Chipping away at the edges: how long does it take?

O-YEAT CHAN and PAWEŁ PRAŁAT

May 22, 2012

Abstract. We introduce the single-node traffic flow process, which is related to both the chip-firing game and the edge searching process. Initially, real-valued weights (instead of chips) are placed on some vertices, and all the edges have zero weight. When a vertex is "fired", the whole content accumulated in this vertex is sent uniformly to all its neighbours, and each edge increases its weight by the amount that is sent through this edge. We would like to discover the shortest firing sequence such that the total amount of traffic that has passed through each edge is at least some fixed value. A complete characterization for complete graphs is presented as well as discussion of other classes of graphs.

2000 AMS Classification Numbers: 05C50, 05C57, 05C78, 05C85
Keywords: Single-node traffic flow process, graph searching, chip-firing game.

1. Introduction

Given a network where traffic flows between nodes, it is often the case that the life of a connection between the nodes is dependent on the total amount of traffic that has passed through it. The question naturally arises, "what is the minimum amount of time that must pass before all the connections have worn out?" A related problem is if the traffic must pay a toll to travel from one node to its neighbour, how long will it take for each connection to recover its costs?

We may model this process as a discrete process where a certain amount of traffic flows between the nodes at each step. In the present treatment we consider the case where the total amount of moveable traffic in the network is fixed, and all the moveable traffic in some node moves evenly to all the neighbouring nodes in each step.

The process we study here is similar to the cleaning process [14, 9, 10, 2, 17, 15, 16, 11], and is related to the chip-firing game (see, for example, [3, 5]) and edge searching (see, for example, [1, 6]). In the past thirty years, the edge-searching problem and similar type problems have attracted researchers from various fields of mathematics and computer science and have been linked to pebble games [12], that model sequential computation; to assuring privacy when using bugged channels [8]; and to VLSI circuit design [7]. The chip-firing game also has rapidly become an important and interesting object of study in structural combinatorics, in part because of its relation to the Tutte polynomial and group theory [13]. It has also been studied in computer science [5], physics [3], and social science [4].

We formally describe the process below.

Definition 1.1. Let $G = (V, E)$ be a connected, undirected graph, and let $\omega_t : V \cup E \to \mathbb{R}_+ \cup \{0\}$ such that $\omega_t(v)$ and $\omega_t(uv)$ denote the weight of a vertex v and an edge uv, respectively at time $t \in \{0, 1, \dots\}$. Define the *single-node traffic flow process* $\mathcal{P} = \mathcal{P}(G, \omega_0) = \{\omega_t\}_{t=0}^{T}$ of G with an *initial configuration* ω_0 and *total weight* of $\omega = \sum_{v \in V} \omega_0(v)$ as follows:

(1) Set $t := 0$ and the weight of all edges to zero (that is, $\omega_0(uv) = 0$ for each $uv \in E$).
(2) If no edge has weight less than one (that is, $\min_{uv \in E} \omega_t(uv) \geq 1$), then stop the process, set $T = t$, return the *firing sequence* of vertices $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_T)$ and the *sequence of weights* $\beta = (\beta_1, \beta_2, \dots, \beta_T)$ that have been used during the process. Otherwise, go to step (3)
(3) Set $t := t + 1$. Then fire the content of any vertex α_t by moving $\beta_t = \omega_{t-1}(\alpha_t)/\deg(\alpha_t)$ units to each neighbour of α_t. Each edge incident to α_t then increases its weight by β_t. More precisely, $\omega_t(\alpha_t) = 0$, for every $v \in N(\alpha_t)$, $\omega_t(\alpha_t v) = \omega_{t-1}(\alpha_t v) + \beta_t$ and $\omega_t(v) = \omega_{t-1}(v) + \beta_t$ (the other values of ω_t remain the same as ω_{t-1}).
(4) Go to step (2).

Note that the total number of moveable units at any step of the process is equal to ω. Note also that it is possible that the process never stops (for example, one can fire the same vertex all the time) but for any connected graph there is a firing sequence that can be used so that the process is finite, unless the total weight is zero.

Theorem 1.2. *Let $G = (V, E)$ be any connected graph and ω_0 be any initial configuration of a non-zero total weight ω. There is a firing sequence α that can be used to finish the process; that is, to get $T < \infty$.*

Proof. Consider the following greedy algorithm that fires a vertex of a maximum weight at each step of the process. It is clear that the weight of each vertex fired is at least the average weight; that is, $\omega_{t-1}(\alpha_t) \geq \omega/|V| > 0$. We show that this algorithm yields $T < \infty$. For a contradiction, suppose that the algorithm runs forever; that is, there is a vertex that is fired an infinite number of times. Note that after $\lceil |V| \deg(v)/\omega \rceil$ firings of a vertex v, each edge adjacent to v must have weight at least one. Therefore there must also be at least one vertex that is fired a finite number of times. Consider a vertex v that is fired a finite number of times and is adjacent to the vertex u that is fired an infinite number of times. Suppose the final firing of v occurs at step $t = T$. Since u is fired an infinite number of times, after $t = T$ each time the content of u is fired v receives at least $\omega/(|V| \deg(u))$ units of traffic. As the total number of moveable units is ω, v eventually accumulates enough units to be fired. Contradiction. □

Based on the Theorem 1.2 the following definition is natural.

Definition 1.3. Let $G = (V, E)$ be any connected graph and ω_0 be any initial configuration of a non-zero total weight ω. Let its *life* $f(G, \omega_0)$ be the minimum value of T that can be realized by the single-node traffic flow process. In particular, $f_{\overline{\omega}}(G) = f(G, \overline{\omega}_0)$,

where $\overline{\omega}_0$ is an initial configuration with total weight $\overline{\omega}$ where every vertex has equal weight; that is, $\overline{\omega}_0(v) = \overline{\omega}/|V|$, $v \in V$.

In light of definition 1.3, the process can also be viewed as an one-person combinatorial game on graphs ($f(G, \omega_0)$ is the score obtained by a perfect player).

The remainder of the paper is organized as follows. In the next section, we investigate the process for complete graphs on n vertices. In particular, we obtain a complete characterization of the life of K_n for any initial configuration ω_0. In Section 3 we extend our results to complete bipartite graphs, obtaining an order-of-magnitude result for the life $f_{\overline{\omega}}(K_{n,n})$. It seems that analyzing other families of graphs is much more sophisticated. As an example, we consider stars in Section 4, and conclude the paper with a few open problems in Section 5.

2. Complete graphs

We begin by proving some results for complete graphs $G = K_{n+1}$ where the initial configuration is $\overline{\omega}_0$. In fact, we prove that the number of steps needed by a greedy algorithm that always fires a vertex of maximum weight is asymptotically equal to $f_{\overline{\omega}}(K_{n+1})$ for large n. To that end, we first prove some properties of the greedy approach.

Let $x_1 \geq x_2 \geq \cdots \geq x_n > 0$ denote a sorted sequence of weights for K_{n+1}. (Note that the vertex that has been fired in the last round has weight zero so n numbers is enough to model the behaviour of K_{n+1}.) The greedy algorithm can be modelled by the map $F : \mathbb{R}^n \to \mathbb{R}^n$ which replaces x_1 by x_1/n and adds x_1/n to all the other $n-1$ numbers, and then sorts them in non-increasing order (again we ignore the vertex of weight zero that results from each firing). That is, F is given by

$$F((x_1, \ldots, x_n)^T) = (x_2 + \tfrac{x_1}{n}, x_3 + \tfrac{x_1}{n}, \ldots, x_n + \tfrac{x_1}{n}, \tfrac{x_1}{n})^T.$$

Since F is linear on the x_i, we may rewrite the map in terms of multiplying the vector $\vec{x} = (x_1, \ldots, x_n)^T$ by an $n \times n$ matrix A_n given by

$$A_n := \begin{pmatrix} 1/n & 1 & 0 & \cdots & 0 \\ 1/n & 0 & 1 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1/n & 0 & 0 & & 1 \\ 1/n & 0 & 0 & \cdots & 0 \end{pmatrix}$$

so that $F^k(\vec{x}) = A_n^k \vec{x}$ for any $k \in \mathbb{N}$. Note that after at most n iterations the internal ordering becomes strictly decreasing.

Theorem 2.1. *Denote by $\vec{x}$ the column vector $(x_1, \ldots, x_n)^T$ and by $\vec{s}$ the vector $(n, n-1, \cdots, 1)^T$. For any $\vec{x} \in \mathbb{C}^n$ we have*

$$\lim_{k \to \infty} A_n^k \vec{x} = \frac{2(x_1 + \cdots + x_n)}{n(n+1)} \vec{s}. \tag{2.1}$$

To prove Theorem 2.1, we use the following theorem from the theory of polynomials [18, p. 255].

Theorem 2.2 (Eneström-Kakeya)**.** *Let $p(z) := \sum_{j=0}^{n} a_j z^j$ be a polynomial of degree n with positive real coefficients. Then all the zeroes of $p(z)$ lie inside the annulus $\alpha \leq |z| \leq \beta$, where*

$$\alpha = \min_{0\leq j<n}\{a_j/a_{j+1}\}, \qquad \textit{and} \qquad \beta = \max_{0\leq j<n}\{a_j/a_{j+1}\}.$$

Proof of Theorem 2.1. Let $A_n(m)$ be the $m \times m$ matrix of the form of A_n; that is, the $m \times m$ matrix with $1/n$ down the first column, ones on the upper off-diagonal, and zeroes everywhere else. The characteristic equation $f_{m,n}(\lambda) = \det(A_n(m) - \lambda I_m)$ is given by

$$\begin{aligned}\det(A_n(m) - \lambda I_m) &= (1/n - \lambda)(-\lambda)^{m-1} - (-\lambda)^{m-2}/n + (-\lambda)^{m-3}/n - \cdots + (-1)^{m-1}/n \\ &= \frac{(-1)^{m-1}}{n}(1 + \lambda + \cdots + \lambda^{m-1} - n\lambda^m).\end{aligned}$$

The case we are interested in is at $m = n$, so that

$$\begin{aligned}f_{n,n}(\lambda) = f_n(\lambda) &= \frac{(-1)^n}{n}(n\lambda^n - \lambda^{n-1} - \cdots - 1) \\ &= \frac{(-1)^n}{n}(\lambda - 1)(n\lambda^{n-1} + (n-1)\lambda^{n-2} + \cdots + 1).\end{aligned}$$

Clearly $\lambda = 1$ is a simple zero of f_n. Since $n(-1)^n(\lambda - 1)f_n = n\lambda^{n+1} - (n+1)\lambda^n + 1$ has derivative $n(n+1)\lambda^{n-1}(\lambda - 1)$, we see that f_n in fact has n distinct simple zeroes. Eneström-Kakeya implies that aside from $\lambda = 1$, the remaining $n - 1$ zeroes of f_n lie in the annulus $1/2 \leq |\lambda| \leq 1 - 1/n$. Thus A_n has n distinct eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_n$, with $\lambda_1 = 1$ and $1/2 \leq |\lambda_i| \leq 1 - 1/n$ for $2 \leq i \leq n$. If we let $\vec{v_i}$ be the eigenvector associated with λ_i, then since the eigenvalues are distinct, the eigenvectors are linearly independent over $\mathbb{C}$ and so any $\vec{x} \in \mathbb{C}^n$ may be expressed as a unique linear combination of the $\vec{v_i}$. That is,

$$\vec{x} = \sum_{i=1}^{n} c_i \vec{v_i}$$

for some constants c_i. Therefore, we find that

$$\begin{aligned}\lim_{k\to\infty} A_n^k \vec{x} = \lim_{k\to\infty} A_n^k \sum_{i=1}^{n} c_i \vec{v_i} &= \lim_{k\to\infty} \sum_{i=1}^{n} c_i \lambda_i^k \vec{v_i} \\ &= c_1\vec{v_1} + \lim_{k\to\infty} \sum_{i=2}^{n} \lambda_i^k \vec{v_i} = c_1\vec{v_1}.\end{aligned}$$

It is easy to check that $\vec{v_1} = \vec{s}$. So all that remains is to calculate c_1. But we note that multiplying any vector $\vec{x}$ by A_n preserves the sum of the components of $\vec{x}$. Thus we have the condition

$$x_1 + x_2 + \cdots + x_n = c_1(n + (n-1) + \cdots + 1).$$

Solving for c_1 completes the proof. □

Remark 2.3. We note that if the sum of the weights $x_1 + \cdots + x_n$ is non-zero, then in fact the algorithm that fires each vertex in turn will be *eventually greedy*, in the sense that the vertex with the largest absolute value of the weight will be fired in each step.

Corollary 2.4. *Let n be fixed and let $\vec{x}$ and $\vec{s}$ be as in Theorem 2.1. Then the rate of convergence, as k tends to ∞, is*

$$\left\| A_n^k \vec{x} - \frac{2(x_1 + \cdots + x_n)}{n(n+1)} \vec{s} \right\| = O\left(n \left(1 - \frac{1}{n} \right)^k \right), \tag{2.2}$$

where the big-O constant depends on $\vec{x}$.

Proof. From the proof of Theorem 2.1 we have that

$$\left\| A_n^k \vec{x} - \frac{2(x_1 + \cdots + x_n)}{n(n+1)} \vec{s} \right\| = \left\| \sum_{i=2}^{n} c_i \lambda_i^k \vec{v_i} \right\|. \tag{2.3}$$

Applying the triangle inequality and using the fact that $\lambda_i \leq 1 - 1/n$ for $2 \leq i \leq n$, we find

$$\left\| \sum_{i=2}^{n} c_i \lambda_i^k \vec{v_i} \right\| \leq (n-1) \left(\max_{2 \leq i \leq n} \|c_i \vec{v_i}\| \right) \left(1 - \frac{1}{n} \right)^k = O\left((n-1) \frac{(n-1)^k}{n^k} \right)$$

as required. □

We also have a lower bound for the error.

Corollary 2.5. *Let n be fixed and let $\vec{x}$ and $\vec{s}$ be as in Theorem 2.1. If $\vec{x} \neq \dfrac{2(x_1 + \cdots + x_n)}{n(n+1)} \vec{s}$, then the difference is bounded below by*

$$\left\| A_n^k \vec{x} - \frac{2(x_1 + \cdots + x_n)}{n(n+1)} \vec{s} \right\| = \Omega\left(2^{-k} \right), \tag{2.4}$$

where, as above, the big-Ω constant depends on $\vec{x}$ and n.

Proof. Let $L = \max_{2 \leq i \leq n} |\lambda_i|$. Applying the triangle inequality to (2.3) we find that

$$\left\| \sum_{i=2}^{n} c_i \lambda_i^k \vec{v_i} \right\| \geq \left| \sum_{\substack{2 \leq i \leq n \\ |\lambda_i| = L,\ c_i \neq 0}} L^k \|c_i \vec{v_i}\| - \sum_{\substack{2 \leq i \leq n \\ |\lambda_i| < L}} \lambda_i^k \|c_i \vec{v_i}\| \right| = \Omega(L^k)$$

as $k \to \infty$, since for fixed n and $\vec{x}$, the c_i and $\vec{v_i}$ are also fixed. Since $L \geq 1/2$ by Theorem 2.2, we have the desired result. □

We can also prove explicit bounds for the special case $\vec{x} = (x, \ldots, x)^T$.

Corollary 2.6. *Let $\vec{x} = (x, \ldots, x)^T$ and denote by $x_1(k)$ the first coordinate of $A_n^k \vec{x}$. Then we have*

$$\left| x_1(k) - 2\frac{nx}{n+1} \right| \leq \frac{xn(n-1)}{n+1} \left(\frac{n-1}{n} \right)^{k+1}. \tag{2.5}$$

Proof. It is easy to check that for each eigenvalue λ_i, the corresponding eigenvector is

$$\vec{v_i} = \begin{pmatrix} n\lambda_i \\ -\lambda_i + n\lambda_i^2 \\ \vdots \\ -\lambda_i - \lambda_i^2 - \cdots - \lambda_i^{n-2} + n\lambda_i^{n-1} \\ 1 \end{pmatrix}. \tag{2.6}$$

To obtain explicit bounds, we need to determine c_i for $i \geq 2$. We claim that $c_i = \dfrac{x}{n+1}$ works. That is, we would like to show that for each $1 \leq j \leq n$ we have

$$\sum_{i=1}^{n} c_i(n\lambda_i^j - \sum_{\ell=1}^{j-1} \lambda_i^\ell) = x$$

for that choice of c_i. Recalling that $c_1 = 2x/(n+1)$ and $\lambda_1 = 1$, we calculate (dividing through by x for simplicity),

$$\sum_{i=1}^{n} c_i(n\lambda_i^j - \sum_{\ell=1}^{j-1} \lambda_i^\ell) = \frac{n-j+1}{n+1} + \sum_{i=1}^{n} \frac{1}{n+1}(n\lambda_i^j - \sum_{\ell=1}^{j-1} \lambda_i^\ell).$$

Let S_k denote the elementary symmetric polynomial of degree k on $\lambda_1, \ldots, \lambda_n$ and P_k denote the sum of the kth powers of λ_i. That is,

$$P_k := \sum_{i=1}^{n} \lambda_i^k.$$

Since the λ_i are all the zeroes of $f_n(\lambda)$, we have $S_k = (-1)^k$(normalized coefficient of λ^{n-k}) $= (-1)^{k-1}/n$ for $1 \leq k \leq n$ and $S_0 = 1$. Now, by the Newton-Girard Formulas [18, p. 8, eq. 1.2.9] we have

$$P_k = (-1)^{k-1} k S_k + \sum_{j=1}^{k-1} (-1)^{j-1} S_j P_{k-j} = \frac{k}{n} + \sum_{j=1}^{k-1} \frac{P_j}{n}. \tag{2.7}$$

Therefore,

$$\begin{aligned} \frac{n-j+1}{n+1} + \sum_{i=1}^{n} \frac{1}{n+1}(n\lambda_i^j - \sum_{\ell=1}^{j-1} \lambda_i^\ell) &= \frac{n-j+1}{n+1} + \frac{nP_j}{n+1} - \frac{1}{n+1} \sum_{\ell=1}^{j-1} P_\ell \\ &= \frac{n-j+1}{n+1} + \frac{j}{n+1} = 1, \end{aligned}$$

as required.

It is now a simple matter to estimate the size of the first coordinate in $A_n^k \vec{x} - c_1 \vec{v_1}$. Since

$$A_n^k \vec{x} - c_1 \vec{v_1} = \sum_{i=2}^{n} \lambda_i^k c_i \vec{v_i},$$

we find that

$$\left|x_1(k) - \frac{2nx}{n+1}\right| = \left|\sum_{i=2}^{n} \lambda_i^k \frac{xn\lambda_i}{n+1}\right| \leq \frac{xn(n-1)}{n+1} \max_{2\leq i\leq n} |\lambda_i|^{k+1}$$
$$\leq \frac{xn(n-1)}{n+1}\left(\frac{n-1}{n}\right)^{k+1}.$$

□

Corollary 2.7. *Let $x_1(k)$ be as in Corollary 2.6. Then we have*

$$|x_1(k)| \leq \frac{xn}{n+1}\left(1+\frac{1}{n}\right)^n < \frac{xne}{n+1} \tag{2.8}$$

for all $k \geq 0$.

Proof. By the proof of Corollary 2.6, we have

$$x_1(k) = \sum_{i=1}^{n} c_i \lambda_i^k n\lambda_i = \frac{xn}{n+1} P_{k+1} + \frac{xn}{n+1}\lambda_1^{k+1}.$$

Since the λ_i satisfy $f_n(\lambda_i) = 0$, we find that for $k \geq n$,

$$\lambda_i^k = \frac{\lambda_i^{k-1} + \cdots + \lambda_i^{k-n}}{n}.$$

Thus

$$P_k = \frac{P_{k-1} + \cdots + P_{k-n}}{n} \tag{2.9}$$

for all $k \geq n$. But using (2.7) we have

$$P_k = \left(1+\frac{1}{n}\right)^k - 1$$

for $1 \leq k \leq n$, so $0 < P_1 < P_2 < \cdots < P_n$. Combining this with (2.9) we find that

$$0 \leq P_k \leq P_n = \left(1+\frac{1}{n}\right)^n - 1$$

for all $k \geq 1$. This gives the desired result. □

We are now ready to prove the main theorem on complete graphs.

Theorem 2.8. *For $n \geq 1$ and $\overline{\omega}$ some function of n, we have*

$$f_{\overline{\omega}}(K_{n+1}) = \frac{n(n+1)^2}{4\overline{\omega}} + O(n\log n).$$

In particular, for $\overline{\omega} = o(n^2/\log n)$

$$f_{\overline{\omega}}(K_{n+1}) = \frac{n^3}{4\overline{\omega}}(1+o(1)).$$

Proof. In order to get an upper bound, we analyze a greedy algorithm that always fires the content of a vertex of maximum weight. Let $V = \{v_1, v_2, \ldots, v_{n+1}\}$ such that $\omega_0(v_1) = \omega_0(v_2) = \cdots = \omega_0(v_{n+1}) = \frac{\overline{\omega}}{n+1}$. It is easy to see that the firing process can be divided into a number of rounds where at each round vertices $v_1, v_2, \ldots, v_{n+1}$ are processed in increasing order of labels. After the first step, we obtain an equivalent system with $\omega(v_2) = \cdots = \omega(v_{n+1}) = \frac{\overline{\omega}}{n+1} + \frac{1}{n}\frac{\overline{\omega}}{n+1} = \frac{\overline{\omega}}{n}$, $\omega(v_1) = 0$. From Corollary 2.6 with $x = \overline{\omega}/n$ it follows that at time t of the process we fire the content of a vertex of weight

$$\omega_{t-1}(\alpha_t) = n\beta_t \geq \overline{\omega}\left(\frac{2}{n+1} - \frac{n-1}{n+1}\left(1 - \frac{1}{n}\right)^t\right) \geq \overline{\omega}\left(\frac{2}{n+1} - e^{-t/n}\right).$$

Thus, during the round k the weight of each vertex processed is at least $\overline{\omega}\left(\frac{2}{n+1} - e^{-k(n+1)/n}\right)$. For early rounds (say, up to and including round k_0), one can use a trivial lower bound of $\frac{\overline{\omega}}{n+1}$. Since at each round each edge increases its weight two times, we find that any k (and k_0) such that the inequality

$$\frac{2\overline{\omega}k_0}{n(n+1)} + \sum_{i=k_0+1}^{k} \frac{2\overline{\omega}}{n}\left(\frac{2}{n+1} - e^{-i(n+1)/n}\right) \geq 1$$

holds will give an upper bound on the number of rounds. Simplifying the left-hand side gives

$$\frac{(4k - 2k_0)\overline{\omega}}{n(n+1)} - \frac{2\overline{\omega}e^{-(k_0+1)(1+1/n)}}{n}\frac{1 - e^{-(k-k_0-1)(1+1/n)}}{1 - e^{-1-1/n}} \geq 1.$$

This is implied if the inequality (with $k_0 = \log n$)

$$\frac{(4k - 2\log n)\overline{\omega}}{n(n+1)} - \frac{2\overline{\omega}}{n^2(e^{1+1/n} - 1)} \geq 1$$

holds. Thus we have

$$k \geq \frac{n(n+1)}{4\overline{\omega}} + \frac{n+1}{2n(e^{1+1/n} - 1)} + \frac{\log n}{2} = \frac{n(n+1)}{4\overline{\omega}} + O(\log n)$$

as $n \to \infty$.

For a lower bound, we use the fact that at the end of the process, each edge has weight at least one; that is,

$$\sum_{t=1}^{N} n\beta_t \geq \binom{n+1}{2}. \tag{2.10}$$

Note that this is a necessary condition for the process to finish but clearly not a sufficient one. Define by $S(\vec{a}, N)$ the maximum over all processes with exactly N steps of the sum

$$\sum_{t=1}^{N} n\beta_t$$

with initial configuration $\vec{a}$, and $S(\vec{a}, N, k)$ the analogous maximum provided the first vertex fired is the kth component of $\vec{a}$. We now show by induction that the greedy algorithm gives $S(\vec{a}, N)$ for all $\vec{a}$ and N. The base case $N = 1$ is trivial. For the

inductive step, consider a process of length $N+1 \geq 2$. We start with an initial configuration ω_0; let $x_i = \omega_0(v_i)$, $i \in [n+1]$ (recall that $x_1 \geq x_2 \geq \cdots \geq x_{n+1}$). We could start the process by firing vertex v_k, $k \geq 2$, to get a sorted sequence of weights

$$\vec{a} := \left(x_1 + \frac{x_k}{n}, \ldots, x_{k-1} + \frac{x_k}{n}, x_{k+1} + \frac{x_k}{n}, \ldots, x_{n+1} + \frac{x_k}{n}\right)^T$$

(where we drop the zero term from the vector) but it appears that firing vertex v_{k-1} gives a result which is not worse than the previous one. After firing v_{k-1} we get

$$\vec{b} := \left(x_1 + \frac{x_{k-1}}{n}, \ldots, x_{k-2} + \frac{x_{k-1}}{n}, x_k + \frac{x_{k-1}}{n}, \ldots, x_{n+1} + \frac{x_{k-1}}{n}\right)^T$$

and in both situations a greedy algorithm has to be used to maximize the sum we consider, by the inductive hypothesis. Let $\Delta = x_{k-1} - x_k \geq 0$. The process starting with $\vec{b}$ can be seen as a combination of two processes: the process starting with $\vec{a}$ and the process starting with

$$\begin{aligned}\vec{b} - \vec{a} &= \left(\frac{\Delta}{n}, \ldots, \frac{\Delta}{n}, \frac{\Delta}{n} - \Delta, \frac{\Delta}{n}, \ldots, \frac{\Delta}{n}\right)^T \\ &= \frac{\Delta}{n}(1, 1, \ldots, 1)^T - \Delta\vec{e}_{k-1} = \Delta(A_n - A_n^{2-k})\vec{e_1}\end{aligned}$$

(here $\vec{e_k}$ is the elementary unit vector with a 1 in the kth component and 0 everywhere else, so that $A_n^{k-1}\vec{e_k} = \vec{e_1}$). In other words, $\omega_i^{\vec{b}}(v)$ in the process starting with $\vec{b}$ can be obtained as a sum of two values $\omega_i^{\vec{a}}(v)$ and $\omega_i^{\vec{b}-\vec{a}}(v)$ for corresponding processes starting with $\vec{a}$ and with $\vec{b} - \vec{a}$, respectively. Applying the greedy algorithm gives the sequence of weight vectors $A_n^j\vec{a}$ and $A_n^j\vec{b} = A_n^j\vec{a} + A_n^j(\vec{b} - \vec{a})$ after j steps. Therefore

$$S(\vec{a}, N) = g\left(\sum_{i=0}^{N-1} A_n^i\vec{a}\right) := \text{first coordinate of } \sum_{i=0}^{N-1} A_n^i\vec{a},$$

and

$$S(\vec{b}, N) = g\left(\sum_{i=0}^{N-1}\left(A_n^i\vec{a} + A_n^i(\vec{b} - \vec{a})\right)\right).$$

Thus the difference $S(\vec{x}, N+1, k-1) - S(\vec{x}, N+1, k)$ is

$$\begin{aligned}S(\vec{x}, N+1, k-1) - S(\vec{x}, N+1, k) &= \Delta + S(\vec{b}, N) - S(\vec{a}, N) \\ &= \Delta + g\left(\sum_{i=0}^{N-1} A_n^i(\vec{b} - \vec{a})\right) \\ &= \Delta\left(1 + g\left(\sum_{i=0}^{N-1}(A_n^{i+1} - A_n^{i+2-k})\vec{e_1}\right)\right).\end{aligned}$$

Define $q_s := \sum_{i=0}^{s} g(A_n^i\vec{e_1})$, with $q_0 = 1$ and $q_{-s} = 0$ for all $s > 0$. It is easy to see that q_s is positive and increasing for $s \geq 0$ since the summands are positive for $i \geq 1$.

Therefore,

$$S(\vec{x}, N+1, k-1) - S(\vec{x}, N+1, k) = \Delta\left(1 + g\left(\sum_{i=0}^{N-1}(A_n^{i+1} - A_n^{i+2-k})\vec{e_1}\right)\right)$$
$$= \Delta(1 + q_N - q_0 - q_{N-1+(2-k)}) = \Delta(q_N - q_{N-(k-1)}) \geq 0.$$

This finishes the inductive proof.

To show the lower estimate, once again apply Corollary 2.6 to show that at time t the maximum weight of the vertices is bounded above by

$$\omega_{t-1}(\alpha_t) = n\beta_t \leq \overline{\omega}\left(\frac{2}{n+1} + \frac{n-1}{n+1}\left(1 - \frac{1}{n}\right)^t\right) \leq \overline{\omega}\left(\frac{2}{n+1} + e^{-t/n}\right).$$

For early steps (say, up to and including step t_0) we can apply Corollary 2.7 to estimate that

$$n\beta_t \leq \frac{\overline{\omega}e}{n+1}.$$

Thus the largest positive integer $T \geq t_0$ that satisfies

$$\sum_{t=1}^{t_0}\frac{\overline{\omega}e}{n+1} + \sum_{t=t_0+1}^{T}\overline{\omega}\left(\frac{2}{n+1} + e^{-t/n}\right) \leq \frac{n(n+1)}{2}$$

gives a lower bound for $f_{\overline{\omega}}(K_{n+1})$. Simplifying we find the left-hand side is bounded above by

$$\frac{\overline{\omega}(2T + (e-2)t_0)}{n+1} + \overline{\omega}e^{-(t_0+1)/n}(T - t_0).$$

Choosing $t_0 = 2n\log n$ we get

$$\frac{\overline{\omega}(2T + 2(e-2)n\log n)}{n+1} + \frac{\overline{\omega}e^{-1/n}(T - 2n\log n)}{n^2} \leq \frac{n(n+1)}{2},$$

and solving for T we find that

$$T \geq \frac{n(n+1)^2}{4\overline{\omega}} - O(n\log n)$$

is necessary for (2.10) to hold. □

The above techniques extend to arbitrary initial configurations $\omega_0 = (x_1, x_2, \cdots, x_n, 0)$ for K_{n+1}.

Theorem 2.9. *For sorted initial configurations $\omega_0 = (x_1, x_2, \ldots, x_n, 0)$ on K_{n+1}, where both x_1 and $\omega = x_1 + \cdots + x_n$ may depend on n, we have*

$$\frac{n(n+1)^2}{4\omega} - O\left(\frac{n^2 x_1}{\omega}\log n\right) \leq f_\omega(K_{n+1}) \leq \frac{n(n+1)^2}{4\omega} + O(n\log(nx_1/\omega)). \quad (2.11)$$

Proof. Beginning with a sorted configuration $\vec{x} = (x_1, x_2, \ldots, x_n)^T$ for K_{n+1} with sum ω, we may decompose it as

$$\vec{x} = \sum_{i=0}^{n} x_i\vec{e_i} = \left(\sum_{i=0}^{n} x_i A_n^{-i}\right)\vec{e_1} = \left(\sum_{i=0}^{n}\frac{x_i A_n^{-i-1}}{n(n+1)}\right)\left(2\vec{v_1} + \sum_{j=2}^{n}\vec{v_j}\right),$$

where as before $\vec{v_i}, 1 \le i \le n$, are the eigenvectors of A_n, corresponding to the eigenvalues λ_i with $\lambda_1 = 1$. This implies

$$\vec{x} = \frac{2\omega}{n(n+1)}\vec{v_1} + \sum_{j=2}^{n}\left(\sum_{i=1}^{n}\frac{x_i\lambda_j^{-i-1}}{n(n+1)}\right)\vec{v_j}.$$

Thus the first coordinate $x_1(k)$ of $A_n^k\vec{x}$ is bounded by

$$\begin{aligned}\left|x_1(k) - \frac{2\omega}{n+1}\right| &= \left|\sum_{j=2}^{n}\left(\sum_{i=1}^{n}\frac{x_i\lambda_j^{k-i-1}}{n(n+1)}\right)n\lambda_j\right| \\ &\le \frac{x_1 n(n-1)}{n+1}\left(1-\frac{1}{n}\right)^{k-n}.\end{aligned}$$

We also have the uniform bound

$$\begin{aligned}|x_1(k)| &= \left|\frac{\omega}{n+1} + \sum_{j=1}^{n}\left(\sum_{i=1}^{n}\frac{x_i\lambda_j^{k-i-1}}{n(n+1)}\right)n\lambda_j\right| \\ &\le \frac{\omega}{n+1} + \left|\sum_{i=1}^{n}\frac{x_i}{n+1}P_{k-i}\right| \le \frac{\omega}{n+1} + \frac{nx_1(e-1)}{n+1}\end{aligned}$$

valid for $k \ge n$. But we know for $k < n$, $x_1(k)$ can be bounded by the first coordinate of the process that starts with $(x_1, x_1, \ldots, x_1)$. Thus the bound

$$|x_1(k)| \le \frac{nx_1e}{n+1}$$

is valid for all $k \ge 0$. Now we apply the same techniques as before to obtain upper and lower bounds for $f_{\omega_0}(K_{n+1})$. The relevant inequality for the upper bound is

$$\frac{(4k-2k_0)\omega}{n(n+1)} - \frac{2x_1e^{-(k_0-1+1)(1+1/n)}}{1-e^{-1-1/n}} \ge 1.$$

Setting $k_0 = \log(nx_1/\omega)$ and solving for k gives the result. For the lower bound the relevant inequality is

$$\sum_{t=1}^{t_0}\frac{nx_1e}{n+1} + \sum_{t=t_0+1}^{T}\left(\frac{2\omega}{n+1} + \frac{nx_1e}{n+1}e^{-t/n}\right) \le \frac{n(n+1)}{2}.$$

Using $t_0 = 2n\log n$ as before works. □

3. Complete bipartite graphs

The case of complete bipartite graphs $K_{n,n}$ is much more delicate compared to the analysis of complete graphs in the previous section. An upper bound is easy to obtain:

Theorem 3.1. *For any $n \ge 1$ and $\overline{\omega}$ a function of n, we have*

$$f_{\overline{\omega}}(K_{n,n}) \le \max\left(\frac{n^3}{\overline{\omega}} + \frac{n}{2}, n\right). \tag{3.1}$$

Proof. Consider the algorithm which fires every vertex of one partite set in some order, then the vertices of the other partite set, and so on. After the first n steps each vertex not fired will have weight ω/n and so after k rounds of firing all n vertices of a partite set (including the first round, of course), each edge has weight

$$\frac{\overline{\omega}}{n^2}(k-1)+\frac{\overline{\omega}}{2n^2}.$$

In order for this to be greater than 1 we require

$$k \geq \frac{n^2}{\overline{\omega}}+\frac{1}{2}$$

rounds. □

We note that the algorithm in the above proof is greedy after n steps. It is natural to ask whether greedy improves this bound, and whether it will give an optimal lower bound as in the case of complete graphs. Unfortunately, even if we ignore the question of how to choose between two vertices of equal weight in asymmetric cases, the following example shows that a greedy algorithm does not necessarily maximize the edge-sum for arbitrary length processes as in the case of complete graphs.

Example 3.2. *Consider the graph $K_{2,2}$ with partite sets (x_1, x_2) and (y_1, y_2) and weights $\omega(x_1) = 1.5, \omega(x_2) = 1, \omega(y_1) = 2, \omega(y_2) = 0$. Then a greedy algorithm would fire vertices in the order y_1, x_1, x_2 in 3 steps, giving an edge sum of 6.5. However, if we fire the vertices in the order x_1, y_1, x_2, then the edge sum becomes 6.625.*

While the methods of the previous section no longer extend directly to $K_{n,n}$, we may still obtain non-trivial lower bounds.

Theorem 3.3. *For any $n \geq 1$ and $\overline{\omega}$ a function of n, we have*

$$f_{\overline{\omega}}(K_{n,n}) \geq \max\left(\frac{n^3}{2\overline{\omega}} - O(n\log n), n\right). \tag{3.2}$$

Proof. The main idea is to reduce the problem to something treatable by our previous methods. To that end, given our complete bipartite graph $G = K_{n,n}$ with partite sets $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ we associate with it a graph G' by contracting each pair of vertices (x_i, y_i) to a single vertex z_i, while keeping the edge x_iy_i as a loop on z_i. At any time t, we let the weight of the vertices in G' be given by $\omega_t(z_i) = \omega_t(x_i) + \omega_t(y_i)$. Then any firing sequence $(v_i)_{i=1}^N$ in G corresponds to some firing sequence in G' where only some fraction $0 < r_t \leq 1$ of $\omega_t(z_i)$ is fired. We claim that a greedy algorithm with $r_t = 1$ for all t maximizes the N-step edge-sum in G' for every N. We note that the greedy algorithm on G' with $r_t = 1$ for all t can be modelled by the same matrix A_n as in the complete graph case. That is, the sorted vector of the n vertex weights at time $t+1$ is given b y multiplying the vector at time t by A_n. Thus, we only need to show that firing any vertex with $r_t = 1$ gives a larger edge-sum than firing that same vertex with $r_t < 1$. The proof now proceeds in exactly the same way as before. Let $S(\vec{a}, N)$, and $S(\vec{a}, N, k)$ be as before, and let $S(\vec{a}, N, k, r)$ be the corresponding maximum given that the vertex z_k is fired first and only $r\omega(z_k)$ is fired. In particular

$S(\vec{a}, N, k) = S(\vec{a}, N, k, 1)$. Starting with a sorted sequence of weights $\vec{w} = (w_1, \ldots, w_n)$, firing a fraction r of vertex k gives a sorted sequence of weights

$$\vec{a} := (w_1, w_2, \ldots, w_{k-1}, w_{k+1}, \ldots, w_j, (1-r)w_k, w_{j+1}, \ldots, w_n)^T + \frac{rw_k}{n}(1, \ldots, 1)^T,$$

where $w_j \geq (1-r)w_k > w_{j+1}$. In the case $r = 1$ we have $j = n$ and we denote this vector by $\vec{b}$. Then

$$\begin{aligned} D &:= S(\vec{w}, N+1, k) - S(\vec{w}, N+1, k, r) = (1-r)w_k + S(\vec{b}, N) - S(\vec{a}, N) \\ &= (1-r)w_k + g\left(\sum_{i=0}^{N-1} A_n^i(\vec{b} - \vec{a})\right), \end{aligned}$$

where as before g extracts the first coordinate of its argument. We decompose $\vec{b} - \vec{a}$ as

$$\begin{aligned} \vec{b} - \vec{a} &= (1-r)\frac{w_k}{n}(1, 1, \ldots, 1)^T + (w_{j+1} - (1-r)w_k)\vec{e_j} + \sum_{i=j+1}^{n}(w_{i+1} - w_i)\vec{e_i} \\ &= (1-r)w_k A_n \vec{e_1} + (w_{j+1} - (1-r)w_k)A_n^{1-j}\vec{e_1} + \sum_{i=j+1}^{n}(w_{i+1} - w_i)A_n^{1-i}\vec{e_1}, \end{aligned}$$

with $w_{n+1} := 0$. Thus,

$$\begin{aligned} D &= (1-r)w_k \\ &\quad + g\left(\sum_{i=0}^{N-1}\left((1-r)w_k A_n^{i+1} + (w_{j+1} - (1-r)w_k)A_n^{i+1-j} + \sum_{m=j+1}^{n}(w_{m+1} - w_m)A_n^{i+1-m}\right)\vec{e_1}\right) \\ &= (1-r)w_k(1 + q_N - q_0) + (w_{j+1} - (1-r)w_k)q_{N-1-(j-1)} + \sum_{m=j+1}^{n}(w_{m+1} - w_m)q_{N-1-(m-1)} \\ &\geq (q_N - q_{N-j})(1-r)w_k + \sum_{m=j+1}^{n} w_m(q_{N-m-1} - q_{N-m}) \geq 0. \end{aligned}$$

Now we apply the same analysis as in the previous section, except the edge-sum must now be at least n^2 rather than $\binom{n+1}{2}$. The result follows immediately. Note that we always need at least n steps since that is the size of the minimum vertex cover. □

It is clear that the lower bound is not best possible, since the process on the complete graph G' with $r_t = 1$ at every step cannot occur in the corresponding $K_{n.n}$, as content fired from one partite set ends up in the other. We therefore suspect that the upper bound is closer to the true value for $f_{\overline{\omega}}(K_{n,n})$. One might ask, can a greedy algorithm yield better bounds than the process given in the proof of Theorem 3.1? It turns out that one can show, using the same linear algebra techniques from the previous section, that

- The greedy algorithm applied directly to $\overline{\omega_0}$ on $K_{n,n}$ is the same as applying the greedy algorithm alternately to each partite set, and this algorithm is *well-defined*. That is, the map can be modelled by the matrix

$$B_n := \begin{pmatrix} 1/n & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & & 0 \\ 1/n & 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & & 1 \\ 1/n & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

  acting on the vector $(x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x_n)^T$, where $x_i \geq x_{i+1}$ belong to one partite set and $y_i \geq y_{i+1} \geq y_n = 0$ belong to the other partite set.
- The greedy algorithm requires at least $n^3/\omega - O(n \log n)$ steps and at most $n^3/\omega + O(n^2)$ steps.

4. Stars

In this section, we consider another simple class of graphs: stars. The lack of symmetry between the two partite sets causes a lot of difficulty, and so we only focus on obtaining upper bounds for $f_{\overline{\omega}}(K_{1,n})$ in this work.

In order to warm up, let us consider a star with 4 rays (vertex v is adjacent to v_1, v_2, v_3, and v_4) with the total weight of 5. We look at the ratio between the increase in weight of each edge to the number of moves in each round, in order to find the most efficient strategy for general initial configurations. Ignoring for the moment the condition that we stop the process when every edge attains weight 1 (or equivalently, scale the initial configuration by an appropriate factor to expose asymptotic behaviour), let us consider the following three different approaches:

- Process vertices in the following order: v_1, v, v_2, v, v_3, v, v_4, v, and so on. The process converges to the stationary distribution on v_i's $(1/2, 1, 3/2, 2)$ (v_i with the highest value fire 2 units and then v fire $1/4$ to each of its neighbours). Each edge increases its weight by 4 during 8 moves. The ratio between the increase and the number of moves is $1/2$.
- Process vertices in the following order: v_1, v_2, v, v_3, v_4, v, and we converge to $(5/6, 5/6, 10/6, 10/6)$. Each edge receives $10/3$ units during 6 moves, so the ratio is $5/9 > 1/2$.
- Process vertices in the following order: v_1, v_2, v_3, v_4, v, getting immediately $(5/4, 5/4, 5/4, 5/4)$. Each edge receives $5/2$ units during 5 moves, and we are back to $1/2$ again.

Trying to discover the best general strategy, let us consider a star with $n = 2^k$ rays for $k \in \mathbb{N}$. Fix an i in $0 \leq i \leq k$ and create 2^{k-i} groups consisting of 2^i vertices each. We process all vertices in a group with maximum content, then v, and repeat. In the stationary distribution, every vertex of group j ($j = 1, 2, \ldots, 2^{k-i}$) has value of $jx/2^{k-i}$

(in particular, the last one has value x). Since the sum over all vertices is $\overline{\omega}$, we get

$$\sum_{j=1}^{2^{k-i}} 2^i \frac{jx}{2^{k-i}} = \overline{\omega},$$

so $x = 2\overline{\omega}/(2^k + 2^i)$. During the whole cycle ($2^k + 2^{k-i}$ moves) each edge receives $2x$ units. The ratio is then $\frac{4\overline{\omega}}{(2^k+2^i)(2^k+2^{k-i})}$, which is maximized for $i = \lfloor k/2 \rfloor$. Therefore, it seems then that the best strategy is to split v_i's into roughly $\sqrt{n}$ sets of cardinalities as close to each other as possible. It is natural to conjecture that

$$f_{\overline{\omega}}(K_{1,n}) = \frac{n^2}{4\overline{\omega}}(1 + \Theta(1/\sqrt{n}))$$

for $\overline{\omega}$ small enough. Unfortunately, this still remains an open problem.

5. A few open problems

We already mentioned about the conjecture for complete bipartite graphs and stars. We would like to finish the paper with one more open problem. Let G be any connected graph on n vertices and consider its life $f_{\overline{\omega}}(G)$ with $\overline{\omega} = n$ (that is, initially each vertex has weight of 1). It seems that $f_{\overline{\omega}}(G)$ should depend on the density of G. It follows from our results that

$$\begin{aligned}
\frac{f_{\overline{\omega}}(K_n)}{|E(K_n)|} &= \frac{1}{2} + o(1), \\
\frac{f_{\overline{\omega}}(K_{n,n})}{|E(K_{n,n})|} &\leq 1 + o(1), \\
\frac{f_{\overline{\omega}}(K_{1,n})}{|E(K_{1,n})|} &\leq \frac{1}{4} + o(1).
\end{aligned}$$

Let $G(n)$ be a family of connected graphs on n vertices. It it natural to ask whether the following limits exist, and if so to find their values.

$$\begin{aligned}
M &= \lim_{n\to\infty} \max_{G\in G(n)} \frac{f_{\overline{\omega}}(G)}{|E(G)|}, \\
m &= \lim_{n\to\infty} \min_{G\in G(n)} \frac{f_{\overline{\omega}}(G)}{|E(G)|}.
\end{aligned}$$

In particular, is it true that $0 < m < M = O(1)$?

References

[1] B. Alspach, Searching and sweeping graphs: a brief survey, *International Conference in Combinatorics, Le Matematiche, Vol LIX* (2004) - Fasc. I-II, 5–37.

[2] N. Alon, P. Prałat, N. Wormald, Cleaning d-regular graphs with brushes, SIAM Journal on Discrete Mathematics **23** (2008), 233–250.

[3] P. Bak, C. Tang, K. Wiesenfeld, Self-organized criticality: an explanation of the $1/f$ noise, Physics Review Letters **59(4)** (1987) 381–384.

[4] N. Biggs, Algebraic potential theory on graphs, The Bulletin of the London Mathematical Society **29** (1997) 641–682.

[5] A. Björner, L. Lovász, W. Shor, Chip-firing games on graphs, European Journal of Combinatorics **12** (1991) 283-291.
[6] D. Dyer, Sweeping graphs and digraphs, Ph.D. thesis, Simon Fraser University, 2004.
[7] M. Fellows, M. Langston, On search, decision and the efficiency of polynomial time algorithm, 21st ACM Symp. on Theory of Computing (STOC 89), (1989) 501–512.
[8] M. Frankling, Z. Galil, M. Yung, Eavesdropping games: A graph-theoretic approach to privacy in distributed systems, Journal of ACM **47** (2000) 225–243.
[9] S. Gaspers, M.E. Messinger, R. Nowakowski, P. Prałat, Clean the graph before you draw it!, Information Processing Letters **109** (2009), 463–467.
[10] S. Gaspers, M.E. Messinger, R. Nowakowski, P. Prałat, Parallel Cleaning of a Network with Brushes, Discrete Applied Mathematics **158** (2010), 467–478.
[11] P. Gordinowicz, R. Nowakowski, and P. Prałat, POLISH-or-Let's play the cleaning game, submitted to Theoretical Computer Science, 17pp.
[12] L.M. Kirousis, C.H. Papadimitriou, Searching and pebbling, Theoretical Computer Science **47** (1986), 205–218.
[13] C. Merino, The chip-firing game, Discrete Mathematics **302** (2005) 188–210.
[14] M.E. Messinger, R.J. Nowakowski, P. Prałat, Cleaning a Network with Brushes, Theoretical Computer Science **399** (2008), 191–205.
[15] M.E. Messinger, R. Nowakowski, P. Prałat, Cleaning with Brooms, Graphs and Combinatorics **27** (2011), 251–267.
[16] P. Prałat, Cleaning random d-regular graphs with Brooms, Graphs and Combinatorics **27** (2011), 567–584.
[17] P. Prałat, Cleaning random graphs with brushes, Australasian Journal of Combinatorics **43** (2009), 237–251.
[18] Q. I. Rahman and G. Schmeisser, *Analytic Theory of Polynomials*, London Mathematics Society Monographs New Series **26**, Oxford University Press, 2002.

10221 Hollymount Drive, Richmond, BC, Canada, V7E 4T5
E-mail address: math@oyeat.com

Department of Mathematics, Ryerson University, Toronto, ON, Canada, M5B 2K3
E-mail address: pralat@ryerson.ca