

Kick the Cliques

Gaétan Berthe 


LIRMM, Université de Montpellier, CNRS, Montpellier, France

Marin Bougeret 

LIRMM, Université de Montpellier, CNRS, Montpellier, France

Daniel Gonçalves 

LIRMM, Université de Montpellier, CNRS, Montpellier, France

Jean-Florent Raymond 

Univ. Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, Lyon, France

Abstract

In the K_r -HITTING problem, given a graph G and an integer k one has to decide if there exists a set of at most k vertices whose removal destroys all r -cliques of G .

In this paper we give an algorithm for K_r -HITTING that runs in subexponential FPT time on graph classes satisfying two simple conditions related to cliques and treewidth. As an application we show that our algorithm solves K_r -HITTING in time

- $2^{O_r(k^{(r+1)/(r+2)} \log k)} \cdot n^{O_r(1)}$ in pseudo-disk graphs and map-graphs;
- $2^{O_{t,r}(k^{2/3} \log k)} \cdot n^{O_r(1)}$ in $K_{t,t}$ -subgraph-free string graphs; and
- $2^{O_{H,r}(k^{2/3} \log k)} \cdot n^{O_r(1)}$ in H -minor-free graphs.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Fixed parameter tractability; Theory of computation → Computational geometry

Keywords and phrases Subexponential FPT algorithms, implicit hitting set problems, geometric intersection graphs

Digital Object Identifier 10.4230/LIPIcs.IPEC.2024.13

Related Version *arXiv Version*: <https://arxiv.org/abs/2407.01465>

Funding *Jean-Florent Raymond*: Supported by the ANR project GRALMECO (ANR-21-CE48-0004).

1 Introduction

In the K_r -HITTING problem, given a graph G and an integer k one has to decide if there are k vertices in G whose deletion yields a K_r -free graph. This problem falls within the general family of (implicit) hitting problems and encompasses several extensively studied problems such as the case $r = 2$ better known under the name VERTEX COVER and the case $r = 3$ that we usually refer to as TRIANGLE HITTING. Already for these small values the problem is NP-complete.

In this paper we are interested in *subexponential parameterized algorithms* for K_r -HITTING, i.e., algorithms that run in *Fixed Parameter Tractable (FPT)* time (that is, time $f(k) \cdot n^{O(1)}$ for some computable function f) and where additionally the contribution of the parameter k is subexponential, in other words $f(k) \in 2^{o(k)}$. Under the Exponential Time Hypothesis of Impagliazzo and Paturi [10], such algorithms do not exist in general for vertex deletion problem to nontrivial hereditary properties [13] (like K_r -HITTING problem) and so we have to focus on particular graph classes.

Historically, subexponential graph algorithms were first obtained for specific problems in sparse graph classes such as planar graphs. The techniques used have then been unified and extended by Demaine, Fomin, Hajiaghayi, and Thilikos in the meta-algorithmic theory of



© Gaétan Berthe, Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond; licensed under Creative Commons License CC-BY 4.0

19th International Symposium on Parameterized and Exact Computation (IPEC 2024).

Editors: Édouard Bonnet and Paweł Rzażewski; Article No. 13; pp. 13:1–13:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

bidimensionality [7], which provides a generic machinery to solve a wide range of problems in subexponential FPT time on H -minor free graphs. Initially bidimensionality was defined for graph classes with some “flatness” property similar to planar graphs, typically graphs of bounded genus and H -minor-free graphs. Over the years, the theory saw several improvements and extensions in order to deal with different settings like map graphs and other classes of intersection graphs, which are initially not sparse as they contain large cliques for example, but where we can branch in subexponential-time to reduce to sparse instances (see for instance the bibliography cited in [4]). However, despite its generality, bidimensionality can only handle the so-called *bidimensional problems* where, informally, as soon as the instance (G, k) contains a large $t \times t$ grid as a minor (for $t \in o(k)$, typically $t = \sqrt{k}$), we know that (G, k) is necessarily a no-instance (or yes-instance depending on the problem). This is the case of VERTEX COVER but unfortunately not of TRIANGLE HITTING (as grids are triangle-free) and more generally not of K_r -HITTING for $r \geq 3$.

The focus of this paper is on this blind spot: subexponential FPT algorithms for a problem that is not bidimensional, namely K_r -HITTING. About this problem, we note that using arguments developed in the context of approximation [9], the following subexponential FPT algorithm can be obtained for apex-minor free graphs (which are sparse).

► **Theorem 1** (from [9]). *For every apex¹ graph H and every $r \in \mathbb{N}$ there is an algorithm solving K_r -HITTING in H -minor-free graphs in time $2^{O_{H,r}(\sqrt{k})} \cdot n^{O_r(1)}$.*

Regarding classes that are not sparse, TRIANGLE HITTING received significant attention in the last years in classes of intersection graphs such as (unit) disk graphs², pseudo-disk graphs, and subclasses of segment graphs³ [14, 2, 4, 3]. We only recall below the most general results and do not mention those that require a geometric representation.

► **Theorem 2.** *There are algorithms that given a parameter k and a n -vertex graph (without a geometric representation) solve TRIANGLE HITTING in time*

1. $2^{O(k^{3/4} \log k)} n^{O(1)}$ in disk graphs [2];⁴
2. $2^{O(k^{3/4} \log k)} n^{O(1)}$ in contact-segment⁵ graphs [4];
3. $2^{O_{t,d}(k^{2/3}) \log k} n^{O(1)}$ in $K_{t,t}$ -subgraph-free d -DIR⁶ graphs [4].

► **Theorem 3** ([4]). *Assuming the Exponential Time Hypothesis, there is no algorithm solving TRIANGLE HITTING in time*

1. $2^{o(n)}$ in 2-DIR graphs;
2. $2^{o(\sqrt{\Delta n})}$ in 2-DIR graphs with maximum degree Δ ; and
3. $2^{o(\sqrt{n})}$ in $K_{2,2}$ -free contact-2-DIR graphs of maximum degree 6.

Our contribution

Our main result is the following subexponential parameterized algorithm for K_r -HITTING in graph classes satisfying two conditions related to cliques and treewidth. Notice that the statement of the following theorem is a simplified version of the actual Theorem 22 that we prove in Section 5.

¹ A graph is *apex* if the deletion of some vertex yields a planar graph.

² (*Unit*) *disk graphs* are intersection graphs of (unit) disks in \mathbb{R}^2 .

³ *Segment graphs* are intersection graphs of segments in \mathbb{R}^2 .

⁴ The published version of the paper gives a bound of $2^{O(k^{4/5} \log k)} n^{O(1)}$ but it can easily be improved to $2^{O(k^{3/4} \log k)} n^{O(1)}$, as confirmed to us by the authors of [2] (private communication).

⁵ *Contact-segment graphs* are the intersection graphs of non-crossing segments in \mathbb{R}^2 .

⁶ A graph is d -DIR if it is the intersection graph of segments of \mathbb{R}^2 with at most d different slopes.

► **Theorem 4.** *Let $r \in \mathbb{N}$, $\alpha \in (0, 1)$, $\mu \in \mathbb{R}_{>0}$ and let \mathcal{G} be a hereditary graph class where every $G \in \mathcal{G}$ with n vertices and clique number ω has $O_r(\omega^\mu n)$ cliques of order less than r and treewidth $O_r(\omega^\mu n^\alpha)$. There exists $\varepsilon < 1$ and an algorithm that solves K_r -HITTING on \mathcal{G} in time $2^{k^\varepsilon} \cdot n^{O_r(1)}$.*

One additional motivation for this work was to generalize to K_r -HITTING the techniques used in previous work to solve TRIANGLE HITTING (in specific graph classes) and to extract the minimal requirements for such an approach to work in more general settings. We believe we met this goal as we actually describe a single generic approach that solves K_r -HITTING on any input graph, for any r . The properties of the class in which the inputs are taken is only used to bound its running time. Such a generalization effort can be fruitful and indeed it allowed us afterwards to identify natural graph classes where subexponential algorithms exist as a consequence of our general result, as we detail now.

In Section 6 we derive from Theorem 22 the following applications.

► **Theorem 5.** *There is an algorithm solving K_r -HITTING in pseudo-disk graphs in time $2^{O_r(k^{(r+1)/(r+2)} \log k)} \cdot n^{O_r(1)}$.*

Pseudo-disk graphs are a classical generalization of disk graphs where to each vertex is associated a *pseudo-disk* (a subset of the plane that is homeomorphic to a disk), two vertices are adjacent if the corresponding pseudo-disks intersect and additionally we require that for any two intersecting pseudo-disks, their boundaries intersect on at most two points. Disk graphs and contact segment graphs are pseudo-disk graphs, so Theorem 5 applies to the two settings handled by the algorithms of [2] and [4] mentioned at items 1 and 2 of Theorem 2. Another application is the following:

► **Theorem 6.** *There is an algorithm solving K_r -HITTING in map graphs⁷ in time $2^{O_r(k^{(r+1)/(r+2)} \log k)} \cdot n^{O_r(1)}$.*

We cannot expect a similar consequence for the more general class of string graphs.⁸ Indeed, there are n -vertex string graphs that are triangle-free and have treewidth $\Omega(n)$, for instance the balanced bicliques.⁹ Note that such graphs prevent string graphs from satisfying the requirement of Theorem 4. Also, and more importantly, by Theorem 3 under ETH there is no $2^{o(n)}$ -time algorithm for K_3 -HITTING in 2-DIR graphs, a restricted subclass of string graphs. As we will show, large bicliques are the only obstructions in the sense that forbidding them in string graphs allows us to solve the problem in subexponential FPT time. For this we use the following light version of Theorem 4 (also consequence of Theorem 22) suited for classes where the clique number is already bounded.

► **Theorem 7.** *Let $r \in \mathbb{N}$, $\alpha \in (0, 1)$ and let \mathcal{G} be a hereditary graph class where every $G \in \mathcal{G}$ with n vertices has $O(n)$ cliques of order less than r and treewidth $O(n^\alpha)$. There exists an algorithm that solves K_r -HITTING on \mathcal{G} in time $2^{O_r(k^{2/(1+1/\alpha)} \log k)} \cdot n^{O_r(1)}$.*

As a consequence we obtain a subexponential FPT algorithm for string graphs excluding large bicliques.

⁷ *Map graphs* are intersection graphs of interior-disjoint regions of \mathbb{R}^2 homeomorphic to disks.

⁸ *String graphs* are intersection graphs of Jordan arcs in \mathbb{R}^2 . They generalize many of the most studied classes of intersection graphs of geometric objects in the plane such as disk graphs, pseudo-disk graphs, segment graphs, chordal graphs, etc.

⁹ $K_{n,n}$ can be drawn as a 2-DIR graph with n horizontal disjoint segments that are all crossed by n vertical disjoint segments.

13:4 Kick the Cliques

► **Theorem 8.** *There is an algorithm solving K_r -HITTING in $K_{t,t}$ -subgraph-free string graphs in time $2^{O_{t,r}(k^{2/3} \log k)} \cdot n^{O_r(1)}$.*

Theorem 8 is a generalization in two directions (the objects to hit and the graph to consider) of item 3 of Theorem 2. We note that under ETH the contribution of k cannot be improved to $2^{o(\sqrt{k})}$, according to Theorem 3. Finally we observe that Theorem 4 can also be applied to certain classes of sparse graphs.

► **Theorem 9.** *For every graph H , there is an algorithm solving K_r -HITTING in H -minor-free graphs in time $2^{O_{H,r}(k^{2/3} \log k)} \cdot n^{O_r(1)}$.*

This is a more general statement than Theorem 1 in the sense that we are not limited to apex-minor free graphs, with the price of a slightly larger time complexity.

Our techniques

Our subexponential algorithm for K_r -HITTING of Theorem 4 is obtained as follows. Given (G, k) , we first perform in Section 3 a preliminary branching step whose objective is to get rid of large cliques (i.e., cliques of order at least k^ε for some $\varepsilon \in (0, 1)$ that we will fix later). This step is a folklore technique which is frequently used for any problem where a solution has to contain almost all vertices of a large clique, like TRIANGLE HITTING (or in general K_r -HITTING), FEEDBACK VERTEX SET, or ODD CYCLE TRANSVERSAL. After this preprocessing has been performed we can assume that the instances to solve have no clique on more than k^ε vertices. Then, we greedily compute an r -approximate K_r -hitting set M . If $|M| > kr$ we can already answer negatively, so in the following we may assume $|M| \leq kr$ and will use the fact that there is no K_r in $G - M$.

Now, the crucial part of the algorithm is Section 4. Informally, the goal of the algorithms described in this section is to extend M into a superset M' together with a new parameter $k' \leq k$, such that $|M'| = O(k^{1+c\varepsilon})$ (for some $c > 0$) and that vertices of $V(G) \setminus M'$ are irrelevant for the problem of hitting r -cliques. In that way, we can remove them, and it remains only to solve $(G[M'], k')$, whose treewidth can be typically bounded by $\sqrt{\omega(G)|M'|}$ in the graph classes we consider. As $\sqrt{\omega(G)|M'|} = O(k^{1/2 + \frac{c+1}{2}\varepsilon})$, this leads to a subexponential algorithm. We stress that the high-level description above is not a kernelization because first we actually do not produce a single reduced instance but instead we have to branch and obtain a subexponential (in k) number of sub-instances and second because the reduction steps are not carried out in polynomial time, but in subexponential (in k) time.

To obtain this set M' , we use lemmatas 14 and 21 that are inspired from the following “virtual branching” procedure of [14, Lemma 6.5]. This routine was introduced for TRIANGLE HITTING and works as follows. It starts with a triangle hitting set M (obtained by greedily packing disjoint triangles), and outputs a slightly larger superset M' such that vertices in $G - M'$ are *almost* useless, in the sense that every triangle has at least two vertices in M' (we do not detail here how to handle the triangles with exactly two vertices in M' and refer to [14]). This is done as follows. For a vertex $v \in M$, consider a maximum matching $M(v) \subseteq N(v) \cap (V(G) \setminus M)$. If for every $v \in M$ such a matching is small, meaning $|M(v)| \leq k^\varepsilon$, then we can define $M' = M \cup \bigcup_{v \in M} M(v)$. We are done as $|M'| = O(k^{1+\varepsilon})$ remains small, and there is no longer a $v \in M'$ with an edge in $N(v) \setminus M'$ (as this would form a triangle outside M). Otherwise, if for some $v \in M$, $|M(v)| > k^\varepsilon$, a solution of TRIANGLE HITTING should either take v , or otherwise hits all edges of $M(v)$. In the second case, it would be too costly to guess which vertex is taken in each $e \in M(v)$, so instead the procedure “absorbs” $M(v)$ by defining $M' = M \cup M(v)$. This absorption increases the size of M , but

“virtually” decreases the parameter k , as it increases by $|M(v)|$ the size of a matching that the solution will have to hit. This leads to a running time typically dominated by the recurrence $f(k) = f(k-1) + f(k-k^\varepsilon)$, which is subexponential in k .

Now, coming back to K_r -HITTING, given a set M , let us say¹⁰ that a *type- i clique* is an r -clique X such that $|X \cap M| = i$. We could remove type-1 cliques by using the previous virtual branching procedure, defining now $M(v)$ as a packing of $r-1$ cliques instead of a packing of edges, but the problem is that, if we want to obtain a set M' as promised (where vertices of $V(G) \setminus M'$ are useless), we also have to remove type- i cliques for $i \in \{2, \dots, r-1\}$. However, there is a first obstacle to remove such type- i cliques: as the part common with M (which was before a single vertex in M) is now an i -clique, we cannot afford to enumerate all possible choices X' of such an i -clique in M . Indeed, already for $i = 2$ we would possibly consider a quadratic number of sets X' , so absorbing every packing (of $(r-i)$ -cliques) $M(X')$ (in the unfortunate case where these are all small) would result in a set $M' = M \cup \bigcup_{X' \subseteq M, |X'|=2} M(X')$ with $|M'| = \Omega(k^2)$, which is too large for our purpose. To circumvent this issue, we identified a key property that holds in many geometric graph classes like pseudo-disk or $K_{t,t}$ -subgraph-free string graphs: in such graphs, there is only a linear (in the number of vertices) number of i -cliques for fixed i , and for fixed clique number ω . In our case, as ω is small, and $i \leq r$ is fixed, this implies that there are $O(|M|)$ such i -cliques in M . Hence, it allows us to control the size of M' . Of course, dealing with r -cliques instead of triangles also raises other problems, in particular related to the way we hit their intersection with M which is no longer a single vertex but a clique. To deal with this issue we had to introduce an annotated variant of the problem where additional sets of vertices have to be hit besides r -cliques.

Organization of the paper

In Section 2 we give the necessary definitions. We describe the first branching in Section 3 and the second in Section 4. The algorithm is given in Section 5. We give applications to selected graph classes in Section 6. We conclude with open questions in Section 7.

2 Preliminaries

Running times

When stating results related to algorithms, the variable n in the running time always refers to the number of vertices of the graph that is part of the input. For any parameter p (typically a graph H , or an integer r) and integer k , and any functions $f(p, k)$ and $g(k)$, we write $f = O_p(g(k))$ to indicate that for any fixed p , the restricted function $k \rightarrow f(p, k)$ is $O(g(k))$. To simplify the presentation we will assume that r is a fixed constant instead of explicitly give it as a parameter in all our algorithms and lemmas.

Graphs

Unless otherwise stated we use standard graph theory terminology. A *clique* in a graph G is a complete subgraph and when there is no ambiguity we also use *clique* to denote a subset of $V(G)$ inducing a complete subgraph. The *clique number* of G is the maximum number of vertices of a clique it contains and we denote it by $\omega(G)$. For any $i \in \mathbb{N}$, an *i -clique* is a

¹⁰This notion of type- i clique will not be used later and is just introduced for this sketch.

13:6 Kick the Cliques

clique on i vertices and a $(< i)$ -clique is a clique on less than i vertices. For any graph G and subset of vertices $X \subseteq V(G)$, we denote $G - X$ the graph whose vertex set is $V(G) \setminus X$ and edge set is $\{e \in E(G) : e \cap X = \emptyset\}$. Let H be a graph. We say that G is H -free if G does not contain H as induced subgraph.

Hypergraphs and hitting sets

A *hypergraph* is simply a collection of sets, where any set is referred as an *hyperedge*. So $|\mathcal{D}|$ refers to the number of sets in the hypergraph \mathcal{D} and we define $V(\mathcal{D}) = \bigcup_{D \in \mathcal{D}} D$. By $\mathcal{K}_r(G)$ (resp. $\mathcal{K}_{<r}(G)$) we denote the hypergraph of r -cliques (resp. $(< r)$ -cliques) of G , i.e. $\mathcal{K}_r(G) = \{X \subseteq V(G), X \text{ is an } r\text{-clique}\}$.

A *hitting set* of \mathcal{D} is a subset $X \subseteq V(\mathcal{D})$ that intersects every hyperedge of \mathcal{D} . A *matching* of \mathcal{D} is a collection of disjoint hyperedges. The maximum size of a matching in \mathcal{D} is denoted by $\nu(\mathcal{D})$. Note that a hitting set of \mathcal{D} has always size at least $\nu(\mathcal{D})$ as it needs to intersect each of the elements of a maximum matching, which are disjoint.

Special cases of hitting sets of hypergraphs are the hitting sets of subgraphs of a graph. For G, H two graphs, an H -hitting set in G is a subset $X \subseteq V(G)$ such that $G - X$ is H -free. In other words it is a hitting set of the hypergraph of the (induced) subgraphs of G isomorphic to H . In the H -HITTING problem, given a graph G and an integer $k \in \mathbb{N}$, one has to decide whether G has a H -hitting set of size at most k .

3 Dealing with large cliques

A K_r -HITTING can be useful to detect large cliques, as we explain now. This will allow us to identify cliques on which to branch.

► **Lemma 10.** *Given a graph G , a K_r -hitting set M , and an integer $p > r$, one can find a p -clique of G , or correctly conclude none exists, in $O(p^2 |M|^p n^{r-1})$ steps.*

Proof. For each choice of $p - r + 1$ vertices of M and $r - 1$ other vertices of G we check whether they form a clique (which takes $O(p^2)$ time), in which case return it and stop. If no clique is found we return that G is K_p -free. The correctness follows from the following observation: as $G - M$ is K_r -free, every p -clique of G has at least $p - r + 1$ of its vertices in M . ◀

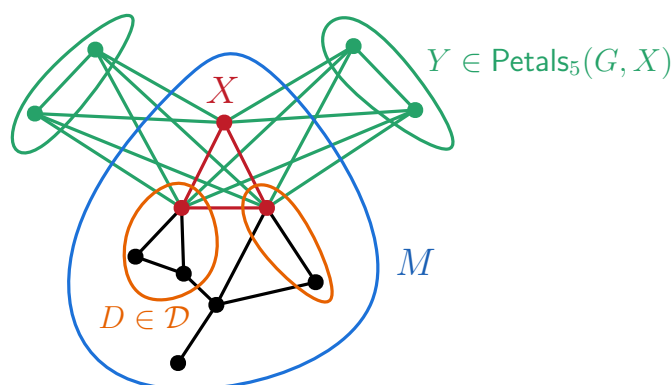
As noted in the proof of Lemma 10, every large clique of the input graph will have most of its vertices in a K_r -hitting set so we can branch on which these are.

► **Lemma 11.** *There is an algorithm that, given two integers $r \geq 1$ and $p > r$, an instance (G, k) of K_r -HITTING and a K_r -hitting set M of G , runs in $2^{kr(\log p)/p} |M|^p n^{O(r)}$ steps and returns a collection \mathcal{Y} of $2^{kr(\log p)/p} \cdot |M|^p$ instances of the same problem such that:*

1. (G, k) is a yes-instance if and only if \mathcal{Y} contains a yes-instance;
2. for every $(G', k') \in \mathcal{Y}$, G' has no p -clique.

Proof. The algorithm is the following:

1. Using the algorithm of Lemma 10 on G and M , we find a p -clique K (if none is found return $\mathcal{Y} = \{(G, k)\}$).
2. We initialize $\mathcal{Y} = \emptyset$.
3. Observe that any solution contains at least $p - r + 1$ vertices from K . For every subset X of K with $p - r + 1 \leq |X| \leq k$ vertices:



■ **Figure 1** A 3-clique X with two 5-petals that live in $G - M$. Here \mathcal{D} contains two hyperedges, represented in orange. Observe that no hyperedge of \mathcal{D} is contained in X , so X is a lush 3-clique.

- a. We look for solutions S that contain X with a recursive call on $(G - X, k - |X|)$ and $M \setminus X$;
 - b. We then add the resulting collection of instances to \mathcal{Y} .
4. Return \mathcal{Y} .

Regarding correctness, Item (2) follows from the base case of the recursion (step 1) and Item (1) can be proved by induction on the number of recursive calls, using the observation that (G, k) has a solution containing a set X if and only if $(G - X, k - |X|)$ has a solution.

We denote by $T_{r,p}(n, k)$ the time complexity of the above algorithm with parameters $r, p, (G, k)$ where $|G| = n$. The time taken by each computation step is the following:

1. Finding a p -clique takes time $O(p^2 |M|^p n^{r-1})$ by Lemma 10.
2. When a p -clique K is found we consider at most p^{r-1} subsets X on which we perform a recursive call of cost at most $T_{r,p}(n - p + r - 1, k - p + r - 1)$.

So

$$T_{r,p}(n, k) \in |M|^p n^{O(r)} + p^{r-1} \cdot T_{r,p}(n - p + r - 1, k - p + r - 1).$$

We deduce

$$\begin{aligned} T_{r,p}(n, k) &\in p^{k(r-1)/(p-r+1)} |M|^p n^{O(r)} \\ &\in 2^{kr(\log p)/p} |M|^p n^{O(r)}, \end{aligned}$$

as desired. Note that a similar recurrence can be used to bound the number of output instances. ◀

4 Picking petals

Let $i \in \{1, \dots, r-1\}$ and let X be an i -clique in a graph G . An r -petal of X is a subset of vertices of $G - X$ that together with X forms an r -clique. We denote by $\text{Petals}_r(G, X)$ the hypergraph of r -petals of X , i.e., $\text{Petals}_r(G, X) = \{Y \subseteq V(G) \setminus X, X \cup Y \in \mathcal{K}_r(G)\}$. See Figure 1 for an illustration.

In order to deal more easily with the recursive steps in our algorithms we introduce ANN- K_r -HITTING, an annotated version of K_r -HITTING where a number of choices have already been made, which is recorded by extra vertex subsets that the solution is required to hit.

13:8 Kick the Cliques

In the ANN- K_r -HITTING problem, one is given a triple (G, \mathcal{D}, k) where G is a graph, $\mathcal{D} \subseteq \mathcal{K}_{<r}(G)$ and k is an integer. A *solution* to this instance is a set of vertices that hits $\mathcal{K}_r(G)$ and \mathcal{D} and has at most k vertices. The question is whether the input instance admits a solution.

In the forthcoming algorithms it will also be more convenient to consider, together with an instance, a non-optimal solution M . This motivates the following definition. A *context* is a pair $((G, \mathcal{D}, k), M)$ where (G, \mathcal{D}, k) is an instance of ANN- K_r -HITTING, M is a K_r -hitting set (possibly larger than k), and $V(\mathcal{D}) \subseteq M$. We say that a context is *positive* if the instance of ANN- K_r -HITTING it contains is a yes-instance, and *negative* otherwise.

Given a context $((G, \mathcal{D}, k), M)$ and $i \in \{1, \dots, r-1\}$, a *lush i -clique* is an i -clique X of $G[M]$ that has an r -petal in $G - M$ and such that no $D \in \mathcal{D}$ is subset of $V(X)$. See Figure 1 for an example.

Informally, if X is a lush clique then we are not guaranteed that hitting \mathcal{D} alone does always also hit the cliques induced by X and its petals, so we have to take care of them separately. Ideally we would like to get rid of lush cliques so that we can focus on \mathcal{D} to solve the problem. We say that the context $((G, \mathcal{D}, k), M)$ is *i -stripped* if for every $i' < i$ it has no lush i' -clique. These notions are motivated by the following easy lemma.

► **Lemma 12.** *Let $((G, \mathcal{D}, k), M)$ be an r -stripped context. The instances (G, \mathcal{D}, k) and $(G[M], \mathcal{D}, k)$ of ANN- K_r -HITTING are equivalent.*

Proof. First, recall that as $((G, \mathcal{D}, k), M)$ is a context, $V(\mathcal{D}) \subseteq M$ so $(G[M], \mathcal{D}, k)$ is indeed a valid instance of ANN- K_r -HITTING. Also as $\mathcal{K}_r(G[M]) \subseteq \mathcal{K}_r(G)$, if (G, \mathcal{D}, k) has a solution then $(G[M], \mathcal{D}, k)$ does. So we only need to show the other direction. Let $v \in V(G) \setminus M$ and suppose that G contains some r -clique X with $v \in X$. The set $M \cap X$ is not empty because M is a K_r -hitting set. Note that $X \cap M$ has a petal $X \setminus M$ disjoint from M . It cannot form a lush i -clique (for $i = |M \cap X| < r$) as this would contradict the assumption that the considered context is r -stripped, so there is some $D \in \mathcal{D}$ that is subset of $X \cap M$. Therefore every solution of $(G[M], \mathcal{D}, k)$ does hit X in G , as it hits D . As this holds for every v and X as above, every solution of $(G[M], \mathcal{D}, k)$ is a solution of (G, \mathcal{D}, k) , as desired. ◀

By the above lemma, if we manage to get rid of lush cliques, we can obtain an equivalent instance whose graph is not larger than $G[M]$. As we will see, we are able to handle lush cliques with the price of slightly increasing the size of M and producing several instances to represent the solutions of the original instance.

► **Lemma 13.** *There is an algorithm that, given an integer $i < r$ and an i -stripped context $((G, \mathcal{D}, k), M)$, runs in time $n^{O(r)}$ and either correctly concludes that the context is $(i+1)$ -stripped, or returns a lush i -clique X .*

Proof. The input context is i -stripped so we only have to check whether it contains a lush i -clique. We iterate over the i -cliques of $G[M]$. For every such clique X , we first check if $D \subseteq V(X)$ for some $D \in \mathcal{D}$. If so X is not a lush i -clique so we can move to the next choice of X . Otherwise we check if the common neighborhood of the vertices of X in $G - M$ has an $(r-i)$ -clique. If so this is an r -petal so X is lush and we can return it. Otherwise we continue to the next choice of X . If the iteration terminates without detecting a lush i -clique, we can safely return that the input context is $(i+1)$ -stripped. The complexity bound follows from the fact that $|\mathcal{D}| = n^{O(r)}$ and that $G[M]$ has $n^{O(r)}$ i -cliques. ◀

The following lemma is the key branching step in our subexponential algorithms for K_r -HITTING.

► **Lemma 14.** *There is an algorithm that, given $i \in \{1, \dots, r-1\}$ and $\lambda \in \{1, \dots, k\}$ and an i -stripped context $((G, \mathcal{D}, k), M)$ where we denote by ζ the number of i -cliques in $G[M]$, runs in time $2^{O((k/\lambda) \cdot \log \zeta)} \cdot n^{O(r)}$ and returns a collection \mathcal{Z} of size $2^{O((k/\lambda) \cdot \log \zeta)}$ of $(i+1)$ -stripped contexts such that:*

1. $((G, \mathcal{D}, k), M)$ is a positive context if and only if \mathcal{Z} contains one; and
2. for every $((G', \mathcal{D}', k'), M') \in \mathcal{Z}$, $M \subseteq M'$, $|M'| \leq |M| + r(\lambda\zeta + k)$, and $k' \leq k$.

Proof. Let us define an auxiliary algorithm that takes as input $(i, \lambda, (G, \mathcal{D}, k), M, \mathcal{P}^*)$, where $(i, \lambda, (G, \mathcal{D}, k), M)$ is as specified in the statement of the lemma, and \mathcal{P}^* is a matching of \mathcal{D} , and output the promised collection \mathcal{Z} . Such an auxiliary algorithm will imply the lemma, as we will run it with parameters $(i, \lambda, (G, \mathcal{D}, k), M, \emptyset)$. For simplicity, as i and λ will not change between recursive calls: we denote by $((G, \mathcal{D}, k), M, \mathcal{P}^*)$ the parameter of this auxiliary algorithm, which is defined as follows:

1. If $k < |\mathcal{P}^*|$, we can immediately return $\mathcal{Z} = \emptyset$.
2. Run the algorithm of Lemma 13 on i and $((G, \mathcal{D}, k), M)$. If no lush i -clique is found then the input context is already $(i+1)$ -stripped so we return $\mathcal{Z} = \{((G, \mathcal{D}, k), M)\}$. Otherwise let X denote the lush i -clique we found.
3. Construct the hypergraph \mathcal{B}_X of those r -petals of X that are subset of $V(G) \setminus M$.
4. Greedily compute a maximal matching \mathcal{P} in \mathcal{B}_X .
5. If $\tilde{\nu}_X \leq \lambda$, we return the result of the recursive call with parameters $((G, \mathcal{D}, k), M \cup V(\mathcal{P}), \mathcal{P}^*)$ (and quit).
6. Otherwise, we investigate the different ways to hit the r -cliques induced by X and \mathcal{B}_X (via X or via \mathcal{B}_X) as follows.
 - a. Solutions containing some (yet unspecified) vertex of X : this is done by a recursive call with parameters $((G, \mathcal{D} \cup \{X\}, k), M, \mathcal{P}^*)$. We call \mathcal{Z}_1 the resulting family.
 - b. Solutions hitting \mathcal{P} .¹¹ As \mathcal{P} is disjoint from M (hence from \mathcal{P}^*), such solutions exist only if $k \geq |\mathcal{P}^*| + |\mathcal{P}|$. In this case we define \mathcal{Z}_2 as the result of the recursive call with parameters

$$((G, \mathcal{D} \cup \mathcal{P}, k), M \cup V(\mathcal{P}), \mathcal{P}^* \cup \mathcal{P}).$$

Otherwise we set $\mathcal{Z}_2 = \emptyset$.

7. We return $\mathcal{Z}_1 \cup \mathcal{Z}_2$.

Observe first that in step (6b), the last parameter $\mathcal{P}^* \cup \mathcal{P}$ is a matching as required, as in particular $V(\mathcal{P}) \subseteq V(G) - M$ and $V(\mathcal{P}^*) \subseteq M$, by definition. We first prove the following fragment of item (2).

▷ **Claim 15.** For every $((G', \mathcal{D}', k'), M') \in \mathcal{Z}$, $M \subseteq M'$ and $k' \leq k$.

Proof. The only places where M is updated are steps 5 and 6b, where new vertices are added to it. Besides we never change the value of the parameter k . ◁

Let us describe the recursion tree T of the above algorithm on some input $((G, \mathcal{D}, k), M, \mathcal{P}^*)$. The nodes of this tree are inputs. The root is $((G, \mathcal{D}, k), M, \emptyset)$ and a node s' is child of a node s if a call of the above algorithm on the input s triggers a call on the input s' . So the leaves of this tree are the inputs that do not trigger any recursive call.

¹¹ Notice that some of these solutions have possibly already been investigated in the previous step. For our purpose it is not an issue however to consider several times the same solution.

13:10 Kick the Cliques

Let us consider a path from the root of T to some leaf. We denote by

$$((G_j, \mathcal{D}_j, k_j), M_j, \mathcal{P}_j^*)_{j \in \{1, \dots, \ell\}}$$

the inputs along this path, and by $C_j = ((G_j, \mathcal{D}_j, k_j), M_j)$ the corresponding contexts, with $((G_1, \mathcal{D}_1, k_1), M_1, \mathcal{P}_1^*) = ((G, \mathcal{D}, k), M, \emptyset)$ and $((G_\ell, \mathcal{D}_\ell, k_\ell), M_\ell, \mathcal{P}_\ell^*)$ corresponding to the aforementioned leaf. Also, for every $j \in \{1, \dots, \ell - 1\}$ we denote by X_j the lush i -clique of C_j that is considered in the corresponding call.

We first show that all lush i -cliques considered along this path belong to the hitting set $M = M_1$ of the initial context.

▷ **Claim 16.** For every $j \in \{1, \dots, \ell - 1\}$, $X_j \subseteq M$.

Proof. Suppose towards a contradiction that for some j , $X_j \not\subseteq M$. Observe that since C_j is not a leaf, \mathcal{B}_{X_j} is not empty so X_j induces an r -clique together with some r -petal $B \in \mathcal{B}_{X_j}$. Recall that B is disjoint from M_j , by definition of \mathcal{B}_{X_j} . As M_j is a superset of M (Claim 15), B is disjoint from M as well. So M intersects X_j otherwise the r -clique $X_j \cup B$ would not be hit by M . Let $i' = |X_j \cap M|$. Note that $X_j \cap M$ is a lush i' -clique of C_1 since it has an r -petal $B \cup (X \setminus M)$ disjoint from M (the fact that no set of \mathcal{D} is subset of $X_j \cap M$ follows from the fact that this property holds for X_j). By our initial assumption and as $|X_j| = i$, we have $i' < i$. This contradicts the fact that C_1 is i -stripped. ◁

With a similar proof we can show the following (so the recursive calls are indeed made on valid inputs).

▷ **Claim 17.** For every $i \in \{1, \dots, \ell\}$, C_i is i -stripped.

Let us now show that each lush clique is only considered once.

▷ **Claim 18.** For every distinct $j, j' \in \{1, \dots, \ell - 1\}$, $X_j \neq X_{j'}$.

Proof. Suppose towards a contradiction that for some $j < j'$ we have $X_j = X_{j'}$. Then in the call on the context C_j , the next context C_{j+1} was obtained at step 5 or 6b (since at step 6a we would include X_j in \mathcal{D}_j , preventing it to be considered in future calls). In any of these two possibilities we set $M_{j+1} = M_j \cup V(\mathcal{P}_j)$, where \mathcal{P}_j denotes the maximal matching of step 4 in the call on context C_j .

Besides, as X_j is a lush i -clique in the context $C_{j'}$, then it has some r -petal B subset of $V(G_{j'}) - M_{j'}$. As $M_{j+1} \supseteq M_{j'}$ (Claim 15), B is also an r -petal of X_j and is subset of $V(G_j) - M_j$ and by the above observation, it is disjoint from \mathcal{P}_j . This contradicts the maximality of \mathcal{P}_j . ◁

Recall that the number of i -cliques in M is ζ . As a consequence of Claim 16 and Claim 18, we get the following.

▷ **Claim 19.** The recursion tree has depth at most ζ .

We can now conclude the proof of item (2).

▷ **Claim 20.** For every $((G', \mathcal{D}', k'), M') \in \mathcal{Z}$, $|M'| \leq |M| + r(\lambda\zeta + k)$.

Proof. When considering the context C_j , and given the chosen maximal matching \mathcal{P}_j of the petals of X_j , the set M_{j+1} is defined from M_j by:

- either adding the at most $\lambda(r - 1)$ new vertices of \mathcal{P}_j , if we make the recursive call at step 5,
- or by adding the vertices of the petals of \mathcal{P}_j , if we recurse at step 6b. Recall that in this case we also have $\mathcal{D}_{j+1} = \mathcal{D}_j \cup \mathcal{P}_j$ and $\mathcal{P}_{j+1}^* = \mathcal{P}_j^* \cup \mathcal{P}_j$

In the later case the number of added vertices is not directly bounded however we have $|\mathcal{P}_{j+1}^*| = |\mathcal{P}_j^*| + |\mathcal{P}_j|$. Because of the stopping condition of step 1, we will overall (from C_1 to C_ℓ) add at most k petals to the hypergraph and each petal has at most $r - 1$ vertices. Hence we get $|M'| \leq |M| + r(\lambda\zeta + k)$, as claimed. \triangleleft

Let us now show that the algorithm is correct, i.e., item 1 of the statement of the lemma. The proof is by induction on the depth of the recursion tree (i.e., $\ell - 1$ with the notation above). When the depth is 0, there is no recursive call. This corresponds to the two base cases in this algorithms: step 1, when the “budget” k is insufficient, and step 2, when the input context is already $(i + 1)$ -stripped. Clearly the outputs in these cases satisfy 1.

So we now consider the case of a run of the algorithm where the recursion tree has depth at least 1 and suppose that item 1 holds for all runs with recursion trees of smaller depth.

- If the recursive call is made at step 5 then item 1 trivially holds because the instance is unchanged.
- Otherwise, note that any solution has to hit the r -cliques induced by X and \mathcal{B}_X . So any solution either contains a vertex of X , or hits \mathcal{B}_X (or both). These are exactly the two branches that are explored in steps 6a and 6b, respectively, by our induction hypothesis.

The above shows that the algorithm is correct. It remains to prove that it has the claimed running time. Note that we do not update the graph neither the parameter between recursive calls: we will always work on the graph G with n vertices and with the parameter k . So the induction proving the time bound will use two different parameters as we explain now. For every $x, y \in \mathbb{N}$, let us denote by $T(x, y)$ the worst-case running time of the above algorithm on an input $((G, \mathcal{D}, k), M, \mathcal{P}^*)$ with $|G| = n$ such that there is at most x (non-necessarily disjoint) lush i -cliques in $G[M]$ and such that $|\mathcal{P}^*| \geq y$. Let $S_r(n)$ be the sum of the worst-case complexity of all subroutines needed in the different steps of item (1) to item (7) to the exception of recursive calls (one such subroutine is the algorithm of Lemma 13, another one is the construction of the hypergraph \mathcal{B}_X). Observe that $T(x, y) \leq S_r(n)$ when $y > k$ (as we fall into base case of step 1) or when $x = 0$ (as we fall into base case of step 2). Notice also that by definition we have $T(x, y) \leq T(x', y)$ for any $x \leq x'$, and $T(x, y) \leq T(x, y')$ for any $y' \leq y$.

If we make a recursive call at step 5, we return in time at most $T(x - 1, y)$ (by induction) as, by Claim 18, no lush i -clique of the original instance is considered twice. Otherwise, we will make recursive calls in step 6 which by induction take time at most $T(x - 1, y) + T(x - 1, y + |\mathcal{P}|)$ as, in the first branch 6a of recursion, Claim 18 implies again that no lush i -clique is considered two times, and in the second branch 6b, we know in addition that the size of the matching given as parameter increases by $|\mathcal{P}|$. Thus, in both cases (and including the other computation steps which take time $S_r(n)$), we obtain the upper bound:

$$\begin{aligned} T(x, y) &\leq T(x - 1, y) + T(x - 1, y + |\mathcal{P}|) + S_r(n) \\ &\leq T(x - 1, y) + T(x, y + \lambda) + S_r(n). \end{aligned}$$

(For the last line recall that T is anti-monotone with respect to its second parameter.)

Let us now show that for every $x, y \in \mathbb{N}$, $T(x, y) \leq xT(x, y + \lambda) + (x + 1)S_r(n)$. The proof is by induction on x . The base case $x = 0$ holds as observed above. Suppose the inequality holds for $x - 1$. As proved above

$$\begin{aligned} T(x, y) &\leq T(x - 1, y) + T(x, y + \lambda) + S_r(n) \\ &\leq (x - 1)T(x - 1, y + \lambda) + xS_r(n) && \text{(by induction)} \\ &\quad + T(x, y + \lambda) + S_r(n) \\ &\leq xT(x, y + \lambda) + (x + 1)S_r(n), && \text{as claimed.} \end{aligned} \tag{1}$$

13:12 Kick the Cliques

We now prove that for every $x, y \in \mathbb{N}$,

$$T(x, y) \leq x^{\frac{k+1-y}{\lambda}} + \left(1 + \frac{k+1-y}{\lambda}\right) (x+1)S_r(n).$$

This time the induction is on y . The base case $y > k$ hold as observed above. Let $x \geq 1$ and $y \leq k$ and suppose the inequality holds for any pair (x', y') with $y' > y$. Then as proved above in Eq. 1,

$$\begin{aligned} T(x, y) &\leq xT(x, y + \lambda) + (x+1)S_r(n) \\ &\leq x \cdot x^{\frac{k+1-y-\lambda}{\lambda}} + \left(1 + \frac{k+1-y-\lambda}{\lambda}\right) (x+1)S_r(n) + (x+1)S_r(n) \\ &\leq x^{\frac{k+1-y}{\lambda}} + \left(1 + \frac{k+1-y}{\lambda}\right) (x+1)S_r(n). \end{aligned}$$

Observe that $S_r(n)$ is dominated by the time spent in the algorithm of Lemma 13, and thus $S_r(n) \in n^{O(r)}$. As we initially have $x \leq \zeta$ and $y \geq 0$, we obtained the claimed running time. A similar analysis can be used to bound the size of the output family \mathcal{Z} . ◀

By iterating the algorithm of Lemma 14 for increasing values of i we can obtain a collection of r -stripped contexts, as we explain now.

► **Lemma 21 (picking petals).** *There is an algorithm that, given $i \in \{1, \dots, r\}$, $\lambda \in \mathbb{R}_{\geq 1}$, a context $((G, \mathcal{D}, k), M)$, where we denote by ζ the number of $(< r)$ -cliques in $G[M]$, runs in time $2^{O((i \cdot k/\lambda) \cdot \log \zeta)} \cdot n^{O(r)}$ and returns a collection \mathcal{Z} of size $2^{O((i \cdot k/\lambda) \cdot \log \zeta)}$ of i -stripped contexts such that:*

1. (G, \mathcal{D}, k) is a yes-instance if and only if some input of \mathcal{Z} contains one; and
2. for every $((G', \mathcal{D}', k'), M') \in \mathcal{Z}$, $M \subseteq M'$, $|M'| \leq |M| + ir(\lambda\zeta + k)$, and $k' \leq k$.

Proof. Again for the sake of clarity we assume λ is a fixed constant.

The proof is by induction on i . For the base case $i = 1$ we simply observe that $((G, \mathcal{D}, k), M)$ is already 1-stripped so there is nothing to do.

So let us now suppose that $i > 1$ and that the statement holds for $i - 1$. So from the input context $((G, \mathcal{D}, k), M)$ and $i - 1$ we can use the induction hypothesis to construct a collection \mathcal{Z}_{i-1} of $(i - 1)$ -stripped contexts satisfying the statement for $i - 1$. Now we apply the algorithm of Lemma 14 to $i - 1$ and each context in \mathcal{Z}_{i-1} and call \mathcal{Z}_i the union of the obtained collections. Item 1 follows from the properties of \mathcal{Z}_{i-1} and the correctness of the algorithm of Lemma 14. Constructing \mathcal{Z}_{i-1} takes time $2^{O(((i-1) \cdot k/\lambda) \cdot \log \zeta)} \cdot n^{O(r)}$ (by induction) and then we run the $(2^{O((k/\lambda) \cdot \log \zeta)} \cdot n^{O(r)})$ -time algorithm of Lemma 14 on each of its $2^{O(((i-1) \cdot k/\lambda) \cdot \log \zeta)}$ contexts. This results in an overall running time of $2^{O((i \cdot k/\lambda) \cdot \log \zeta)} \cdot n^{O(r)}$, as claimed. In each context of \mathcal{Z}_{i-1} the set M' has size at most $|M| + (i - 1)r(\lambda\zeta + k)$ (by induction) and after the run of the algorithm Lemma 14, the corresponding set in the produced instances has at most $r(\lambda\zeta + k)$ vertices more so we get the desired bound. Finally, as in the proof of Lemma 14 the value of k never changes. ◀

5 Kick the cliques

For every $\phi, \gamma \in \mathbb{R}_{\geq 0}$ and $\alpha \in (0, 1)$, we say that graph class \mathcal{G} has property $P_r(\phi, \gamma, \alpha)$ if there are functions $f(x) \in O(x^\phi)$ and $g(x) \in O(x^\gamma)$ such that for $G \in \mathcal{G}$ with n vertices and clique number less than ω ,

- (P1) G has at most $f(\omega) \cdot n$ cliques of order less than r ; and
- (P2) $\text{tw}(G) \leq g(\omega)n^\alpha$.

Our main contribution is the following.

► **Theorem 22.** *For every hereditary graph class \mathcal{G} that has property $P_r(\phi, \gamma, \alpha)$ for some $\alpha \in (0, 1)$, $\phi, \gamma \in \mathbb{R}_{\geq 0}$, there is an algorithm that solves K_r -HITTING on \mathcal{G} in time*

$$2^{O_{r,\phi}(k^\varepsilon \log k)} \cdot n^{O_r(1)} \quad \text{with} \quad \varepsilon = \frac{\gamma + \alpha(\phi + 2)}{\gamma + \alpha(\phi + 1) + 1} < 1,$$

i.e., subexponential FPT time.

Proof. Let $f: x \mapsto c_f \cdot x^\phi$ and $g: x \mapsto c_g \cdot x^\gamma$ (for some $c_f, c_g > 0$) be as in the definition of property $P_r(\phi, \gamma, \alpha)$. For the sake of readability we will here use (when deemed useful) \exp to denote the function $x \mapsto 2^x$ defined over reals. Given an instance (G, k) of K_r -HITTING, we consider the context $((G, \mathcal{D}, k), M)$ where $\mathcal{D} = \emptyset$ and M is an r -approximation of a K_r -hitting set computed in $n^{O(r)}$ time by greedily packing disjoint r -cliques. If $|M| > kr$ we can already answer negatively, so in what follows we suppose that $|M| \leq kr$. We run the algorithm of Lemma 11 with $p = \lceil k^\varepsilon \rceil$ for some constant $\varepsilon \in (0, 1)$ that we will fix later. In time

$$2^{O(rk^{1-\varepsilon} \log k)} n^{O(r)}$$

we obtain a set \mathcal{Y} of $2^{O(rk^{1-\varepsilon} \log k)}$ contexts that have no p -clique. For each such context we apply the petal-picking algorithm of Lemma 21 with

$$\begin{aligned} \lambda &= k^\varepsilon & \text{and} \\ \zeta &= f(p) \cdot |M| \\ &\in O(rk^{1+\varepsilon\phi}). \end{aligned}$$

Let \mathcal{Z} denote the union of the outputs families of these algorithms. Computing this set then takes time

$$\begin{aligned} |\mathcal{Y}| \cdot 2^{O((r \cdot k/\lambda) \cdot \log \zeta)} \cdot n^{O(r)} &\in 2^{O(r \cdot k^{1-\varepsilon} (\log k + \log \zeta))} \cdot n^{O(r)} \\ &\in 2^{O(r^2 \cdot (1+\varepsilon\phi)k^{1-\varepsilon} \log k)} \cdot n^{O(r)}. \end{aligned}$$

For every $((G', \mathcal{D}', k), M') \in \mathcal{Z}$ we have

$$\begin{aligned} |M'| &\leq |M| + r^2(\lambda\zeta + k) && \text{by Lemma 21} \\ &\in O(kr + r^2(rk^{1+\varepsilon\phi+\varepsilon} + k)) \\ &\in O(r^3k^{1+\varepsilon(\phi+1)}). \end{aligned}$$

So given any such context, we can decide whether it is positive or not in time

$$\begin{aligned} 2^{g(p)|M'|^\alpha} n^{O(1)} &\in \exp\left(O\left(p^\gamma \cdot \left(r^3k^{1+\varepsilon(\phi+1)}\right)^\alpha\right)\right) \cdot n^{O_r(1)} \\ &\in \exp\left(O\left(r^{3\alpha} \cdot k^{\varepsilon\gamma + \alpha(1+\varepsilon(\phi+1))}\right)\right) \cdot n^{O_r(1)} \end{aligned}$$

as follows: first we use Lemma 12 to delete irrelevant vertices and obtain an equivalent instance H on $|M'|$ vertices, then we use property $P_r(\phi, \gamma, \alpha)$ to bound the treewidth of H and then we solve the problem by dynamic programming on an approximate tree-decomposition in $2^{O(\text{tw}(H))} n^{O(1)}$ time by noting that every r -clique and every $D' \in \mathcal{D}'$ (which is an $(< r)$ -clique) has to be contained in a bag.

13:14 Kick the Cliques

So overall, computing \mathcal{Z} and solving the problem in each sub-instance takes time

$$\begin{aligned} & \exp\left(O_r\left((1+\varepsilon\phi)k^{1-\varepsilon}\log k + k^{\varepsilon\gamma+\alpha(1+\varepsilon(\phi+1))}\right)\right) \cdot n^{O_r(1)} \\ & \in \exp\left(O_r\left((1+\varepsilon\phi)k^{1-\varepsilon}\log k + k^{\varepsilon(\gamma+\alpha+\alpha\phi)+\alpha}\right)\right) \cdot n^{O_r(1)}. \end{aligned}$$

As we aim for algorithms where the contribution of k to the time complexity is of the form $2^{o(k)}$, the above bound sets the following constraint: $\varepsilon < \frac{1-\alpha}{\gamma+\alpha+\alpha\phi}$. Let $\varepsilon = \frac{1-\alpha-\delta}{\gamma+\alpha+\alpha\phi}$ for some constant $\delta \in (0, 1)$ that we will fix later. Then the above complexity becomes:

$$\exp\left(O_r\left((1+\varepsilon\phi)k^{1-\varepsilon}\log k + k^{1-\delta}\right)\right) \cdot n^{O_r(1)}.$$

We optimize (ignoring logarithmic factors) by choosing the value of δ so that $1 - \varepsilon = 1 - \delta$, i.e., $\delta = \frac{1-\alpha}{\gamma+\alpha(\phi+1)+1}$. This gives the following overall time bound:

$$2^{O_r((1+\varepsilon\phi)k^{\varepsilon'}\log k)} \cdot n^{O_r(1)} \quad \text{with } \varepsilon' = \frac{\gamma + \alpha(\phi + 2)}{\gamma + \alpha(\phi + 1) + 1}.$$

As $\alpha < 1$ we have $\varepsilon' < 1$ so the algorithm runs in subexponential FPT time, as desired. \blacktriangleleft

6 Applications

In this section we give applications of Theorem 22 to specific graph classes. First we have to show that the considered classes satisfy property P_r (recall that this property is defined at the beginning of Section 5).

6.1 Pseudo-disk graphs and map graphs

In this subsection we will prove the following lemma.

► **Lemma 23.** *Pseudo-disk graphs and map graphs have the property $P_r(r-2, 1/2, 1/2)$.*

As a consequence we get the two following results.

► **Theorem 5.** *There is an algorithm solving K_r -HITTING in pseudo-disk graphs in time $2^{O_r(k^{(r+1)/(r+2)}\log k)} \cdot n^{O_r(1)}$.*

► **Theorem 6.** *There is an algorithm solving K_r -HITTING in map graphs¹² in time $2^{O_r(k^{(r+1)/(r+2)}\log k)} \cdot n^{O_r(1)}$.*

To prove Lemma 23, we first need to state some external results. For $d \in \mathbb{N}$ we say that a graph G is d -degenerate if every subgraph of G (including G itself) has a vertex of degree at most d . In order to bound the number of small cliques in the considered graphs we can bound their degeneracy and then rely on the following result of Chiba and Nishizeki.

► **Theorem 24 ([6]).** *Any string graph G with n vertices and degeneracy d has $O(id^{i-1}n)$ i -cliques.*

Actually [6] gives a time bound for the enumeration of i -cliques in graphs of arboricity d . As arboricity and degeneracy are linearly bounded by each other and since the time bound implies a bound on the number of enumerated objects (up to a constant factor), we get the above statement.

¹²Map graphs are intersection graphs of interior-disjoint regions of \mathbb{R}^2 homeomorphic to disks.

► **Theorem 25** ([3]). *Pseudo-disk graphs on n vertices with clique number ω have at most $3e\omega n$ edges and treewidth $O(\sqrt{\omega n})$. In particular they are $(3e\omega)$ -degenerate.*

► **Theorem 26** ([5]). *Map graphs on n vertices with clique number ω have at most $7\omega n$ edges. In particular they are (7ω) -degenerate.*

To bound the treewidth of map graphs we use the following combination of results on balanced separators of string graphs of Lee and the links between separators and treewidth of Dvořák and Norin.

► **Theorem 27** ([12] and [8]). *Any m -edge string graph has treewidth $O(\sqrt{m})$.*

As a consequence of Theorem 26 and Theorem 27 we get the following.

► **Corollary 28.** *Map graphs on n vertices with clique number ω have treewidth $O(\sqrt{\omega n})$.*

Proof of Lemma 23. Let G be a pseudo-disk graph with n vertices and clique number ω . By Theorem 25 the pseudo-disk graphs with n vertices and clique number ω are $(3e\omega)$ -degenerate, so by Theorem 24 they have $O((r-1)^2(3e\omega)^{r-2}n)$ cliques of order less than r . So property P1 holds with $\phi = r - 2$. By Theorem 25 property P2 is satisfied with $\alpha = 1/2$ and $\gamma = 1/2$. The proof for map graphs is very similar, using Theorem 26 and Corollary 28. ◀

6.2 String graphs

We now move to string graphs where, as discussed in the introduction, forbidding large bicliques is necessary. Actually for $K_{t,t}$ -subgraph free string graphs the branching of Lemma 10 to reduce the clique number is not necessary in the algorithm of Theorem 22 as the number of small cliques and the treewidth are already suitably bounded. This explains the zeroes in Lemma 29 hereafter. In this subsection we will prove the following lemma.

► **Lemma 29.** *$K_{t,t}$ -subgraph-free string graphs have the property $P_r(0, 0, 1/2)$.*

As a consequence we get the following result.

► **Corollary 30.** *There is an algorithm solving K_r -HITTING in $K_{t,t}$ -subgraph-free string graphs in time*

$$2^{O_{t,r}(k^{2/3} \log k)} \cdot n^{O_r(1)}.$$

The contributions of t and r to the complexity in Corollary 30 are not explicit due to the way we stated the bound of Theorem 22 but can be extracted from the proof of this theorem.

► **Theorem 31** ([12]). *For every $t \in \mathbb{N}$, $K_{t,t}$ -subgraph-free string graphs on n vertices have degeneracy $O(t \log t)$ and treewidth $O(\sqrt{n \cdot t \log t})$.*

Proof of Lemma 29. Combining Theorem 24 and Theorem 31 we get that the number of $(< r)$ -cliques in $K_{t,t}$ -subgraph-free string graphs is $O_r((t \log t)^{r-2}n)$, i.e. P1 holds with $\phi = 0$. Theorem 31 gives P2 with $\alpha = 1/2$ and $\gamma = 0$. ◀

6.3 Minor-closed classes

In this subsection we will prove the following lemma.

► **Lemma 32.** *For every graph H , H -minor-free graphs have the property $P_r(0, 0, 1/2)$.*

13:16 Kick the Cliques

As a consequence we get the following result.

► **Theorem 9.** *For every graph H , there is an algorithm solving K_r -HITTING in H -minor-free graphs in time $2^{O_{H,r}(k^{2/3} \log k)} \cdot n^{O_r(1)}$.*

To prove Lemma 32, we first need to state some external results.

► **Theorem 33** (see [16]). *Every d -degenerate graph with $n \geq d$ vertices has at most $2^d(n - d + 1)$ cliques.*

► **Theorem 34** ([11, 15]). *For every h -vertex graph H there is a constant $d = O(h\sqrt{\log h})$ such that H -minor-free graphs are d -degenerate.*

► **Theorem 35** ([1]). *For every graph H , n -vertex H -minor-free graphs have treewidth $O_H(\sqrt{n})$.*

Proof of Lemma 32. As a consequence of Theorem 34 and Theorem 33, H -minor-free graphs have a linear number of cliques (regardless of their clique number). Hence they satisfy P1 with $\phi = 0$. By Theorem 35 they also satisfy P2 with $\alpha = 1/2$ and $\gamma = 0$. ◀

7 Open problems

As discussed in the introduction, our main result provides a generic way to obtain subexponential parameterized algorithms for K_r -HITTING, which can in particular be applied to several graph classes for which such algorithms were known from previous work (for TRIANGLE HITTING, the special case $r = 3$). Nevertheless there is still a gap between the running times of these different applications of our algorithm and the best time bounds for these specific classes. One can for instance compare our Theorem 5 (resp. Theorem 9) with the previous results corresponding to items 1 and 2 of Theorem 2 (resp. Theorem 1). It would be nice to match these known bounds, or to improve them when possible. More generally we can ask about the infimum ε such that K_r -HITTING can be solved in time $2^{O(k^\varepsilon)} n^{O(1)}$ in the classes we considered. We recall that under ETH, K_3 -HITTING cannot be solved in time $2^{o(\sqrt{n})}$ (so $\varepsilon \geq 1/2$) even for a very restricted subclass of string graphs (Theorem 3).

A second research direction is to understand for which graphs H our results can be extended to the H -HITTING problem (where one wants to hit any subgraph isomorphic to H). In disk graphs for example, it is already known [14] that there exist subexponential FPT algorithms for P_ℓ -HITTING when $\ell \leq 5$.

Recall that in this paper we gave sufficient conditions for a hereditary graph class to admit a subexponential FPT algorithm for K_r -HITTING. It remains an open problem to characterize such classes.

References

- 1 Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990. URL: <http://www.jstor.org/stable/1990903>.
- 2 Shinwoo An, Kyungjin Cho, and Eunjin Oh. Faster algorithms for cycle hitting problems on disk graphs. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures - 18th International Symposium, WADS 2023, Montreal, QC, Canada, July 31 - August 2, 2023, Proceedings*, volume 14079 of *Lecture Notes in Computer Science*, pages 29–42, Berlin, Heidelberg, 2023. Springer. doi:10.1007/978-3-031-38906-1_3.

- 3 Gaétan Berthe, Marin Bougeret, Daniel Gonçalves, and Jean-Florent Raymond. FVS for pseudo-disk graphs in subexponential FPT time. In *Proceedings of WG 2024*. LNCS, Springer, 2024.
- 4 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. doi:10.4230/LIPIcs.SWAT.2024.11.
- 5 Zhi-Zhong Chen, Michelangelo Grigni, and Christos H Papadimitriou. Map graphs. *Journal of the ACM (JACM)*, 49(2):127–138, 2002. doi:10.1145/506147.506148.
- 6 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223, 1985. doi:10.1137/0214017.
- 7 Erik D Demaine, Fedor V Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. *Journal of the ACM (JACM)*, 52(6):866–893, 2005. doi:10.1145/1101821.1101823.
- 8 Zdeněk Dvořák and Sergey Norin. Treewidth of graphs with balanced separations. *J. Comb. Theory B*, 137:137–144, 2019. doi:10.1016/j.jctb.2018.12.007.
- 9 Fedor V Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 748–759. SIAM, 2011. doi:10.1137/1.9781611973082.59.
- 10 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
- 11 A Kostochka. On the minimum of the Hadwiger number for graphs with given average degree. *Metody Diskret. Analiz.*, 38:37–58, 1982. English translation: AMS Translations (2), 132:15–32, 1986.
- 12 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 für Informatik. doi:10.4230/LIPIcs.ITCS.2017.1.
- 13 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.
- 14 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. Full version available at <https://sites.cs.ucsb.edu/~daniello/papers/subexpDiskOCTandFriends.pdf>. doi:10.1137/1.9781611977073.80.
- 15 Andrew Thomason. An extremal function for contractions of graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 95(2), pages 261–265. Cambridge University Press, 1984.
- 16 David R Wood. On the maximum number of cliques in a graph. *Graphs and Combinatorics*, 23(3):337–352, 2007. doi:10.1007/s00373-007-0738-8.