepsilon$ with $\varepsilon = \frac{1-\delta}{c_7+1}$ we obtain a running time $2^{\tilde{\mathcal{O}}\left(k_0^{1-\varepsilon}\right)} n_0^{\mathcal{O}(1)}$. To get rid of the hidden logarithmic factors in the exponent and obtain the wanted result it then suffices to replace ε with $0 < \varepsilon' < \varepsilon$. □

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.

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. Let $r \geq 2$, G be a $K_{r,r}$-free pseudo-disk graph, $(\mathcal{D}_v)_{v \in V(G)}$ be a representation of G as pseudo-disks, $A \subseteq V(G)$, and $\mathcal{T}$ a family of disjoint non-adjacent trees of $G - A$. We would like to apply Theorem 5.1 on the set A and the family of subsets of the plane obtained by taking for each tree $T \in \mathcal{T}$ the union $\mathcal{D}_T = \cup_{v \in T} \mathcal{D}_v$. Because we are considering pseudo-disks and trees, this union is homeomorphic to a disk. Moreover as the trees are non-adjacent the obtained unions do not intersect each other, and so are a pseudo-disk system trivially. Now applying Theorem 5.1 with the families $(\mathcal{D}_v)_{v\in A}$ and $(\mathcal{D}_T)_{T\in\mathcal{T}}$ directly gives that there exists a constant c_2 such that if for all $T\in\mathcal{T}$ we have $d_A(T)\leq p$ then $|\{N_A(T),\ T\in\mathcal{T}\}|\leq c_2p^3|A|$ as wanted for the second bound. Observing that we can take $p=|A|$ as an upper bound for $d_A(T)$ with $T\in\mathcal{T}$ gives the first wanted bound. □

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 $d_r=c_4r\log r$.*

Proof. Most of the conditions were already stated with Theorem 1.2, Theorem 1.3 and Lemma 5.2. The only property that remains to check is that in a pseudo-disk graph G, contracting an edge between degree-two vertices u and v that do not belong to a triangle, obtaining a new vertex w, preserves being a pseudo-disk graph. By considering $(\mathcal{D}_q)_{q\in V(G)}$ a pseudo-disk representation of G, we can take $\mathcal{D}_w=\mathcal{D}_u\cup\mathcal{D}_v$ as it is homeomorphic to a disk and its boundary will cross the second neighbor of u (respectively v) as many time than the boundary of $\mathcal{D}_u$ (respectively $\mathcal{D}_v$). □

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 $\alpha=4,f_1(r)=c_1s^4r\log r$ and $f_2(r,p,m)=c_2(sr\log r)^4(p+m)^3$.*

Proof. The proof is very similar to the proof of Lemma 5.2 with some additional tweaks. Let G be a $K_{r,r}$-free s-string graph for some $r\geq 2$ and $(\mathcal{S}_v)_{v\in V(G)}$ be an s-string representation of G. For $T\in\mathcal{T}$, again we consider the region $\mathcal{D}_T=\bigcup_{v\in V(T)}\mathcal{S}_v$. In this case however, even so T is a tree, the complement of this region is not necessarily connected (for example if two strings of T intersect on several points). In order to obtain a region of the plane homeomorphic to a disk, we modify $\mathcal{D}_T$ by cutting small sections where no intersections take place (not even with a string of A) and obtain a new set $\mathcal{D}'_T\subseteq\mathcal{D}_T$ that is connected and such that there is exactly one arc-connected region in $\mathbb{R}^2\setminus\mathcal{D}'_T$ and which intersects the same strings in A as $\mathcal{D}_T$ (see Figure 5). Moreover we add a small thickness to $\mathcal{D}'_T$ while preserving the wanted property about the number of regions in $\mathbb{R}^2\setminus\mathcal{D}'_T$ in order to obtain a geometric shape homeomorphic to a disk. Observe that as there are no edges between vertices of distinct members of $\mathcal{T}$, by taking the thickness small enough the connected sets of the form $\mathcal{D}'_T$ (for $T\in\mathcal{T}$) are disjoint and so they trivially form a system of pseudo-disks.

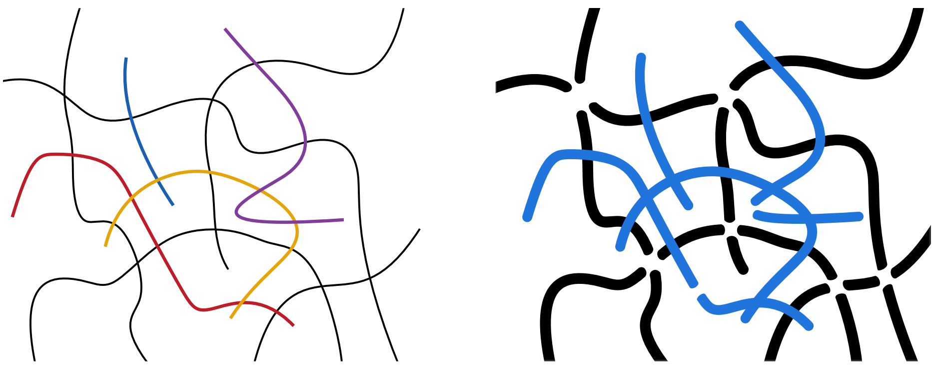

FIGURE 5. Transformation of a union of s-strings to a region homeomorphic to a disk, while keeping the same neighborhood in A. The strings of A are depicted in black.

Now we would like to construct another system of pseudo-disks representing the vertices of A. We do this by cutting the strings in $(\mathcal{S}_u)_{u\in A}$ whenever two of them cross each other. Observe that by Theorem 1.3

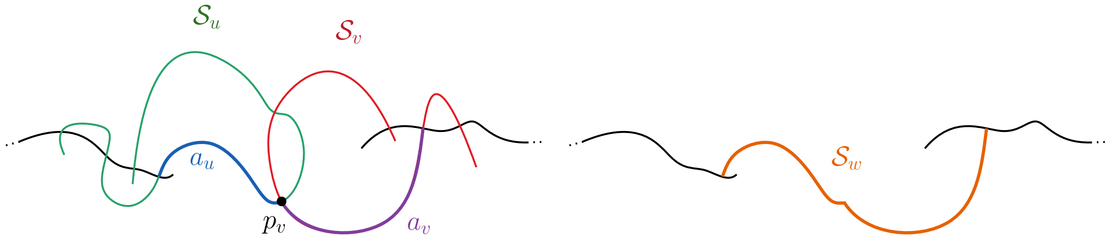

FIGURE 6. Illustration of the construction used in the proof of Lemma 5.5 for contracting an edge between adjacent degree-two vertices without a common neighbor.

there exists a constant c such that by denoting $d_r = cr\log r$ we have at most $d_r|A|$ intersecting pairs, and remember that each pair intersects each other at most s times as we consider a s-string representation. So we have at most $sd_r|A|$ cuts which gives an upper bound of $(sd_r+1)|A|$ for the number of sections of strings between intersection points. We denote C the set of such sections. Again we define a family $(\mathcal{R}_v)_{v\in C}$ by considering the sections (which are disjoint by construction) and giving them some thickness, while preventing the sections to intersect each other. This is clearly a system of pseudo-disks as its elements are not crossing. By Theorem 5.1, the number of distinct sets $N_C(\mathcal{D}'_T) = \{v \in C,\ \mathcal{D}'_T \text{ intersects } \mathcal{R}_v\}$, for $T \in \mathcal{T}$, is $\mathcal{O}(x^3|C|)$ where x is the maximum, over every $T \in \mathcal{T}$, of the number of pseudo-disks in $(\mathcal{R}_v)_{v\in C}$ crossed by D'_T. This upper bound gives the second result of this lemma by observing that if a pair of pseudo-disks $\mathcal{D}'_{T_1}, \mathcal{D}'_{T_2}$ intersect exactly the same set of $\mathcal{R}_v$, then $N_A(T_1) = N_A(T_2)$. We obtain the first result of the lemma by using the trivial bound $x \le |C|$. □

Lemma 5.5. *There exist constants c_1, c_2, c_3, c_4 such that for every $s \ge 1$ 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. Again most of the conditions were already stated with Theorem 1.2, Theorem 1.3 and Lemma 5.4. The only remaining property to prove is that given an s-string graph G, contracting an edge between degree-two vertices u and v that do not belong to a triangle preserves being a s-string graph. We do this by replacing in a s-string representation $(\mathcal{S}_q)_{q\in V(G)}$ of G the strings $\mathcal{S}_u$ and $\mathcal{S}_v$ with a new string $\mathcal{S}_w$ while preserving the global property of being an s-string family. $\mathcal{S}_u$ contains a Jordan arc a_u linking its two neighbors, without being intersected by any string in its interior. Let denote p_v the extremity of a_u in $\mathcal{S}_v$. Then $\mathcal{S}_v$ contains a Jordan arc a_v linking p_v to the string representing its second neighbor, without being intersected by any other string than $\mathcal{S}_u$. The string $\mathcal{S}_w$ is then obtained by taking the union of a_u and a_v. This construction ensures that the number of crossings between two strings other than $\mathcal{S}_u$ and $\mathcal{S}_v$ of the initial representation remain the same. We now have to bound the number of crossing of $\mathcal{S}_w$ with others strings. Recalling that u and v do not share a common neighbor, and that by construction w has 2 crossings (one for each distinct neighbor), w does not have more than 1 crossings with another string. So after replacing $\mathcal{S}_u$ and $\mathcal{S}_v$ by $\mathcal{S}_w$ we still have a s-string family. □

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.

[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.

[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*.

[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`