Discrete Mathematics and Theoretical Computer Science DMTCS vol. **15:1**, 2013, 121–132

A chip-firing variation and a new proof of Cayley's Formula

P. Mark Kayll[1†] and Dave Perkins[2‡]

[1] *Department of Mathematical Sciences, University of Montana, Missoula MT, USA*
[2] *Department of Mathematics, Luzerne County Community College, Nanticoke PA, USA*

received 13th July 2012, revised 12th February 2013, accepted 20th February 2013.

We introduce a variation of chip-firing games on connected graphs. These 'burn-off' games incorporate the loss of energy that may occur in the physical processes that classical chip-firing games have been used to model. For a graph $G = (V, E)$, a configuration of 'chips' on its nodes is a mapping $C: V \to \mathbb{N}$. We study the configurations that can arise in the course of iterating a burn-off game. After characterizing the 'relaxed legal' configurations for general graphs, we enumerate the 'legal' ones for complete graphs K_n. The number of relaxed legal configurations on K_n coincides with the number t_{n+1} of spanning trees of K_{n+1}. Since our algorithmic, bijective proof of this fact does not invoke Cayley's Formula for t_n, our main results yield secondarily a new proof of this formula.

Keywords: chip-firing, burn-off games, relaxed legal configurations, Cayley's Formula

1 Introduction

Chip-firing games on graphs by now enjoy a rich literature. This is due not only to their surprising array of mathematical connections but also to their utility in modeling certain kinds of physical systems. For the former, we find ties, *e.g.*, to discrepancy theory [21], the Tutte polynomial [16], critical groups of graphs [5], G-parking functions [3], and stochastic processes [11]. For the latter, these games connect to earthquakes [1], sandpiles [10], traffic [19], and the brain [22].

This article continues the published account of [20]—initiated in [15]—where we study a variant of chip-firing in which all games have finite length. These 'burn-off' games incorporate the loss of energy that may occur in the physical processes that classical chip-firing games have been used to model. Whereas, classically, when a node v 'fires', it sends one chip to each of its neighbors, our game modifies this action by additionally eliminating one chip at v; see Section 1.1 for a precise definition of these games.

It's worth noting the analogy between burn-off games and graph pebbling, sometimes used in modeling the movement of expendable resources. Here, 'pebbles' are placed on the nodes of a graph, then moved

---

[†]Email: `mark.kayll@umontana.edu`; contact author
[‡]Email: `dperkins@luzerne.edu`; part of this work appears in the author's PhD dissertation [20].

around subject to the condition that if v contains (at least) two pebbles, then one of them may be moved to a neighbor of v at a cost of permanently removing another from v; see [14] for an early pebbling survey and [13] for a dynamic record of related references. Though we don't make use of the pebbling/burn-off game connection, the recent explosion in pebbling research suggests that our chip-firing approach could also prove to be fruitful.

Björner *et al.* [7] are generally credited with initiating the study of chip-firing games on general (connected, undirected) graphs, although a subset of these authors in [6] points to antecedents in [11] and [21]. It's historically interesting that an early result for general graphs, in [23], predates the purported initiator [7]. We offer some of this as evidence that a thorough literature review is beyond our scope and instead point the reader to [17] for a chip-firing survey, to [12] for a textbook account of the algebraic aspects of chip firing, and to [3] for a few more recent related citations.

This paper is organized as follows. Section 2 presents our results for general graphs, while Section 3 focuses on complete graphs. After characterizing the 'relaxed legal' chip configurations (§2.1), we give an algorithm to check a configuration for 'legality' (§2.2) and then prove that the set of 'legal' configurations is up-closed in the poset of all configurations on a fixed graph (§2.3). In §3.1, we enumerate the 'legal' chip configurations for complete graphs K_n. The number of relaxed legal configurations on K_n coincides with the number t_{n+1} of spanning trees of K_{n+1}. Since our algorithmic, bijective proof of this fact (§3.2) does not invoke Cayley's Formula for t_n, our results in Section 3 yield a new proof of this formula.

We are not the first authors to observe a connection between chip-firing games and spanning tree enumeration. Indeed, Biggs and Winkler [4] present bijections between their set of 'critical' configurations and the set of trees spanning their underlying graph. Benson *et al.* [3], who "present the[ir] article in an expository self-contained form", provide somewhat of a related mini-survey, including a citation to [4]. (Their scope also includes connections with parking functions and sandpile models.) Even more recently, Baker and Shokrieh [2] further explore these latter connections and obtain a bijective proof of the Matrix-Tree Theorem. We ourselves, in [15], have even touched on the chip-firing/spanning tree connection. Nevertheless, since burn-off games comprise a non-trivial variation of chip-firing, it seems worthwhile to illuminate this connection in the new context.

Notation and terminology

We generally follow 'standard' graph theory conventions and make attempts to produce a self-contained treatment of our results. For any basics we may have omitted, we point the reader to [8].

1.1 Description of the game

The 'game board' is a connected graph $G = (V, E)$. For each node $v \in V$, we begin with a nonnegative number $C(v)$ of chips on v; the function $C\colon V \to \mathbb{N}$ is a *configuration*. Each $v \in V$ has an associated *critical number* $k_v \geq \deg_G(v)$. Every node v for which $C(v) \geq k_v$ is *live* and may be 'fired'. Formally, when v *fires*, C is modified to a configuration C' such that

$$C'(u) = \begin{cases} C(v) - \deg_G(v) & \text{if } u = v, \\ C(u) + 1 & \text{if } uv \in E(G), \\ C(u) & \text{if } v \neq u \not\sim v; \end{cases} \tag{1}$$

loosely speaking, v sends a chip to each of its neighbors. A configuration in which no node is live is *relaxed*. Starting with a configuration C on G, a *chip-firing game* is played as follows. If there exists a live node, fire it; this constitutes a *turn* of the game. The game proceeds in turns, wherein live nodes are successively chosen and fired. If, at any point in this process, no node is live, the game ends, its *length* being the number of turns taken starting from C until the end. If C is relaxed, the game has length zero; otherwise, the process of passing from C through a C' and eventually to a relaxed configuration on G is *relaxing* C.

Björner *et al.* [7] took each k_v to be the minimum natural value $\deg_G(v)$. They observed that some chip-firing games may be of infinite length. For example, if the total number of chips on G exceeds twice the number of edges, then, by the pigeonhole principle, one can always find at least one live node. This paper also established, among other results, that if the total number of chips on G is less than $|E|$, then every chip-firing game on G is of finite length.

Another way to ensure finite game lengths is to decrease the total number of chips at each turn. A 'burn-off game' differs from the chip-firing game of [7] in two ways: first, we take each k_v, for $v \in V$, to be $\deg_G(v) + 1$; second, we modify the firing rule (1) to

$$C'(u) = \begin{cases} C(v) - \deg_G(v) - 1 & \text{if } u = v, \\ C(u) + 1 & \text{if } uv \in E(G), \\ C(u) & \text{if } v \neq u \not\sim v; \end{cases} \tag{2}$$

notice that when v is fired, one of its chips effectively vanishes. A *burn-off game* is a chip-firing game with the adjusted critical numbers and following the modified firing rule (2). The game just defined is equivalent to the 'dollar game' of Biggs [5] in the particular case when his 'government' node is adjacent to every other node in the underlying graph; see also [4]. In our burn-off games, we call a node v *supercritical* when $C(v)$ takes a value larger than $\deg_G(v)$.

During a chip-firing or burn-off game, chips from the neighbors of a node v may cause v to become supercritical. If, during one of these games, several nodes simultaneously become live in this fashion, one may wonder how to decide which node to fire next. The paper [7] established that for the chip-firing game described above, this decision has no bearing on the game length or the final chip configuration. Using a similar approach, this paper's second author showed in [20] that the analogous result for burn-off games also holds.

1.2 Reverse-firing and legal configurations

Our primary purpose is to study the configurations that can arise in the course of iterating a burn-off game, *i.e.*, incrementing C at a chosen node, relaxing the resulting configuration, and repeating these two steps indefinitely. Of course, the set of configurations that arise in this way depends on the initial configuration. Though it may seem natural to begin with the empty configuration, $C \equiv 0$, this configuration will never recur during a burn-off game sequence because each firing event redistributes chips to the neighbors of the firing node.

Instead, we seek to begin with a configuration typical of those that will be encountered in a long sequence of burn-off games. Here, we follow [1], wherein the authors investigated an earthquake model based on chip-firing on a grid.

We define a configuration to be *supercritical* if every node is supercritical. In the spirit of [1], we shall focus on the configurations that can result from relaxing a supercritical configuration. To understand these, it is instructive to consider what happens when a burn-off game is played in reverse.

A *reverse-firing game* is defined so as to undo the firing rule of the chip-firing game under consideration. Thus, for burn-off games, considering (2), we see that to *reverse-fire* a node v (each of whose neighbors

u of necessity satisfies $C'(u) \geq 1$) means to modify C' to a configuration C such that

$$C(u) = \begin{cases} C'(v) + \deg_G(v) + 1 & \text{if } u = v, \\ C'(u) - 1 & \text{if } uv \in E(G), \\ C'(u) & \text{if } v \neq u \not\sim v. \end{cases} \tag{3}$$

Informally, starting from C', when a node v reverse-fires, it pulls one chip from each of its neighbors and one out of thin air.

Now a configuration C is *legal* if there exists a reverse-firing sequence starting with C and ending with a supercritical configuration. A legal configuration containing no live nodes is a *relaxed legal* configuration. Notice that the empty configuration is relaxed but not legal, while the configuration with $C(v) = \deg_G(v) + 1$, for each $v \in V$, is legal but not relaxed. We leave it as an (easy) exercise to construct a configuration that is neither legal nor relaxed.

2 Results for general graphs

2.1 Characterizing relaxed legal configurations

Our first result characterizes the relaxed legal configurations on general (connected simple) graphs. Its statement uses N_G to denote the 'earlier neighbor set'; *i.e.*, given an ordering $(w_1, w_2 \ldots, w_n)$ of V, we define $N_G(w_i) := \{w_h \in V : w_h w_i \in E(G) \text{ and } h < i\}$.

Proposition 2.1 *A relaxed configuration $C : V \to \mathbb{N}$ is legal if and only if it is possible to relabel V as $w_1, w_2, \ldots, w_n$ so that*

$$C(w_i) \geq |N_G(w_i)| \text{ for } 1 \leq i \leq n. \tag{4}$$

Proof: Suppose that we have a relaxed legal configuration C. Since C is legal, it can be reverse-fired into a configuration where all nodes are supercritical. Further, since C is relaxed, no node is supercritical. Thus, in any reverse-firing sequence that certifies the legality of C, all nodes must reverse-fire (this being the only way for a node to gain chips during a reverse-firing game). Consider such a reverse-firing sequence. Listing only the first time each node reverse-fires during the game, suppose that they are reverse-fired in the order $w_1, w_2, \ldots, w_n$. Any given node w_j reverse-fires only after each node $w_1, w_2, \ldots, w_{j-1}$ reverse-fires at least once. Each of these nodes which is a neighbor of w_j takes a chip from w_j when it reverse-fires. Thus, we must have $C(w_j) \geq |N_G(w_j)|$, and (4) follows since j was arbitrary.

To establish the converse, suppose that it is possible to relabel the nodes $w_1, w_2, \ldots, w_n$ so that (4) holds. We claim that reverse-firing the nodes in increasing subscript order results in each node increasing its chip-count by one.

Consider a node w_j for some j with $1 \leq j \leq n$, and define $s := |N_G(w_j)|$. Each of the s nodes in $N_G(w_j)$ reverse-fires before w_j does, and each reverse-firing will pull one chip from w_j. This leaves $C(w_j) - s \geq 0$ chips on w_j. When w_j reverse-fires, it pulls a chip from each of its $\deg(w_j)$ neighbors and receives one extra chip for the reverse burn-off. Finally, the nodes $w_{j+1}, \ldots, w_n$ reverse-fire, and each neighbor of w_j in this set (say there are ℓ of these) pulls a chip from w_j. Therefore, after each node has been reverse-fired once, the number of chips on w_j is decreased by $s + \ell = \deg(w_j)$ and increased by $\deg(w_j) + 1$ (while never becoming negative), for a net increase of one, as claimed.

Notice that reverse-firing each node once, as described in the preceding paragraph, preserves (4). Thus, this process may be repeated until all nodes become supercritical. That is, if $t := \max_{v \in V}\{\deg(v) -$

$C(v)\}$, then repeating the process $t+1$ times will result in every node v containing at least $\deg(v)+1$ chips. Therefore, the original configuration was legal. $\square$

2.2 Checking the legality of a configuration

Given a configuration C (not necessarily relaxed) on a graph G, we may check if C is legal using the following algorithm. The proof of its efficacy leans on Proposition 2.1.

Algorithm 2.2

INPUT:	a graph $G = (V, E)$ and a chip configuration $C\colon V \to \mathbb{N}$ on G
OUTPUT:	an answer to the question 'Is C legal?'
(1)	Let $G^* = G$.
(2)	If $C(v) < \deg_{G^*}(v)$ for all $v \in V(G^*)$, then stop. Output 'No.'
(3)	Choose any $v \in V(G^*)$ with $C(v) \geq \deg_{G^*}(v)$.
(4)	Delete v and all incident edges from G^* to create a graph G^-.
(5)	If $V(G^-) = \varnothing$, then stop. Output 'Yes.'
(6)	Let $G^* = G^-$ and go to step 2.

In the proof of Proposition 2.4, which asserts that Algorithm 2.2 works correctly, we also need the following lemma. For a configuration $C\colon V(G) \to \mathbb{N}$ on G and a subgraph H of G, the notation $C|_{V(H)}$ as usual denotes the configuration restricted to H.

Lemma 2.3 *If C is legal on G, then $C|_{V(H)}$ is legal on H.*

Proof: By Proposition 2.1, the legality of C on G implies that it is possible to relabel the nodes $w_1, \ldots, w_n$ so that each w_i (considered in G) contains at least as many chips as it has neighbors with smaller subscripts. In H, a node w_i may have fewer such neighbors, but it certainly cannot have more. Thus, $C(w_i) \geq |N_G(w_i)| \geq |N_H(w_i)|$, so (4) holds for each w_i (considered now in H). Therefore, Proposition 2.1 shows that $C|_{V(H)}$ is legal on H. $\square$

Now we are ready to establish the correctness of Algorithm 2.2.

Proposition 2.4 *Given a graph $G = (V, E)$ and a configuration $C\colon V \to \mathbb{N}$ on G, Algorithm 2.2 correctly determines whether C is a legal configuration on G.*

Proof: First, we show that if at any point during the operation of Algorithm 2.2 (say, when we have arrived at a subgraph G^*) every node v contains fewer than $\deg_{G^*}(v)$ chips, then the original configuration is not legal. Suppose, for a contradiction, that an original configuration C^* leading to this situation on G^* is legal. By Lemma 2.3, $C^*|_{V(G^*)}$ is legal; letting $k := |V(G^*)|$, then by Proposition 2.1 the nodes of G^* may be relabeled $w_1, w_2, \ldots, w_k$ so that $C(w_i) \geq |N_{G^*}(w_i)|$ for all i with $1 \leq i \leq k$. But $C(w_k) \geq |N_{G^*}(w_k)| = \deg_{G^*}(w_k)$ contradicts our assumption in the first sentence. Thus, if every $v \in V(G^*)$ contains fewer than $\deg_{G^*}(v)$ chips, then C^* is not legal.

Second, we show that if Algorithm 2.2 proceeds until all nodes are deleted, then C is legal. Suppose we relabel the nodes so that the algorithm's deletion order is $u_n, u_{n-1}, \ldots, u_1$. For $n \geq j \geq 1$, let $G^*(u_j)$ be the subgraph of G remaining just before the deletion of u_j. For u_j to be deleted from $G^*(u_j)$

by Algorithm 2.2, it must contain at least as many chips as it has neighbors in G^*. Thus, the relabeling $u_n, u_{n-1}, \dots, u_1$ that gives the deletion order also suffices to show that (4) holds for C. Therefore, C is legal. $\square$

2.3 The poset of legal configurations

In the next section, we shall find it useful to consider the set $\mathcal{B}$ of all configurations on a fixed graph G as a poset $(\mathcal{B}, \preceq)$ whose ordering relates to the numbers of chips on the nodes of G as follows: for $P, Q \in \mathcal{B}$ and $\leq$ the usual (total) ordering on $\mathbb{N}$, let

$$P \preceq Q \text{ if and only if each } v \in V(G) \text{ satisfies } P(v) \leq Q(v).$$

Proposition 2.5 *If P is a legal configuration, then any Q with $P \preceq Q$ is also legal.*

Proof: We clearly need only consider those configurations Q with $P \prec Q$ (*i.e.*, $P \preceq Q$ but $P \neq Q$). Such a Q has at least as many chips on any given node v as does P, and since $P \prec Q$, there exists a node x with $P(x) < Q(x)$. Starting from the configuration P, add one chip to x to create a new configuration P'.

Since P is legal, there exists a reverse-firing sequence that results in a supercritical configuration. 'Freeze' the new chip on x, and carry out the same reverse-firing sequence starting with P'. The frozen chip will not affect the reverse-firing game, and once P is reverse-fired to a supercritical configuration, the chip may be 'thawed'. The resulting supercritical configuration shows that P' is legal.

If $P' = Q$, the assertion is proved; if not, the argument above can be repeated with P' in the role of P. $\square$

3 Results for complete graphs

After specializing Proposition 2.1 to complete graphs, we enumerate certain legal configurations on these graphs. Then we present a pair of algorithms that give a one-to-one correspondence between relaxed legal configurations on K_n and spanning trees of K_{n+1}. These algorithms provide our new proof of Cayley's Formula (see [9] or, *e.g.*, [8]).

3.1 Enumerating legal configurations

Lemma 3.1 *A relaxed configuration $C \colon V \to \mathbb{N}$ on K_n is legal if and only if it is possible to relabel the nodes $w_1, w_2, \dots, w_n$ so that*

$$C(w_i) \geq i - 1 \text{ for } 1 \leq i \leq n. \tag{5}$$

Proof: We simply observe that condition (5) is equivalent to condition (4) because in a complete graph, each w_i satisfies $|N_{K_n}(w_i)| = i - 1$. $\square$

The following formulation of Lemma 3.1 is needed in Section 3.2.

Corollary 3.2 *A relaxed configuration $C \colon V \to \mathbb{N}$ on K_n is legal if and only if for each $\ell \in \{1, 2, \dots, n\}$, at least $n - \ell$ nodes contain at least ℓ chips.*

Now we develop a formula for $L(K_n)$, the number of relaxed legal configurations C on K_n. We first count the legal (but not necessarily relaxed) configurations; here our determination includes a parameter to bound the maximum value of $C(v)$ for $v \in V := V(K_n)$. For $n \geq 1$ and $m \geq n-1$, let $L_{n,m}$ be the number of legal configurations satisfying $C(v) \leq m$ for each $v \in V$. For convenience, we also define $L_{0,m} := 1$ for all $m \geq 0$. The proof of the next result makes implicit use of Proposition 2.5.

Theorem 3.3 *For all $n \geq 1$ and $m \geq n-1$, we have*

$$L_{n,m} = (m-n+2)(m+2)^{(n-1)}. \tag{6}$$

Proof: We proceed by induction on n and m. Since $L_{0,m} := 1$, this satisfies (6). For each $m \geq 0$, we include in our base case $L_{1,m}$, which counts the number of legal configurations on K_1. By (5), the single node must contain at least zero chips. Thus, the number of chips occupying this node lies in the set $\{0, 1, \ldots, m\}$; so $L_{1,m} = m+1$, which satisfies (6). Finally, we observe that for $n \geq 2$, the symbol $L_{n,n-2}$ enumerates the legal configurations in which all n nodes contain at most $n-2$ chips; since (5) requires $C(w_n) \geq n-1$, we have $L_{n,n-2} = 0$ for all $n \geq 2$. This satisfies (6), and we include it in our base case.

Now fix $n \geq 2$ and $m \geq n-1$, and assume that (6) is valid for each $L_{n-k,m-1}$ with $k \in \{0, 1, \ldots, n\}$. To determine $L_{n,m}$, we let $k \in \{0, 1, \ldots, n\}$ count the number of nodes containing exactly m chips; there are $\binom{n}{k}$ ways to choose these k nodes. Consider the configuration on the remaining $n-k$ nodes of K_n. Focusing on these nodes, we know from Lemma 2.3 that $C|_{V(K_{n-k})}$ must be legal on K_{n-k}, with at most $m-1$ chips on each node. Since the number of such configurations is $L_{n-k,m-1}$, we have

$$L_{n,m} = \sum_{k=0}^{n} \binom{n}{k} L_{n-k,m-1}.$$

Now we apply our inductive hypothesis to simplify the sum:

$$\begin{aligned}
L_{n,m} &= \sum_{k=0}^{n} \binom{n}{k} ((m-1)-(n-k)+2)\,((m-1)+2)^{(n-k)-1} \\
&= \sum_{k=0}^{n} \binom{n}{k} \left(\frac{k}{n}[(m+1)-(m-n+1)] + (m-n+1)\right)(m+1)^{n-k-1} \\
&= \sum_{k=0}^{n} \left[\frac{k}{n}\binom{n}{k}(m+1) + \frac{n-k}{n}\binom{n}{k}(m-n+1)\right](m+1)^{n-k-1} \\
&= \sum_{k=1}^{n} \binom{n-1}{k-1}(m+1)^{n-k} + \sum_{k=0}^{n-1} \binom{n-1}{k}(m-n+1)(m+1)^{n-k-1} \\
&= (m-n+2) \sum_{k=0}^{n-1} \binom{n-1}{k}(m+1)^{(n-1)-k} \\
&= (m-n+2)((m+1)+1)^{n-1};
\end{aligned}$$

induction gives the result. □

Since $L(K_n) = L_{n,n-1}$, we obtain the immediate

Corollary 3.4 $L(K_n) = (n+1)^{n-1}$.

This last observation led us to the connection between burn-off games on complete graphs and the enumeration of spanning trees therein.

3.2 Connections with spanning trees

A direct proof that the number of relaxed legal configurations on K_n coincides with the number of spanning trees of K_{n+1}, without resort to Cayley's Formula, will yield a proof of the latter.

Theorem 3.5 *The number of relaxed legal configurations on K_n equals the number of spanning trees of K_{n+1}.*

Proof: We establish algorithmically injections between the set $\mathcal{R}$ of relaxed legal configurations on K_n and the set $\mathcal{S}$ of spanning trees of K_{n+1}. Define $A\colon \mathcal{R} \to \mathcal{S}$ via Algorithm 3.6 and $B\colon \mathcal{S} \to \mathcal{R}$ via Algorithm 3.7 (both below). Regarding these algorithms, *score* refers to a numeric label assigned to a node. Define $F\colon V(K_n) \to \mathbb{N}$ as the function that makes this assignment.

Our first algorithm injectively maps $\mathcal{R}$ to $\mathcal{S}$.

Algorithm 3.6

INPUT:	a complete graph K_n and relaxed legal configuration $C\colon V(K_n) \to \mathbb{N}$
OUTPUT:	a spanning tree of K_{n+1}
(1)	Let $V(K_n) = \{v_1, v_2, \ldots, v_n\}$.
(2)	Delete all edges from K_n and introduce a new node v_0.
(3)	Let $M_0 = (v_0)$ and $\overline{M}_0 = \{v_0\}$.
(4)	Let $F(v_0) = n-1$.
(5)	Let $Q_0 = 0$ and $i = 0$.
	Until all v_k have been included in some sequence M_i, do the following:
(6)	$i \leftarrow i+1$.
(7)	Let $Q_i = Q_{i-1} + \lvert\overline{M}_{i-1}\rvert$.
(8)	Let $M_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_t})$, for some $t \geq 1$, be the sequence (in increasing subscript order) of all nodes v for which $C(v) = F(u)$ for some $u \in \overline{M}_{i-1}$. Let $\overline{M}_i = \{x : x$ is an entry of $M_i\}$.
(9)	Add an edge from each v_k in $\overline{M}_i$ to the node $u \in \overline{M}_{i-1}$ for which $C(v_k) = F(u)$.
(10)	For each $j = 1, 2, \ldots, \lvert\overline{M}_i\rvert$, let $F(v_{i_j}) = n - Q_i - j$.

The purpose of the variables Q_i is to record the number of nodes that have been included in an earlier M_j. In fact,

$$Q_i = \left| \bigcup_{j=0}^{i-1} \overline{M}_j \right|.$$

Thus, in step (10), each score is given to exactly one node. Therefore, "the node" selected in step (9) is indeed unique.

Proof that A is well-defined: Algorithm 3.6 will fail if, during any iteration, $\overline{M}_i$ is empty, because no further edges can then be added in step (9). We demonstrate below that no $\overline{M}_i$ is ever empty, but assume for now that this is true. Step (9) adds an edge from each $v_k \in \overline{M}_i$ to a node that is already part of a single growing component of the subgraph of K_{n+1} being constructed. The algorithm continues until all nodes of K_{n+1} are members of some $\overline{M}_i$, so all nodes of K_{n+1} are eventually connected to the growing component. Note also that exactly n edges are created by step (9), one for each node except v_0. A spanning connected subgraph (of an $(n+1)$-node graph G) with n edges is necessarily a spanning tree of G; thus, the algorithm certainly constructs a spanning tree of K_{n+1}.

It remains to prove that no $\overline{M}_i$ is empty. We proceed by induction. It is clear in step (3) that $\overline{M}_0$ is not empty; suppose that $\overline{M}_{i-1}$ is not empty for some fixed $i > 0$. (By definition of $\overline{M}_{i-1}$, it is clear that $\overline{M}_{i-2}$ is also not empty.) As Algorithm 3.6 proceeds, the scores assigned to the nodes in step (10) descend from $n-1$. We must show that at least one value of $C(v_k)$, for $1 \le k \le n$, is large enough to equal the score of one of the nodes in $\overline{M}_{i-1}$. This will guarantee that $\overline{M}_i$ contains at least one element.

When, in step (8), the nodes are checked to see if they will be members of M_i, the algorithm inspects the scores assigned to the nodes in M_{i-1} (our induction hypothesis ensures that there are nodes in M_{i-1} to inspect). These scores were assigned (during the preceding iteration) in the following order:

$$n - Q_{i-1} - 1, n - Q_{i-1} - 2, \ldots, n - Q_{i-1} - |\overline{M}_{i-1}| = n - Q_i.$$

Thus, the lowest score assigned to a node of M_{i-1} is $n - Q_i$.

Since C is a legal configuration on K_n, we know that for each $\ell \in \{1, 2, \ldots, n\}$, at least $n - \ell$ nodes contain at least ℓ chips (see Corollary 3.2). Substituting $n - Q_i$ for ℓ, we see that our legal configuration C is such that at least Q_i nodes contain at least $n - Q_i$ chips.

Since $Q_i - 1$ nodes have been assigned scores (we subtract 1 because $v_0 \notin V(K_n)$), there is at least one unassigned node v_k containing at least $n - Q_i$ chips (*i.e.*, $C(v_k) \ge n - Q_i$); this number is at least as big as the lowest score found in M_{i-1}. We also know that $C(v_k)$ is one of the scores assigned to a node in M_{i-1}, for if $C(v_k)$ exceeded all of those scores, then v_k would have already been assigned to an earlier sequence. It follows by induction that no $\overline{M}_i$ is empty and, by our earlier remarks, that A is well-defined. □

Proof that A is an injection: Let $C\colon V(K_n) \to \mathbb{N}$ and $C^*\colon V(K_n) \to \mathbb{N}$ be two distinct relaxed legal configurations on K_n. Let $A(C) = T$ and $A(C^*) = T^*$. We will prove that A is an injection by showing that T and T^* must be distinct.

As $C \ne C^*$, Algorithm 3.6 must encounter $C(v_i) \ne C^*(v_i)$ for some $i \in \{1, 2, \ldots, n\}$. Choose i to index the earliest such encounter; since Algorithm 3.6 assigns the scores in order of decreasing value, i is the index witnessing $\max\{\max\{C(v_j), C^*(v_j)\} : 1 \le j \le n \text{ and } C(v_j) \ne C^*(v_j)\}$. Without loss of generality, we may suppose that $C^*(v_i) > C(v_i)$.

We will show that v_i has different neighbors in T and T^*, so that $T \ne T^*$. Suppose that as Algorithm 3.6 operates on C, the sequences constructed in step (8) are $M_1, M_2, \ldots, M_j$, and suppose that as it operates on C^*, the corresponding sequences are $M_1^*, M_2^*, \ldots, M_k^*$. Now suppose that $v_i \in \overline{M}_s$ (for some $s \in \{1, 2, \ldots, j\}$) and $v_i \in \overline{M}_t^*$ (for some $t \in \{1, 2, \ldots, k\}$). We distinguish two cases.

Case 1: $s = t$.

Since $C(v_i)$ and $C^*(v_i)$ are the earliest unequal entries considered, we have $\overline{M}_{s-1} = \overline{M}^*_{t-1}$. Since $C(v_i) \neq C^*(v_i)$, the node v_i must be assigned different neighbors, say x, y, from among the nodes of $\overline{M}_{s-1}$ and $\overline{M}^*_{t-1}$, respectively. Then the edge $\{v_i, x\}$ belongs to $E(T) \smallsetminus E(T^*)$, implying that $T \neq T^*$.

Case 2: $s \neq t$.

Algorithm 3.6 adds an edge between v_i and some node w^* in $\overline{M}^*_{t-1}$. Since $C(v_i)$ and $C^*(v_i)$ are the earliest unequal entries encountered, we know that $\overline{M}_{t-1} = \overline{M}^*_{t-1}$. But we have assumed that $C^*(v_i) > C(v_i)$, so $t < s$; this implies that $\overline{M}_{s-1} \neq \overline{M}^*_{t-1}$. Thus, the edge $\{v_i, w^*\}$ belongs to $E(T^*) \smallsetminus E(T)$, again implying that $T \neq T^*$.

□

Our second algorithm injectively maps $\mathcal{S}$ to $\mathcal{R}$.

Algorithm 3.7

INPUT:	a spanning tree T of K_{n+1}
OUTPUT:	a relaxed legal configuration $C\colon V(K_n) \to \mathbb{N}$ on K_n
(1)	Fix a node v_0 in K_{n+1}.
(2)	Label the remaining nodes $v_1, v_2, \ldots, v_n$.
(3)	Let $N_0 = (v_0)$, $\overline{N}_0 = \{v_0\}$, and $i = 0$.
	Until all v_k have been included in some sequence N_i, repeat steps (4) and (5):
(4)	$i \leftarrow i + 1$.
(5)	Define $N_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_t})$, for $t \geq 1$, as the sequence (in increasing subscript order) of all unassigned nodes that are neighbors in T of a node in N_{i-1}. Let $\overline{N}_i = \{x : x \text{ is an entry of } N_i\}$.
(6)	Define $N = (u_k)_{k=1}^{n+1}$ as the concatenation of all the N_i's, in their natural order.
(7)	For $k = 1, 2, \ldots, n$, set $F(u_k) = n - k$ (note that the $(n+1)^{st}$ entry is not assigned a score).
(8)	a. For each $i = 1, 2, \ldots, n$, the node v_i is an entry of some N_j, and is thus the neighbor in T of some $v_k \in \overline{N}_{j-1}$. b. Let $C(v_i) = F(v_k)$.

It's worth noting that in step (5) we are performing a breadth-first search (see, *e.g.*, [8]) from v_0 to determine the N_i's.

Proof that B is well-defined: Step (7) makes it clear that C is relaxed, since the maximum assigned score is $n - 1$. In order to show that C is legal, we demonstrate that for each $\ell \in \{1, 2, \ldots, n\}$, at least $n - \ell$ nodes contain at least ℓ chips (again, see Corollary 3.2). In step (7), a node v_k receives the score ℓ' after $(n-1) - \ell'$ other nodes have been assigned scores. As v_k has n neighbors in K_{n+1}, it will have

$n - ((n-1) - \ell') = \ell' + 1$ unlabeled neighbors in T. These are the only nodes that can be assigned C-values at most $F(v_k) = \ell'$ in step (8b). Thus, for $\ell' \in \{0, 1, \ldots, n-1\}$, at most $\ell' + 1$ nodes of K_n are assigned C-values at most ℓ'. Complementation and Corollary 3.2 imply that C is indeed legal. □

Proof that B is an injection: Let T and T^* be two distinct spanning trees of K_{n+1}. Let $V(K_{n+1}) = \{v_0, v_1, \ldots, v_n\}$. Let $B(T) = C$ and $B(T^*) = C^*$. We will prove that B is an injection by showing that the configurations C and C^* must be distinct.

For $v \in V(T)$, let $\Gamma_T(v)$ denote the set of neighbors of v that are assigned the value $F(v)$ in step (8b); in other words, $\Gamma_T(v)$ is the set of nodes adjacent to v, and one edge further from v_0, in T. Define $\Gamma_{T^*}(v)$ analogously.

Since $T \neq T^*$, we infer that $\Gamma_T(v) \neq \Gamma_{T^*}(v)$ for some $v \in V(K_{n+1})$. With N_i, N_i^* denoting the sequences constructed by step (5) for T, T^* respectively, let $r := \min\{i :$ there exists $v \in \overline{N}_i \cap \overline{N}_i^*$ with $\Gamma_T(v) \neq \Gamma_{T^*}(v)\}$. Choose any $v \in \overline{N}_r \cap \overline{N}_r^*$ with $\Gamma_T(v) \neq \Gamma_{T^*}(v)$. Step (8b) assigns the value $F(v)$ to all $x \in \Gamma_T(v)$ and the value $F^*(v)$ to all $y \in \Gamma_{T^*}(v)$. By the choice of r (as a minimum), we have $F(v) = F^*(v)$; by step (7), we know that this score is assigned exclusively to v. To receive this score in step (8b) (as a chip count), a node must be a member of either $\Gamma_T(v)$ or $\Gamma_{T^*}(v)$. Since $\Gamma_T(v) \neq \Gamma_{T^*}(v)$, we have $C \neq C^*$; thus, B is injective. □

Now that we have demonstrated injections between the set $\mathcal{R}$ of relaxed legal configurations on K_n and the set $\mathcal{S}$ of spanning trees of K_{n+1}, we have $|\mathcal{R}| = |\mathcal{S}|$, which finally completes the proof of Theorem 3.5. □

As noted at the start of this section, our results yield a new proof of Cayley's Formula, for combining Theorem 3.5 with Corollary 3.4 gives $|\mathcal{S}| = |\mathcal{R}| = L(K_n) = (n+1)^{n-1}$. We view this as somewhat of a curiosity because it is certainly not the most efficient published proof of this identity (see, *e.g.*, [18]). Nevertheless, it ties our results on burn-off games to spanning tree enumeration much as conventional chip-firing games have been likewise linked.

We intend to present further results on burn-off games from [20] in a future paper, for which the present article will form a foundation.

Acknowledgements

Our thanks go to the referee for the careful read, for sharing the reference [4], and especially for pointing out the connection between our burn-off game and the Biggs dollar game.

References

[1] P. Bak and C. Tang. Earthquakes as a self-organized critical phenomenon. *J. Geophys. Res.*, 94(B11):15635–15637, 1989. ISSN 2156-2202. doi: 10.1029/JB094iB11p15635. URL `http://dx.doi.org/10.1029/JB094iB11p15635`.

[2] M. Baker and F. Shokrieh. Chip-firing games, potential theory on graphs, and spanning trees. *J. Combin. Theory Ser. A*, 120(1):164–182, 2013. ISSN 0097-3165. doi: 10.1016/j.jcta.2012.07.011. URL `http://dx.doi.org/10.1016/j.jcta.2012.07.011`.

[3] B. Benson, D. Chakrabarty, and P. Tetali. G-parking functions, acyclic orientations and spanning trees. *Discrete Math.*, 310(8):1340–1353, 2010. ISSN 0012-365X. doi: 10.1016/j.disc.2010.01.002. URL `http://dx.doi.org/10.1016/j.disc.2010.01.002`.

[4] N. Biggs and P. Winkler. Chip-firing and the chromatic polynomial. Research Report LSE-CDAM-97-03, Centre for Discrete and Applicable Mathematics, London School of Economics, London, U.K., 1997. 10pp.

[5] N. L. Biggs. Chip-firing and the critical group of a graph. *J. Algebraic Combin.*, 9(1):25–45, 1999. ISSN 0925-9899. doi: 10.1023/A:1018611014097. URL http://dx.doi.org/10.1023/A:1018611014097.

[6] A. Björner and L. Lovász. Chip-firing games on directed graphs. *J. Algebraic Combin.*, 1(4):305–328, 1992. ISSN 0925-9899. doi: 10.1023/A:1022467132614. URL http://dx.doi.org/10.1023/A:1022467132614.

[7] A. Björner, L. Lovász, and P. W. Shor. Chip-firing games on graphs. *European J. Combin.*, 12(4):283–291, 1991. ISSN 0195-6698.

[8] J. A. Bondy and U. S. R. Murty. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, New York, 2008. ISBN 978-1-84628-969-9. doi: 10.1007/978-1-84628-970-5. URL http://dx.doi.org/10.1007/978-1-84628-970-5.

[9] A. Cayley. A theorem on trees. In *The Collected Mathematical Papers of Arthur Cayley*, volume 13, pages 26–28. Cambridge University Press, Cambridge, 1897. (Quart. J. Math. **23** (1889), 376–378).

[10] D. Dhar, P. Ruelle, S. Sen, and D.-N. Verma. Algebraic aspects of abelian sandpile models. *J. Phys. A*, 28(4): 805–831, 1995. ISSN 0305-4470. URL http://stacks.iop.org/0305-4470/28/805.

[11] A. Engel. The probabilistic abacus. *Educ. Stud. Math.*, 6(1):1–22, 1975. ISSN 00131954. URL http://www.jstor.org/stable/3482156.

[12] C. Godsil and G. Royle. *Algebraic graph theory*, volume 207 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2001. ISBN 0-387-95241-1; 0-387-95220-9. doi: 10.1007/978-1-4613-0163-9. URL http://dx.doi.org/10.1007/978-1-4613-0163-9.

[13] G. H. Hurlbert. The graph pebbling page. URL mingus.la.asu.edu/~hurlbert/pebbling/pebb.html. Accessed 5 March 2013.

[14] G. H. Hurlbert. A survey of graph pebbling. In *Proceedings of the Thirtieth Southeastern International Conference on Combinatorics, Graph Theory, and Computing (Boca Raton, FL, 1999)*, volume 139, pages 41–64, 1999.

[15] P. M. Kayll and D. Perkins. Combinatorial proof of an Abel-type identity. *J. Combin. Math. Combin. Comput.*, 70:33–40, 2009. ISSN 0835-3026.

[16] C. Merino. Chip firing and the Tutte polynomial. *Ann. Comb.*, 1(3):253–259, 1997. ISSN 0218-0006. doi: 10.1007/BF02558479. URL http://dx.doi.org/10.1007/BF02558479.

[17] C. Merino. The chip-firing game. *Discrete Math.*, 302(1-3):188–210, 2005. ISSN 0012-365X. doi: 10.1016/j.disc.2004.07.033. URL http://dx.doi.org/10.1016/j.disc.2004.07.033.

[18] J. W. Moon. Various proofs of Cayley's formula for counting trees. In F. Harary, editor, *A seminar on Graph Theory*, pages 70–78. Holt, Rinehart and Winston, New York, 1967.

[19] K. Nagel and M. Paczuski. Emergent traffic jams. *Phys. Rev. E*, 51(4):2909–2918, 1995. doi: 10.1103/PhysRevE.51.2909. URL http://link.aps.org/doi/10.1103/PhysRevE.51.2909.

[20] D. Perkins. *Investigations of a chip-firing game*. ProQuest LLC, Ann Arbor, MI, 2005. ISBN 978-0542-13672-6. URL http://proquest.umi.com/pqdlink?did=920935351&Fmt=7&clientId=79356&RQT=309&VName=PQD. Thesis (Ph.D.)–University of Montana.

[21] J. Spencer. Balancing vectors in the max norm. *Combinatorica*, 6(1):55–65, 1986. ISSN 0209-9683. doi: 10.1007/BF02579409. URL http://dx.doi.org/10.1007/BF02579409.

[22] D. Stassinopoulos and P. Bak. Democratic reinforcement: A principle for brain function. *Phys. Rev. E*, 51(5): 5033–5039, 1995. URL http://link.aps.org/doi/10.1103/PhysRevE.51.5033.

[23] G. Tardos. Polynomial bound for a chip firing game on graphs. *SIAM J. Discrete Math.*, 1(3):397–398, 1988. ISSN 0895-4801. doi: 10.1137/0401039. URL http://dx.doi.org/10.1137/0401039.