Discrete Applied Mathematics 207 (2016) 1–14

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Game brush number

William B. Kinnersley [a], Paweł Prałat [b,*]

[a] Department of Mathematics, University of Rhode Island, Kingston, RI, 02881, USA

[b] Department of Mathematics, Ryerson University, Toronto, ON, Canada, M5B 2K3

ARTICLE INFO

Article history:
Received 25 June 2015
Received in revised form 26 January 2016
Accepted 8 February 2016
Available online 16 March 2016

Keywords:
Brush number
Cleaning process
Random graphs

ABSTRACT

We study a two-person game based on the well-studied brushing process on graphs. Players Min and Max alternately place brushes on the vertices of a graph. When a vertex accumulates at least as many brushes as its degree, it sends one brush to each neighbor and is removed from the graph; this may in turn induce the removal of other vertices. The game ends once all vertices have been removed. Min seeks to minimize the number of brushes played during the game, while Max seeks to maximize it. When both players play optimally, the length of the game is the *game brush number* of the graph G, denoted $b_g(G)$.

By considering strategies for both players and modeling the evolution of the game with differential equations, we provide an asymptotic value for the game brush number of the complete graph; namely, we show that $b_g(K_n) = (1 + o(1))n^2/e$. Using a fractional version of the game, we couple the game brush numbers of complete graphs and the binomial random graph $\mathcal{G}(n, p)$. It is shown that for $pn \gg \ln n$ asymptotically almost surely $b_g(\mathcal{G}(n, p)) = (1 + o(1))pb_g(K_n) = (1 + o(1))pn^2/e$. Finally, we study the relationship between the game brush number and the (original) brush number.

1. Introduction

Imagine a network of pipes that must be periodically cleaned of a regenerating contaminant, say algae. In *cleaning* such a network, there is an initial configuration of brushes on vertices, and 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 purposes in this paper, we may think about clean vertices as if they were removed from the graph.) The goal is to clean all vertices (and hence also all edges) of the graph using as few brushes as possible. The minimum number of brushes needed to clean a graph G is the *brush number* of G, denoted $b(G)$.

Fig. 1 illustrates the cleaning process for a graph G where there are initially 2 brushes at vertex a. The solid edges indicate dirty edges while the dotted edges indicate clean edges. For example, the process starts with vertex a being cleaned, sending a brush to each of vertices b and c.

This model, which was introduced in [15], is tightly connected to the concept of *minimum total imbalance* of a graph, which is used in graph drawing theory. The cleaning process has been well studied, especially on random graphs [1,17]. (See also [9,13] for algorithmic aspects, [16,18] for a related model of cleaning with brooms, [6] for a variant with no edge capacity restrictions, [4] for a variant in which vertices can send out no more than k brushes, and [11] for a combinatorial

* Corresponding author.
E-mail addresses: billk@mail.uri.edu (W.B. Kinnersley), pralat@ryerson.ca (P. Prałat).

http://dx.doi.org/10.1016/j.dam.2016.02.011
0166-218X/

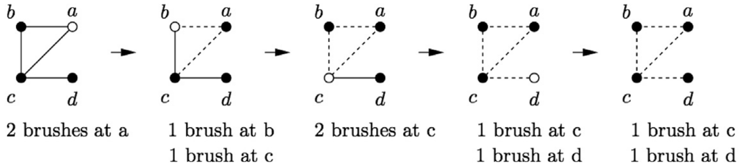

Fig. 1. An example of the cleaning process for graph G.

game.) Owing to inspiration from chip-firing processes [2,14], brushes disperse from an individual vertex 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 [10] for more details. In contrast, *sequential cleaning models* mandate that vertices get cleaned one at a time. The variant considered in [15] and the one we consider in this paper are sequential in nature.

The *brushing game* that we introduce in this paper is a two-player game played on a graph G. Initially, every vertex and edge is dirty and there are no brushes on any vertices. The players, Max and Min, alternate turns; on each turn, a player adds one brush to a vertex of his or her choosing. When a vertex accumulates at least as many brushes as it has dirty neighbors, it *fires*, sending one brush to each dirty neighbor. All edges incident to this vertex become clean, and the vertex itself becomes clean. This may in turn make other vertices ready to fire, so the process continues until we obtain a stable configuration. (It is known that the sequence in which vertices fire does not affect the distribution of brushes on dirty vertices—see below for more details.) The game ends when all vertices (and so all edges as well) are clean. Max aims to maximize the number of brushes played before this point, while Min aims to minimize it. When Min starts and both players play optimally, the length of the game on G is the *game brush number* of G, denoted $b_g(G)$; we use $\widehat{b_g}(G)$ to denote the variant of the game in which Max plays first.

The game brush number follows in the same spirit as the game matching number [7], game chromatic number [8], game domination number [5], toppling number [3], etc., in which players with conflicting objectives together make choices that produce a feasible solution to some optimization problem. The general area can be called *competitive optimization*. Competitive optimization processes can also be viewed as on-line problems in which one wants to build a solution to some optimization problem, despite not having complete control over its construction. In this context, the game models adversarial analysis of an algorithm for solving the problem: one player represents the algorithm itself, the other player represents the hypothetical adversary, and the outcome of the game represents the algorithm's worst-case performance.

Throughout this paper, we consider only finite, simple, undirected graphs. For background on graph theory, the reader is directed to [19].

1.1. Main results

Let us start with the following convenient bound proved in Section 2.

Theorem 1. *Always* $\left|b_g(G) - \widehat{b_g}(G)\right| \le 1$.

Theorem 1 is best possible. For the star $K_{1,3k-1}$, we have $b_g(K_{1,3k-1}) = 2k - 1$ and $\widehat{b_g}(K_{1,3k-1}) = 2k$; in particular, we have $b_g(P_3) = 1$ and $\widehat{b_g}(P_3) = 2$. On the other hand, for $n \ge 3$ we have $b_g(C_n) = 3$ and $\widehat{b_g}(C_n) = 2$. (We remark without proof that the graph G_k obtained by taking $5k$ triangles and identifying a single vertex of each has $b_g(G_k) = 8k + 1$ and $\widehat{b_g}(G_k) = 8k$; this yields another family – with unbounded brush number – witnessing sharpness of the bound. We leave the details to the reader.)

For the remainder of the paper, we will be concerned primarily with the asymptotics of b_g over various families of graphs. Hence the difference between b_g and $\widehat{b_g}$ is unimportant; we use whichever is most convenient (but prefer b_g in general).

Because the game produces a feasible solution to the original problem, the value of the game parameter is bounded by that of the original optimization parameter.

Proposition 2. *Always* $b(G) \le b_g(G) \le 2b(G) - 1$ *and* $b(G) \le \widehat{b_g}(G) \le 2b(G)$.

In Section 2, we show that these bounds, though elementary, are best possible in a strong sense:

Theorem 3. *For every rational number* r *in* $[1, 2)$, *there exists a graph* G *such that*

$$\frac{b_g(G)}{b(G)} = r.$$

We next turn our attention to complete graphs. Our next main result (proved in Section 3) provides the asymptotic behavior of the game brush number for K_n.

Theorem 4. $b_g(K_n) = (1 + o(1))n^2/e$