Hitting and Harvesting Pumpkins

Gwenaël Joret[1], Christophe Paul[2], Ignasi Sau[2],
Saket Saurabh[3], and Stéphan Thomassé[4]

[1] Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium
gjoret@ulb.ac.be
[2] AlGCo project-team, CNRS, LIRMM, Montpellier, France
{paul,sau}@lirmm.fr
[3] The Institute of Mathematical Sciences, Chennai, India
saket@imsc.res.in
[4] Laboratoire LIP (U. Lyon, CNRS, ENS Lyon, INRIA, UCBL), Lyon, France
stephan.thomasse@ens-lyon.fr

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: On the one hand, we provide 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 the *single-exponential* FPT algorithms for Vertex Cover and Feedback Vertex Set, which correspond to the cases $c = 1, 2$ respectively. For this, we use a combination of iterative compression and a kernelization-like technique. On the other hand, we present an $\mathcal{O}(\log n)$-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. Our main ingredient here is a combinatorial lemma saying that any *properly reduced* n-vertex graph has a c-pumpkin-model of size at most $f(c)\log n$, for a function f depending only on c.

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 few vertices, 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

C. Demetrescu and M.M. Halldórsson (Eds.): ESA 2011, LNCS 6942, pp. 394–407, 2011.

on FPT algorithms for the parameterized version of the hitting problem, as well as on poly-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 for short)
> *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?

When $c = 1$, the p-c-HIT problem is the p-VERTEX COVER problem [2, 8]. For $c = 2$, it is the p-FEEDBACK VERTEX SET problem [16,9]. When $c = 3$, this corresponds to the recently introduced p-DIAMOND HITTING SET problem [12].

The p-c-HIT problem can also be seen as a particular case of the following problem, recently introduced by Fomin *et al.* [14] 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 [14] 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.* [8]. On the other hand, the question about the existence of single-exponential algorithms for p-FEEDBACK VERTEX SET was open for a while and was finally settled independently by Guo *et al.* [16] (using iterative compression) and by Dehne *et al.* [9].

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 $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ [7, 17]. It is worth mentioning here that a similar quantitative approach was taken by Lampis [20] 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 for short)
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 $\lvert S \rvert$.

Maximum c-Pumpkin-Packing (Max c-Pack for short)
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 $\lvert \mathcal{M} \rvert$.

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 an $\mathcal{O}(\log n)$ factor [19]. For $c = 3$, Min 3-Hit is the Diamond Hitting Set problem studied by Fiorini *et al.* in [12], 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 factor $\mathcal{O}(\log n)$ for every fixed $c \geq 1$. Note that this algorithm matches the best existing algorithms for Max c-Pack for $c = 2$ [19]. 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 [14]. However, for the Max c-Pack problem, there was no approximation algorithm known before except for the case $c = 2$. 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 an $\mathcal{O}(\log^{1/2-\varepsilon} n)$ factor: For $c = 2$ this was shown by Friggstad and Salavatipour [15], and their result can be extended to the case $c > 2$ in the following straightforward way: given an instance G of Max 2-Hit, we build an instance of Max c-Hit by replacing each edge of G with $c - 1$ parallel edges.

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.* [12], who dealt with the $c = 3$ case.

2 Preliminaries

Graphs. We use standard graph terminology, see for instance [10]. 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. 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.

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 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
(ii) 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 $G \setminus X$ 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 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 $k \cdot \Delta(H) \cdot s$.*

Tree-width. We refer the reader to Diestel's book [10] for the definition 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. *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.*

Theorem 1 (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 1. *Every n-vertex graph 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 [14] that given an instance (G, k) of p-$\mathcal{F}$-Hit such that $\mathcal{F}$ contains 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. (We assume that the reader is familiar with the basic concepts of Parameterized Complexity; c.f. for instance [11].) This kernel leads to the following simple FPT algorithm for p-$\mathcal{F}$-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 any of the graphs in $\mathcal{F}$ as a minor, using the poly-time algorithm of Robertson and Seymour [22] for every graph in $\mathcal{F}$. If for some S we have that $K[V(K) \setminus S]$ contains no graph from $\mathcal{F}$ 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, or p-c-Disjoint Hit for short. In p-c-Disjoint Hit, apart from a graph G and a positive integer k, we are also given a set S of size at most $k+1$ such that $G \setminus S$ does not contain the c-pumpkin as a minor. Here the objective is to find a set $S' \subseteq V(G) \setminus S$ such that $|S'| \leq k$ and $G \setminus S'$ 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 ($\star$[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 $(\eta(c)+1)^k \cdot n^{\mathcal{O}(1)}$.*

Lemma 3 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.

Overview of the algorithm. The algorithm for p-c-Disjoint Hit is based on a combination of branching and poly-time preprocessing. Let (G, S, k) be the given instance of p-c-Disjoint Hit and let $V := V(G)$. Our main objective is to eventually show that $A := \{v \in V \setminus S : N_G(v) \cap S \neq \emptyset\}$ has cardinality $\mathcal{O}(k)$. As far as we cannot guarantee this upper bound, we will see that we can branch suitably to obtain a few subproblems. Once we have the desired upper bound, we use a protrusion-based reduction rule from [14] to give a poly-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 in this particular case. Notice that once we have a linear vertex kernel of size αk for p-c-Disjoint Hit, we can solve the problem in $\binom{\alpha k}{k} \cdot k^{\mathcal{O}(1)}$. So the overall algorithm consists of a recursion tree where in leaf nodes we obtain a linear kernel and then solve the problem using brute force enumeration.

We can now proceed to the formal description of the algorithm. Let

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

Linear kernel. We start off with a procedure, called *protrusion rule*, that bounds the size of our graph when $|V_1 \cup V_{\geq 2}| = \mathcal{O}(k)$. 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 $|\partial_G(X)| \leq r$. We now state the protrusion rule.

P Let (G, S, k) be an instance and let $\gamma : \mathbb{N} \to \mathbb{N}$ be the function defined in [14, Lemma 5] (originally shown in [4]). If G contains a $4c$-protrusion Y of size at least $\gamma(4c)$, then replace Y to obtain an equivalent instance (G^*, S, k^*) such that $|V(G^*)| < |V(G)|$.

The proof of the next lemma is identical to the proof of [14, Theorem 1].

Lemma 4 ($\star$). *If $|V_1 \cup V_{\geq 2}| = \mathcal{O}(k)$ then p-c-Disjoint Hit has a kernel with $\mathcal{O}(k)$ vertices.*

Branching procedure. For our branching algorithm we take the following measure:

$$\mu \;=\; \mathbf{cc}(G[S]) + k. \tag{1}$$

[1] The proofs of the results marked with "$\star$" can be found in [18].

For simplicity we call the vertices in V_1 *white*. We start with some simple reduction rules (depending on c) which will be applied in the compression routine. We also prove, together with the description of each rule, that they are valid for our problem.

R1. Let C be a connected component of $G[V \setminus S]$. If C does not have any neighbor in S, we can safely delete C, as its vertices will never participate in a c-pumpkin-model.

R2. Let C be a connected component of $G[V \setminus (S \cup \{v\})]$ for some vertex $v \in V \setminus S$. If C does not have any neighbor in S, we can safely delete C, as its vertices will never participate in a minimal c-pumpkin-model.

R3. Let C be a connected component of $G[V \setminus S]$. If C has exactly one neighbor v in S and $G[V(C) \cup \{v\}]$ is c-pumpkin-free, we can safely delete it, as its vertices will never participate in a minimal c-pumpkin-model.

R4. Let B be a connected induced subgraph (not necessarily a connected component) of $G[V \setminus S]$, let $v \in V(B)$, and let $P_1, \ldots, P_\ell$ be the connected components of $G[V(B) \setminus \{v\}]$. Assume that $\ell \geq c$ and that the following conditions hold:

(i) For $1 \leq i \leq \ell$, all neighbors of vertices in P_i are in $S \cup V(P_i) \cup \{v\}$.
(ii) For $1 \leq i \leq \ell$, there exists a vertex $u_i \in S$ such that $\bigcup_{w \in V(P_i)} N_G(w) \cap S = \{u_i\}$.
(iii) For $1 \leq i \leq \ell$, $G[V(P_i) \cup \{u_i\}]$ is c-pumpkin-free.
(iv) The vertices $u_1, \ldots, u_\ell$ belong to the same connected component D of $G[S]$.

Then we include vertex v in the solution and we decrease the parameter by one. Indeed, note that as $\ell \geq c$ and by conditions (ii) and (iv), $G[V(B) \cup V(D)]$ contains a c-pumpkin, so any solution needs to contain at least one vertex of B, since we are looking for a solution that does not include vertices from S. On the other hand, by condition (iii) every minimal c-pumpkin-model intersecting B necessarily contains vertex v, and condition (i) guarantees that no minimal c-pumpkin-model of $G \setminus \{v\}$ contains a vertex of B. Therefore, we can safely include vertex v in the solution and decrease the parameter accordingly. See Fig. 1 for an illustration for $c = 4$.

We say that the instance (G, S, k) is (S, c)-*reduced* if rules **R1**, **R2**, **R3**, or **R4** cannot be applied anymore. Note that these reduction rules can easily be applied in polynomial time.

We now describe a branching rule which will be used in our compression routine.

B Let P be a simple path in $G[V \setminus S]$ with $|V(P)| = \ell$ and let v_1 and v_2 be the endpoints of P. Suppose that v_1 (resp. v_2) has a neighbor in a connected component C_1 (resp. C_2) of $G[S]$, with $C_1 \neq C_2$. Then we branch for all 2^ℓ subsets of vertices in P. Note that in every case we decrease the measure of the iterative compression, as either we include at least one vertex in the solution or we decrease the number of connected components of $G[S]$.

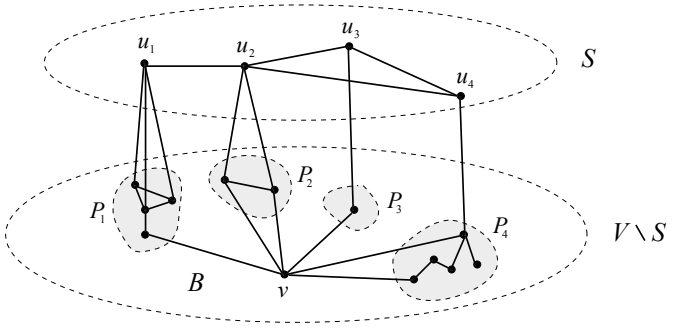

Fig. 1. Illustration of reduction rule **R4** for $c = 4$

The branching rule above will be used on paths P that are "small enough". The following Lemmas 5 and 8 are key to our algorithm. We also need two intermediate technical results, which will be used in the proof of Lemma 8.

Lemma 5 ($\star$). *There is a function $f(c)$ such that if (G, S, k) is an (S, c)-reduced* YES*-instance to the p-c-*DISJOINT HIT *problem, then $|V_{\geq 2}| \leq f(c) \cdot k$.*

Lemma 6 ($\star$). *There is a function $g(c)$ such that if (G, S, k) is an (S, c)-reduced* YES*-instance to the p-c-*DISJOINT HIT *problem and $\mathcal{C}$ is a collection of disjoint connected subgraphs of $G[V \setminus S]$ such that each subgraph has at least two distinct neighbors in S, then $|\mathcal{C}| \leq g(c) \cdot k$.*

Lemma 7 ($\star$). *In an (S, c)-reduced* YES*-instance (G, S, k), the number of connected components of $G[V \setminus S]$ is $\mathcal{O}(k)$.*

Now we prove our key structural lemma.

Lemma 8. *There is a function $h(c)$ such that if (G, S, k) is an (S, c)-reduced* YES*-instance to the p-c-*DISJOINT HIT *problem, then either $|V_1| \leq h(c) \cdot k$, or we can apply protrusion rule* **P***, or we can branch according to branching rule* **B***.*

Proof: We proceed to find a packing of disjoint connected subgraphs $\mathcal{P} = \{B_1, \ldots, B_\ell\}$ of $G[V \setminus S]$ 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 $H \subseteq G[V \setminus S]$, 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 either bound the number of white vertices, or to apply protrusion rule **P**, or to branch according to branching rule **B**.

We start with $\mathcal{P}$ containing all the connected components C of $G[V \setminus S]$ such that $w(C) > c^3$, and we recursively try to refine the current packing. By Lemma 7, 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 $G[V \setminus S]$ by setting $A := \{v\}$ and $B := V(P)$, contradicting the fact that $G[V \setminus S]$ is c-pumpkin-free. □

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.

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

Claim 2 ($\star$). *In the packing $\mathcal{P}$ constructed above, the number of suitable or unsuitable blocks is $\mathcal{O}(k)$.*

Claim 3 ($\star$). *In an (S, c)-reduced instance, either the total number of pieces in the packing $\mathcal{P}$ constructed above is $\mathcal{O}(k)$ or we can branch according to branching rule* **B**.

More precisely, the proof of Claim 3 shows that if the number of pieces is not $\mathcal{O}(k)$, then we can apply protrusion rule **P** (and possibly reduction rules **R2** and **R3**) to eventually branch according to branching rule **B**.

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 unless we can branch according to branching rule **B**, so if we cannot branch the number of white vertices contained in pieces of unsuitable blocks is also $\mathcal{O}(k)$.□

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

Theorem 2. *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 it is sufficient to obtain an algorithm for p-c-Disjoint Hit. Recall the measure $\mu = \mathbf{cc}(G[S]) + k$. Now by Lemma 8 either we branch according to branching rule **B**, or we can apply protrusion rule **P** and get a smaller instance, or we have that $|V_1| \leq h(c) \cdot k$. In the first case we branch into $\alpha(c)$ ways and in each branch the measure decreases by one as either we include a vertex in our potential solution or we decrease the number of connected components of $G[S]$. Here $\alpha(c)$ is a constant that only depends on c. This implies that the number of nodes in the branching tree is upper-bounded by $\alpha(c)^{\mu} \leq \alpha(c)^{2k+1} = 2^{\mathcal{O}(k)}$. In the second case, we get a smaller instance in polynomial time. Finally, in the third case, using Lemma 5 we get that $|V_1 \cup V_{\geq 2}| = \mathcal{O}(k)$. Thus using Lemma 4 we can get 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. □

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 an $\mathcal{O}(\log n)$-approximation algorithm for both problems.

Reduction rules. We describe two reduction rules for hitting/packing c-pumpkin-models, which given an input graph G, 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.

An *outgrowth* of a graph G is a triple (C, u, v) such that (i) u, v are two distinct vertices of G; (ii) C is a connected component of $G \setminus \{u, v\}$ with $|V(C)| \geq 2$; and (iii) u and v both have at least one neighbor in C in the graph G.

Given an outgrowth (C, u, v) of a graph G, we let $\Gamma(C, u, v)$ denote the subgraph of G induced by $V(C) \cup \{u, v\}$ where all the edges between u and v are removed. We let $\Lambda(C, u, v)$ be the graph $\Gamma(C, u, v)$ where an edge between u and v has been added. Also, we define $\gamma(C, u, v)$ as the largest integer k such that $\Gamma(C, u, v)$ has a k-pumpkin-model $\{A, B\}$ with $u \in A$ and $v \in B$, and $\lambda(C, u, v)$ as the largest integer k such that $\Lambda(C, u, v)$ has a k-pumpkin-model $\{A, B\}$ with $u, v \in A$.[2] See Fig. 2 for an illustration.

Fig. 2. The graph $\Gamma(C, u, v)$ of two outgrowths (C, u, v). We have $\lambda(C, u, v) = 8 < 9 = \gamma(C, u, v)$ in the left one, while $\lambda(C, u, v) = 7 > 5 = \gamma(C, u, v)$ holds for the right one.

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

Z1 Suppose v is a vertex of G such that no block (here, a *block* is a maximal 2-connected component) of G containing v has a c-pumpkin-minor. Then define H as the graph obtained from G by removing v.

Z2 Suppose (C, u, v) is an outgrowth of G such that $\Gamma(C, u, v)$ has no c-pumpkin-minor. Assume further that **Z1** cannot be applied on G. Let $\gamma := \gamma(C, u, v)$ and $\lambda := \lambda(C, u, v)$. If $\lambda \leq \gamma$, define H as the graph obtained from $G \setminus V(C)$ by adding γ parallel edges between u and v. Otherwise, define H as the graph obtained from $G \setminus V(C)$ by adding a new vertex v_C and linking v_C to u with γ parallel edges, and to v with $\lambda - \gamma$ parallel edges.

See Fig. 3 for an illustration of **Z2**. 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 9 ($\star$). *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 $|\mathcal{M}| = |\mathcal{M}'|$.

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 $\mathcal{O}_c(\log n)$. We will use the following recent result by Fiorini *et al.* [13] about the existence of small minors in *simple* graphs with large average degree.

[2] We note that $\lambda(C, u, v)$ can equivalently be defined as the largest integer k such that $\Lambda(C, u, v)/uv$ has a k-pumpkin-model $\{A, B\}$ with $u \in A$. Thus $\lambda(C, u, v)$ can be computed in polynomial time.

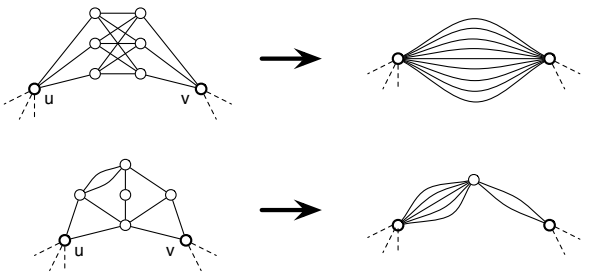

Fig. 3. Illustration of reduction rule **Z2** on the two outgrowths from Fig. 2

Theorem 3 (Fiorini *et al.* [13]). *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.*

Even though the authors of [13] do not mention it explicitly, we note that the proof given in that paper can easily be turned into a poly-time algorithm finding such a K_t-model. 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 3, which is central in the proof of Lemma 10.

Corollary 2. *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 c-reduced graphs; its proof strongly relies on a graph structure that we call *hedgehog* (see [18] for more details). The idea is that a hedgehog witnesses the existence of either a small c-pumpkin-model or an outgrowth, which is impossible in a c-reduced graph.

Lemma 10 ($\star$). *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.*

Algorithmic consequences. Lemma 10 can be used to obtain $\mathcal{O}(\log n)$-approximation algorithms for both the Minimum c-Pumpkin-Hitting and the Maximum c-Pumpkin-Packing problems for every fixed $c \geq 1$, as we now show.

Theorem 4 ($\star$). *Given an n-vertex graph G, an $\mathcal{O}(\log n)$-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 $c \geq 1$.*

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

Algorithm 1. An $\mathcal{O}(\log n)$-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 9(b) a c-packing $\mathcal{M}$ of G with $|\mathcal{M}| = |\mathcal{M}'|$
    Compute using Lemma 9(a) a c-hitting set X of G with $|X| \leq |X'|$
    Return $\mathcal{M}$, X
  Else:
    Compute using Lemma 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

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). 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 [21]. 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 tree-width bounded by some constant. 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 an $\mathcal{O}(\log n)$-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 $c \leq 3$.

References

1. Bafna, V., Berman, P., Fujito, T.: A 2-approximation algorithm for the undirected feedback vertex set problem. SIAM Journal on Discrete Mathematics 12(3), 289–297 (1999)
2. Balasubramanian, R., Fellows, M., Raman, V.: An improved fixed parameter algorithm for Vertex Cover. Information Processing Letters 65, 163–168 (1998)

3. Becker, A., Geiger, D.: 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. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (Meta) kernelization. In: Proc. of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 629–638 (2009)
5. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 635–646. Springer, Heidelberg (2009)
6. Bodlaender, H.L., van Leeuwen, J., Tan, R.B., Thilikos, D.M.: On interval routing schemes and treewidth. Information and Computation 139(1), 92–109 (1997)
7. Chen, J., Chor, B., Fellows, M., Huang, X., Juedes, D.W., Kanj, I.A., Xia, G.: Tight lower bounds for certain parameterized NP-hard problems. Information and Computation 201(2), 216–231 (2005)
8. Chen, J., Kanj, I.A., Xia, G.: Improved upper bounds for vertex cover. Theoretical Computer Science 411(40-42), 3736–3756 (2010)
9. Dehne, F.K.H.A., Fellows, M.R., Langston, M.A., Rosamond, F.A., Stevens, K.: An $O(2^{O(k)}n^3)$ FPT Algorithm for the Undirected Feedback Vertex Set Problem. Theory of Computing Systems 41(3), 479–492 (2007)
10. Diestel, R.: Graph Theory, vol. 173. Springer, Heidelberg (2005)
11. Downey, R.G., Fellows, M.R.: Parameterized complexity. Springer, Heidelberg (1999)
12. Fiorini, S., Joret, G., Pietropaoli, U.: Hitting Diamonds and Growing Cacti. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 191–204. Springer, Heidelberg (2010)
13. Fiorini, S., Joret, G., Theis, D.O., Wood, D.R.: Small Minors in Dense Graphs (2010) (manuscript), http://arxiv.org/abs/1005.0895
14. Fomin, F.V., Lokshtanov, D., Misra, N., Philip, G., Saurabh, S.: Hitting forbidden minors: Approximation and Kernelization. In: Proc. of the 28th Symposium on Theoretical Aspects of Computer Science (STACS). LIPIcs, vol. 9, pp. 189–200 (2011)
15. Friggstad, Z., Salavatipour, M.: Approximability of packing disjoint cycles. Algorithmica 60, 395–400 (2011)
16. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. Journal of Computer and System Sciences 72(8), 1386–1396 (2006)
17. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? Journal of Computer and System Sciences 63(4), 512–530 (2001)
18. Joret, G., Paul, C., Sau, I., Saurabh, S., Thomassé, S.: Hitting and Harvesting Pumpkins (2011) (manuscript), http://arxiv.org/abs/1105.2704
19. Krivelevich, M., Nutov, Z., Salavatipour, M.R., Yuster, J., Yuster, R.: Approximation algorithms and hardness results for cycle packing problems. ACM Transactions on Algorithms 3(4) (2007)
20. Lampis, M.: Algorithmic meta-theorems for restrictions of treewidth. In: de Berg, M., Meyer, U. (eds.) ESA 2010. LNCS, vol. 6346, pp. 549–560. Springer, Heidelberg (2010)
21. Lokshtanov, D., Marx, D., Saurabh, S.: Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal. In: Proc. of the 22nd annual ACM-SIAM Symposium on Discrete algorithms (SODA), pp. 777–789 (2011)
22. Robertson, N., Seymour, P.: Graph Minors. XIII. The Disjoint Paths Problem. Journal of Combinatorial Theory, Series B 63(1), 65–110 (1995)