HITTING AND HARVESTING PUMPKINS

GWENAËL JORET, CHRISTOPHE PAUL, IGNASI SAU,
SAKET SAURABH, AND STÉPHAN THOMASSÉ

ABSTRACT. The *c-pumpkin* is the graph with two vertices linked by $c \geq 1$ parallel edges. A *c-pumpkin-model* in a graph G is a pair $\{A, B\}$ of disjoint subsets of vertices of G, each inducing a connected subgraph of G, such that there are at least c edges in G between A and B. We focus on hitting and packing c-pumpkin-models in a given graph in the realm of approximation algorithms and parameterized algorithms. We give an FPT algorithm running in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ deciding, for any fixed $c \geq 1$, whether all c-pumpkin-models can be hit by at most k vertices. This generalizes known single-exponential FPT algorithms for VERTEX COVER and FEEDBACK VERTEX SET, which correspond to the cases $c = 1, 2$ respectively. Finally, we present a $\mathcal{O}(\log n)$-approximation algorithm for both the problems of hitting all c-pumpkin-models with a smallest number of vertices, and packing a maximum number of vertex-disjoint c-pumpkin-models.

Keywords: Hitting and packing; parameterized complexity; approximation algorithm; single-exponential algorithm; iterative compression; graph minors.

1. INTRODUCTION

The *c-pumpkin* is the graph with two vertices linked by $c \geq 1$ parallel edges. A *c-pumpkin-model* in a graph G is a pair $\{A, B\}$ of disjoint subsets of vertices of G, each inducing a connected subgraph of G, such that there are at least c edges in G between A and B. In this article we study the problems of hitting all c-pumpkin-models of a given graph G with as few vertices as possible, and packing as many disjoint c-pumpkin-models in G as possible. As discussed below, these problems generalize several well-studied problems in algorithmic graph theory. We focus on FPT algorithms for the parameterized version of the hitting problem, as well as on polynomial-time approximation algorithms for the optimization version of both the packing and hitting problems.

FPT algorithms. From the parameterized complexity perspective, we study the following problem for every fixed integer $c \geq 1$.

p-c-PUMPKIN-HITTING (*p-c*-HIT)	
Instance:	A graph G and a non-negative integer k.
Parameter:	k.
Question:	Does there exist $S \subseteq V(G)$, $|S| \leq k$, such that $G \setminus S$ does not contain the c-pumpkin as a minor?

Several special cases of this problem are well studied in the parameterized complexity. When $c = 1$, the p-c-HIT problem is the p-VERTEX COVER problem [2, 9]. For $c = 2$, it is the p-FEEDBACK VERTEX SET problem [7, 12, 21]. When $c = 3$, this corresponds to the recently introduced p-DIAMOND HITTING SET problem [16].

The p-c-HIT problem can also be seen as a particular case of the following problem, recently introduced by Fomin *et al.* [19] and studied from the kernelization perspective: Let $\mathcal{F}$ be a finite set of graphs. In the p-$\mathcal{F}$-HIT problem, we are given an n-vertex graph G and an integer k as input, and asked whether at most k vertices can be deleted from G such that the resulting graph does not contain any graph from $\mathcal{F}$ as a minor. Among other results, it is proved in [19] that if $\mathcal{F}$ contains a c-pumpkin for some $c \geq 1$, then p-$\mathcal{F}$-HIT admits a kernel of size $\mathcal{O}(k^2 \log^{3/2} k)$. As discussed in Section 3, this kernel leads to a simple FPT algorithm for p-$\mathcal{F}$-HIT in this case, and in particular for p-c-HIT, with running time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$. A natural question is whether there exists an algorithm for p-c-HIT with running time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ for every fixed $c \geq 1$. Such algorithms are called *single-exponential*. For the p-VERTEX COVER problem the existence of single-exponential algorithms is well-known since almost the beginnings of the field of Parameterized Complexity [2], the best current algorithm being by Chen *et al.* [9]. On the other hand, the question about the existence of single-exponential algorithms for p-FEEDBACK VERTEX SET was open for a long time and was finally settled independently by Guo *et al.* [21] (using iterative compression) and by Dehne *et al.* [12]. The current champion algorithm for p-FEEDBACK VERTEX SET runs in time $\mathcal{O}(3.83^k k \cdot n^2)$ [7].

We present in Section 3 a single-exponential algorithm for p-c-HIT for every fixed $c \geq 1$, using a combination of a kernelization-like technique and iterative compression. Notice that this generalizes the above results for p-VERTEX COVER and p-FEEDBACK VERTEX SET. We remark that asymptotically these algorithms are optimal, that is, it is known that unless ETH fails neither p-VERTEX COVER nor p-FEEDBACK VERTEX SET admit an algorithm with running time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ [8, 22]. It is worth mentioning here that a similar quantitative approach was taken by Lampis [26] for graph problems expressible in MSOL parameterized by the sum of the formula size and the size of a minimum vertex cover of the input graph.

Approximation algorithms. For a fixed integer $c \geq 1$, we define the following two optimization problems.

MINIMUM c-PUMPKIN-HITTING (MIN c-HIT)
    Input: A graph G.
    Output: A subset $S \subseteq V(G)$ such that $G \setminus S$
               does not contain the c-pumpkin as a minor.
    Objective: Minimize $|S|$.

MAXIMUM c-PUMPKIN-PACKING (MAX c-PACK)
    Input: A graph G.
    Output: A collection $\mathcal{M}$ of vertex-disjoint subgraphs of G,
               each containing the c-pumpkin as a minor.
    Objective: Maximize $|\mathcal{M}|$.

Let us now discuss how the above problems encompass several well-known problems. For $c = 1$, MIN 1-HIT is the MINIMUM VERTEX COVER problem, which can be easily 2-approximated by finding any maximal matching, whereas MAX 1-PACK corresponds to finding a MAXIMUM MATCHING, which can be done in polynomial time. For $c = 2$,

Min 2-Hit is the Minimum Feedback Vertex Set problem, which can be also 2-approximated [1,3], whereas Max 2-Pack corresponds to Maximum Vertex-Disjoint Cycle Packing, which can be approximated to within a $\mathcal{O}(\log n)$ factor [25]. For $c = 3$, Min 3-Hit is the Diamond Hitting Set problem studied by Fiorini *et al.* in [16], where a 9-approximation algorithm is given.

We provide in Section 4 an algorithm that approximates both the Min c-Hit and the Max c-Pack problems to within a $\mathcal{O}(\log n)$ factor for every fixed $c \geq 1$. Note that this algorithm matches the best existing algorithms for Max c-Pack for $c = 2$ [25]. For the Min c-Hit problem, our result is only a slight improvement on the $\mathcal{O}(\log^{3/2} n)$-approximation algorithm given in [19]. However, for the Max c-Pack problem, there was no approximation algorithm known before except for the $c \leq 2$ case. Also, let us remark that, for $c \geq 2$ and every fixed $\varepsilon > 0$, Max c-Pack is quasi-NP-hard to approximate to within a $\mathcal{O}(\log^{1/2-\varepsilon} n)$ factor. For $c = 2$ this was shown by Friggstad and Salavatipour [20], and their result can be extended to the case $c > 2$ in the following straightforward way. Given an instance G of Max 2-Pack, we build an instance of Max c-Pack by replacing each edge of G with $c - 1$ parallel edges. This approximation preserving transformation shows that Max c-Pack is quasi-NP-hard to approximate to within a $\mathcal{O}(\log^{1/2-\varepsilon} n)$ factor for any $c \geq 2$.

The main ingredient of our approximation algorithm is the following combinatorial result: We show that every n-vertex graph G either contains a small c-pumpkin-model or has a structure that can be reduced in polynomial time, giving a smaller equivalent instance for both the Min c-Hit and the Max c-Pack problems. Here by a "small" c-pumpkin-model, we mean a model of size at most $f(c) \cdot \log n$ for some function f independent of n. This result extends one of Fiorini *et al.* [16], who dealt with the case $c = 3$.

2. Preliminaries

Graphs. We use standard graph terminology, see for instance [13]. All graphs in this article are finite and undirected, and may have parallel edges but no loops. We will sometimes restrict our attention to simple graphs, that is, graphs without parallel edges.

Given a graph G, we denote the vertex set of G by $V(G)$ and the edge set of G by $E(G)$. We use the shorthand $|G|$ for the number of vertices in G. For a subset $X \subseteq V(G)$, we use $G[X]$ to denote the subgraph of G induced by X. For a subset $Y \subseteq E(G)$ we let $G[Y]$ be the graph with $E(G[Y]) := Y$ and with $V(G[Y])$ being the set of vertices of G incident with some edge in Y. For a subset $X \subseteq V(G)$, we may use the notation $G \setminus X$ to denote the graph $G[V(G) \setminus X]$.

The set of neighbors of a vertex v of a graph G is denoted by $N_G(v)$. The *degree* $\deg_G(v)$ of a vertex $v \in V(G)$ is defined as the number of edges incident with v (thus parallel edges are counted). We write $\deg_G^*(v)$ for the number of neighbors of v, that is, $\deg_G^*(v) := |N_G(v)|$. Similarly, given a subgraph $H \subseteq G$ with $v \in V(H)$, we can define in the natural way $N_H(v)$, $\deg_H(v)$, and $\deg_H^*(v)$, that is, $N_H(v) = N_G(v) \cap V(H)$, $\deg_H(v)$ is the number of edges incident with v with both endpoints in H, and $\deg_H^*(v) = |N_H(v)|$. In these notations, we may drop the subscript if the graph is clear from the context. By the *neighbors of a subgraph* $H \subseteq G$ we mean the set of vertices in $V(G) \setminus V(H)$ that have at least one neighbor in H. The minimum degree of a vertex in a graph G is denoted $\delta(G)$, and the maximum degree of a vertex in G is denoted $\Delta(G)$. We use the notation $\mathbf{cc}(G)$ to denote the number of connected components of G. Also, we let $\mu(G)$ denote the maximum multiplicity of an edge in G. A graph is said to be a *multipath* if its underlying simple graph (without parallel edges) is isomorphic to a path.

Minors and models. Given a graph G and an edge $e \in E(G)$, let $G \backslash e$ be the graph obtained from G by removing the edge e, and let G/e be the graph obtained from G by contracting e (we note that parallel edges resulting from the contraction are kept but self loops are deleted). If H can be obtained from a subgraph of G by a (possibly empty) sequence of edge contractions, we say that H is a *minor* of G, and we denote it by $H \preceq_m G$. A graph G is *H-minor-free*, or simply *H-free*, if G does not contain H as a minor. A *model* of a graph H, or simply *H-model*, in a graph G is a collection $\{S_v \subseteq V(G) \mid v \in V(H)\}$ such that

(i) $G[S_v]$ is connected for every $v \in V(H)$;

(ii) S_v and S_w are disjoint for every two distinct vertices v, w of H; and

(iii) there are at least as many edges between S_v and S_w in G as between v and w in H, for every $vw \in E(H)$.

The *size* of the model is defined as $\sum_{v \in V(H)} |S_v|$. Clearly, H is a minor of G if and only if there exists a model of H in G. In this paper, we will almost exclusively consider H-models with H being isomorphic to a c-pumpkin for some $c \geq 1$. Thus a c-pumpkin-model in a graph G is specified by an unordered pair $\{A, B\}$ of disjoint subsets of vertices of G, each inducing a connected subgraph of G, such that there are at least c edges in G between A and B. A c-pumpkin-model $\{A, B\}$ of G is said to be *minimal* if there is no c-pumpkin-model $\{A', B'\}$ of G with $A' \subseteq A$, $B' \subseteq B$, and $|A'| + |B'| < |A| + |B|$.

A subset X of vertices of a graph G such that $G \backslash X$ has no c-pumpkin-minor is called a *c-pumpkin-hitting set*, or simply *c-hitting set*. We denote by $\tau_c(G)$ the minimum size of a c-pumpkin-hitting set in G. A collection $\mathcal{M}$ of vertex-disjoint subgraphs of a graph G, each containing a c-pumpkin-model, is called a *c-pumpkin-packing*, or simply *c-packing*. We denote by $\nu_c(G)$ the maximum size of a c-pumpkin-packing in G. Obviously, for any graph G it holds that $\nu_c(G) \leq \tau_c(G)$, but the converse is not necessarily true.

The following lemma on models will be useful in our algorithms. The proof is straightforward and hence is omitted.

Lemma 2.1. *Suppose G' is obtained from a graph G by contracting some vertex-disjoint subgraphs of G, each of diameter at most k. Then, given an H-model in G' of size s, one can compute in polynomial time an H-model in G of size at most $k \cdot \Delta(H) \cdot s$.*

Parameterized algorithms. A *parameterized problem* Π is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet Γ. An instance of a parameterized problem consists of a pair (x, k), where k is called the *parameter*. A central notion in parameterized complexity is *fixed parameter tractability (FPT)*, which means, for a given instance (x, k), solvability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size.

A *kernelization algorithm* or, in short, a *kernel* for a parameterized problem $\Pi \subseteq \Gamma^* \times \mathbb{N}$ is an algorithm that given $(x, k) \in \Gamma^* \times \mathbb{N}$ outputs in time polynomial in $|x| + k$ a pair $(x', k') \in \Gamma^* \times \mathbb{N}$ such that

(i) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$; and

(ii) $\max\{|x'|, k'\} \leq g(k)$,

where g is some computable function. The function g is referred to as the *size* of the kernel. If $g(k) = k^{\mathcal{O}(1)}$ or $g(k) = \mathcal{O}(k)$, then we say that Π admits a polynomial kernel and a linear kernel, respectively.

Iterative compression is a tool that has been used successfully in finding fast FPT algorithms for a number of parameterized problems. The main idea behind iterative compression is an algorithm which, given a solution of size $k + 1$ for a problem, either compresses it to a solution of size k or proves that there is no solution of size k. This technique was first introduced by Reed *et al.* to solve the ODD CYCLE TRANSVERSAL

problem [28], where one is interested in finding a set of at most k vertices whose deletion makes the graph bipartite [28]. Since then, it has been extensively used in the literature, see for instance [10, 18, 21].

See [14] for detailed introduction to Parameterized Complexity.

Tree-width. We briefly recall the definition of the tree-width of a graph. A *tree decomposition* of a graph G is an ordered pair $(T, \{W_x \mid x \in V(T)\})$ where T is a tree and $\{W_x \mid x \in V(T)\}$ a family of subsets of $V(G)$ (called *bags*) such that

(i) $\bigcup_{x \in V(T)} W_x = V(G)$;
(ii) for every edge $uv \in E(G)$, there exists $x \in V(T)$ with $u, v \in W_x$; and
(iii) for every vertex $u \in V(G)$, the set $\{x \in V(T) \mid u \in W_x\}$ induces a subtree of T.

The *width* of tree decomposition $(T, \{W_x \mid x \in V(T)\})$ is $\max\{|W_x| - 1 \mid x \in V(T)\}$. The *tree-width* $\mathbf{tw}(G)$ of G is the minimum width among all tree decompositions of G. We refer the reader to Diestel's book [13] for an introduction to the theory of tree-width. It is an easy exercise to check that the tree-width of a simple graph is an upper bound on its minimum degree. This implies the following lemma.

Lemma 2.2. *Every n-vertex simple graph with tree-width k has at most $k \cdot n$ edges.*

We will need the following result of Bodlaender *et al.* [6].

Theorem 2.3 (Bodlaender *et al.* [6]). *Every graph not containing a c-pumpkin as a minor has tree-width at most $2c - 1$.*

The following corollary is an immediate consequence of the above theorem.

Corollary 2.4. *Every n-vertex graph (may contain parallel edges) with no minor isomorphic to a c-pumpkin has at most $(c - 1) \cdot (2c - 1) \cdot n$ edges.*

3. A SINGLE-EXPONENTIAL FPT ALGORITHM

As mentioned in the introduction, it is proved in [19] that given an instance (G, k) of p-$\mathcal{F}$-HIT such that $\mathcal{F}$ consists of only a c-pumpkin for some $c \geq 1$, one can obtain in polynomial time an equivalent instance with $\mathcal{O}(k^2 \log^{3/2} k)$ vertices. This kernel leads to the following simple FPT algorithm for p-c-HIT: First compute the kernel K in polynomial time, and then for every subset $S \subseteq V(K)$ of size k, test whether $K[V(K) \setminus S]$ contains a c-pumpkin as a minor, using the polynomial-time algorithm of Robertson and Seymour [29] (or the recent faster version of Kawarabayashi *et al.* [23]). If for some S we have that $K[V(K) \setminus S]$ does not contain c-pumpkin as a minor, we answer YES; otherwise the answer is NO. The running time of this algorithm is clearly bounded by $\binom{k^2 \log^{3/2} k}{k} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$.

In this section we give an algorithm for p-c-PUMPKIN-HITTING that runs in time $d^k \cdot n^{\mathcal{O}(1)}$ for any fixed $c \geq 1$, where d only depends on the fixed constant c. Towards this, we first introduce a variant of p-c-PUMPKIN-HITTING, namely p-c-PUMPKIN-DISJOINT HITTING, formally defined as follows.

---

p-c-PUMPKIN-DISJOINT HITTING (p-c-DISJOINT HIT for short)

Instance: A graph G, a non-negative integer k, and a set $S \subseteq V(G)$ with $|S| \leq k + 1$, such that $G \setminus S$ does not contain the c-pumpkin as a minor.

Parameter: k.

Question: Does there exist $S' \subseteq V(G) \setminus S$, with $|S'| \leq k$, such that $G \setminus S$ does not contain the c-pumpkin as a minor?

Next we show a lemma that allows us to relate the two problems mentioned above.

Lemma 3.1. *If p-c-DISJOINT HIT can be solved in time $\eta(c)^k \cdot n^{\mathcal{O}(1)}$, then p-c-HIT can be solved in time $(\eta(c) + 1)^k \cdot n^{\mathcal{O}(1)}$.*

Proof. Let $\mathcal{A}$ be an FPT algorithm which solves the p-c-DISJOINT HIT problem in time $\eta(c)^k \cdot n^{\mathcal{O}(1)}$. Let (G, k) be an input graph for the p-c-HIT problem, and let $v_1, \ldots, v_n$ be an arbitrary ordering of $V(G)$. Let V_i and G_i, respectively, denote the subset $\{v_1, \ldots, v_i\}$ of vertices and the induced subgraph $G[V_i]$. We iterate over $i = 1, \ldots, n$ in the following way. At the i-th iteration, suppose we have a c-hitting set $S_i \subseteq V_i$ of G_i of size at most k. At the next iteration, we set $S_{i+1} := S_i \cup \{v_{i+1}\}$ (note that S_{i+1} is a c-hitting set for G_{i+1} of size at most $k+1$). If $|S_{i+1}| \leq k$, we can safely move on to the $(i+2)$-th iteration. If $|S_{i+1}| = k+1$, we look at every subset $S \subseteq S_{i+1}$ and check whether there is a c-hitting set W of size at most k such that $W \cap S_{i+1} = S_{i+1} \setminus S$. To do this, we use the FPT algorithm $\mathcal{A}$ for p-c-DISJOINT HIT on the instance (H, S), with $H = G_{i+1} \setminus (S_{i+1} \setminus S)$. If $\mathcal{A}$ returns a c-hitting set W of H with $|W| < |S|$, then observe that the vertex set $(S_{i+1} \setminus S) \cup W$ is a c-hitting set of G of size strictly smaller than S_{i+1}. If there is a c-hitting set of G_{i+1} of size strictly smaller than S_{i+1}, then for some $S \subseteq S_{i+1}$, there is a small c-hitting set in $G_{i+1} \setminus (S_{i+1} \setminus S)$ disjoint from S, and $\mathcal{A}$ correctly returns a solution. If no such small c-hitting set exists, the algorithm returns NO. Let us now argue about the running time of this algorithm. The time required to execute $\mathcal{A}$ for every subset S at the i-th iteration is $\sum_{i=0}^{k+1} \binom{k+1}{i} \cdot \eta(c)^i \cdot n^{\mathcal{O}(1)} = (\eta(c) + 1)^{k+1} \cdot n^{\mathcal{O}(1)}$. That is, we have an algorithm for p-c-HIT running in time $(\eta(c) + 1)^k \cdot n^{\mathcal{O}(1)}$, as we wanted to prove. $\square$

Lemma 3.1 allows us to focus on p-c-DISJOINT HIT. In what follows we give an algorithm for p-c-DISJOINT HIT that runs in single-exponential time. In fact, we obtain a linear kernel for p-c-DISJOINT HIT, which clearly yields a single-exponential algorithm.

Overview of the algorithm. The algorithm for p-c-DISJOINT HIT is based on a combination of polynomial-time preprocessing and a protrusion-based reduction rule. Let (G, S, k) be the given instance of p-c-DISJOINT HIT and let $V := V(G)$. Our main objective is to show that, after applying some simple polynomial-time reduction rules, $\{v \in V \setminus S : N_G(v) \cap S \neq \emptyset\}$ has cardinality $\mathcal{O}(k)$; the proof of this fact, specially Lemma 3.5, is the most technical part of this section. Once we have the desired upper bound, we use a protrusion-based reduction rule adapted from [19] to give a polynomial-time procedure that, given an instance (G, S, k) of p-c-DISJOINT HIT, returns an equivalent instance (G', S, k') such that G' has $\mathcal{O}(k)$ vertices. That is, we obtain a linear vertex kernel for p-c-DISJOINT HIT[1]. Notice that once we have a linear vertex kernel of size αk for p-c-DISJOINT HIT, we can solve the problem in $\binom{\alpha k}{k} \cdot k^{\mathcal{O}(1)}$. We can now proceed to the detailed description of the algorithm.

Pre-processing step. We start by defining two sets. Our first objective is to upper bound the cardinality of these two sets by $\mathcal{O}(k)$.

$$
\begin{aligned}
V_1 &:= \{v \in V \setminus S : |N_G(v) \cap S| = 1\} \\
V_{\geq 2} &:= \{v \in V \setminus S : |N_G(v) \cap S| \geq 2\}.
\end{aligned}
$$

We start with some simple polynomial-time reduction rules (depending on c) that will be applied in the compression routine. We also prove, together with the description of each rule, that they are valid for our problem.

---

[1]This was not the case in the conference version of this article, in which the algorithm for p-c-DISJOINT HIT was considerably more complicated, involving in particular a branching procedure and a more extensive usage of the protrusion-based reduction rule.
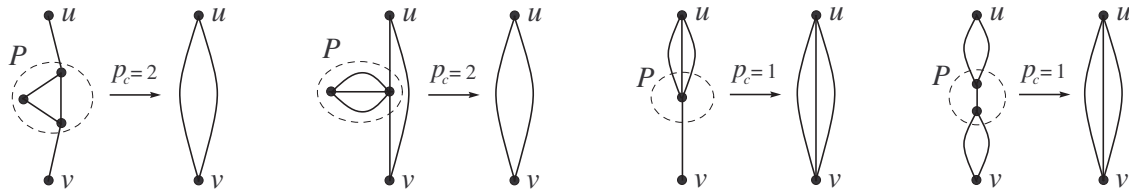
FIGURE 1. Four examples of reduction rule **R3** for $c = 4$.

R1 Suppose that C is a connected component of $G \setminus S$ with no neighbor in S. Then delete C.

Proof of correctness. The deletion of C can be safely done, as its vertices will never participate in a c-pumpkin-model. □

R2 Suppose that C is a connected component of $G \setminus S$ with exactly one neighbor v in S, and such that $G[V(C) \cup \{v\}]$ is c-pumpkin-free. Then delete C.

Proof of correctness. In this case C can be also safely deleted, as its vertices will never participate in a minimal c-pumpkin-model. □

R3 Let $u \in S$, $v \in V(G) \setminus S$, and P be a (non-empty) connected component of $G \setminus \{u, v\}$ entirely contained in $G \setminus S$. Suppose that $G[V(P) \cup \{u\}]$ is c-pumpkin-free. We distinguish two cases.

    (i) If $G[V(P) \cup \{u, v\}]$ contains a c-pumpkin-minor, include vertex v in the solution, and decrease the parameter by one.

    (ii) Otherwise, let H_p be the graph obtained from $G[V(P) \cup \{u, v\}]$ by adding p parallel edges between u and v, and let p_c be the smallest integer p such that H_p contains a c-pumpkin-minor. Then replace P with $c - p_c$ parallel edges between u and v. See Figure 1 for some small examples for $c = 4$.

Proof of correctness. In case (i), since $G[V(P) \cup \{u\}]$ is c-pumpkin-free and $G[V(P) \cup \{u, v\}]$ contains a c-pumpkin-minor, then at least one vertex in $V(P) \cup \{v\}$ needs to be included in the solution, and there is no need to include any vertex of P. Therefore, vertex v can be safely included in the solution.

In case (ii), note also that in order to hit all c-pumpkins-models intersecting P, there is no need to include any vertex of P in the solution, as any such vertex could be replaced with v, obtaining another solution with equal or smaller size. We say that two c-pumpkins-models in G are *P-equivalent* if they coincide except, possibly, for vertices in P. Let G' be the graph obtained from G by replacing P with $c - p_c$ parallel edges between u and v. By construction, G and G' contain exactly the same c-pumpkins-models modulo the P-equivalence relation. As we can assume that no vertex of P is included in the solution, we conclude that the reduction rule yields an equivalent instance. □

We would like to stress that Lemma 4.1 in Section 4.1 proves a stronger statement that the correctness of rule **R3**.

We say that the instance (G, S, k) is (S, c)-*reduced* if rules **R1**, **R2**, or **R3** cannot be applied anymore. Note that reduction rule **R1** can easily be applied in polynomial time. For reduction rules **R2** and **R3**, we have to test whether the considered graph contains a c-pumpkin-minor. Towards this, note that the graphs on which we need to test whether they contain a c-pumpkin-minor have treewidth upper-bounded by $2c$. Thus in linear time we can test whether such graphs contain a c-pumpkin-minor [11]. This implies that reduction rules **R2** and **R3** can also be applied in polynomial time. Note that

the existence of a c-pumpkin-minor can also be tested, in a less efficient way, using the polynomial-time algorithm of Robertson and Seymour [29].

The following Lemmas 3.2 and 3.5 are key to the analysis of our algorithm. We also need two intermediate technical results stated in Lemmas 3.3 and 3.4, which will be used in the proof of Lemma 3.5.

Lemma 3.2. *There is a function $f \colon \mathbb{N} \to \mathbb{N}$ such that if (G, S, k) is a* Yes-*instance to the p-c-*Disjoint Hit *problem, then $|V_{\geq 2}| \leq f(c) \cdot k$.*

Proof. In order to upper-bound $|V_{\geq 2}|$, we build from $G[S]$ the following auxiliary graph H: we start with $H = (S, E(G[S]))$, and for each vertex $v \in V_{\geq 2}$ with neighbors $u_1, \ldots, u_\ell$, $\ell \geq 2$, we add to H an edge e_v between two arbitrary neighbors u_1, u_2 of v. Note that $H \preceq_m G$, and that for each vertex $v \in V_{\geq 2}$, $H \setminus e_v \preceq_m G \setminus v$. We now argue that $|E(H)|$ is linearly bounded by k, which implies the desired result as by construction $|V_{\geq 2}| \leq |E(H)|$. If G is a Yes-instance, there must be a set $S' \subseteq V \setminus S$, $|S'| \leq k$, such that $G \setminus S'$ is c-pumpkin-free. By construction of H, the removal of each vertex $v \in S' \cap V_{\geq 2}$ corresponds to the removal of the edge $e_v \in E(H)$. Let $H' = H \setminus \{e_v : v \in S' \cap V_{\geq 2}\}$, and note that $H' \preceq_m G \setminus S'$, so H' is c-pumpkin-free. Therefore, by Corollary 2.4 it follows that $|E(H')| \leq (c - 1) \cdot (2c - 1) \cdot |V(H')| \leq (c - 1) \cdot (2c - 1) \cdot (k + 1)$. As $|E(H)| \leq |E(H')| + k$, we conclude that $|V_{\geq 2}| \leq |E(H)| \leq (c-1) \cdot (2c-1) \cdot (k+1) + k$. $\square$

Lemma 3.3. *There is a function $g \colon \mathbb{N} \to \mathbb{N}$ such that if (G, S, k) is a* Yes-*instance to the p-c-*Disjoint Hit *problem and $\mathcal{C}$ is a collection of disjoint connected subgraphs of $G \setminus S$ such that each subgraph has at least two distinct neighbors in S, then $|\mathcal{C}| \leq g(c) \cdot k$.*

Proof. The proof is very similar to the proof of Lemma 3.2. In order to upper-bound $|\mathcal{C}|$, we build from $G[S]$ the following auxiliary graph H: we start with $H = (S, E(G[S]))$, and for each subgraph $C \in \mathcal{C}$ with neighbors $u_1, \ldots, u_\ell$, $\ell \geq 2$, we add to H an edge e_C between two arbitrary neighbors u_1, u_2 of C. Note that $H \preceq_m G$, and that for each subgraph $C \in \mathcal{C}$, $H \setminus e_C \preceq_m G \setminus C$. We now argue that $|E(H)|$ is linearly bounded by k, which implies the desired result as by construction $|\mathcal{C}| \leq |E(H)|$. If G is a Yes-instance, there must be a set $S' \subseteq V \setminus S$, $|S'| \leq k$, such that $G \setminus S'$ is c-pumpkin-free. By construction of H, the removal of a vertex v in a subgraph $C \in \mathcal{C}$ corresponds to the removal of at most one edge $e_C \in E(H)$ (as maybe the edge e_C can still be *simulated* after the removal of v). Let H' be the subgraph obtained from H after the removal of those edges, and note that $H' \preceq_m G \setminus S'$, so H' is c-pumpkin-free. Therefore, by Corollary 2.4 it follows that $|E(H')| \leq (c - 1) \cdot (2c - 1) \cdot |V(H')| \leq (c - 1) \cdot (2c - 1) \cdot (k + 1)$. As $|E(H)| \leq |E(H')| + k$, we conclude that $|\mathcal{C}| \leq |E(H)| \leq (c-1) \cdot (2c-1) \cdot (k+1) + k$. $\square$

Lemma 3.4. *In an (S, c)-reduced* Yes-*instance (G, S, k), the number of connected components of $G \setminus S$ is $\mathcal{O}(k)$.*

Proof. Note that by reduction rules **R1** and **R2**, we can assume that each connected component C of $G \setminus S$ has some neighbor in S, and that if C has exactly one neighbor v in S, then $G[V(C) \cup \{v\}]$ has a c-pumpkin. On the one hand, the number of components C that have exactly one neighbor v in S and such that $G[V(C) \cup \{v\}]$ contains the c-pumpkin as a minor is at most k, as any solution needs to contain at least one vertex from each such connected component. On the other hand, the number of components that have at least two neighbors in S is $\mathcal{O}(k)$ by Lemma 3.3. $\square$

Now we prove our key structural lemma.

Lemma 3.5. *There is a function $h \colon \mathbb{N} \to \mathbb{N}$ such that if (G, S, k) is an (S, c)-reduced* Yes-*instance to the p-c-*Disjoint Hit *problem, then $|V_1| \leq h(c) \cdot k$.*

Proof. For simplicity we call the vertices in V_1 *white*. We proceed to find a packing of disjoint connected subgraphs $\mathcal{P} = \{B_1, \ldots, B_\ell\}$ of $G \setminus S$ containing all white vertices except for $\mathcal{O}(k)$ of them. This will help us in bounding $|V_1|$. We call the subgraphs in $\mathcal{P}$ *blocks*. For a graph $H \subseteq G \setminus S$, let $w(H)$ be the number of white vertices in H. The idea is to obtain, as far as possible, blocks B with $c \leq w(B) \leq c^3$; we call these blocks *suitable*, and the other blocks are called *unsuitable*. If at some point we cannot refine the packing anymore in order to obtain suitable blocks, we will argue about its structural properties, which will allow us to bound the number of white vertices.

We start with $\mathcal{P}$ containing all the connected components C of $G \setminus S$ such that $w(C) > c^3$, and we recursively try to refine the current packing. By Lemma 3.4, we know that the number of connected components is $\mathcal{O}(k)$, and hence the number of white vertices that are not included in $\mathcal{P}$ is $\mathcal{O}(c^3 k) = \mathcal{O}(k)$.

More precisely, for each block B with $w(B) > c^3$, we build a spanning tree T of B, and we orient each edge $e \in E(T)$ towards the components of $T \setminus \{e\}$ containing at least c white vertices. Note that, as $w(B) > c^3$, each edge gets at least one orientation, and that edges may be oriented in both directions. If some edge $e \in E(T)$ is oriented in both directions, we replace in $\mathcal{P}$ block B with the subgraphs induced by the vertices in each of the two subtrees. We stop this recursive procedure whenever we cannot find more suitable blocks using this orientation trick. Let $\mathcal{P}$ be the current packing.

Now let B be an unsuitable block in $\mathcal{P}$, that is, $w(B) > c^3$ and no edge of its spanning tree T is oriented in both directions. This implies that there exists a vertex $v \in V(T)$ with all its incident edges pointing towards it. We call such a vertex v a *sink*. Let $T_1, \ldots, T_p$ be the connected components of $T \setminus \{v\}$. Note that as v is a sink, $w(T_i) < c$ for $1 \leq i \leq p$, using the fact that $w(B) > c^3$ we conclude that $p \geq c^2$. Now let $P_1, \ldots, P_\ell$ be the connected components of $G[V(T_1) \cup \cdots \cup V(T_p)] = G[V(B) \setminus \{v\}]$, and note that $\ell \leq p$. We call these subgraphs P_i the *pieces* of the unsuitable block B. For each unsuitable block, we delete the pieces with no white vertex. This completes the construction of $\mathcal{P}$. The next claim bounds the number of white vertices in each piece of an unsuitable block in $\mathcal{P}$.

Claim 1. *Each of the pieces of an unsuitable block contains less than c^2 white vertices.*

Proof. Assume for contradiction that there exists a piece P of an unsuitable block with $w(P) \geq c^2$, and let v be the sink of the unsuitable block obtained from tree T. By construction, $V(P)$ is the union of the vertices in some of the trees in $T \setminus \{v\}$; let without loss of generality these trees be $T_1, \ldots, T_q$. As $w(P) \geq c^2$ and $w(T_i) < c$ for $1 \leq i \leq q$, it follows that $q \geq c$. As v has at least one neighbor in each of the trees T_i, $1 \leq i \leq q$, and P is a connected subgraph of G, we can obtain a c-pumpkin-model $\{A, B\}$ in $G \setminus S$ by setting $A := \{v\}$ and $B := V(P)$, contradicting the fact that $G \setminus S$ is c-pumpkin-free. See Figure 2(a) for an example for $c = 3$. $\qquad\square$

Hence in the packing $\mathcal{P}$ we are left with a set of suitable blocks with at most c^3 white vertices each, and a set of unsuitable blocks, each one broken up into pieces linked by a sink in a star-like structure. By Claim 1, each piece of the remaining unsuitable blocks contains at most c^2 white vertices. See Figure 2(b) for an example of an unsuitable block B for $c = 3$. Note that in this figure, piece P_2 must have at least one neighbor in another block B' (suitable or unsuitable), as otherwise we could have applied case (i) of reduction rule **R3**.

Now we need two claims about the properties of the constructed packing.

Claim 2. *In the packing $\mathcal{P}$ constructed above, the number of suitable or unsuitable blocks is $\mathcal{O}(k)$.*
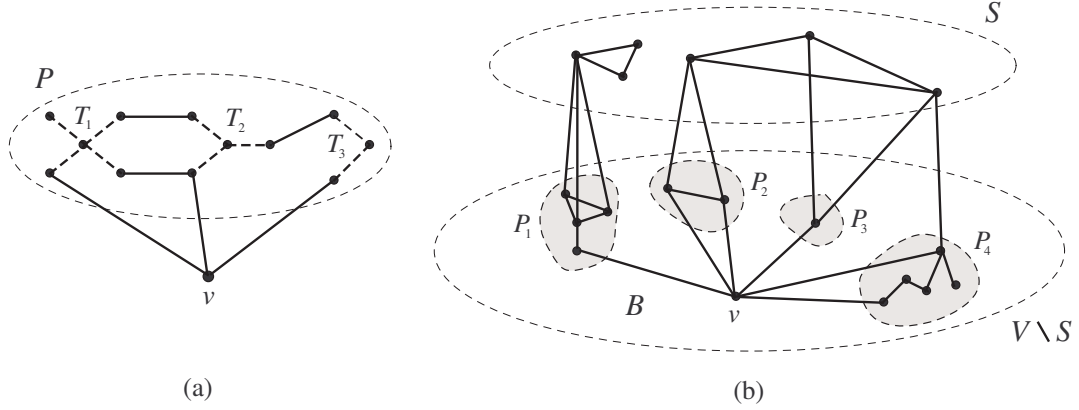
FIGURE 2. (a) Example for $c = 3$ of the contradiction in the proof of Claim 1. The trees T_1, T_2, and T_3 are defined by the dashed edges. (b) Example for $c = 3$ of an unsuitable block B in the packing $\mathcal{P}$ built in the proof of Lemma 3.5.

Proof. We first bound the number of suitable blocks, and for this we distinguish between two types of suitable blocks.

- Type 1: blocks that have exactly one neighbor in S. For each such block we need to include a vertex of it in the c-hitting set (as each suitable block contains at least c white vertices), so their number is at most k.
- Type 2: blocks that have at least two distinct neighbors in S. The number of such blocks is $\mathcal{O}(k)$ by Lemma 3.3.

The proof for unsuitable blocks is similar. We distinguishing between the same two types of blocks, and use the fact that each unsuitable block contains at least $c^3 \geq c$ white vertices. This concludes the proof. $\square$

Claim 3. *In an (S, c)-reduced YES-instance, the total number of pieces in the packing $\mathcal{P}$ constructed above is $\mathcal{O}(k)$.*

Proof. We distinguish between three types of pieces.

- Type 1: pieces that have at least two distinct neighbors in S (see piece P_3 in Figure 2(b) for $c = 3$). The number of pieces of this type is $\mathcal{O}(k)$ by Lemma 3.3.
- Type 2: pieces that are not of type 1 and that have at least one neighbor in some suitable block or in another unsuitable block (note that by construction a piece cannot have any neighbor in other pieces of the same unsuitable block). We construct an auxiliary graph H as follows: we start with the packing $\mathcal{P}$, and we add all the edges in $G \setminus S$ between vertices in different blocks of $\mathcal{P}$ (suitable or unsuitable). Then we contract each block to a single vertex, and let H be the resulting graph. By Claim 2, $|V(H)| = \mathcal{O}(k)$. As $H \preceq_m G \setminus S$ and $G \setminus S$ is c-pumpkin-free, by Corollary 2.4 we have that $|E(H)| = \mathcal{O}(k)$. Since each piece of type 2 is incident with at least one edge of H after uncontracting the blocks, it follows that the number of pieces of type 2 is at most $2|E(H)| = \mathcal{O}(k)$.
- Type 3: the remaining pieces. That is, these are pieces P that see exactly one vertex u in S, and that are connected to the rest of $G \setminus S$ only through the corresponding sink v. In other words, such a piece P is a connected component of $G \setminus \{u, v\}$. We distinguish two subcases.

○ If $G[V(P) \cup \{u\}]$ contains a c-pumpkin-minor, then any c-hitting set needs to contain at least one vertex in P (see piece P_1 in Figure 2(b) for $c = 3$). We conclude that the number of pieces of this subtype is at most k.

○ Otherwise, all the conditions of case (ii) of reduction rule **R3** are fulfilled, and we conclude that such a piece cannot exist in an (S, c)-*reduced* instance.

□

To conclude, recall that the constructed packing $\mathcal{P}$ contains all but $\mathcal{O}(k)$ white vertices, either in suitable blocks or in pieces of unsuitable blocks. As by construction each suitable block has at most c^3 white vertices and by Claim 2 the number of such blocks is $\mathcal{O}(k)$, it follows that the number of white vertices contained in suitable blocks is $\mathcal{O}(k)$. Similarly, by Claim 1 each piece contains at most c^2 white vertices, and the total number of pieces is $\mathcal{O}(k)$ by Claim 3, so the number of white vertices contained in pieces of unsuitable blocks is also $\mathcal{O}(k)$. □

Linear kernel. We now proceed to describe a procedure, called *protrusion rule*, that bounds the size of our graph when $|V_1 \cup V_{\geq 2}| = \mathcal{O}(k)$. We first need some definitions.

Many graph optimization problems can be expressed as finding an optimal number of vertices or edges satisfying a property expressible in Monadic Second Order (MSO) logic. A parameterized graph problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is given with a graph G and an integer k as an input. When the goal is to decide whether there exists a subset W of at most k vertices for which an MSO-expressible property $P_\Pi(G, W)$ holds, we say that Π is a *p-min-MSO* graph problem. One can easily check that the p-c-HIT problem is p-min-MSO. In the (parameterized) *disjoint version* Π^d of a p-min-MSO problem Π, we are given a triple (G, S, k), where G is a graph, S a subset of $V(G)$ and k the parameter, and we seek for a solution set W which is disjoint from S, and whose size is at most k.

Given $R \subseteq V(G)$, we define $\partial_G(R)$ as the set of vertices in R that have a neighbor in $V(G) \setminus R$. Thus the neighborhood of R is $N_G(R) = \partial_G(V(G) \setminus R)$. We say that a set $X \subseteq V(G)$ is an *r-protrusion* of G if $\mathbf{tw}(G[X]) \leq r$ and $|\partial_G(X)| \leq r$.

An important concept when applying protrusion-based reduction rules is *strong monotonicity* of a problem, which we do not define here (see for instance [4]). What we will need is the following fact, which can be found in [4, proof of Lemma 13]: if $\mathcal{F}$ is a finite set of connected planar graphs, then the p-$\mathcal{F}$-HIT problem is strongly monotone. As the c-pumpkin is a connected planar graph for any $c \geq 1$, we immediately have the following lemma.

Lemma 3.6. *The p-c-HIT problem is strongly monotone.*

The following lemma is key to our protrusion-based reduction rule. Its proof follows basically from the framework introduced in [4], although some details need to be modified when dealing with the disjoint version of a problem, as it is our case. A self-contained proof for the specific case of disjoint problems can be found in [24, Lemma 12]. The general statement deals with the disjoint version of a general (parameterized) strongly monotone p-min-MSO problem. As the p-c-HIT problem is p-min-MSO, and it is strongly monotone by Lemma 3.6, we only state the lemma for the specific case of our problem.

Lemma 3.7 (Kim *et al.* [24]). *Let Π^d be the disjoint version of p-c-HIT. There exists a computable function $\gamma : \mathbb{N} \to \mathbb{N}$ and an algorithm that given:*

- *an instance (G, S, k) of Π^d such that $G \setminus S$ is c-pumpkin-minor-free, and*
- *a t-protrusion X of G such that $|X| > \gamma(2t + 1)$ and $X \cap S = \emptyset$,*

in time $\mathcal{O}(|X|)$ outputs an instance (G', S, k') such that $|V(G')| < |V(G)|$, $k' \leq k$, $(G', S, k') \in \Pi^d$ if and only if $(G, S, k) \in \Pi^d$, and $G' \setminus S$ is c-pumpkin-minor-free.

We are now ready the state the protrusion rule. It follows as a corollary of Lemma 3.7 that the following reduction rule for p-c-DISJOINT HIT is safe.

> **P** Let (G, S, k) be an instance of p-c-DISJOINT HIT and let $\gamma : \mathbb{N} \to \mathbb{N}$ be the computable function given by Lemma 3.7. Let X be a $4c$-protrusion of G with $|X| > \gamma(8c + 1)$ and such that $X \cap S = \emptyset$. Then use the algorithm given by Lemma 3.7 to compute in time $\mathcal{O}(|X|)$ an equivalent instance (G', S, k') such that $G[S]$ and $G'[S]$ are isomorphic, $G' \setminus S$ is c-pumpkin-minor-free, $|V(G')| < |V(G)|$, and $k' \le k$.

Before describing how to obtain the linear kernel for p-c-DISJOINT HIT, we need the following lemma, corresponding to [19, Lemma 6].

Lemma 3.8 (Fomin *et al.* [19])**.** *There is a linear time algorithm that given an n-vertex graph G and a set $S \subseteq V(G)$ such that $\mathbf{tw}(G \setminus S) \le d$, outputs a $2(d + 1)$-protrusion in $G \setminus S$ of size at least $\frac{n - |S|}{4|N(S)| + 1}$, where $N(S)$ is the set of vertices in $V(G) \setminus S$ with at least one neighbor in S. Here d is some constant.*

The proof of the next lemma is similar to the proof of [19, Theorem 1].

Lemma 3.9. *If $|V_1 \cup V_{\ge 2}| = \mathcal{O}(k)$, then p-c-DISJOINT HIT has a kernel with $\mathcal{O}(k)$ vertices.*

Proof. Let for this proof $V_\star = V_1 \cup V_{\ge 2}$, and let (G, S, k) be an instance of p-c-DISJOINT HIT such that $|V_\star| = \mathcal{O}(k)$. As by Theorem 2.3 we have that $\mathbf{tw}(G \setminus S) \le 2c - 1$, we can apply Lemma 3.8 and obtain in linear time a $2((2c - 1) + 1) = 4c$-protrusion Y of size at least $\frac{|V(G)| - |S|}{4|V_\star| + 1}$ in $G \setminus S$. Let $\gamma : \mathbb{N} \to \mathbb{N}$ be the function defined in Lemma 3.7. If $\frac{|V(G)| - |S|}{4|V_\star| + 1} > \gamma(8c + 1)$, then using protrusion rule **P** we obtain in time $\mathcal{O}(|Y|)$ an instance (G', S, k') such that $G[S]$ and $G'[S]$ are isomorphic, $G' \setminus S$ is c-pumpkin-minor-free, $|V(G')| < |V(G)|$, $k' \le k$, and such that (G', S, k') is a YES-instance of p-c-DISJOINT HIT if and only if (G, S, k) is a YES-instance of p-c-DISJOINT HIT. We continue applying Lemma 3.8 and protrusion rule **P** to the newly obtained instance as far as there is a $4c$-protrusion of size strictly greater than $\gamma(8c + 1)$. We would like to note that in the whole process we never delete either the vertices of S or $V_\star$.

Let (G^*, S, k^*) be the reduced instance obtained after this procedure. It follows that there is no $4c$-protrusion of size greater than $\gamma(8c + 1)$ in $G^* \setminus S$, so protrusion rule **P** no longer applies. Note that $k^* \le k$. We claim that the number of vertices in this graph G^* is bounded by $\mathcal{O}(k)$. Indeed, since we cannot apply protrusion rule **P**, we have that $\frac{|V(G^*)| - |S|}{4|V_\star| + 1} \le \gamma(8c + 1)$. Therefore, we have that $|V(G^*)| \le \gamma(8c + 1) \cdot (4|V_\star| + 1) + |S|$. Since $|S| = \mathcal{O}(k)$ and by hypothesis $|V_\star| = \mathcal{O}(k)$, we have that $|V(G^*)| = \mathcal{O}(k)$. This completes the proof. □

Lemma 3.9 clearly implies that, if $|V_1 \cup V_{\ge 2}| = \mathcal{O}(k)$, then p-c-DISJOINT HIT can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$. Nevertheless, the above proof only shows that the *decision version* of p-c-DISJOINT HIT can be solved in single-exponential time, as we have applied reduction rules that may modify the instance. But in the iterative compression routine (see proof of Lemma 3.1), we need to be able to obtain an *explicit solution* $S' \subseteq V(G) \setminus S$ of p-c-DISJOINT HIT in the original instance, with $|S'| = k$, if it exists.

We can get this explicit solution (if it exists) by repeatedly applying the single-exponential algorithm for the decision version as follows. Suppose that G is a YES-instance of p-c-DISJOINT HIT, and let an ordering of the vertices of $V(G) \setminus S$ be $u_1, u_2, \ldots, u_q$. Set $i := 1$ and $U := \emptyset$. Repeat the following two steps for $i = 1, \ldots, q$.

(1) Check whether $G \setminus U$ is c-pumpkin-free. If it is the case, then return U as the desired solution. Otherwise, go to the next step.

(2) Using the single-exponential algorithm for the decision version of p-c-Disjoint Hit, check whether $G \setminus (\{u_i\} \cup U)$ contains a solution $S^* \subseteq V(G) \setminus (S \cup U \cup \{u_i\})$ of size $k - (|U| + 1)$. If it is the case, then set $U := U \cup \{u_i\}$. Set $i := i + 1$.

This concludes the description of the algorithm to obtain the desired explicit solution U in the compression step.

Final algorithm. Finally we combine everything to obtain the following result.

Theorem 3.10. *For any fixed $c \geq 1$, the p-c-Pumpkin-Hitting problem can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$.*

Proof. To obtain the desired result, by Lemma 3.1 and the procedure described after the proof of Lemma 3.9, it is sufficient to obtain a single-exponential algorithm for p-c-Disjoint Hit. To this end, after applying reduction rules **R1**, **R2**, and **R3** in polynomial time, by Lemmas 3.2 and 3.5 we have that $|V_1 \cup V_{\geq 2}| = \mathcal{O}(k)$. Thus, using Lemma 3.9 we get, also in polynomial time, an equivalent instance (G^*, S, k^*) with $\mathcal{O}(k)$ vertices, and hence the problem can be solved by enumerating all subsets of size at most k^* of $G^* \setminus S$. This concludes the proof. $\square$

4. An approximation algorithm for hitting and packing pumpkins

In this section we show that every n-vertex graph G either contains a small c-pumpkin-model or has a structure that can be reduced, giving a smaller equivalent instance for both the Minimum c-Pumpkin-Hitting and the Maximum c-Pumpkin-Packing problems. Here by a "small" c-pumpkin-model, we mean a model of size at most $f(c) \cdot \log n$ for some function f independent of n. We finally use this result to derive a $\mathcal{O}(\log n)$-approximation algorithm for both problems.

This section is organized as follows. We first describe in Section 4.1 our reduction rules and prove their validity for both hitting and packing problems (see Lemma 4.1). The existence of small c-pumpkin-models in c-reduced graphs is provided in Section 4.3 (see Lemma 4.10); its proof strongly relies on a graph structure that we call *hedgehog*, which we study in Section 4.2. We finally focus in Section 4.4 on the algorithmic consequences of our results (see Theorem 4.11).

4.1. Reduction rules.
We describe two reduction rules for hitting/packing c-pumpkin-models, which given an input graph G satisfying some specific conditions, produce a graph H with less vertices than G and satisfying $\tau_c(G) = \tau_c(H)$ and $\nu_c(G) = \nu_c(H)$. Moreover, these operations are defined in such a way that, for both problems, an optimal (resp. approximate) solution for G can be retrieved in polynomial time from an optimal (resp. approximate) solution for H.

Given a graph G and two distinct vertices u, v of G, we write $G +_k uv$ for the graph obtained from G by adding k parallel edges linking u to v. A *c-outgrowth* of a graph G is a triple (K, u, v) such that

   (i) u, v are two distinct vertices of G;
   (ii) K is a connected component of $G \setminus \{u, v\}$ with $|K| \geq 1$;
   (iii) u and v both have at least one neighbor in K in the graph G; and
   (iv) the graph $\Gamma(K, u, v)$ obtained from $G[V(K) \cup \{u, v\}]$ by removing all the edges between u and v has no c-pumpkin minor.

Given a c-outgrowth (K, u, v) we let $\gamma(K, u, v) := c - k$, where k is the smallest integer such that $\Gamma(K, u, v) +_k uv$ has a c-pumpkin minor. Observe that, when adding k parallel edges between u and v to $\Gamma(K, u, v)$, there are two distinc "types" of c-pumpkin models $\{A, B\}$ that could appear: Exchanging A and B if necessary, we either have $u \in A$ and $v \in B$ (first type), or $u, v \in A$ (second type). (Possibly both types of models coexist.)

FIGURE 3. The graph $\Gamma(K, u, v)$ of two c-outgrowths (K, u, v) for $c = 11$ (left) and $c = 7$ (right), respectively. We have $\gamma(K, u, v) = 9$ for the left one, and $\gamma(K, u, v) = 6$ for the right one.
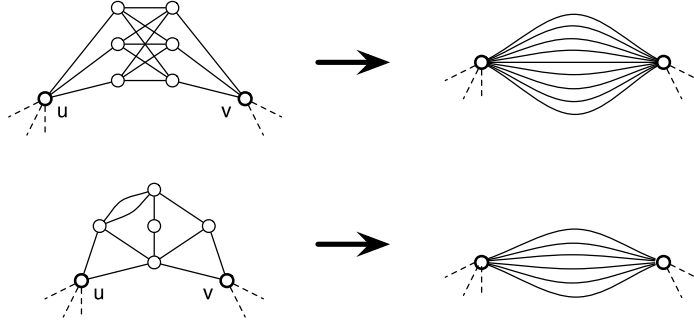

FIGURE 4. Illustration of reduction rule **Z2** on the two outgrowths from Figure 3.

Note that we always have $\gamma(K, u, v) = c - 1$ when $\Gamma(K, u, v) +_k uv$ contains a c-pumpkin model of the second type. See Figure 3 for an illustration.

Now that we are equipped with these definitions and notations, we may describe the two reduction rules, which depend on the value of the positive integer c.

Z1 Suppose v is a vertex of G such that no block of G containing v has a c-pumpkin-minor. Then define H as the graph obtained from G by removing v.

Z2 Suppose that (K, u, v) is a c-outgrowth of G. Then define H as the graph obtained from $G \setminus V(K)$ by adding $\gamma(K, u, v)$ parallel edges between u and v.

See Figure 4 for an illustration of **Z2**.

We note that testing for the existence of a c-pumpkin-minor in a graph G can be done in polynomial time when c is fixed, and if there is one an explicit c-pumpkin-model can be computed. This follows from classical results of Robertson and Seymour [29], and will be used implicitly in the subsequent proofs. Note that, in particular, testing whether a vertex v is in a block containing a c-pumpkin-minor can be done in polynomial time. Similarly, testing whether a triple (K, u, v) with K a component of $G - \{u, v\}$ is a c-outgrowth can be done in polynomial time, and the parameter $\gamma(K, u, v)$ can be computed in polynomial time as well. Therefore, we can check in polynomial time if **Z1** or **Z2** can be applied to a given graph, and each of these two reduction rules can be realized in polynomial time.

A graph G is said to be *c-reduced* if neither **Z1** nor **Z2** can be applied to G. The next lemma shows the validity of these reduction rules.

Lemma 4.1. *Let c be a fixed positive integer. Suppose that H results from the application of **Z1** or **Z2** on a graph G. Then*

(a) *$\tau_c(G) = \tau_c(H)$ and moreover, given a c-hitting set X' of H, one can compute in polynomial time a c-hitting set X of G with $|X| \leq |X'|$.*

(b) *$\nu_c(G) = \nu_c(H)$ and moreover, given a c-packing $\mathcal{M}'$ of H, one can compute in polynomial time a c-packing $\mathcal{M}$ of G with $|\mathcal{M}| = |\mathcal{M}'|$.*

In order to prove Lemma 4.1, we first need to introduce a few technical lemmas; the validity of the reduction rules is shown in Lemmas 4.4 and 4.5 at the end of this section, which correspond to Lemma 4.1(a) and Lemma 4.1(b), respectively.

Lemma 4.2. *Let c be a fixed positive integer. Suppose H is obtained by applying rule* **Z2** *on a c-outgrowth (K, u, v) of a graph G. Let X be an arbitrary subset of vertices of $V(G) \setminus (V(K) \cup \{u, v\})$. Then, given a c-pumpkin-model of $H \setminus X$, one can find in polynomial time a c-pumpkin-model of $G \setminus X$.*

Proof. Let $\Gamma := \Gamma(K, u, v)$, $\gamma := \gamma(K, u, v)$, and $k := c - \gamma$. Let $\{A, B\}$ denote the given c-pumpkin-model of $H \setminus X$.

If $u \notin A \cup B$ or $v \notin A \cup B$, then $\{A, B\}$ is a c-pumpkin-model in $G \setminus X$ and we are done. Thus, exchanging A and B if necessary, we may assume that either $u, v \in A$, or $u \in A$ and $v \in B$. In the first case, since $G[A \cup V(K)]$ is connected, $\{A \cup V(K), B\}$ is a c-pumpkin-model in $G \setminus X$. Now suppose that $u \in A$ and $v \in B$. We need to consider which types of c-pumpkin-models appear in $\Gamma +_k uv$.

If $\Gamma +_k uv$ contains a c-pumpkin-model $\{A', B'\}$ with $u \in A'$ and $v \in B'$ then there are exactly γ edges linking A' to B' in the graph Γ, and hence $\{A \cup A', B \cup B'\}$ is a c-pumpkin-model in $G \setminus X$, as desired.

If $\Gamma +_k uv$ has a c-pumpkin-model $\{A', B'\}$ with $u, v \in A'$, then $k = 1$ and $\gamma = c - 1$. In $H[A \cup B]$ there is a path P linking u to v that avoids the $c - 1$ edges that resulted from the application of **Z2** on the c-outgrowth (K, u, v). (Note that P could possibly consists of a single edge linking u to v.) Then $\{A' \cup V(P), B'\}$ is a c-pumpkin-model in $G \setminus X$. $\qquad\square$

Next we show that the converse of the above lemma also holds.

Lemma 4.3. *Let c be a fixed positive integer. Suppose H is obtained by applying rule* **Z2** *on a c-outgrowth (K, u, v) of a graph G. Let X be an arbitrary subset of vertices of $V(G) \setminus (V(K) \cup \{u, v\})$. Then, given a c-pumpkin-model of $G \setminus X$, one can find in polynomial time a c-pumpkin-model of $H \setminus X$.*

Proof. Let $\Gamma := \Gamma(K, u, v)$, $\gamma := \gamma(K, u, v)$, and $k := c - \gamma$. Let $\{A, B\}$ denote the given c-pumpkin-model of $G \setminus X$. We may assume that this model is minimal (if not, one can obviously make it minimal in polynomial time).

If $u \notin A \cup B$ or $v \notin A \cup B$, then by minimality of $\{A, B\}$ both A and B avoid $V(K)$. Thus $\{A, B\}$ is a c-pumpkin-model in $H \setminus X$, and we are done. Hence, exchanging A and B if necessary, we may assume that either $u, v \in A$, or $u \in A$ and $v \in B$. In the second case, at most γ edges between A and B in $G \setminus X$ are included in Γ. Since there are γ extra edges between u and v in H compared to G, it follows that $\{A \setminus V(K), B \setminus V(K)\}$ is a c-pumpkin-model in $H \setminus X$.

Now suppose that $u, v \in A$. If $B \subseteq V(K)$ then all edges between A and B in $G \setminus X$ are in Γ. Let P be a path in $G[A]$ linking u to v. (Note that the path P possibly consists of a single edge.) Then P is disjoint from $V(K)$, as otherwise $P \subseteq \Gamma$ and $\{A \cap V(\Gamma), B\}$ would be a c-pumpkin-model in Γ. Thus in particular $k = 1$ and $\gamma = c - 1$. Since there are $c - 1$ extra edges between u and v in H compared to G, and P avoids all these edges, $\{\{u\}, \{v\} \cup (V(P) \setminus \{u\})\}$ is a c-pumpkin-model in $H \setminus X$.

If $B \not\subseteq V(K)$ then B is disjoint from $V(K)$. Since u and v are linked by at least $\gamma \geq 1$ edges in H, the graph $H[A \setminus V(K)]$ is connected, and it follows that $\{A \setminus V(K), B\}$ is a c-pumpkin-model in $H \setminus X$. $\qquad\square$

Lemma 4.4. *Let c be a fixed positive integer. Suppose H results from the application of* **Z1** *or* **Z2** *on a graph G. Then $\tau_c(G) = \tau_c(H)$. Moreover, every c-hitting set X' of H is also a c-hitting set of G.*

This lemma implies that an optimal solution to the MINIMUM c-PUMPKIN-HITTING problem on G can be computed given one for H, and similarly that an approximate solution for G can be obtained from an approximate solution for H. This will be used in our approximation algorithms in Section 4.4.

Proof of Lemma 4.4. First suppose H results from the application of **Z1** on G with vertex v. We trivially have $\tau_c(G) \geq \tau_c(H)$. Let X' be a given c-hitting set of H. If X' is not a c-hitting set of G, then $G \setminus X'$ has a c-pumpkin-model; let $\{A, B\}$ be a minimal one. We have $v \in A \cup B$ since otherwise $\{A, B\}$ would be a c-pumpkin-model in $H \setminus X'$. By the minimality of $\{A, B\}$, we must have $A \cup B \subseteq V(K)$ for some block K of G. But then K is a block of G including v and containing a c-pumpkin-minor, contradicting the assumptions of **Z1**. Therefore X' is a c-hitting set of G, and $\tau_c(G) \leq \tau_c(H)$ also holds, implying $\tau_c(G) = \tau_c(H)$.

Now assume H has been obtained by applying **Z2** on G with c-outgrowth (K, u, v), and let $\Gamma := \Gamma(K, u, v)$.

First we show $\tau_c(G) \geq \tau_c(H)$. Let X be a minimum c-hitting set of G. If $u \in X$ or $v \in X$, then X is trivially a c-hitting set of H, so let us assume $u, v \notin X$. Moreover, we may suppose that X has no vertex in K, since otherwise we could replace all such vertices with the vertex u (or equivalently v). Since $X \subseteq V(G) \setminus V(\Gamma)$ and $G \setminus X$ has no c-pumpkin-minor, it follows from Lemma 4.2 that $H \setminus X$ has no c-pumpkin-minor either, that is, X is a c-hitting set of H. This shows $\tau_c(G) \geq \tau_c(H)$.

Now we prove that $\tau_c(G) \leq \tau_c(H)$ also holds. Here we show that, given a c-hitting set X' of H, the set X' is also a c-hitting set of G. Hence, this will also prove the second part of the lemma. If $u \in X'$ or $v \in X'$, then X' is trivially a c-hitting set of G. If $u, v \notin X'$, then Lemma 4.3 implies that $G \setminus X'$ has no c-pumpkin-minor, that is, that X' is a c-hitting set of G. This shows $\tau_c(G) \leq \tau_c(H)$, and therefore $\tau_c(G) = \tau_c(H)$.   □

We conclude this section with a lemma similar to Lemma 4.4 for c-packings.

Lemma 4.5. *Let c be a fixed positive integer. Suppose H results from the application of **Z1** or **Z2** on a graph G. Then $\nu_c(G) = \nu_c(H)$. Moreover, given a c-packing $\mathcal{M}'$ of H one can compute in polynomial time a c-packing $\mathcal{M}$ of G with $|\mathcal{M}| = |\mathcal{M}'|$.*

Proof. First suppose H results from the application of **Z1** on G with vertex v. Clearly, every c-packing of H is a c-packing for G. Thus $\nu_c(G) \geq \nu_c(H)$, and it is enough to show the reverse inequality. Consider a c-packing of G. We may assume that every c-pumpkin-model in that packing is minimal. Thus each such model is contained in some block of G, and hence avoids the vertex v. Therefore the packing also exists in H, implying $\nu_c(G) \leq \nu_c(H)$ and $\nu_c(G) = \nu_c(H)$, as desired.

Now assume H has been obtained by applying **Z2** on G with outgrowth (K, u, v).

First we show $\nu_c(G) \geq \nu_c(H)$. Let $\mathcal{M}' = \{M'_1, \ldots, M'_k\}$ be a given c-packing of H. We show that a packing of the same size in G can be computed in polynomial time, which will prove the second part of the lemma. If every M'_i avoids at least one of u, v then the packing $\mathcal{M} := \mathcal{M}'$ is a c-packing in G and we are done. So assume one model in the collection, say without loss of generality M'_1, includes both u and v. Let X be the union of the vertices in $M'_2, \ldots, M'_k$. Since M'_1 is a c-pumpkin-model in $H \setminus X$, using Lemma 4.2 we can compute in polynomial time a c-pumpkin-model M_1 in $G \setminus X$. Hence $\mathcal{M} := \{M_1, M'_2, \ldots, M'_k\}$ is a c-packing of the desired size in G.

In order to prove $\nu_c(G) = \nu_c(H)$ it remains to show $\nu_c(G) \leq \nu_c(H)$. Let $\{M_1, \ldots, M_k\}$ be a c-packing of G. We may assume that each M_i is minimal. Thus if some M_i contains some vertex of K then M_i contains both u and v. If there is no such model in the packing then $\{M_1, \ldots, M_k\}$ is also of c-packing of H and we are done. We may thus assume that
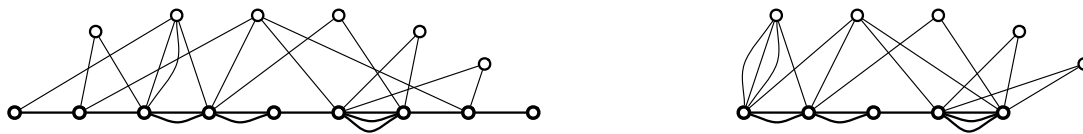
FIGURE 5. A hedgehog (H, P) (left) and a contraction (H', Q) of (H, P) (right). The multipaths P and Q are drawn in bold.

some model in the packing, say without loss of generality M_1, contains both u and v. As before, let X be the union of the vertices in $M_2, \ldots, M_k$. Using Lemma 4.3 with M_1 and X we find a c-pumpkin-model M_1' in $H \setminus X$. Thus $\{M_1', M_2, \ldots, M_k\}$ is a c-packing of size k in H, as desired. □

4.2. **Hedgehogs.** Recall that a graph is said to be a *multipath* if its underlying simple graph is isomorphic to a path. If P is a multipath and $u, v \in V(P)$, we write uPv for the subgraph of P induced by the vertices on a u–v path in P (thus edges in uPv have the same multiplicities as in P).

A *hedgehog* is a pair (H, P), where H is a graph and P is an induced multipath of H with $|P| \geq 2$ and such that

(i) the (possibly empty) set $S := V(H) \setminus V(P)$ is a stable set of H; and

(ii) every vertex in S has at least two neighbors in P.

(Let us recall that a *stable set* is a set of vertices such that no two of them are adjacent.)

Consider a hedgehog (H, P). Its *size* is defined as $|P|$, the number of vertices in P. A *bad cutset* of (H, P) is a set $X = \{u, v\}$ of two *internal* vertices of P such that $H \setminus X$ has a connected component K avoiding both endpoints of P. This definition is motivated by reduction rule **Z2**: First, if K is such a component, then u and v each have at least one neighbor in K. This is because either K contains the subpath of P strictly between u and v, or K consists of a unique vertex of $V(H) \setminus V(P)$ which is then adjacent to u and v (by condition (ii) in the definition of hedgehogs). Hence either (K, u, v) is a c-outgrowth of H, or one can find a c-pumpkin-minor in $H[V(K) \cup \{u, v\}]$.

A *rooted c-pumpkin-model* of (H, P) is a c-pumpkin-model $\{A, B\}$ of H with the extra property that A and B both contain an endpoint of P.

Given a hedgehog (H, P) and a connected induced subgraph Q of P with $|Q| \geq 2$, one can define a hedgehog (H', Q) as follows: First, remove from H every vertex not in P that has no neighbor in Q. Then contract every edge of P not included in Q. Finally, remove from the graph every vertex not in Q that has only one neighbor in Q. This defines the graph H'. We leave it to the reader to check that (H', Q) is indeed a hedgehog; we say that (H', Q) is the *contraction* of (H, P) on the multipath Q. See Figure 5 for an illustration of this operation. The following lemma is a direct consequence of the definition.

Lemma 4.6. *If (H', Q) is a contraction of a hedgehog (H, P) and X is a bad cutset of (H', Q), then X is also a bad cutset of (H, P).*

We show that every big enough hedgehog has a rooted c-pumpkin-model or a bad cutset. This fact will be useful in the subsequent proofs.

Lemma 4.7. *Let c be a fixed positive integer. Then every hedgehog (H, P) of size at least $(2c)^{2c}$ contains a rooted c-pumpkin-model or a bad cutset, either of which can be found in polynomial time.*

Proof. The proof is by induction on c. The base case $c = 1$ is trivial since P directly gives a rooted 1-pumpkin-model. For the inductive step, assume $c > 1$. Define $f(k)$, for

a positive integer k, as $f(k) := (2k)^{2k}$. Let $S := V(H) \setminus V(P)$. Let a, b be the endpoints of P.

If a vertex $v \in S$ has at least c neighbors in P, then let w be the neighbor of v that is closest to a on P. Then $A := V(aPw) \cup \{v\}$ and $B := V(P) \setminus A$ both induce a connected subgraph of H. Moreover, there are at least $c - 1$ edges from v to B, and at least one from $A \setminus \{v\}$ to B (because of P). Since $a \in A$ and $b \in B$, we deduce that $\{A, B\}$ is a rooted c-pumpkin-model of (H, P). Thus we may assume that every vertex in S has at most $c - 1$ neighbors in P. In particular we have $c \geq 3$, since every vertex in S has at least two neighbors in P.

The multipath P, seen from its endpoint a, induces a natural linear ordering of the neighbors of a given vertex in S; we say that two such neighbors are *consecutive* if they are consecutive in that ordering.

Suppose that there exists a vertex $v \in S$ with two consecutive neighbors x, y such that $|xPy| \geq f(c-1) + 2$. Consider the contraction (H', Q) of (H, P) on the multipath $Q := xPy \setminus \{x, y\}$. Since $|Q| \geq f(c-1)$, by induction (H', Q) has a rooted $(c-1)$-pumpkin-model $\{A', B'\}$ or a bad cutset X. If the latter holds, then by Lemma 4.6 the set X is also a bad cutset of (H, P) and we are done. Thus we may assume the former holds. In the graph H, the vertex v has no neighbor in Q, thus v is not included in H'. Hence, we can obtain a rooted c-pumpkin-model $\{A, B\}$ in (H, P) by setting $A := A' \cup V(aPx) \cup \{v\}$ and $B := B' \cup V(yPb)$. Therefore we can assume that, for every vertex $v \in S$, every two consecutive neighbors of v are at distance at most $f(c-1)$ on P.

Let us enumerate the vertices of P in order as $p_1, p_2, \ldots, p_k$, with $p_1 = a$ and $p_k = b$. We may assume that, for every $i \in \{3, \ldots, k-2\}$,

$$(1) \qquad\qquad p_i \text{ is adjacent to some vertex in } S.$$

Indeed, if not then $\{p_{i-1}, p_{i+1}\}$ would be a bad cutset of (H, P). Since $k = |P| \geq f(c) \geq f(3) \geq 5$, this implies in particular that S is not empty.

Define an *open* interval $I_v = (i, j)$ for every vertex $v \in S$, where i (j) is the smallest (largest, respectively) index t such that p_t is a neighbor of v in H. (Observe that $i < j$ since v has at least two neighbors.) Now, let G be the interval graph defined by these open intervals, that is, let $V(G) := S$, and for every two distinct vertices $v, w \in S$, make v adjacent to w in G if and only if $I_v \cap I_w \neq \emptyset$.

For a connected subgraph G' of G, we define $I(G')$ as the union of the intervals of vertices in G', that is, $I(G') := \bigcup \{I_v : v \in V(G')\}$. Observe that, since G' is connected, we have $I(G') = (i, j)$ for some integers i, j with $1 \leq i < j \leq k$.

First suppose that G has at least three connected components. The ordering $p_1, \ldots, p_k$ of the vertices of P induces an ordering of these components; let C, C', C'' be three consecutive connected components in that ordering. Let $(i, j) := I(C)$, $(i', j') := I(C')$, and $(i'', j'') := I(C'')$. Then we have $1 \leq i < j \leq i' < j' \leq i'' < j'' \leq k$, and every vertex of S that is adjacent to some vertex strictly between $p_{i'}$ and $p_{j'}$ on P has all its neighbors in the set $\{p_{i'}, p_{i'+1}, \ldots, p_{j'}\}$. Thus, for each $w \in V(C')$, the component K of $H - \{p_{i'}, p_{j'}\}$ that contains w avoids both endpoints of P. It follows that $\{p_{i'}, p_{j'}\}$ is a bad cutset of (H, P). Hence, we may assume that G has at most two connected components.

Since G has at most two connected components, using (1) we deduce that G has a connected component C with $I(C) = (x, y)$ such that

$$(2) \qquad\qquad y - x + 1 \geq \frac{|P| - 4}{2} \geq \frac{f(c) - 4}{2}.$$
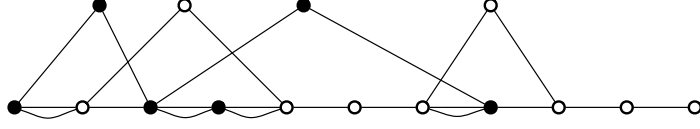
FIGURE 6. Illustration of the coloring for $d = 2$. The vertices in $\{z_1, \ldots, z_{2d}\}$ are on top, the path Q at the bottom, and each breakpoint p_i of Q is linked to the vertices z_j such that $j \in J(p_i)$.

Let $Q := p_x P p_y$ and let (H', Q) be the contraction of (H, P) on Q. (Note that possibly $Q = P$, in which case $(H', Q) = (H, P)$.) We will show that (H', Q) contains a rooted c-pumpkin-model. The lemma will then follow, since such a model can be extended straightforwardly to one of (H, P).

First let us observe that H' is an induced subgraph of H. This is because, by our choice of Q, every vertex of S that is adjacent to at least two vertices of Q, or to at least one internal vertex of Q, has all its neighbors in Q. Let $S' := V(H') \setminus V(Q) = V(C)$. For a vertex $u \in S'$, let us denote by $\ell(u)$ and $r(u)$ the two integers such that $I_u = (\ell(u), r(u))$.

It follows from our assumptions on (H, P) that, for each $u \in S'$, the vertex u has at most $c - 1$ neighbors in Q and every two consecutive neighbors of u are at distance at most $f(c-1)$ on Q. This implies

$$(3) \qquad\qquad r(u) - \ell(u) \leq (c-2)f(c-1)$$

for each $u \in S'$.

In H', the vertices p_x and p_y each have at least one neighbor in S'. Let $v \in S'$ be a neighbor of p_x maximizing $r(v)$, and let $w \in S'$ be a neighbor of p_y minimizing $\ell(w)$. Let Z be a shortest v–w path in the interval graph G; enumerate the vertices of Z as $z_1, z_2, \ldots, z_m$ with $z_1 = v$ and $z_m = w$.

By our choice of v, w and the fact that Z is a shortest v–w path, we have

$$(4) \qquad\qquad \ell(z_j) < \ell(z_{j+1})$$
$$(5) \qquad\qquad r(z_j) < r(z_{j+1})$$

for each $j \in \{1, \ldots, m-1\}$, and

$$(6) \qquad\qquad r(z_j) \leq \ell(z_{j+2})$$

for each $j \in \{1, \ldots, m-2\}$.

Since $I(Z) = I(C) = (x, y)$ we have $y \leq x + \sum_{j=1}^{m}(r(z_j) - \ell(z_j))$. Hence $y - x \leq m(c-2)f(c-1)$ by (3). Using (2) we then obtain

$$(7) \qquad m \geq \frac{y - x}{(c-2)f(c-1)} \geq \frac{f(c) - 6}{2(c-2)f(c-1)} = \frac{(2c)^{2c} - 6}{2(c-2)(2c-2)^{2c-2}} \geq c.$$

Let $d := \lfloor m/2 \rfloor$. Define, for $i \in \{x, \ldots, y\}$, the set $J(p_i)$ as the set of indices $j \in \{1, \ldots, 2d\}$ such that $i \in \{\ell(z_j), r(z_j)\}$. We say that p_i is a *breakpoint* of Q if $J(p_i)$ is not empty. (Thus p_x is a breakpoint in particular.) It is a consequence of (4), (5), and (6) that, if $|J(p_i)| > 1$, then $J(p_i) = \{j, j+2\}$ for some $j \in \{1, \ldots, 2d-2\}$. In particular, the numbers in $J(p_i)$ always have the same parity.

We color the vertices in $V(Q) \cup \{z_1, \ldots, z_{2d}\}$ in *black* or *white* as follows. First, for every $j \in \{1, \ldots, 2d\}$, color z_j black if j is odd, white if j is even. Next color every breakpoint p_i of Q with the color corresponding to the parity of the numbers in $J(p_i)$ (namely, black for odd and white for even). Finally, color every uncolored vertex of Q with the color of the closest breakpoint of Q in the direction of p_x. See Figure 6 for an illustration of the coloring.

Let A and B be the set of black and white vertices, respectively. By construction, $p_x \in A$ and $p_y \in B$, and each of A, B induces a connected subgraph of H'. Moreover, there are $2d + 1 \geq m \geq c$ edges of Q whose endpoints received distinct colors. It follows that $\{A, B\}$ is a rooted c-pumpkin-model of (H', Q), as desired.

We have shown that (H, P) always has a rooted c-pumpkin-model or a bad cutset. Moreover, it is easily seen from the proof given above that each of these can be found in polynomial time. This concludes the proof of the lemma. $\qquad\square$

We note that no effort has been made in order to optimize the constants in Lemma 4.7.

4.3. **Small pumpkins in reduced graphs.** Our goal is to prove that every n-vertex c-reduced graph G has a c-pumpkin-model of size $\mathcal{O}(\log n)$, where c is a fixed constant. We will use the following recent result by Fiorini *et al.* [17] about the existence of small minors in *simple* graphs with large average degree.

Theorem 4.8 (Fiorini *et al.* [17]). *There is a function h such that every n-vertex simple graph G with average degree at least 2^t contains a K_t-model with at most $h(t) \cdot \log n$ vertices. Moreover, such a model can be computed in polynomial time.*

Since a K_t-model in a graph directly gives a c-pumpkin-model of the same size for $c = (\lfloor t/2 \rfloor)^2$, we have the following corollary from Theorem 4.8, which is central in the proof of Lemma 4.10.

Corollary 4.9. *There is a function h such that every n-vertex simple graph G with average degree at least $2^{2\sqrt{c}+1}$ contains a c-pumpkin-model with at most $h(c) \cdot \log n$ vertices. Moreover, such a model can be computed in polynomial time.*

The next lemma states the existence of small c-pumpkin-models in a c-reduced graph G. Its proof relies on Lemma 4.7 on hedgehogs and Corollary 4.9. The proof can be briefly summarized as follows. An hedgehog in G which is large enough so that Lemma 4.7 can be applied to it, but at the same time not too big, witnesses the existence of either a small c-pumpkin-model or a c-outgrowth. Hence we may assume that no such hedgehog exists in G. The latter fact is then used to either find directly a small c-pumpkin-model, or a dense-enough minor that is not "too far" from G in the sense that it is obtained by contracting disjoint connected subgraphs of G of bounded radius. In the latter case, we apply Corollary 4.9 on the minor, yielding a small c-pumpkin-model which we then lift back to G, incurring only a constant-factor increase in its size.

Lemma 4.10. *There is a function f such that every n-vertex c-reduced graph G contains a c-pumpkin-model of size at most $f(c) \cdot \log n$. Moreover, such a model can be computed in polynomial time.*

Proof. Let

$$k := c^2 \left\lceil 2^{2\sqrt{c}+1} \right\rceil$$
$$r := (2c)^{2c} k$$
$$b := k^r.$$

We will prove the lemma with f defined as

$$f(c) := \max\{krb, 3rc \cdot h(c)\},$$

where h is the function in Corollary 4.9. Throughout the proof, a c-pumpkin-model of G is said to be *small* if it has the required size, that is, if it has at most $f(c) \log n$ vertices.

Recall that $\mu(G)$ denotes the maximum multiplicity of any edge in G. The lemma trivially holds if $\mu(G) \geq c$, so we may assume $\mu(G) < c$. Let W be the (possibly empty) subset of vertices of G having degree at least k.
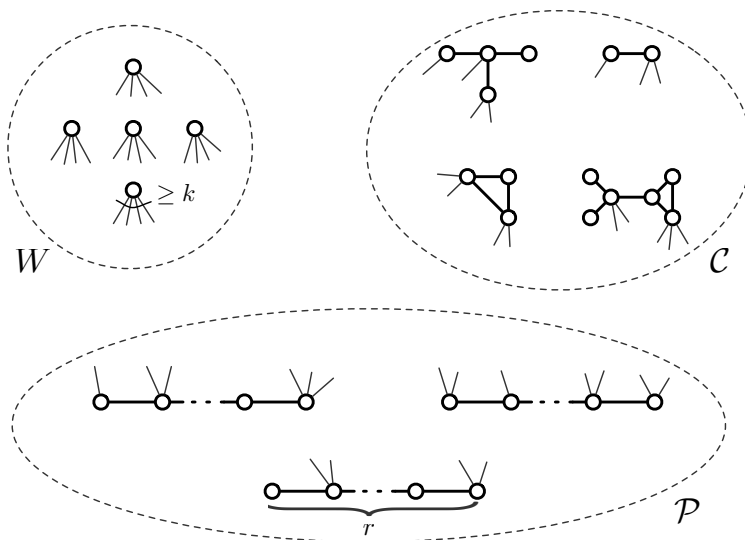
FIGURE 7. The sets W, $\mathcal{P}$, and $\mathcal{C}$ in the graph G.

We build a collection $\mathcal{P}$ of vertex-disjoint induced subgraphs of $G \setminus W$, each isomorphic to a multipath on r vertices. Initially, we let $\mathcal{P} := \emptyset$ and $G' := G \setminus W$. Then, as long as G' has a connected component C with diameter at least $r-1$, we do the following: First, we consider two vertices at distance $r-1$ in C and compute a shortest path Q between these two vertices. Note that the subgraph P of G induced by $V(Q)$ is a multipath on r vertices. Next, we add P to $\mathcal{P}$. Finally, we remove from G' the r vertices in P.

When the above procedure is finished, every connected component of G' has diameter less than $r-1$ and maximum degree less than k. Hence each such component has bounded size: at most $k^r = b$ vertices. Let $\mathcal{C}$ denote the collection of connected components of G'.

An illustration of the sets W, $\mathcal{P}$, and $\mathcal{C}$ in the graph G is given in Figure 7.

If some subgraph $C \in \mathcal{C}$ contains a c-pumpkin-model, then the size of the model is at most $|C| \leq b \leq f(c) \leq f(c) \log n$, and we are done. (Note that $n \geq 2$ since the model has at least two vertices.) Thus we may assume that no subgraph $C \in \mathcal{C}$ contains a c-pumpkin-minor.

Let J be the graph obtained from G by contracting each subgraph $C \in \mathcal{C}$ into a single vertex v_C. Consider a subgraph $C \in \mathcal{C}$. We cannot have $\deg^*_J(v_C) = 0$, because otherwise we could have applied **Z1** on any vertex of C in G (since C has no c-pumpkin-minor). If $\deg^*_J(v_C) = 1$, then let v be an arbitrary vertex of C, and let w be the unique vertex in $V(G) \setminus V(C)$ having a neighbor in $V(C)$ in the graph G. Since **Z1** cannot be applied on G with vertex v, there is a block B of G that includes v and containing a c-pumpkin-model. Since $V(B) \subseteq V(C) \cup \{w\}$, this model has size at most $|B| \leq b+1 \leq f(c)$, that is, we have found a small c-pumpkin-model of G. Therefore, we may assume

$$\deg^*_J(v_C) \geq 2 \tag{8}$$

for every $C \in \mathcal{C}$.

Let K be the graph obtained from J by contracting each subgraph $P \in \mathcal{P}$ into a single vertex v_P. If two vertices of K are linked by at least c parallel edges, then we directly find a c-pumpkin-model in G of size at most $\max\{b+1, r+1, 2r, b+r\} \leq f(c)$.

Thus we may assume

$$\mu(K) < c. \tag{9}$$

We have $\deg_K^*(v_C) \geq 1$ for every $C \in \mathcal{C}$. Let us say that a subgraph $C \in \mathcal{C}$ is *bad* if $\deg_K^*(v_C) = 1$, and *good* otherwise.

We color the vertices of each multipath $P \in \mathcal{P}$ as follows: a vertex $v \in V(P)$ is colored *black* if, in the graph G, all its neighbors outside P belong to bad subgraphs of $\mathcal{C}$; the vertex v is colored *white* otherwise. (We remark that v could possibly have no neighbor outside P, in which case v is colored black by our definition.)

Claim 4. *If some multipath $P \in \mathcal{P}$ contains $(2c)^{2c}$ consecutive black vertices, then one can find a small c-pumpkin-model in G.*

Proof. Let Q be the subgraph of P induced by these $(2c)^{2c}$ black vertices. Let $\mathcal{C}'$ be the subset of subgraphs $C \in \mathcal{C}$ such that v_C is adjacent to an *internal* vertex of Q in the graph J. Let $S := \{v_C : C \in \mathcal{C}'\}$. Since all vertices of Q are colored black, it follows that internal vertices of Q are only adjacent in J to vertices in $V(P) \cup S$, and that every subgraph $C \in \mathcal{C}'$ is bad.

Let H be the graph obtained from $J[V(P) \cup S]$ by contracting every edge of P not included in Q. Since every subgraph $C \in \mathcal{C}'$ is bad, it follows from (8) that, in H, every vertex in S has at least two neighbors in Q. Hence (H, Q) is a hedgehog of size $|Q| = (2c)^{2c}$.

The graph H is a minor of the subgraph G^* of G induced by

$$V(P) \cup \bigcup \{V(C) : C \in \mathcal{C}'\}.$$

Since vertices of P have degree at most k in G and $|C| \leq b$ for every $C \in \mathcal{C}'$, we have

$$|G^*| \leq r + r(k-1)b \leq f(c). \tag{10}$$

We claim that G^* contains a c-pumpkin-minor. By (10), such a minor directly yields a small c-pumpkin-model of G. Arguing by contradiction, assume G^* has no c-pumpkin-minor. Thus H has no c-pumpkin-minor either.

Applying Lemma 4.7 on (H, Q), we obtain either a bad cutset X of (H, Q) or a c-pumpkin-model of H. The latter case cannot happen since H has no c-pumpkin-minor, so assume the former holds and let $\{u, v\} := X$. Consider a connected component T of $H \setminus X$ that avoids both endpoints of Q. Let Z be the subgraph of G induced by $(V(T) \cap V(Q)) \cup \bigcup \{V(C) : v_C \in V(T)\}$. It follows from the definition of H and our choice of T that Z is a connected component of $G \setminus X$ such that u and v are both adjacent to some vertex in Z. Since G^* has no c-pumpkin-minor, it follows that (Z, u, v) is a c-outgrowth of G. But this implies that we could have applied **Z2** on G with the c-outgrowth (Z, u, v), a contradiction. $\square$

By Claim 4, we may assume that, for every $P \in \mathcal{P}$, the number $w(P)$ of white vertices in P satisfies

$$w(P) \geq \frac{r}{(2c)^{2c}} = k. \tag{11}$$

Our aim now is to use (11) to define a minor of K with large minimum degree. First, for every good subgraph $C \in \mathcal{C}$, "assign" v_C to an arbitrary neighbor of v_C in K. Next, for every $w \in W$, contract all edges $v_C w$ of K into the vertex w for all vertices v_C assigned to w. Similarly, for every $P \in \mathcal{P}$, contract all edges $v_C v_P$ into the vertex v_P for all vertices v_C assigned to v_P. Finally, remove the vertex v_C for every bad subgraph $C \in \mathcal{C}$. The resulting graph is denoted L.

For every vertex of L there is a natural induced subgraph of G that corresponds to it, namely the subgraph defined by all the edges that were contracted into w. Let S_w

and S_P be the (induced) subgraph of G that corresponds to the vertex $w \in W$ and v_P ($P \in \mathcal{P}$) of L, respectively. The subgraphs S_w ($w \in W$) and S_P ($P \in \mathcal{P}$) of G have diameter at most $2r$ and $3r$, respectively. Thus, by Lemma 2.1, a c-pumpkin-model of L of size q can be turned into one of G of size at most $3rc \cdot q$. Hence, in order to conclude the proof, it is enough to find a c-pumpkin-model in L of size at most $h(c) \log |L|$, since

$$ h(c) \log |L| \leq \frac{f(c)}{3rc} \log |L| \leq \frac{f(c)}{3rc} \log n. $$

To do so, we will show that L has large minimum degree.

First consider a vertex $w \in W$. Let a be the number of edges incident with w in K such that the other endpoint is a vertex of the form v_C ($C \in \mathcal{C}$) that was assigned to w. By (8), w cannot be adjacent in K to a vertex v_C corresponding to a bad subgraph $C \in \mathcal{C}$. Thus, it follows from the definitions of good subgraphs and L that

$$ \deg_L(w) \geq \frac{a}{\mu(K)} + (\deg_K(w) - a). $$

(The $\frac{a}{\mu(K)}$ term above comes from the fact that each vertex v_C that was assigned to w contributes at least one to the degree of w in L, while in K there were at most $\mu(K)$ edges between v_C and w.) Using (9) we obtain

$$ (12) \qquad \deg_L(w) > \frac{a}{c} + (\deg_K(w) - a) \geq \frac{\deg_K(w)}{c} \geq \frac{k}{c}. $$

Now consider a multipath $P \in \mathcal{P}$. Let a' be the number of edges incident with v_P in K such that the other endpoint is a vertex of the form v_C ($C \in \mathcal{C}$) that was assigned to v_P. Let b' be the number of edges incident with v_P in K that are not of the previous form and also not incident with a vertex v_C such that C is bad. By the definition of white vertices, we have $a' + b' \geq w(P)$ (recall that $w(P)$ is the number of white vertices in P). Using (11), it follows

$$ (13) \qquad \deg_L(v_P) \geq \frac{a'}{\mu(K)} + b' > \frac{a'}{c} + b' \geq \frac{w(P)}{c} \geq \frac{k}{c}. $$

It follows from (12) and (13) that L has minimum degree at least k/c. If $\mu(L) \geq c$, then L has a c-pumpkin-model of size two and we are trivially done, so let us assume $\mu(L) < c$. Then the underlying simple graph L' of L has minimum degree at least $k/c^2 \geq 2^{2\sqrt{c}+1}$. Using Corollary 4.9 on L', we find a c-pumpkin-model in L of the desired size, that is, of size at most $h(c) \log |L|$.

Finally, we note that each step of the proof can easily be realized in polynomial time. Therefore, a small c-pumpkin-model of G can be found in polynomial time. $\qquad \square$

4.4. **Algorithmic consequences.** Lemma 4.10 can be used to obtain $\mathcal{O}(\log n)$-approximation algorithms for both the MINIMUM c-PUMPKIN-HITTING and the MAXI-MUM c-PUMPKIN-PACKING problems for every fixed $c \geq 1$, as we now show.

Theorem 4.11. *Given an n-vertex graph G, a $\mathcal{O}(\log n)$-approximation for both the* MINIMUM c-PUMPKIN-HITTING *and the* MAXIMUM c-PUMPKIN-PACKING *problems on G can be computed in polynomial time using Algorithm 1, for any fixed integer $c \geq 1$.*

Proof. Consider Algorithm 1, where f is the function in Lemma 4.10. We will show that this algorithm provides a $\mathcal{O}(\log n)$-approximation for the two problems under consideration.

It should be clear that the collection $\mathcal{M}$ returned by Algorithm 1 is a c-packing of G, and similarly that the set X is a c-hitting set of G. Thus it is enough to show that they satisfy $|X| \leq (f(c) \log |G|) \cdot |\mathcal{M}|$ as claimed in the description of the algorithm. Indeed, since $|\mathcal{M}| \leq \nu_c(G) \leq \tau_c(G) \leq |X|$ and $f(c)$ is a constant depending only on c, this

---

Algorithm 1 A $\mathcal{O}(\log n)$-approximation algorithm.

INPUT: An arbitrary graph G
OUTPUT: A c-packing $\mathcal{M}$ of G and a c-hitting set X of G s.t. $|X| \leq (f(c)\log|G|)\cdot|\mathcal{M}|$
$\mathcal{M} \leftarrow \emptyset$; $X \leftarrow \emptyset$
If $|G| \leq 1$: Return $\mathcal{M}$, X /* G cannot have a c-pumpkin-minor */
Else:
    If G is not c-reduced:
        Apply a reduction rule on G, giving a graph H
        Call the algorithm on H, giving a packing $\mathcal{M}'$ and a c-hitting set X' of H
        Compute using Lemma 4.1(b) a c-packing $\mathcal{M}$ of G with $|\mathcal{M}| = |\mathcal{M}'|$
        Compute using Lemma 4.1(a) a c-hitting set X of G with $|X| \leq |X'|$
        Return $\mathcal{M}$, X
    Else:
        Compute using Lemma 4.10 a c-pumpkin-model $M = \{A, B\}$ of G with
            $|A \cup B| \leq f(c)\log|G|$
        $H \leftarrow G \setminus (A \cup B)$
        Call the algorithm on H, giving a packing $\mathcal{M}'$ and a c-hitting set X' of H
        $\mathcal{M} \leftarrow \mathcal{M}' \cup \{M\}$
        $X \leftarrow X' \cup A \cup B$
        Return $\mathcal{M}$, X

---

implies that the approximation factor of Algorithm 1 is $\mathcal{O}(\log n)$ for both the MINIMUM c-PUMPKIN-HITTING and the MAXIMUM c-PUMPKIN-PACKING problems.

We prove the inequality $|X| \leq (f(c)\log|G|)\cdot|\mathcal{M}|$ by induction on $|G|$. The inequality is obviously true in the base case, namely when $|G| \leq 1$, so let us assume $|G| > 1$.

If G is not c-reduced, then by induction the packing $\mathcal{M}'$ and the c-hitting set X' of H considered by the algorithm satisfy $|X'| \leq (f(c)\log|H|)\cdot|\mathcal{M}'|$, and we obtain

$$|X| \leq |X'| \leq (f(c)\log|H|)\cdot|\mathcal{M}'| = (f(c)\log|H|)\cdot|\mathcal{M}| \leq (f(c)\log|G|)\cdot|\mathcal{M}|$$

as desired.

If G is c-reduced, then by induction the packing $\mathcal{M}'$ and the c-hitting set X' of H satisfy $|X'| \leq (f(c)\log|H|)\cdot|\mathcal{M}'|$, and we have

$$\begin{aligned}
|X| &= |X'| + |A \cup B| \\
&\leq (f(c)\log|H|)\cdot|\mathcal{M}'| + f(c)\log|G| \\
&\leq (f(c)\log|G|)\cdot(|\mathcal{M}'| + 1) \\
&= (f(c)\log|G|)\cdot|\mathcal{M}|.
\end{aligned}$$

Thus $|X| \leq (f(c)\log|G|)\cdot|\mathcal{M}|$ holds in all cases.

Finally, we observe that there are at most n recursive calls during the whole execution of the algorithm, which implies that its running time is polynomial in n. $\qquad\square$

5. CONCLUDING REMARKS

On the one hand, we provided an FPT algorithm running in time $2^{\mathcal{O}(k)}\cdot n^{\mathcal{O}(1)}$ deciding, for any fixed $c \geq 1$, whether all c-pumpkin-models of a given graph can be hit by at most k vertices. In our algorithms we used protrusions but it may be possible to avoid it by further exploiting the structure of the graphs during the iterative compression routine (for example, a graph excluding the 3-pumpkin is a forest of cacti). We did not focus on optimizing the constants involved in our algorithms; it may be worth doing it, as well as enumerating all solutions, in the same spirit of [21] for FEEDBACK VERTEX SET.

It is natural to ask whether there exist faster algorithms for sparse graphs. Also, it would be interesting to have lower bounds for the running time of parameterized algorithms for this problem, in the spirit of those recently provided in [27]. One could as well consider other containment relations, like topological minor, induced minor, or contraction minor.

A more difficult problem seems to find single-exponential algorithms for the problem of deleting at most k vertices from a given graph so that the resulting graph has treewidth bounded by some constant c. Note that the case $c = 0$ (resp. $c = 1$) corresponds to p-Vertex Cover (resp. p-Feedback Vertex Set). Very recently, this problem has been solved for $c = 2$ [24], the cases $c \geq 3$ being still open. One could also consider the parameterized version of packing disjoint c-pumpkin-models, as it has been done for $c = 2$ in [5].

On the other hand, we provided a $\mathcal{O}(\log n)$-approximation for the problems of packing the maximum number of vertex-disjoint c-pumpkin-models, and hitting all c-pumpkin-models with the smallest number of vertices. It may be possible that the hitting version admits a constant-factor approximation; so far, such an algorithm is only known for $c \leq 3$.

As mentioned in the introduction, for the packing version there is a lower bound of $\Omega(\log^{1/2-\varepsilon} n)$ on the approximation ratio (under reasonable complexity-theoretic assumptions). In fact, this lower bound applies to both the vertex-disjoint packing and the edge-disjoint packing [20]. For $c = 2$, the problem of packing a maximum number of edge-disjoint cycles admits a $\mathcal{O}(\sqrt{\log n})$-approximation, whereas up to date $\mathcal{O}(\log n)$ is the best approximation ratio known for vertex-disjoint cycles [25]. Therefore, one might expect to get better approximation algorithms for packing edge-disjoint c-pumpkin models.

Finally, a class of graphs $\mathcal{H}$ has the *Erdős-Pósa property* if there exists a function f such that, for every integer k and every graph G, either G contains k vertex-disjoint subgraphs each isomorphic to a graph in $\mathcal{H}$, or there is a set $S \subseteq V(G)$ of at most $f(k)$ vertices such that $G \setminus S$ has no subgraph in $\mathcal{H}$. Given a connected graph H, let $\mathcal{M}(H)$ be the class of graphs that can be contracted to H. Robertson and Seymour [30] proved that $\mathcal{M}(H)$ satisfies the Erdős-Pósa property if and only if H is planar. Therefore, for every $c \geq 1$, the class of graphs that can be contracted to the c-pumpkin satisfies the Erdős-Pósa property. But the best known function f is super-exponential (see [13]), so it would be interesting to find a better function for this case. The only known lower bound on f is $\Omega(k \log k)$ when $c \geq 2$, which follows from the $\Omega(k \log k)$ lower bound given by Erdős and Pósa in their seminal paper [15] for $c = 2$.

References

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

[2] R. Balasubramanian, M. Fellows, and V. Raman. An improved fixed parameter algorithm for Vertex Cover. *Information Processing Letters*, 65:163–168, 1998.

[3] A. Becker and D. Geiger. Optimization of pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83:167–188, 1996.

[4] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) Kernelization. In *Proc. of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 629–638, 2009.

[5] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *Proc. of the 17th Annual European Symposium on Algorithms (ESA)*, volume 5757 of *LNCS*, pages 635–646, 2009.

[6] H. L. Bodlaender, J. van Leeuwen, R. B. Tan, and D. M. Thilikos. On interval routing schemes and treewidth. *Information and Computation*, 139(1):92–109, 1997.

[7] Y. Cao, J. Chen, and Y. Liu. On feedback vertex set new measure and new structures. In *Proc. of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 6139 of *LNCS*, pages 93–104, 2010.

[8] J. Chen, B. Chor, M. Fellows, X. Huang, D. W. Juedes, I. A. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005.

[9] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

[10] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. In *Proc. of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 177–186, 2008.

[11] B. Courcelle. The monadic second-order logic of graphs iii: tree-decompositions, minor and complexity issues. *ITA*, 26:257–286, 1992.

[12] F. K. H. A. Dehne, M. R. Fellows, M. A. Langston, F. A. Rosamond, and K. Stevens. An $O(2^{O(k)}n^3)$ FPT Algorithm for the Undirected Feedback Vertex Set Problem. *Theory of Computing Systems*, 41(3):479–492, 2007.

[13] R. Diestel. *Graph Theory*, volume 173. Springer-Verlag, 2005.

[14] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer-Verlag, 1999.

[15] P. Erdős and L. Pósa. On independent circuits contained in a graph. *Canadian Journal of Mathematics*, 17:347–352, 1965.

[16] S. Fiorini, G. Joret, and U. Pietropaoli. Hitting Diamonds and Growing Cacti. In *Proc. of the 14th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 6080 of *LNCS*, pages 191–204, 2010.

[17] S. Fiorini, G. Joret, D. O. Theis, and D. R. Wood. Small minors in dense graphs. *European Journal of Combinatorics*, 33(6):1226–1245, 2012.

[18] F. V. Fomin, S. Gaspers, D. Kratsch, M. Liedloff, and S. Saurabh. Iterative compression and exact algorithms. *Theoretical Computer Science*, 411(7-9):1045–1053, 2010.

[19] F. V. Fomin, D. Lokshtanov, N. Misra, G. Philip, and S. Saurabh. Hitting forbidden minors: Approximation and Kernelization. In *Proc. of the 28th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 9 of *LIPIcs*, pages 189–200, 2011.

[20] Z. Friggstad and M. Salavatipour. Approximability of packing disjoint cycles. *Algorithmica*, 60:395–400, 2011.

[21] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006.

[22] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[23] K.-i. Kawarabayashi, Y. Kobayashi, and B. A. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, `doi:10.1016/j.jctb.2011.07.004`. To appear.

[24] E. Kim, C. Paul, and G. Philip. A single-exponential FPT algorithm for the K_4-minor cover problem. Manuscript available at `http://arxiv.org/abs/1204.1417`, to appear in the *Proc. of SWAT*, 2012.

[25] M. Krivelevich, Z. Nutov, M. R. Salavatipour, J. Yuster, and R. Yuster. Approximation algorithms and hardness results for cycle packing problems. *ACM Transactions on Algorithms*, 3(4), 2007.

[26] M. Lampis. Algorithmic meta-theorems for restrictions of treewidth. In *Proc. of the 18th Annual European Symposium on Algorithms (ESA)*, volume 6346 of *LNCS*, pages 549–560, 2010.

[27] D. Lokshtanov, D. Marx, and S. Saurabh. Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal. In *Proc. of the 22nd annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 777–789, 2011.

[28] B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

[29] N. Robertson and P. Seymour. Graph Minors. XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[30] N. Robertson and P. D. Seymour. Graph Minors. V. Excluding a Planar Graph. *Journal of Combinatorial Theory, Series B*, 41(1):92–114, 1986.

Département d'Informatique
Université Libre de Bruxelles
Brussels, Belgium
   E-mail address: `gjoret@ulb.ac.be`

AlGCo project-team
CNRS, LIRMM
Montpellier, France
  E-mail address: {paul,sau}@lirmm.fr

Laboratoire LIP
Université de Lyon, CNRS, ENS Lyon, INRIA, UCBL
Lyon, France
  E-mail address: stephan.thomasse@ens-lyon.fr

The Institute of Mathematical Sciences
Chennai, India
  E-mail address: saket@imsc.res.in