Theoretical Computer Science 557 (2014) 76–86

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Brushing with additional cleaning restrictions

Piotr Borowiecki [a,1], Dariusz Dereniowski [a,2], Paweł Prałat [b,*,3]

[a] Department of Algorithms and System Modeling, Gdańsk University of Technology, Gdańsk, Poland
[b] Department of Mathematics, Ryerson University, Toronto, ON, Canada, M5B 2K3

ARTICLE INFO

Article history:
Received 17 December 2013
Received in revised form 28 August 2014
Accepted 8 September 2014
Available online 16 September 2014
Communicated by G. Ausiello

Keywords:
Graph searching
Brush number
Chip firing

ABSTRACT

In graph cleaning problems, brushes clean a graph by traversing it subject to certain rules. We consider the process where at each time step, a vertex that has at least as many brushes as incident, contaminated edges, sends brushes down these edges to clean them. Various problems arise, such as determining the minimum number of brushes (called the brush number) that are required to clean the entire graph. Here, we study a new variant of the problem in which no more than k brushes can be sent at any time step.

1. Introduction

Imagine a network of pipes that must be periodically cleaned of a regenerating contaminant, say algae. In *cleaning* such a network (see [11] for a paper that introduced this process), there is an initial configuration of brushes on vertices; every vertex and edge is initially regarded as *dirty*. A vertex is ready to be cleaned if it has at least as many brushes as incident dirty edges. When a vertex is cleaned, it sends one brush along each incident dirty edge; these edges are now said to be *clean*. (No brush ever traverses a clean edge.) The vertex is also deemed clean. Excess brushes remain on the clean vertex and take no further part in the process. (In fact, for our purpose in this paper, we may think about clean vertices as if they were removed from the graph.) Fig. 1 illustrates the cleaning process for a graph G where there are initially two brushes at vertex b and one brush at vertex c. The solid edges indicate dirty edges while the dashed edges indicate clean edges. For example, the process starts with vertex b being cleaned, sending a brush to each of vertices a and d.

This model, perhaps surprisingly, corresponds to the *minimum total imbalance* of the graph which is used in graph drawing theory [5] and is well-studied, in particular, for random graphs [1,15]. (See also [9] for algorithmic aspects, [12,14] for a related model of cleaning with brooms, [4] for a variant with no edge capacity restrictions, [7] for a combinatorial game, and [13] for a relation to more general family of perfect vertex elimination schemes leading to upper-locally distributed lattices.) Having been inspired by chip firing processes [3,10], the manner in which brushes disperse from an individual vertex is such that they do so in unison, provided that their vertex meets the criteria to be cleaned. Models in which multiple vertices may be cleaned simultaneously are called *parallel cleaning models* (see [6] for more details). In contrast,

* Corresponding author.
E-mail addresses: pborowie@eti.pg.gda.pl (P. Borowiecki), deren@eti.pg.gda.pl (D. Dereniowski), pralat@ryerson.ca (P. Prałat).

[1] The first author was partially supported by National Science Centre grant DEC-2011/02/A/ST6/00201.
[2] The second author was partially supported by National Science Centre grant DEC-2011/02/A/ST6/00201 and a scholarship for outstanding young researchers founded by the Polish Ministry of Science and Higher Education.
[3] The third author was supported by NSERC and Ryerson University.

http://dx.doi.org/10.1016/j.tcs.2014.09.005
0304-3975/

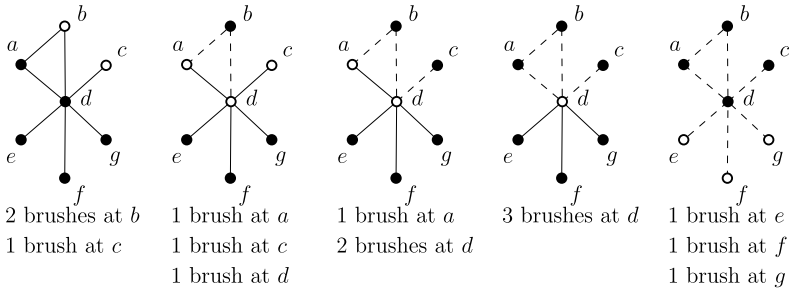

Fig. 1. An example of the cleaning process for graph G.

sequential cleaning models mandate that vertices are cleaned one at a time. The variant considered in [11] and the one we consider in this paper are sequential in nature.

In this paper, we consider the variant of the process in which there is one additional restriction, namely, it is not allowed to clean vertices with more than k dirty neighbours (k is a fixed parameter). For example, 3 brushes were enough to clean the graph presented on Fig. 1. However, if at most $k = 2$ brushes can be sent in each time step, vertex d cannot be cleaned; one additional brush has to be introduced (say, at vertex e) and our task can be accomplished with 4 brushes in total.

The formal definition of the k-restricted process and th k-brush number, $b_k(G)$, is introduced in Section 2 ($b(G)$ denotes the original brush number introduced in [11] and also defined in Section 2). In Section 3, we characterize graphs that can be cleaned in this process. Moreover, we show that for any $k \in \mathbb{N}$, $x \in [1, 2]$, and $\varepsilon > 0$, there exists a tree T such that $|b_k(T)/b(T) - x| < \varepsilon$. This result is sharp, that is, this property does not hold for other values of x. Similarly, it is shown that for any $k \in \mathbb{N} \setminus \{1\}$, $x \in [1, \infty)$, and $\varepsilon > 0$, there exists a graph G such that $|b_k(G)/b(G) - x| < \varepsilon$. In Section 4, we study a polynomial-time algorithm that, given any tree T and any $k \in \mathbb{N}$, computes the k-brush number of T. The paper is concluded with a few open problems (see Section 5).

Throughout, we consider only finite, simple, undirected graphs in the paper. For background on graph theory, the reader is directed to [16].

2. Definitions

As already noted, the graph cleaning model we consider here differs from the one presented in [11] in that one is not allowed to clean vertices with more than k dirty neighbours. Now, we formally define the cleaning process we are considering in this paper. Let $k \in \mathbb{N}$ and let $G = (V, E)$ be any graph. The initial configuration of brushes is given by the function $\omega_0 : V \to \mathbb{N} \cup \{0\}$, where $\omega_0(v)$ is the number of brushes initially at vertex v, and all vertices and edges of the graph are initially dirty. At the end of each step t of the process, $\omega_t(v)$ denotes the number of brushes at vertex $v \in V$, and $D_t \subseteq V$ denotes the set of dirty vertices. An edge $uv \in E$ is dirty if and only if both u and v are dirty; that is, $\{u, v\} \subseteq D_t$. Finally, let $D_t(v)$ denote the number of dirty edges incident to v at the end of step t; that is,

$$D_t(v) = \begin{cases} |N(v) \cap D_t| & \text{if } v \in D_t \\ 0 & \text{otherwise} \end{cases}$$

(where $N(v)$ denotes, as usual, the neighbourhood of v).

Definition 1. Let $k \in \mathbb{N}$. The **k-restricted cleaning process** $\mathfrak{P}(G, k, \omega_0) = \{(\omega_t, D_t)\}_{t=0}^{L}$ of an undirected graph $G = (V, E)$ with an **initial configuration of brushes** ω_0 is as follows:

(0) Initially, all vertices are dirty: $D_0 = V$; set $t := 0$.

(1) Let α_{t+1} be any vertex in D_t such that $\omega_t(\alpha_{t+1}) \geq D_t(\alpha_{t+1})$ and $D_t(\alpha_{t+1}) \leq k$. If no such vertex exists, then stop the process (set $L := t$), return the **cleaning sequence** $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_L)$, the **final set of dirty vertices** D_L, and the **final configuration of brushes** ω_L.

(2) Clean α_{t+1} and all dirty incident edges by moving a brush from α_{t+1} to each dirty neighbour. More precisely, $D_{t+1} = D_t \setminus \{\alpha_{t+1}\}$, $\omega_{t+1}(\alpha_{t+1}) = \omega_t(\alpha_{t+1}) - D_t(\alpha_{t+1})$, and for every $v \in N(\alpha_{t+1}) \cap D_t$, $\omega_{t+1}(v) = \omega_t(v) + 1$, the other values of ω_{t+1} remain the same as in ω_t.

(3) Set $t := t + 1$ and go back to **(1)**.

In the model considered in [11], there is no restriction that $D_t(\alpha_{t+1}) \leq k$. All other rules remain the same. It was shown in [11] that, given a graph G and an initial configuration ω_0, the cleaning process of [11] returns a unique final set of dirty vertices. Consequently, if for a given ω_0, there exists a cleaning sequence that cleans G, we know that every cleaning sequence will clean G. This is also the case here and easily follows from the simple observation that once a vertex v is ready to be cleaned it stays in this state until it is actually cleaned.

Finally, let us note that, for a fixed natural number k, some graphs cannot be cleaned at all. However, it is straightforward to obtain a characterization of graphs that can be cleaned (see Theorem 4). Thus, the following definition is natural.

Definition 2. Let $k \in \mathbb{N}$. A graph $G = (V, E)$ **can be cleaned** (by the k-restricted cleaning process) by the initial configuration of brushes ω_0 if the cleaning process $\mathfrak{P}(G, k, \omega_0)$ returns an empty final set of dirty vertices (that is, $D_L = \emptyset$).

Let the **k-brush number**, $b_k(G)$, be the minimum number of brushes needed to clean G if G can be cleaned; otherwise, $b_k(G) = \infty$. That is, if G can be cleaned, then

$$b_k(G) = \min_{\omega_0 : V \to \mathbb{N} \cup \{0\}} \left\{ \sum_{v \in V} \omega_0(v) : G \text{ can be cleaned by } \omega_0 \right\}.$$

Recall that the brush number for the model in [11] was denoted by $b(G)$. Since there were less restrictions for this process than for the k-restricted one which we are considering here, we immediately get that for every $k \in \mathbb{N}$ we have $b_k(G) \geq b(G)$. As we will see, the difference can be arbitrarily large although there are some classes of graphs for which $b_k(G) = b(G)$ (consider, for example, paths). It also follows immediately from the definition that $b_{k_1}(G) \geq b_{k_2}(G)$ whenever $1 \leq k_1 < k_2$. On the other hand, it is clear that the additional restriction is trivially satisfied for $k \geq \Delta(G)$ and so $b_k(G) = b(G)$ for every $k \geq \Delta(G)$.

When a graph G is cleaned using the cleaning process, each edge of G is traversed by exactly one brush. Note that no brush may return to a vertex it has already visited, motivating the following definition.

Definition 3. The **brush path** of a given brush is the path formed by the set of edges cleaned by that brush.

By definition, G can be decomposed into $b(G)$ brush paths. The minimum number of paths a graph G can be decomposed into therefore yields a lower bound for $b(G)$ (and so for $b_k(G)$ as well). This is only a lower bound because some path decompositions would not be valid in the cleaning process. For example, K_4 can be decomposed into 2 undirected paths but $b(K_4) = 4$.

Finally, let us mention that for any graph G consisting of ℓ components $G_1, G_2, \ldots, G_\ell$, we have $b_k(G) = \sum_{i=1}^{\ell} b_k(G_i)$ and so, without loss of generality, we will always assume that G is connected.

3. Characterization of graphs that can be cleaned and relation to the ordinary brush number

As we already mentioned, not every graph can be cleaned in the new model. For a given $k \in \mathbb{N}$, let $\mathcal{F}_k$ be the family of all (connected) graphs that can be cleaned by the k-restricted cleaning process. Clearly, this family is ascending; that is, $\mathcal{F}_k \subset \mathcal{F}_{k+1}$ for every $k \in \mathbb{N}$. We start this section with characterizing graphs that can be cleaned. However, before we do it, we need to introduce a few more definitions.

Let $k \in \mathbb{N}$. The **k-core** of a graph is the largest subgraph of minimum degree at least k. The k-core can be found by the **vertex deletion algorithm** that repeatedly deletes vertices with degree less than k. This algorithm always terminates with the k-core of the graph, which is possibly empty. This concept is related to the degeneracy one. A graph G is **k-degenerate** if every subgraph of G has a vertex of degree at most k. The **degeneracy** of a graph is the smallest value of k for which it is k-degenerate. In other words, the degeneracy of a graph is the largest value of k such that it has non-empty k-core.

Since cleaning vertices can be viewed as deleting them, the connection between the two processes is straightforward. In order to decide if a graph G can be cleaned, it is enough to start with, say, $\omega_0(v) = \deg(v)$ for each $v \in V$ so that the only restriction is the additional restriction that $D_t(\alpha_{t+1}) \leq k$. (We do not care about optimizing the number of brushes used at this point.) Vertex v can be cleaned if the number of dirty neighbours is at most k, which is equivalent to the property that the degree of v is at most k (in the graph induced by the set of dirty vertices). We get immediately the following result.

Theorem 4. *Let $k \in \mathbb{N}$. A graph $G \in \mathcal{F}_k$ if and only if G is k-degenerate.*

In other words, Theorem 4 says that a graph G can be cleaned by the k-restricted cleaning process if and only if $(k+1)$-core of G is empty.

We will be investigating the following sets. For $k \in \mathbb{N}$, let

$$A_k = \left\{ \frac{b_k(G)}{b(G)} : G \in \mathcal{F}_k \right\},$$

and

$$B_k = \left\{ \frac{b_k(G)}{b(G)} : G \in \mathcal{F}_1 \right\} \subseteq A_k.$$

Clearly $A_1 = B_1$. Since $b_k(G) \geq b(G)$ for any graph G, we get that $B_k \subseteq A_k \subseteq [1, \infty)$. We will show that, in fact, the closure of B_k is $[1, 2]$, regardless of the value of $k \in \mathbb{N}$ (see Theorem 6 for the $k = 1$ case and Theorem 7 for the $k \geq 2$ case). Moreover, we will show that the closure of A_k is $[1, \infty)$ for any $k \geq 2$ (see Theorem 8). Recall that we restrict ourselves to connected graphs; the corresponding properties for families containing disconnected graphs could be analysed much easier.

3.1. 1-restricted cleaning process

In this section, we investigate 1-restricted cleaning process. The analysis is relatively simple, since $\mathcal{F}_1$ consists of trees only. It is known (see [11]) that for every tree T we have $b(T) = d_o(T)/2$, where $d_o(T)$ denotes the number of vertices of T of odd degree. (Note that $d_o(G)/2$ is a trivial lower bound that holds for every graph G, since each vertex of odd degree has to start or finish at least one brush path.) It turns out that $b_1(T)$ might be substantially larger.

Theorem 5. *For any tree T with $d_\ell(T)$ leaves, $b_1(T) = d_\ell(T) - 1$.*

Proof. Let T be any tree with $d_\ell(T)$ leaves. If T consists of an isolated vertex, then the claim trivially holds ($b_1(K_1) = 0$). Suppose then that T has at least 2 vertices (and so at least 2 leaves).

First, we will show that T can be cleaned with $d_\ell(T) - 1$ brushes. We put exactly one brush on all leaves but one (the leaf with no brush can be chosen arbitrarily). Since T has at least 2 leaves, at least one leaf contains a brush. We clean arbitrarily selected leaf with a brush and after that the desired property still holds; that is, all but one leaf contain at least one brush. We continue this process until we are left with an isolated vertex and the process can be easily finished.

Now, we will show that T cannot be cleaned with $d_\ell(T) - 2$ brushes. For a contradiction, suppose that it is possible. Let v_1 and v_2 be any two leaves that have no brush in the initial configuration (if more than 2 leaves have no brush, then the two leaves can be chosen arbitrarily). Note that when either vertex is cleaned (say, v_1), the only neighbour of v_1 is already cleaned, exactly one brush occupies v_1, and no brush is moved. Moreover, without loss of generality, we may assume that v_1 is cleaned before v_2.

Let us consider a graph G that is obtained from T by adding an edge v_1v_2; that is, $V(G) = V(T)$ and $E(G) = E(T) \cup \{v_1v_2\}$. It is clear that G can be cleaned under the same model—the same initial configuration and cleaning sequence can be used; when v_1 is cleaned it sends now exactly one brush to v_2 but this causes no problem. However, this gives us the desired contradiction, since G contains a cycle and so does *not* belong to $\mathcal{F}_1$; in other words, G cannot be cleaned by 1-restricted cleaning process. The lower bound of $d_\ell(T) - 1$ holds and the proof is finished. □

It follows immediately from Theorem 5 that for every tree T we have

$$\frac{b_1(T)}{b(T)} = \frac{d_\ell(T) - 1}{d_o(T)/2} \leq 2\frac{d_\ell(T) - 1}{d_\ell(T)} < 2, \tag{1}$$

so the closure of B_1 ($= A_1$) is contained in $[1, 2]$. We will construct a family of trees that can be used to get the ratio arbitrarily close to any real number $x \in [1, 2]$. This will prove the following result.

Theorem 6. *The closure of B_1 is $[1, 2]$.*

Proof. Consider first a complete binary tree Z_h of height $h \geq 1$ (there are 2^h leaves and $2^h - 1$ internal vertices). Since all vertices but the root are odd, we have

$$\frac{b_1(Z_h)}{b(Z_h)} = \frac{d_\ell(Z_h) - 1}{d_o(Z_h)/2} = \frac{2^h - 1}{(2^h + (2^h - 1) - 1)/2} = 1. \tag{2}$$

Hence $1 \in A_1$.

Let $x \in (1, 2]$. We will show that x belongs to the closure of A_1. Suppose that ℓ is a large integer of the form 2^h for some integer h. Put $i = \lfloor \ell(2 - x)/x \rfloor$. Since $(2 - x)/x \in [0, 1)$ and so $i \leq \ell - 2$ for ℓ large enough, we may assume that this is the case. Our goal is to construct a tree with ℓ leaves, $i + 1$ internal vertices that are all (but perhaps one vertex) of odd degree. We start with $\ell = 2^h$ isolated vertices. We create (one by one) 2^{h-1} complete binary trees of height one (that is, P_3's). After that we continue 'gluing' binary trees (again, one by one) creating larger complete binary trees. We stop once the number of internal vertices reaches i (recall that $i \leq \ell - 2$ so we are done before the complete binary tree with ℓ leaves is created). We finish the construction with adding a root that is adjacent to all binary trees created up to this point, and call this graph $T = T(x, h)$. It follows

$$\frac{b_1(T)}{b(T)} = \frac{d_\ell(T) - 1}{d_o(T)/2} = \frac{\ell - 1}{(\ell + i + a)/2},$$

where $a = 1$ if the root has odd degree and $a = 0$ otherwise. In either case, it is clear that the ratio tends to x as $h \to \infty$. Hence, x belongs to the closure of A_1 and the proof is complete. □

3.2. k-restricted cleaning process for trees with $k \geq 2$

In this section, we continue investigating trees but this time we focus on k-restricted cleaning process for $k \geq 2$. Since for every graph G we have $b_{k_1}(G) \geq b_{k_2}(G)$ whenever $1 \leq k_1 \leq k_2$, it follows immediately that for every tree T and $k \in \mathbb{N}$, we have

$$\frac{b_k(T)}{b(T)} \leq \frac{b_1(T)}{b(T)} < 2.$$

(See (1) to justify the second inequality.) Hence, the closure of B_k is contained in $[1, 2]$.

For $k = 1$ we were able to find an explicit formula for $b_1(T)$ and then to construct a family of trees that can be used to get the ratio arbitrarily close to any real number $x \in [1, 2]$. Investigating $b_k(T)$ for $k \geq 2$ seems to be more complicated and perhaps no simple formula can be found. However, we are going to use a simple non-constructive argument that will prove the following result. Alternatively, one could construct a family of trees for which both $b_k(T)$ and $b(T)$ can be calculated (or, at least, estimated) but a non-constructive argument might be of independent interest.

Theorem 7. *Let $k \geq 2$ be an integer. The closure of B_k is* $[1, 2]$.

Proof. First let us observe the following property. For every graph $G \in \mathcal{F}_k$ and any edge $uv \in E(G)$, we have $G - uv \in \mathcal{F}_k$ and

$$b_k(G - uv) \leq b_k(G) + 1. \tag{3}$$

($G - uv$ denotes a graph with an edge uv removed; that is, $V(G - uv) = V(G)$ and $E(G - uv) = E(G) \setminus \{uv\}$.) Indeed, suppose that a cleaning sequence α can be used to clean G with an initial configuration ω_0 of $b_k(G)$ brushes. Without loss of generality, we may assume that vertex v is cleaned before u. Note that the same cleaning sequence can be used to clean $G - uv$. The required condition for the number of brushes sent in each step is trivially satisfied. However, it might happen that a brush sent from v to u in the process for G (that is *not* sent in the process for $G - uv$) might be necessary to be able to clean u. Adding one additional brush to the initial configuration (to vertex u) solves this potential problem and so the claim holds.

Similarly, for every graph $G \in \mathcal{F}_k$ and any vertex $v \in V(G)$, we have $G_v \in \mathcal{F}_k$ and

$$b_k(G_v) \leq b_k(G) + 1. \tag{4}$$

(G_v denotes a graph with an edge attached to v; that is, $V(G_v) = V(G) \cup \{u\}$ and $E(G_v) = E(G) \cup \{uv\}$.) To see this one can add an additional brush to a new vertex u, clean this vertex first and use the cleaning sequence for G that yields $b_k(G)$ to finish the process. Clearly, the required condition is satisfied.

Finally, consider a complete binary tree Z_h of height $h \geq 1$ and a star S_h with $2^{h+1} - 2$ leaves (note that both trees have $2^{h+1} - 1$ vertices). We already know that

$$1 \leq \frac{b_k(Z_h)}{b(Z_h)} \leq \frac{b_1(Z_h)}{b(Z_h)} = 1 \tag{5}$$

(see (2)) and so $b_k(Z_h)/b(Z_h) = 1$. It is also clear that

$$\frac{b_k(S_h)}{b(S_h)} = \frac{d_\ell(S_h) - k}{d_o(S_h)/2} = \frac{d_\ell(S_h) - k}{d_\ell(S_h)/2} = 2 - \frac{2k}{d_\ell(S_h)} = 2 - \frac{k}{2^h - 1} \to 2,$$

as $h \to \infty$.

Now, we have all ingredients to finish the proof. We fix $h \in \mathbb{N}$ and consider the following graph process starting with $G_0 = Z_h$, binary tree rooted at vertex v. In every step $t \geq 1$, graph G_t is obtained from G_{t-1} by removing any leaf at distance at least 2 from v and adding an edge attached to v. Clearly, the number of vertices does not change during this process and the process terminates at time L with $G_L = S_h$, star with $2^{h+1} - 2$ leaves. Moreover, the number of leaves does not decrease during this process; as a result, $d_o(G_t) \geq d_\ell(G_t) \geq d_\ell(G_0) = 2^h$ for each $t \geq 1$. We concentrate on the sequence of numbers $(r_t)_{t=0}^{L}$, where $r_t = b_k(G_t)/b(G_t)$. As we already mentioned, $r_0 = 1$ and $r_L = 2 - k/(2^h - 1)$. It follows from (3) and (4) that $b_k(G_{t+1}) \leq b_k(G_t) + 2$. Moreover, $b(G_{t+1}) = d_o(G_{t+1})/2 \geq (d_o(G_t) - 1)/2 = b(G_t) - 1/2$. Hence,

$$\begin{aligned} r_{t+1} = \frac{b_k(G_{t+1})}{b(G_{t+1})} &\leq \frac{b_k(G_t) + 2}{b(G_t)(1 - \frac{1}{2b(G_t)})} = \frac{b_k(G_t) + 2}{b(G_t)}\left(1 + \frac{1}{2b(G_t)} + \left(\frac{1}{2b(G_t)}\right)^2 + \dots\right) \\ &\leq \frac{b_k(G_t) + 2}{b(G_t)}\left(1 + \frac{1}{b(G_t)}\right) = \frac{b_k(G_t)}{b(G_t)}\left(1 + \frac{1}{b(G_t)}\right) + \frac{2}{b(G_t)}\left(1 + \frac{1}{b(G_t)}\right) \\ &< \frac{b_k(G_t)}{b(G_t)} + \frac{2}{b(G_t)} + \frac{4}{b(G_t)} = r_t + \frac{12}{d_o(G_t)} \leq r_t + 12 \cdot 2^{-h} \end{aligned}$$

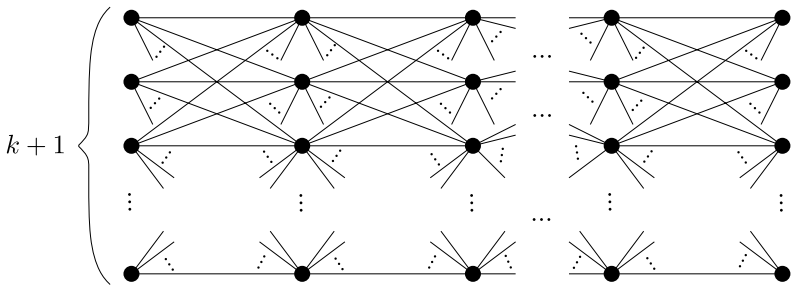

Fig. 2. The graph G_n with $b_k(G_n)/b(G_n) \geq 2(n-2)/(k+1)$.

(the second inequality follows from the fact that $1/(1-x) = 1 + x + x^2 + \ldots \leq 1 + 2x$ for $x \in [0, 1/2]$; the second last inequality follows from the fact that $b_k(G_t)/b(G_t) < 2$ and, trivially, we have $1 + 1/b(G_t) \leq 2$). Hence, for every real number $x \in [1, 2]$ there is an integer $0 \leq t \leq L$ such that $|x - r_t| \leq \delta(h) := \max\{k/(2^h - 1), 12 \cdot 2^{-h}\}$. Since $\delta(h) \to 0$ as $h \to \infty$, we are guaranteed to have a graph with the ratio arbitrarily close to x. The proof is complete. □

3.3. k-restricted cleaning process for graphs with $k \geq 2$

In this section, we continue investigating k-restricted cleaning process for $k \geq 2$ but we take into account all graphs that can be cleaned (that is, graphs from $\mathcal{F}_k$). It turns out that graphs with cycles can be used to obtain the ratio of $b_k(G)/b(G)$ larger than 2. In fact, the ratio can be arbitrarily close to any real number at least 1.

We will need one more definition. The **lexicographic product** $G \cdot H$ of graphs G and H is a graph such that the vertex set of $G \cdot H$ is the cartesian product $V(G) \times V(H)$; and any two vertices (u, v) and (x, y) are adjacent in $G \cdot H$ if and only if either u is adjacent with x in G or $u = x$ and v is adjacent with y in H. Note that the lexicographic product is in general noncommutative, that is, it might be that $G \cdot H \neq H \cdot G$.

Theorem 8. *Let $k \geq 2$ be an integer. The closure of A_k is $[1, \infty)$.*

Proof. We start the proof by showing that the ratio of $b_k(G)/b(G)$ can be arbitrarily large. Consider a graph $P_n \cdot \overline{K_{k+1}}$ (see Fig. 2). A graph G_n is obtained from $P_n \cdot \overline{K_{k+1}}$ after subdividing each edge. Graph G_n has $2(k+1)$ vertices of degree $(k+1)$, $(n-2)(k+1)$ vertices of degree $2(k+1)$, and $(n-1)(k+1)^2$ vertices of degree 2.

It is clear that $b(G_n) \leq (k+1)^2$, since one can put $(k+1)$ brushes on vertices of degree $(k+1)$ that lie on one 'side' of the graph G_n, and clean G_n layer by layer, along P_n. (Let us note that it should be straightforward to find the exact value of $b(G_n)$ but it is not needed for our argument.) On the other hand, when vertex of degree $2(k+1)$ is cleaned in a k-restricted process, at most k brushes are sent and so at least 2 brushes get stuck on this vertex. (Note that a lot of brushes start on vertices of degree 2 due to the restriction on the number of brushes sent at each step.) Hence, no matter how we clean the graph, at least $2(n-2)(k+1)$ brushes get stuck on these vertices. It follows that $b_k(G_n) \geq 2(n-2)(k+1)$ and so $b_k(G_n)/b(G_n) \geq 2(n-2)/(k+1)$ which can be made arbitrarily large by taking n large enough. In fact, it is straightforward to show that it is possible to clean the graph so that exactly 2 brushes are left on vertices of degree $2(k+1)$ and no brush is stuck on vertex of degree 2. By counting the number of brushes after the process we get the following upper bound:

$$b_k(G) \leq 2(n-2)(k+1) + 2(k+1)^2.$$

(Again, let us stress the fact that it should be possible to determine $b_k(G)$ precisely but it would be tedious and is not needed for our purpose.)

Now, we are ready to finish the proof. We already showed that for a complete binary tree Z_h of height $h \geq 1$, we have $b_k(Z_h) = 2^h - 1$ (see (2) and (5)). Hence $1 \in A_k$.

Let $x \in (1, \infty)$. Our goal is to show that x belongs to the closure of A_k. For a positive integer h, let us take

$$n = n(h) = \left\lceil \frac{x-1}{2(k+1)} 2^h \right\rceil \geq 1,$$

and consider graph G_n constructed above. Let $Z_h + G_n$ denote a graph obtained from Z_h and G_n after connecting them with an edge (arbitrarily). It is clear that

$$b(Z_h + G_n) = b(Z_h) + b(G_n) + O(1) = 2^h + O(1),$$

where the constant hidden in $O(\cdot)$ notation is a function of k but does not depend on n. Similarly,

$$b_k(Z_h + G_n) = b_k(Z_h) + b_k(G_n) + O(1) = 2^h + 2n(k+1) + O(1) = x2^h + O(1).$$

Hence,

$$\frac{b_k(Z_h + G_n)}{b(Z_h + G_n)} = \frac{x2^h + O(1)}{2^h + O(1)} \to x,$$

as $h \to \infty$. This finishes the proof of this theorem. □

4. An efficient algorithm for trees

In this section we describe a polynomial-time algorithm that, given any tree T and any $k \in \mathbb{N}$, computes the k-brush number of T. In the following we consider T to be rooted at an arbitrarily selected leaf. Then, for any vertex v of T define $T[v]$ to be the subtree of T rooted at v, i.e., the subtree of T induced by v and all its descendants in T. Define $T^+[v]$ to be a tree obtained from $T[v]$ by adding one additional vertex that is the parent of v; the additional vertex of $T^+[v]$ is denoted by r_v. Note that $T^+[v]$ is a subtree of T for each vertex v of T except for the root. Finally, for any $X \subseteq V(T)$, we write for brevity $\omega_0(X) = \sum_{v \in X} \omega_0(v)$.

We start with an informal description of our method. Our algorithm uses a dynamic programming approach. More precisely, a bottom-up tree processing allows us to compute a label for each vertex of T, except for the root. The label of a vertex v, formally defined below, consists of two integers giving the minimum number of brushes needed to clean $T^+[v]$. We use the two integers to distinguish 'directions' of cleaning the edge $r_v v$. Once all labels are computed, the label of the only child of the root of T contains our final answer.

We define a **label** of a vertex v of T as a pair of integers (l_1, l_2), where l_1 (respectively l_2) is the minimum i such that there exists a k-restricted cleaning process $\{(\omega_t, D_t)\}_{t=0}^{L}$ of T with $\omega_0(V(T^+[v])) = i$ and having the property that the edge $r_v v$ is cleaned by a brush that moves from r_v to v (from v to r_v, respectively). Note that the label is well defined because for each $k \geq 1$ and for any pair of adjacent vertices u and u' there exists a k-restricted process (not necessarily an optimal one) that cleans u prior to u'.

Lemma 9. *Let T be a rooted tree and suppose that $u \in V(T)$ is not the root of T. Let $\mathfrak{P} = \mathfrak{P}(T, k, \omega_0)$ be a k-restricted cleaning process of T and let (l_1, l_2) be the label of u in T. Then,*

(i) *if u is cleaned by $\mathfrak{P}$ prior to its parent, then $\omega_0(V(T[u])) \geq l_2$,*
(ii) *if u is cleaned by $\mathfrak{P}$ after its parent, then $\omega_0(V(T[u])) \geq l_1 - 1$.*

Proof. Follows directly from the definition of the label. □

Procedure CL (*Compute Label*) given below computes a label of a given vertex v of T, provided that the integer k and the labels of the children $v_1, \ldots, v_p$ of v are given.

Procedure CL (*Compute Label*).

Input: A rooted tree T, a vertex v of T, an integer k, and the labels $(l_1^1, l_2^1), \ldots, (l_1^p, l_2^p)$ of the children $v_1, \ldots, v_p$ of v in T.
Output: The label of v.
1: Find a permutation π of $\{1, \ldots, p\}$ such that $l_2^{\pi(i)} - l_1^{\pi(i)} \geq l_2^{\pi(i+1)} - l_1^{\pi(i+1)}$ for each $i \in \{1, \ldots, p-1\}$.
2: Let $x_1 := +\infty$ and $x_2 := +\infty$.
3: **for each** $i \in \{0, \ldots, p\}$ **do**
4: **for each** $j \in \{1, 2\}$ **do**
5: $z := \max\{i - p + 2 - j, -i + j - 1\} + \sum_{t=1}^{i} l_2^{\pi(t)} + \sum_{t=i+1}^{p} l_1^{\pi(t)}$
6: **if** $z \leq x_j$ **and** $p - i + j - 1 \leq k$ **then**
7: $x_j := z$
return (x_1, x_2)

We now give some intuitions on Procedure CL and then we formally prove its correctness. The permutation π in Line 1 of Procedure CL provides a non-decreasing order of the children of v with respect to the value of $l_2^i - l_1^i$. The permutation π has the following property: if $\pi(i) < \pi(i')$ for some $i, i' \in \{1, \ldots, p\}$ and in some k-restricted process of $T^+[v]$ the vertex v is cleaned after $v_{i'}$ but before v_i, then there exists another k-restricted cleaning process $\mathfrak{P}(T, k, \omega_0)$ of $T^+[v]$ that does not use more brushes than the former one and in which v is cleaned after v_i and before $v_{i'}$. The values of x_1 and x_2, initially set to $+\infty$ in line 2, give the label of v at the end of the computation. In lines 5, 6 and 7 of Procedure CL, for a particular choice of $i \in \{1, \ldots, p\}$ and $j \in \{1, 2\}$, the algorithm tests the existence of a k-restricted cleaning process of $T^+[v]$ with several constraints:

- the value of j indicates the 'direction' of cleaning the edge $r_v v$,
- i equals the number of children of v cleaned prior to v,
- the value of z computed in line 5 equals the number of brushes needed to clean $T^+[v]$,

- the condition in line 6 verifies whether the cleaning process with the above restrictions is k-restricted and whether the value of z improves the best bound found so far.

Note that, if i children of v need to be selected as the ones cleaned prior to v, then we pick them greedily according to the permutation π.

In Lemma 10 we formally prove the properties that the value of z has. In Lemma 11, we prove the correctness of Procedure CL. Then, we state our main algorithm that computes the k-brush number of a given tree. Finally, Theorem 12 summarizes the main result of this section.

Lemma 10. *Suppose that z is computed in line 5 of Procedure CL for a particular choice of $i \in \{0, \ldots, p\}$ and $j \in \{1, 2\}$. If $p - i + j - 1 \leq k$, then there exists a k-restricted cleaning process $\mathfrak{P}(T^+[v], k, \omega_0)$ of $T^+[v]$ that satisfies:*

(a) $\omega_0(V(T^+[v])) = z$,
(b) *if $j = 1$ (respectively $j = 2$) then the edge $r_v v$ is cleaned by a brush that moves from r_v to v (from v to r_v, respectively),*
(c) *there exist exactly i children of v that are cleaned prior to v.*

Proof. Let T, $v \in V(T)$, an integer k and the labels $(l_1^1, l_2^1), \ldots, (l_1^p, l_2^p)$ of the children $v_1, \ldots, v_p$ of v in T be the input to Procedure CL. Let π be the permutation selected in line 1.

We first define a cleaning process $\mathfrak{P} = \mathfrak{P}(T^+[v], k, \omega_0)$ of $T^+[v]$ and then we prove that it is indeed a cleaning process that is k-restricted and satisfies conditions (a), (b) and (c). We define the process by first stating the initial brush placement ω_0. Let

$$\omega_0(r_v) = 2 - j \quad \text{and} \quad \omega_0(v) = \max\{0, p - 2i + 2j - 3\}.$$

The placement on the remaining vertices of $T^+[v]$ is 'derived' from the definition of the label. In particular, for each $t \in \{1, \ldots, i\}$, there exists a k-restricted cleaning process $\mathfrak{P}_{\pi(t)}(T^+[v_{\pi(t)}], k, \omega_0^{\pi(t)})$ of $T^+[v_{\pi(t)}]$ that uses $l_2^{\pi(t)}$ brushes and cleans $v_{\pi(t)}$ prior to its parent; note that such a process places no brushes on the root of $T^+[v_{\pi(t)}]$. Similarly, for each $t \in \{i + 1, \ldots, p\}$, there exists a k-restricted cleaning process $\mathfrak{P}_{\pi(t)}(T^+[v_{\pi(t)}], k, \omega_0^{\pi(t)})$ of $T^+[v_{\pi(t)}]$ that uses $l_2^{\pi(t)}$ brushes and cleans $v_{\pi(t)}$ after its parent; note that such a process places one brush on the root of $T^+[v_{\pi(t)}]$. Let for each $t \in \{1, \ldots, p\}$ and for each $u \in V(T[v_t])$, $\omega_0(u) = \omega_0^t(u)$.

The cleaning process $\mathfrak{P}$ cleans $T^+[v]$ as follows:

1. if $j = 1$, then the root is fired, i.e., the brush at r_v is moved from r_v to v,
2. for each $t \in \{1, \ldots, i\}$ (in any order) the subtree $T^+[v_{\pi(t)}]$ is cleaned by cleaning its vertices in the same order as in $\mathfrak{P}_{\pi(t)}(T[v_{\pi(t)}], k, \omega_0)$,
3. the vertex v is cleaned, and
4. for each $t \in \{i + 1, \ldots, p\}$ (in any order) the subtree $T[v_{\pi(t)}]$ is cleaned by cleaning its vertices in the same order as in $\mathfrak{P}_{\pi(t)}(T[v_{\pi(t)}], k, \omega_0)$.

We now prove that $\mathfrak{P}$ is a k-restricted cleaning process of $T^+[v]$. To that end it is enough to argue that firing v results in cleaning all dirty edges incident to v and that the number of those dirty edges is at most k. We have that

$$a_1 = i + 2 - j$$

equals the number of brushes that 'arrive' at v due to the vertices cleaned prior to v. The number of edges to be cleaned by brushes moving away from v is

$$a_2 = p - i + j - 1.$$

Since $\omega_0(v) = \max\{0, a_2 - a_1\}$, we obtain that there are enough brushes present at v when it is cleaned by $\mathfrak{P}$. Note that the condition $p - i + j - 1 \leq k$ is equivalent to $a_2 \leq k$. Thus, $\mathfrak{P}$ is a valid k-restricted cleaning process of $T^+[v]$.

Clearly, $\mathfrak{P}$ satisfies (b) and (c). It remains to prove (a). By construction of $\mathfrak{P}$,

$$\sum_{t=1}^{i} \omega_0\big(V\big(T[v_{\pi(t)}]\big)\big) = \sum_{t=1}^{i} l_2^{\pi(t)}.$$

Moreover, the value of $l_1^{\pi(t)}$ for $t > i$ already accommodates pre-placing a brush at v that then moves from v to $v_{\pi(t)}$, which means that the sum $\sum_{t=i+1}^{p} l_1^{\pi(t)}$ accommodates pre-placing $p - i$ brushes at v in total, i.e.,

$$\sum_{t=i+1}^{p} \omega_0\big(V\big(T[v_{\pi(t)}]\big)\big) = -(p - i) + \sum_{t=i+1}^{p} l_1^{\pi(t)}.$$

Thus, according to the definition of z,

$$\begin{aligned}\omega_0\big(V\big(T^+[v]\big)\big) &= \omega_0(v) + \omega_0(r_v) - (p-i) + \sum_{t=1}^{i} l_2^{\pi(t)} + \sum_{t=i+1}^{p} l_1^{\pi(t)} \\ &= \max\{i-p+2-j, -i+j-1\} + \sum_{t=1}^{i} l_2^{\pi(t)} + \sum_{t=i+1}^{p} l_1^{\pi(t)} \\ &= z.\end{aligned}$$

This proves (a). □

Lemma 11. *Given a tree T rooted at its leaf, an integer k, a vertex v of T and the labels of the children of v, Procedure* CL *returns the label of v.*

Proof. Let (x_1, x_2) be the pair of integers returned by Procedure CL. Fix the integer $j \in \{1,2\}$ arbitrarily. We prove that whenever $\mathfrak{P}_j = \mathfrak{P}_j(T^+[v], k, \omega_0)$ is a k-restricted process of $T^+[v]$ in which the root of $T^+[v]$ is cleaned prior to v if $j=1$ and after v if $j=2$, then

$$x_j \le \omega_0\big(V\big(T^+[v]\big)\big). \tag{6}$$

Let $i \in \{0,\dots,p\}$ be the number of children of v cleaned prior to v in $\mathfrak{P}_j$. Suppose without loss of generality that $v_1,\dots,v_i$ are the children of v cleaned prior to v in $\mathfrak{P}_j$. Note that $\omega_0(r_v) \ge 2-j$ and $\omega_0(v) \ge \max\{0, p-2i+(2j-3)\}$. Thus, the number of brushes used by $\mathfrak{P}_j$ satisfies

$$\begin{aligned}\omega_0\big(V\big(T^+[v]\big)\big) &\ge 2-j+\max\big\{0, p-2i+(2j-3)\big\} + \sum_{t=1}^{p} \omega_0\big(V\big(T[v_t]\big)\big) \\ &= \max\{2-j, p-2i+j-1\} - (p-i) + \sum_{t=1}^{i} \omega_0\big(V\big(T[v_t]\big)\big) + \sum_{t=i+1}^{p} \big(1+\omega_0\big(V\big(T[v_t]\big)\big)\big) \\ &= \max\{i-p+2-j, -i+j-1\} + \sum_{t=1}^{i} \omega_0\big(V\big(T[v_t]\big)\big) + \sum_{t=i+1}^{p} \big(1+\omega_0\big(V\big(T[v_t]\big)\big)\big).\end{aligned} \tag{7}$$

Note that the above inequality is strict only in case when $\mathfrak{P}_j$ does not use the minimum number of brushes. We prove that the value of z computed in line 5 of Procedure CL for the given values of i and j satisfies

$$z \le \omega_0\big(V\big(T^+[v]\big)\big). \tag{8}$$

Let π be the permutation selected in line 1 of Procedure CL. Now we define four sets $A, \bar{A}, B, \bar{B}$ such that $A \cup \bar{A} = \{1,\dots,i\}$ and $B \cup \bar{B} = \{i+1,\dots,p\}$. Let

$$A = \big\{t \in \{1,\dots,i\} \mid \pi(t) \le i\big\}, \qquad \bar{A} = \big\{t \in \{1,\dots,i\} \mid \pi(t) > i\big\},$$

$$B = \big\{t \in \{i+1,\dots,p\} \mid \pi(t) \ge i+1\big\}, \qquad \bar{B} = \big\{t \in \{i+1,\dots,p\} \mid \pi(t) < i+1\big\}.$$

Now we bound $\sum_{t=1}^{i} l_2^{\pi(t)} + \sum_{t=i+1}^{p} l_1^{\pi(t)}$. By Lemma 9(i),

$$\sum_{t\in A} l_2^{\pi(t)} \le \sum_{t\in A} \omega_0\big(V\big(T[v_{\pi(t)}]\big)\big), \tag{9}$$

and by Lemma 9(ii),

$$\sum_{t\in \bar{A}} l_2^{\pi(t)} = \sum_{t\in \bar{A}} \big(l_2^{\pi(t)} - l_1^{\pi(t)}\big) + \sum_{t\in \bar{A}} l_1^{\pi(t)} \le \sum_{t\in \bar{A}} \big(l_2^{\pi(t)} - l_1^{\pi(t)}\big) + \sum_{t\in \bar{A}} \big(1+\omega_0\big(V\big(T[v_{\pi(t)}]\big)\big)\big). \tag{10}$$

Using Lemma 9(ii) again we get

$$\sum_{t\in B} l_1^{\pi(t)} \le \sum_{t\in B} \big(1+\omega_0\big(V\big(T[v_{\pi(t)}]\big)\big)\big), \tag{11}$$

and by Lemma 9(i),

$$\sum_{t\in\bar{B}} l_1^{\pi(t)} = \sum_{t\in\bar{B}} \left(l_1^{\pi(t)} - l_2^{\pi(t)}\right) + \sum_{t\in\bar{B}} l_2^{\pi(t)} \leq \sum_{t\in\bar{B}} \left(l_1^{\pi(t)} - l_2^{\pi(t)}\right) + \sum_{t\in\bar{B}} \omega_0\left(V\left(T[v_{\pi(t)}]\right)\right). \tag{12}$$

By (9), (10), (11), and (12),

$$\sum_{t=1}^{i} l_2^{\pi(t)} + \sum_{t=i+1}^{p} l_1^{\pi(t)} \leq |\bar{A}| + |B| + \sum_{t=1}^{p} \omega_0\left(V\left(T[v_{\pi(t)}]\right)\right) + \sum_{t\in\bar{A}} \left(l_2^{\pi(t)} - l_1^{\pi(t)}\right) - \sum_{t\in\bar{B}} \left(l_2^{\pi(t)} - l_1^{\pi(t)}\right).$$

We have that $|\bar{A}| + |B| \leq p - i$ because π is a permutation. Also, $|\bar{A}| = |\bar{B}|$ and by the definitions of $\bar{A}$ and $\bar{B}$, $\pi(t) > i \geq \pi(t')$ for each $t \in \bar{A}$ and $t' \in \bar{B}$. Therefore, due to the choice of π in line 1 of Procedure CL,

$$\sum_{t\in\bar{A}} \left(l_2^{\pi(t)} - l_1^{\pi(t)}\right) \leq \sum_{t\in\bar{B}} \left(l_2^{\pi(t)} - l_1^{\pi(t)}\right).$$

Thus,

$$\sum_{t=1}^{i} l_2^{\pi(t)} + \sum_{t=i+1}^{p} l_1^{\pi(t)} \leq p - i + \sum_{t=1}^{p} \omega_0\left(V\left(T[v_{\pi(t)}]\right)\right).$$

Hence, by (7) and the definition of z in line 5 of Procedure CL we obtain

$$z \leq \max\{i - p + 2 - j, -i + j - 1\} + p - i + \sum_{t=1}^{p} \omega_0\left(V\left(T[v_{\pi(t)}]\right)\right) \leq \omega_0\left(V\left(T^+[v]\right)\right).$$

This proves (8). Since $\mathfrak{P}_j$ is k-restricted, the condition in line 6 of Procedure CL is satisfied. This implies that the value of z is assigned to x_j in line 7 and hence (6) holds. Thus, Lemma 10 gives us that (x_1, x_2) returned by Procedure CL is the label of v. □

Having a procedure that computes the label of a vertex provided that the labels of its children are given, we can formulate Procedure RC (*Restricted Cleaning*).

Procedure RC (*Restricted Cleaning*).

Input: A tree T and an integer k.
Output: The value of $b_k(T)$.
1: Root T at an arbitrary leaf u.
2: Find an ordering $v_1, \ldots, v_n$ of the vertices of T such that each $v \in V(T) \setminus \{u\}$ is ordered prior to its parent.
3: **for** $i := 1$ **to** $n - 1$ **do**
4: Call CL to compute the label (x_1^i, x_2^i) of v_i.
return $\min\{x_1^{n-1}, x_2^{n-1}\}$

Theorem 12. *Given a tree T and an integer k, Procedure* RC *computes in linear time the k-brush number of T.*

Proof. By Lemma 11 and by the ordering of vertices of T chosen in line 2 of Procedure RC, we obtain that (x_1^i, x_2^i) computed in line 4 is the label of v_i. Note that T is rooted at a leaf (see line 1 of Procedure RC). Thus, by the definition of the label, $b_k(T) = \min\{x_1^{n-1}, x_2^{n-1}\}$.

Now we prove that the running time of Procedure RC is $O(n)$. Note that it is enough to argue that the complexity of Procedure CL is linear in the degree $p + 1$ of the input vertex v.

It remains to analyse the running time of Procedure CL and we focus on the running time of the sorting performed in line 1 as it is straightforward to see that the remaining instructions can be implemented in time $O(p)$. Observe that if (l_1, l_2) is the label of a vertex u of T, then $l_2^i - l_1^i \in \{-1, 0, 1\}$. Indeed, if $T^+[u]$ can be cleaned by a k-restricted process that cleans u prior to its root, then with one extra brush placed at the root, $T^+[u]$ can be cleaned by a k-restricted process that cleans u after the root. Therefore, the sorting in line 6 of Procedure CL can be implemented in time $O(p)$, which completes the proof. □

5. Open problems

In Theorem 5, we derive an explicit formula for the 1-restricted brush number of any tree $T \in \mathcal{F}_1$. Unfortunately, we were less successful for $b_k(T)$ for $k \geq 2$ and we leave it as an open problem. Similarly, not much is known for $b_k(G)$ for $k \geq 2$, $G \in \mathcal{F}_k$. It is not reasonable to expect an explicit formula here but it would be interesting to find some general upper and lower bounds, and to determine the k-restricted brush number for some known classes of graphs.

The problem of determining whether $b(G) \leq l$ for the given G and l is NP-complete for planar graphs of maximum degree 4 and for 5-regular graphs [5,8]. On the other hand, the problem can be solved in polynomial time for graphs of maximum degree 3 [2]. We can immediately conclude that the problem of determining whether $b_k(G) \leq l$ for the given G and l is NP-complete for planar graphs with maximum degree 4 when $k \geq 4$ (with fixed k, i.e., k is part of the problem, not part of the input) and for 5-regular graphs when $k \geq 5$. An interesting open problem lies in determining more accurate border between computationally easy and hard instances for the k-brush number. Also, of interest are non-trivial classes of graphs for which the complexity of the two problems differs.

References

[1] N. Alon, P. Prałat, N. Wormald, Cleaning regular graphs with brushes, SIAM J. Discrete Math. 23 (2008) 233–250.
[2] T. Biedl, T. Chan, Y. Ganjali, M. Hajiaghayi, D. Wood, Balanced vertex-orderings of graphs, Discrete Appl. Math. 148 (2005) 27–48.
[3] A. Bjorner, L. Lovasz, P.W. Shor, Chip-firing games on graphs, European J. Combin. 12 (1991) 283–291.
[4] D. Bryant, N. Francetic, P. Gordinowicz, D. Pike, P. Prałat, Brushing without capacity restrictions, Discrete Appl. Math. 170 (2014) 33–45.
[5] S. Gaspers, M.-E. Messinger, R.J. Nowakowski, P. Prałat, Clean the graph before you draw it!, Inform. Process. Lett. 109 (2009) 463–467.
[6] S. Gaspers, M.-E. Messinger, R.J. Nowakowski, P. Prałat, Parallel cleaning of a network with brushes, Discrete Appl. Math. 158 (2009) 467–478.
[7] P. Gordinowicz, R. Nowakowski, P. Prałat, POLISH—let us play the cleaning game, Theoret. Comput. Sci. 463 (2012) 123–132.
[8] J. Kára, K. Kratochvíl, D. Wood, On the complexity of the balanced vertex ordering problem, Discrete Math. Theor. Comput. Sci. 9 (2007) 193–202.
[9] D. Lokshtanov, N. Misra, S. Saurabh, Imbalance is fixed parameter tractable, Inform. Process. Lett. 113 (2013) 714–718.
[10] C. Merino, The chip-firing game, Discrete Math. 302 (2005) 188–210.
[11] M.-E. Messinger, R.J. Nowakowski, P. Prałat, Cleaning a network with brushes, Theoret. Comput. Sci. 399 (2008) 191–205.
[12] M.-E. Messinger, R.J. Nowakowski, P. Prałat, Cleaning with Brooms, Graphs Combin. 27 (2011) 251–267.
[13] M.-E. Messinger, R.J. Nowakowski, P. Prałat, Elimination schemes and lattices, Discrete Math. 328 (2014) 63–70.
[14] P. Prałat, Cleaning random d-regular graphs with Brooms, Graphs Combin. 27 (2011) 567–584.
[15] P. Prałat, Cleaning random graphs with brushes, Australas. J. Combin. 43 (2009) 237–251.
[16] D.B. West, Introduction to Graph Theory, 2nd edition, Prentice Hall, 2001.