CHIP-FIRING GAMES, G-PARKING FUNCTIONS, AND AN EFFICIENT BIJECTIVE PROOF OF THE MATRIX-TREE THEOREM

FARBOD SHOKRIEH

Abstract. Kirchhoff's matrix-tree theorem states that the number of spanning trees of a graph G is equal to the value of the determinant of the reduced Laplacian of G. We outline an efficient bijective proof of this theorem, by studying a canonical finite abelian group attached to G whose order is equal to the value of same matrix determinant. More specifically, we show how one can efficiently compute a bijection between the group elements and the spanning trees of the graph. The main ingredient for computing the bijection is an efficient algorithm for finding the unique G-parking function (reduced divisor) in a linear equivalence class defined by a chip-firing game. We also give applications, including a new and completely algebraic algorithm for generating random spanning trees. Other applications include algorithms related to chip-firing games and sandpile group law, as well as certain algorithmic problems about the Riemann-Roch theory on graphs.

1. Introduction

1.1. **Overview.** Every graph G has a canonical finite abelian group attached to it. This group has appeared in the literature under many different names; in theoretical physics it was first introduced as the "abelian sandpile group" or "abelian avalanche group" in the context of self-organized critical phenomena ([4, 20, 21]). In arithmetic geometry, this group appeared as the "group of components" in the study of degenerating algebraic curves ([30]). In algebraic graph theory this group appeared under the name "Jacobian group" or "Picard group" in the study of flows and cuts in graphs ([3]). The study of a certain chip-firing game on graphs led to the definition of this group under the name "critical group" ([9, 10]).

The order of this group is equal to the value of the determinant of the reduced Laplacian of G (see, e.g. Lemma 2.3). We know from Kirchhoff's famous matrix-tree theorem that the value of the same determinant gives the number of spanning trees of the graph ([27]). So, one might wonder whether there is a nice and explicit bijection between the elements of the group and the spanning trees of the graph; existence of such a bijection would independently prove the matrix-tree theorem, and might have other algorithmic consequences. It is rather clear that such a bijection cannot be fully canonical, as that would imply a particular spanning tree is distinguished, and corresponds to the identity of the group. Therefore, one needs to make some choices to be able to write down a bijection. If one fixes a vertex q, then there is a canonical representative for each element of the group, called the *G-parking function* (based at q) or *q-reduced divisor* (see, e.g., [21, 19, 31, 5] , or Proposition 2.5). The main result of this paper is an algorithm to find this canonical representative efficiently. Once we have this, we can use one of the bijections in [12, 18, 16, 7] to find the corresponding spanning tree.

This easy-to-compute bijection from the "Jacobian group" to the set of spanning trees of the graph gives new insight into the matrix-tree theorem. It also may lead to new algorithms for graphs and

Date: April 12, 2009.

2000 *Mathematics Subject Classification.* 05C05, 05C50, 05C85.

I would like to thank Matthew Baker for recommending this problem to me, and for many helpful discussions. Thanks also to Richard Lipton, Prasad Tetali, and Arash Asadi for their helpful comments.

their spanning trees. For example, as an immediate corollary, if one picks a random element of the group - which is a trivial task - and constructs the bijection, then one gets a random spanning tree. This yields a new and completely algebraic approach for sampling a random spanning tree from the set of spanning trees of the graph. With this approach, it is very easy to sample multiple spanning trees with certain "joint" distribution. We believe there may be other algorithmic applications for this bijection.

The theory of G-parking functions relates nicely to a number of problems in theory of chip-firing games , and to the Riemann-Roch theory on finite graphs. As a result of our main algorithm, we are able to give efficient algorithms for a number of problems in these areas. The question of finding reduced divisors (G-parking functions) was first posed by Henrik Lenstra to Baker and Norine, in connection with Riemann-Roch theory on finite graphs in [5], through private communication. The problem of finding the "sandpile prediction" in polynomial-time was listed as an open problem by László Babai in [2] (§10.3). Our main algorithm also settles this open problem, as the sandpile predictions and critical configurations are, in a sense, dual to the reduced divisors; see Remark 3.14.

1.2. **Some related work.** The term "G-parking functions" was first introduced in [31]. The reason for this terminology is that they can be considered as a natural generalization of "parking functions". The theory of parking functions was first considered in connection with hash functions ([28]). The original problem was phrased in terms of cars and parking spots. The theory of parking functions has since been developed, with links many different areas including priority queues ([22]), representation theory ([24]), and noncrossing partitions ([32]). Although the explicit definition was first given in [31], the concept had appeared (sometimes in disguise) in many previous work (see, e.g.,[20, 21, 9, 11, 19]). The fact that G-parking functions provide a canonical representative for each element of the Jacobian appear implicitly in [21, 19], and explicitly in [31, 5].

The problem of giving explicit bijection between G-parking functions and spanning trees of graph is studied extensively in the literature (see, e.g., [12, 18, 16, 7]).

The relationship between chip-firing games and the Jacobian group is studied in [9, 10]. Some algorithmic aspects of the chip-firing games are studied in [34, 13, 33]; see §A.2 for a discussion on how [34] relates to our work.

The *Uniform Spanning Tree* (UST) problem has been extensively studied in the literature and there are two known types of algorithms: determinant based algorithms (e.g. [23, 17, 29]), and random walk based algorithms (e.g. [15, 1, 35]).

More related work is given in the paper.

The paper is structured as follows. In §2 we provide the relevant definitions and tools. In §3 we first state and prove Dhar's algorithm which can efficiently check whether a given divisor is reduced. Then we give our main algorithm for computing the reduced divisors and prove its correctness and efficiency. We then give the explicit and efficient bijection between the Jacobian group and the set of spanning trees of the graph. In §A we outline some applications of our main results, including a new approach to the *Uniform Spanning Tree* problem, efficient computation of the group law in the critical and sandpile group, efficient algorithm for the chip-firing game of Baker and Norine, and the relation to the Riemann-Roch theory on finite graphs.

2. Definitions and background

2.1. **Notation and Terminology.** Throughout this paper, a *graph* means a finite, unweighted multigraph with no loops. All graphs are assumed to be connected. For a graph G, the set of vertices is denoted by $V(G)$, and the set of edges is denoted by $E(G)$. Throughout this paper, n and m denote the number of vertices and edges, respectively.

Let $\mathrm{Div}(G)$ be the free abelian group generated by $V(G)$. One can think of elements of $\mathrm{Div}(G)$ as formal integer linear combination of vertices

$$\mathrm{Div}(G) = \{ \sum_{v \in V(G)} a_v(v) \ : \ a_v \in \mathbb{Z} \} \ .$$

By analogy with the algebraic curve case, elements $\mathrm{Div}(G)$ are called *Divisors* on G. For a divisor D, the coefficient a_v of (v) in D is denoted by $D(v)$.

$\mathcal{M}(G) = \mathrm{Hom}(V(G), \mathbb{Z})$ is, by definition, the abelian group consisting of all integer-valued functions on the vertices. For $A \subseteq V(G)$, $\chi_A \in \mathcal{M}(G)$ denotes the $\{0,1\}$-valued characteristic function of A. Note that $\{\chi_{\{v\}}\}_{v \in V(G)}$ generates $\mathcal{M}(G)$ as a $\mathbb{Z}$-module.

The *Laplacian operator* $\Delta : \mathcal{M}(G) \to \mathrm{Div}(G)$ is given by the formula

$$\Delta(f) = \sum_{v \in V(G)} \Delta_v(f)(v) \ ,$$

where

$$\Delta_v(f) = \sum_{\{v,w\} \in E(G)} (f(v) - f(w)) \ .$$

Let $\{v_1, \ldots, v_n\}$ be an ordering of $V(G)$. With respect to this ordering, the *Laplacian matrix* Q associated to G is the $n \times n$ matrix $Q = (q_{ij})$, where q_{ii} is the degree of vertex v_i, and $-q_{ij}$ $(i \neq j)$ is the number of edges connecting v_i and v_j. It is well-known (and easy to verify) that Q is symmetric, has rank $n-1$, and that the kernel of Q is spanned by $\mathbf{1}$, the all-one vector[1] (see, e.g., [8, 14]).

Using the ordering $\{v_1, \ldots, v_n\}$ of vertices, there exist isomorphisms between abelian groups $\mathrm{Div}(G), \mathcal{M}(G)$, and the $\mathbb{Z}$-module of $n \times 1$ column vectors having integer coordinates. Under these isomorphisms the Laplacian operator $\Delta : \mathcal{M}(G) \to \mathrm{Div}(G)$ coincides with the $\mathbb{Z}$-module homomorphism $Q : \mathbb{Z}^n \to \mathbb{Z}^n$. More specifically, if $[D]$ denotes the column vector corresponding to $D \in \mathrm{Div}(G)$, and $[f]$ denotes the column vector corresponding to $f \in \mathcal{M}(G)$, then $[\Delta(f)] = Q[f]$.

2.2. **Chip-firing games on graphs.** Following [5] we define an equivalence relation $\sim$ on the group $\mathrm{Div}(G)$ as follows.

Definition. For $D_1, D_2 \in \mathrm{Div}(G)$, $D_1 \sim D_2$ if and only if $D_1 - D_2$ is in the image of $\Delta : \mathcal{M}(G) \to \mathrm{Div}(G)$.

It is an easy exercise to show that this is indeed an equivalence relation. This equivalence relation is closely related to notion of "chip-firing games" played on the vertices of the graph (see, e.g., [13, 9, 10, 5]). For a given divisor $D \in \mathrm{Div}(G)$, the integer $D(v)$ can be viewed as the amount of *dollars* that is assigned to the vertex v, and D can be viewed as a *configuration* of the economy in the society modeled by the graph. If $D(v) < 0$ then the vertex v is said to be *in debt*[2]. A *move* in this "one-player game" consists of choosing a vertex and having it borrow one dollar from each of its neighbors, or give ("fire") one dollar to each of its neighbors.

The following easy lemma is proved in [5].

Lemma 2.1. *For $D_1, D_2 \in \mathrm{Div}(G)$, $D_1 \sim D_2$ if and only if starting from the configuration D_1 one can reach to the configuration D_2 , through a sequence of moves.*

[1]We again emphasize that G has no loops.

[2]Since negative assignments to vertices are permitted, we prefer to use the term "dollar" instead of "chip". This term was first used by Biggs in [9, 10].

2.3. **The Jacobian of a finite graph.** Consider the group homomorphism $\deg : \mathrm{Div}(G) \to \mathbb{Z}$ defined by $\deg(D) = \sum_{v \in V(G)} D(v)$. Denote by $\mathrm{Div}^0(G)$ the kernel of this homomorphism, consisting of *divisors of degree zero*. Denote by $\mathrm{Prin}(G)$ the image of the Laplacian operator $\Delta : \mathcal{M}(G) \to \mathrm{Div}(G)$.

Lemma 2.2. $\mathrm{Prin}(G) \subseteq \mathrm{Div}^0(G)$*, and both* $\mathrm{Prin}(G)$ *and* $\mathrm{Div}^0(G)$ *are free* $\mathbb{Z}$*-modules of rank* $n-1$*.*

Proof is easy and is left as an exercise.

As a corollary, the quotient group

$$\mathrm{Jac}(G) = \mathrm{Div}^0(G)/\mathrm{Prin}(G)$$

is well-defined and is a finite abelian group. Following [3], it is called the *Jacobian* of G. By the fundamental theorem of finitely generated abelian groups, this group is isomorphic to direct sum of finite cyclic groups.

If we choose an ordering of vertices it is easy to see that $\mathrm{Jac}(G)$ is the torsion part[3] of the cokernel of the $\mathbb{Z}$-module homomorphism $Q : \mathbb{Z}^n \to \mathbb{Z}^n$. The following result is an easy consequence of this observation.

Lemma 2.3. *The order of* $\mathrm{Jac}(G)$ *is equal to the value of the determinant of the reduced Laplacian of* G*. Invariant factors and generators of* $\mathrm{Jac}(G)$ *can efficiently be computed by finding the Smith normal form of the Laplacian matrix of the graph* G*.*

Remark 2.4. Of course the value of the determinant of the reduced Laplacian of G gives the number of spanning trees of G. This is Kirchhoff's *matrix-Tree Theorem* ([27]). We will not assume the knowledge of this famous theorem and, instead, we will use Lemma 2.3 to give a new bijective proof, with computable bijection, of the matrix-tree theorem in §3.5. Indeed, one can think of Lemma 2.3 as the generalization of Kirchhoff's theorem; the matrix-tree theorem only computes the order of the group and forgets the structure!.

2.4. **Reduced divisors or G-parking functions.** For $A \subseteq V(G)$ and $v \in A$, let $\mathrm{outdeg}_A(v)$ denote the number of edges of G having v as one endpoint and whose other endpoint lies in $V(G)\backslash A$.

Definition. Fix a vertex $q \in V(G)$. A function $f : V(G)\backslash\{q\} \longrightarrow \mathbb{Z}$ is called a q-reduced function (or G-parking function based at q) , if it satisfies the following two conditions:

- $f(v) \geq 0$ for all $v \in V(G)\backslash\{q\}$.
- For every non-empty set $A \subseteq V(G)\backslash\{q\}$, there exists a vertex $v \in A$ such that $f(v) < \mathrm{outdeg}_A(v)$.

This definition might seem unmotivated and rather technical for the moment. Its significance will only be clear after Proposition 2.5.

In this paper, following [5], we use the term *q-reduced*. Moreover, a divisor $D \in \mathrm{Div}(G)$ is called *q-reduced* if the map $v \mapsto D(v)$, defined for $v \in V(G)\backslash\{q\}$, is a G-parking function (relative to the base vertex q).

The significance of q-reduced divisors is mainly due to the following proposition. This was discovered by several different authors (see, e.g., [21, 19, 31, 5])

Proposition 2.5. *Fix a base vertex* $q \in V(G)$*. Then for every* $D \in \mathrm{Div}(G)$*, there exists a unique* q*-reduced divisor* $D' \in \mathrm{Div}(G)$ *such that* $D' \sim D$*.*

Remark 2.6. (i) None of the proofs in the literature gives to an efficient algorithm for computing such $D' \sim D$. This work is done in the current paper.

[3]The full cokernel group gives the full "Picard group" $\mathrm{Pic}(G) = \mathrm{Div}(G)/\mathrm{Prin}(G)$, which is isomorphic to $\mathbb{Z} \oplus \mathrm{Jac}(G)$

(ii) There is a close relationship between q-reduced divisors and q-critical configurations (or sandpile predictions); see Remark 3.14 for more detail.
(iii) It follows from Proposition 2.5 that the Jacobian of G induces a group structure on the set of q-reduced divisor, or on the set of q-critical configurations (see §A.2 for more about this observation). However, the group law on these sets will be somehow "artificial".

2.5. **Generalized inverses.** A matrix can have an inverse only if it is square and its columns (or rows) are linearly independent. But one can still get "partial inverse" of any matrix.

Definition. Let A be a matrix (not necessarily square). Any matrix L satisfying $ALA = A$ is called a generalized inverse of A.

It is somehow surprising that for every matrix A there exists at least one generalized inverse. In fact, more is true; any matrix has a unique *Moore-Penrose pseudoinverse*[4].

Let Q be the Laplacian matrix of a connected graph. Since it has rank $n-1$ it cannot have an inverse. But there are many ways to obtain generalized inverses. Two examples are given below:

Example 2.7. Fix an integer $1 \leq i \leq n$. Let Q_i be the $(n-1) \times (n-1)$ matrix obtained from Q by deleting i^{th} row and i^{th} column[5]. Then Q_i is a full rank matrix and has an inverse Q_i^{-1}. Let $L_{(i)}$ be the $n \times n$ matrix obtained from Q_i^{-1} by inserting a zero row after $(i-1)^{\text{th}}$ row and inserting a zero column after $(i-1)^{\text{th}}$ column. Then $L_{(i)}$ is a generalized inverse of Q. One can check

$$QL_{(i)} = I + R_{(i)}$$

where I is the identity matrix, and $R_{(i)}$ has -1 entries in i^{th} row and is zero everywhere else. As $R_{(i)}Q = 0$, we get $QL_{(i)}Q = Q$.

Example 2.8. Let J be the $n \times n$ all one matrix. Then $Q + \frac{1}{n}J$ is nonsingular and $Q^+ = (Q + \frac{1}{n}J)^{-1} - \frac{1}{n}J$ is a generalized inverse of Q. In fact it is the unique *Moore-Penrose pseudoinverse* of Q. It is easy to check $QQ^+ = Q^+Q = I - \frac{1}{n}J$.

3. Algorithm for computing reduced divisors

3.1. **Dhar's algorithm: the decision problem.** Given a divisor on the graph G, definition suggests that the verifier needs to check whether there exists a vertex $v \in A$ such that $D(v) < \text{outdeg}_A(v)$ for *all subsets* $A \subseteq V(G)\backslash\{q\}$. But, in fact, there is an elegant algorithm to do this quickly, which is called the Dhar's algorithm (after Dhar [20]). Both the algorithm and its proof of correctness are given here.

Algorithm. (Dhar's algorithm)
Input: A divisor $D \in \text{Div}(G)$, and a vertex $q \in V(G)$.
Output: TRUE if D is q-reduced, and FALSE if D is not q-reduced.
Let $A_0 = V(G)$ and $v_0 = q$.

(0) Check $D(v) \geq 0$ for all $v \in V(G)\backslash\{q\}$.
(1) For $1 \leq i \leq n-1$, let $A_i = A_{i-1} \setminus \{v_{i-1}\}$. Find a $v_i \in A_i$ with $D(v_i) < \text{outdeg}_{A_i}(v_i)$ - If no such v_i exists, output FALSE and stop.
(2) Output TRUE.

[4]Moore-Penrose pseudoinverse of A is a generalized inverse of A with three extra properties; see [6] for an extensive study of the subject.

[5]Q_i is called a reduced laplacian of the graph.

Analysis of the Dhar's algorithm. If the algorithm outputs FALSE, D cannot be q-reduced by definition. Now assume the algorithm outputs TRUE. We need to show that for an *arbitrary* subset $A \subseteq V(G)\backslash\{q\}$, there exists a vertex $v \in A$ such that $D(v) < \text{outdeg}_A(v)$. Let v_k be the minimal element of A, with respect to the ordering induced by the algorithm. Then $A \subseteq A_k$ and therefore $\text{outdeg}_{A_k}(v_k) \leq \text{outdeg}_A(v_k)$. But we know $D(v_k) < \text{outdeg}_{A_k}(v_k)$ by construction. Hence $D(v_k) < \text{outdeg}_A(v_k)$ and v_k is the desired element in A.

The Dhar's algorithm clearly runs in time $O(n^2)$. □

3.2. **The main algorithm: the search problem.** Fix a base vertex $q \in V(G)$. Here is the problem we want to solve.

Given a divisor $D \in \text{Div}(G)$, we want to find the unique q-reduced divisor $D' \sim D$.

A basic idea is to try to make the Dhar's algorithm to output TRUE; anytime it is about to output FALSE, "modify" the divisor to avoid it. However, there is a major problem with this naive idea; this process can take exponential time in the size of input. Of course, one also needs to figure out how to "modify" the divisor when the Dhar's algorithm is about to fail.

For us, the multigraph can be described by the number of vertices n and edges m in the graph. A divisor D can be presented by roughly $\sum_{v \in V(G)} \log(D(v))$ bits, which is easily seen to be less than $n \cdot \log(\deg(D))$ bits.

Remark 3.1. In fact, it is not too hard to find an algorithm that finds the q-reduced divisors in time that runs in polynomial in n, m, and $\deg(D)$ using the existing techniques appeared in [13, 33, 25]. But this is exponential in the "size of input"! Our algorithm will have a running time polynomial in m and n, and there will be some $\log(\deg(D))$-bit computations involved.

We will now give an algorithm for finding the reduced divisor and prove its correctness in §3.3. In §3.4 we give a deterministic upper bounds for its running time which show the algorithm is indeed efficient.

Notation. $\lfloor \cdot \rfloor$ denotes the *floor* function. For a vector X, we denote by $\lfloor X \rfloor$ a vector whose entries are the *floor* of entries of X. Recall from §2.1 that $\chi_A \in \mathcal{M}(G)$ denotes the $\{0,1\}$-valued characteristic function of $A \subseteq V(G)$. To simplify the presentation, we pick an ordering of vertices, and use $[\cdot]$ for column vector presentation of divisors and functions.

Algorithm. (Finding the reduced divisor)
Input: The Laplacian matrix Q of graph G. A divisor $D \in \text{Div}(G)$, and a vertex $q \in V(G)$.
Output: The unique q-reduced divisor $D' \sim D$.

Step 1. Find the generalized inverse $L = L_{(q)}$ of Q, as in Example 2.7. Compute the divisor $[D'] = [D] - Q\lfloor L_{(q)}[D] \rfloor$.

Step 2. Find a vertex $v \neq q$ with $D'(v) < 0$ and substitute $[D']$ with $[D'] + Q[\chi_{\{v\}}]$ (i.e. have v borrow one dollar from each of its neighbors). Repeat Step 2 until no such "negative" vertex $v \neq q$ exists anymore.

Step 3. Now let $A_0 = V(G)$ and $v_0 = q$.
(1) Set $i = 1$.
(2) While $i \leq n-1$, let $A_i = A_{i-1} \smallsetminus \{v_{i-1}\}$.
* If there exists a $v_i \in A_i$ such that $D'(v_i) < \text{outdeg}_{A_i}(v_i)$: increase i by 1 and go back to (2)
* Else, compute

$$k = \min_{\substack{v_i \in A_i \\ \text{outdeg}_{A_i}(v_i) \neq 0}} \lfloor D'(v_i)/\text{outdeg}_{A_i}(v_i) \rfloor$$

and replace $[D']$ with $[D'] - Q[k \cdot \chi_{A_i}]$ (i.e. have all vertices in A_i fire enough times to get $D'(v_i) < \text{outdeg}_{A_i}(v_i)$ for at least one vertex in A_i). Then go back to (1).

3.3. **Correctness of the algorithm.** Here is the high-level idea of the the steps in the algorithm

- The task of Step 1 is to bound the values of the divisor on vertices $v \neq q$. Bounding these values is important for bounding the running time of Step 2 and Step 3.
- The task of Step 2 is to make the values of the divisor on $v \neq q$ non-negative while still keeping the values bounded.
- The task of Step 3 is to "force" the divisor to pass the Dhar's algorithm.

Assume for the moment that the algorithm actually terminates and produces an output. It is easy to see that the output would be equivalent to D. Also, the output would certainly pass the Dhar's algorithm and therefore is q-reduced. Therefore, for the correctness of the algorithm, we only need to show that it always terminates and produces an output.

Step 1. The following proposition clarifies the task of Step 1.

Proposition 3.2. *If $[D'] = [D] - Q\lfloor L_{(q)}[D]\rfloor$, then $|D'(v)| < \deg(v)$ for all $v \neq q$. $D'(q)$ is such that $\deg(D') = \deg(D)$.*

Proof. Recall from Example 2.7 that $QL_{(q)} = I + R_{(q)}$, where I is the identity matrix and $R_{(q)}$ has -1 entries in q^{th} row and is zero everywhere else. Therefore $[D] = QL_{(q)}[D] + \deg(D) \cdot \mathbf{e}_q$, where $\mathbf{e}_q$ is the column vector which is zero everywhere except at position q, which is 1. Now

$$\begin{aligned}[D'] &= [D] - Q\lfloor L_{(q)}[D]\rfloor \\ &= Q(L_{(q)}[D] - \lfloor L_{(q)}[D]\rfloor) + \deg(D) \cdot \mathbf{e}_q \\ &= Q\mathbf{f} + \deg(D) \cdot \mathbf{e}_q\end{aligned}$$

where $\mathbf{f} = L_{(q)}[D] - \lfloor L_{(q)}[D]\rfloor$ is a vector with entries from the interval $[0, 1)$. It is now easy to show that the absolute values of the entries of $Q\mathbf{f}$ are bounded by the degree of the corresponding vertices. The second statement is trivial. □

Step 2. Perhaps it is not even clear that Step 2 will ever terminate, let alone in a polynomially bounded number of iterations. The following Proposition justifies this fact.

Proposition 3.3. *Given a divisor and a fixed vertex q, there exists a sequence of borrowing by vertices in debt (i.e. having negative numbers) which takes all vertices $v \neq q$ out of debt (i.e. there exists a sequence of vertices such that Step 2 of the algorithm terminates).*

Moreover, any sequence of borrowing by vertices with in debt terminates in the same number of steps, with the same terminal configuration.

Proof. This is follows from the proof of Lemma 5.3 and discussion in page 24 of [5].

□

Remark 3.4. Step 2 has very nice features; by Proposition 3.2, $|D'(v)| < \deg(v)$ for all $v \neq q$. So for any $v \neq q$ with $D'(v) < 0$ only one borrowing is needed to make the vertex positive. Moreover, the resulting positive number will clearly be less than the vertex degree again. This fact, together with Proposition 3.2 guarantees that the output of Step 2 satisfies $0 \leq D'(v) < \deg(v)$ for all $v \neq q$.

Step 3. In Step 3, the algorithm forces the divisor to pass the Dhar's algorithm. If for a subset A_i no $v_i \in A_i$ with $D'(v_i) < \text{outdeg}_{A_i}(v_i)$ exists (i.e. a counter-example for being q-reduced is

found), the algorithm computes

$$k = \min_{\substack{v_i \in A_i \\ \mathrm{outdeg}_{A_i}(v_i) \neq 0}} \lfloor D'(v_i)/\mathrm{outdeg}_{A_i}(v_i) \rfloor \ . \tag{3.5}$$

Since

$$k \leq D'(v_i)/\mathrm{outdeg}_{A_i}(v_i)$$

or

$$0 \leq D'(v_i) - k \cdot \mathrm{outdeg}_{A_i}(v_i)$$

for all $v_i \in A_i$ with $\mathrm{outdeg}_{A_i}(v_i) \neq 0$, replacing $[D']$ with $[D'] - Q[k \cdot \chi_{A_i}]$ will not make any vertex negative. Let $w \in A_i$ be the vertex that achieves the equality (3.5). Then

$$D'(w)/\mathrm{outdeg}_{A_i}(w) - 1 < k$$

or

$$D'(w) - k \cdot \mathrm{outdeg}_{A_i}(w) < \mathrm{outdeg}_{A_i}(w) \ .$$

This means that replacing $[D']$ with $[D'] - Q[k \cdot \chi_{A_i}]$ will result in $D'(w) < \mathrm{outdeg}_{A_i}(w)$. Algorithm then restarts the Dhar algorithm.

It is easy to see Step 3 eventually terminates; as vertex q never "fires", it should eventually stop receiving from its neighbors at some point. This, in turn, means that its neighbors will stop firing at some point and therefore, eventually, will stop receiving from their neighbors. Iterating this argument shows that the whole process will stop at a finite number of steps.

3.4. **Bounds on the total number of chip-firing moves.** It seems to be a hard problem to find a good estimate of the running time of the algorithm, as it depends on the structure of the graph and the given divisor in a rather complicated way. In practice, the algorithm seems to run much faster than what is predicted by the upper bound given in this section. One reason for this is that the technique is rather blind to the algorithm itself. This point will be clear shortly.

The following lemma is a generalization of the standard Cauchy-Schwarz inequality which can be proved similarly.

Lemma 3.6. *For any positive semidefinite matrix A with largest eigenvalue η, and for all vectors x and y,*

$$|x^T A y| \leq \eta \|x\|_2 \|y\|_2$$

Let $A = L_q$ as in Example 2.7. Non-zero eigenvalues of L_q are the reciprocal of eigenvalues of the reduced Laplacian matrix Q_q. So $L_{(q)}$ is positive semidefinite and its the largest eigenvalue is $1/\lambda_2$, where λ_2 is the smallest eigenvalue of Q_q.

Let $\mathbf{y} = Q\mathbf{x}$. Multiplying both sides by L_q, we get $L_q\mathbf{y} = (I + R^T_{(q)})\mathbf{x}$, or

$$L_q\mathbf{y} = \mathbf{x} - \mathbf{x}(q)\mathbf{1} \tag{3.7}$$

where $\mathbf{1}$ denotes the all-one vector. Then $\mathbf{1}^T L_q \mathbf{y} = \sum_i \mathbf{x}(i) - n \cdot \mathbf{x}(q)$. Now using Lemma 3.6

$$|\sum_i \mathbf{x}(i) - n \cdot \mathbf{x}(q)| \leq \frac{1}{\lambda_2} \|\mathbf{1}\|_2 \|\mathbf{y}\|_2 = \frac{\sqrt{n}}{\lambda_2} \|\mathbf{y}\|_2$$

In particular, if $\mathbf{x}(q) = 0$ we have

$$|\sum_i \mathbf{x}(i)| \leq \frac{\sqrt{n}}{\lambda_2} \|\mathbf{y}\|_2 \tag{3.8}$$

In fact something stronger is true.

Proposition 3.9. *Let $\mathbf{y} = Q\mathbf{x}$ where Q is the Laplacian matrix of the graph G. If $\mathbf{x}(q) = 0$ then*

$$|\sum_i \mathbf{x}(i)| \leq \frac{\sqrt{n}}{\lambda_2}\|\mathbf{y}\|_2'$$

where $\|\mathbf{y}\|_2' = \sqrt{\sum_{i\neq q}\mathbf{y}(i)^2}$ (i.e. one can assume $\mathbf{y}(q) = 0$ when computing $\|\mathbf{y}\|_2$ in inequality (3.8)).

Proof. This follows from the above discussion, together with the following observation; in (3.7), since the q^{th} column of $L_{(q)}$ is zero, one can freely change the value of $\mathbf{y}(q)$. Choosing $\mathbf{y}(q) = 0$ we are done.

□

If $D_1 \sim D_2$ via a sequence of moves that q never participates, then there exists a vector $\mathbf{x}$ with $\mathbf{x}(q) = 0$, such that $[D_1] - [D_2] = Q\mathbf{x}$. Moreover, if all moves are borrowing or all moves are firings, then all $\mathbf{x}(i)$'s all the same sign, and $|\sum_i \mathbf{x}(i)|$ precisely counts the number of vertex moves. By Proposition 3.9 an upper bound on the number of vertex moves is given by $|\sum_i \mathbf{x}(i)| \leq \frac{\sqrt{n}}{\lambda_2}\|[D_1] - [D_2]\|_2'$. Since $\|\cdot\|_2 \leq \|\cdot\|_1$ for (finite-dimensional) vectors we have the following.

Proposition 3.10. *Assume D_2 is obtained from D_1 with a sequence of moves so that*

- *q never makes a move.*
- *all moves are of the same type; vertices only fire or only borrow throughout the process.*

Then an upper bound on the number of vertex moves is given by

$$\frac{\sqrt{n}}{\lambda_2}(\|[D_1]\|_1' + \|[D_2]\|_1')$$

where $\|\mathbf{y}\|_1' = \sum_{i\neq q}|\mathbf{y}(i)|$.

Corollary 3.11. *Each of Step 2 and Step 3 in the algorithm terminates in at most $\frac{4\sqrt{n}m}{\lambda_2}$ vertex moves.*

Proof. The input and output of Step 2 satisfy $|D'(v)| < \deg(v)$ (for $v \neq q$) by Proposition 3.2 and Remark 3.4. Therefore $\|\cdot\|_1'$ of both input and output is less than $\sum_{v\neq q}|\deg(v)| < 2m$. Since vertices only borrow in Step 2 and q never makes a move, Proposition 3.10 applies. Step 3 follows similarly because the input and output of Step 3 satisfy $|D'(v)| < \deg(v)$ (for $v \neq q$) by Remark 3.4 and definition of q-reduced divisor. Vertices only fire in Step 3 and q never makes a move. □

Remark 3.12. (i) Step 3 in the algorithm performs many moves at the same time (e.g., set firings can be considered as combination vertex firings). So the actual number of iterations in Step 3 is, in practice, a fraction of what is given in Corollary 3.11. Similarly, it is possible to modify Step 2 to have combination moves. We omit the details here.

(ii) In Step 1, computing the generalized inverse $L_{(q)}$ is the expensive part, which takes time $O(n^\omega)$, where $\omega \leq 3$ is the exponent for matrix multiplication. But we only need to compute it once and save it. The rest of Step 1 takes $O(n^2)$ operations.

Remark 3.13. It is possible to give bound on the running time that involve the diameter or the maximum effective resistance instead of algebraic connectivity, using the techniques appeared in [33, 25]. We omit this discussion here.

3.5. **Efficient bijective proof of the matrix-tree theorem.** By Lemma 2.3 the order of $\mathrm{Jac}(G)$ is equal to the value of the determinant of the reduced Laplacian of G. In order to have a bijective proof of the matrix-tree theorem, one needs to find an explicit bijection between the elements of the group $\mathrm{Jac}(G)$ and the spanning trees of G. It is rather clear that such a bijection cannot be fully canonical, as that would imply a particular spanning tree is distinguished and corresponds to the identity! Therefore, one needs to make some choices to write down a bijection.

By definition $\mathrm{Jac}(G) = \mathrm{Div}^0(G)/\mathrm{Prin}(G)$. So the Jacobian of G is the set of equivalence classes of divisors of degree zero. We know by Proposition 2.5, once we fix a vertex q, there is a unique representative in each class that is q-reduced.

Now, one can ask again whether there is a nice and explicit bijection between these q-reduced divisors and spanning trees of G. It turns out to have such a bijection, one needs to make a few more choices, for example choose an ordering on the edges of the graph. The problem of giving explicit bijection between reduced divisors (G-parking functions) and spanning trees of graph is studied extensively in the literature (see, e.g., [12, 18, 16, 7]). For our work, we use the bijection given by Cori and Le Borgne in [18]. In fact, a variation of their bijection algorithm can be viewed as the Dhar algorithm with some extra features. The running time of the algorithm is then roughly the same as the running time of the Dhar's algorithm. We omit further details in this paper.

Remark 3.14. Some of the bijections in the literature are given between the q-critical configurations (or sandpile predictions) and spanning trees. For a fixed vertex q, q-critical configurations provide another set of unique representatives for equivalence classes of divisors (see, e.g., [9, 10]). There is a rather simple relationship between reduced and critical divisors; they add up to $\deg(v) - 1$ for any $v \neq q$, and their value on q is forced by the degree of the divisor (see [21, 11, 19, 31, 5]).

We have shown in §3.2 that this unique representative can be computed efficiently. Therefore we have the following "constructive and efficient" version of Kirchhoff's matrix-tree theorem.

Theorem 3.15. *Size of the group* $\mathrm{Jac}(G)$ *is equal to the number of spanning trees of* G*. Therefore the value of the determinant of the reduced Laplacian of* G *gives the number of spanning trees of* G*. Moreover, there exists an efficiently computable bijection between elements of* $\mathrm{Jac}(G)$ *and spanning trees of of* G*.*

Appendix A. Some applications

A.1. **Uniform spanning trees.** First application of our results is to give a new, completely algebraic method for picking uniform spanning trees of a given graph G. The running time of this algorithm is bounded deterministically. Proving that the output of the algorithm is indeed (uniformly) random comes for free. The new algorithm is taking full advantage of the Jacobian of the graph which, as we observed in §2.3, is a natural group attached to the set of spanning trees of the graph.

Here is the algorithm for picking a (uniformly) random spanning tree. Proof of correctness is trivial.

Algorithm. (Picking a uniform spanning tree)
Input: A graph G.
Output: A uniform spanning tree of G.

(1) Fix an arbitrary vertex $q \in V(G)$, and an arbitrary ordering on $E(G)$.
(2) Compute the invariant factors $\{n_1, \ldots, n_s\}$ and the corresponding generators $\{\mathbf{g}_1, \ldots, \mathbf{g}_s\}$ for $\mathrm{Jac}(G)$ using a Smith normal form algorithm on Q ($\mathbf{g}_i$'s are presented by divisors of degree zero).

(3) For $1 \leq i \leq s$, pick a random integer $0 \leq a_i \leq n_i - 1$, and compute the divisor $D = \sum_{i=1}^{s} a_i \mathbf{g}_s$.
(4) Use Algorithm 3.2 to find the unique q-reduced divisor in the equivalence class of D.
(5) Use the Cori and Le Borgne algorithm in [18] (or its variation) to find the corresponding spanning tree.

To our knowledge, the fastest Smith normal form algorithm given to date is given in [26], which runs in $(n^{2.697263} \log \|Q\|)^{1+o(1)}$, where $\|Q\|$, for our application, means the maximal degree of a vertex. Note that for repeated sampling, one can perform steps (1) and (2) only once, and save invariant factors and generators. Step (3) takes $O(\frac{\sqrt{n}m}{\lambda_2})$ chip-firing moves. Step 5 can be done in less than $O(n^2)$.

With this algorithm, unlike previous methods, it is very easy to sample multiple spanning trees with certain joint distribution. For example, very few random bits are required to generate the pairwise spanning trees. We omit the details here.

The bound on the running time of our algorithm does not beat the current best known running time $O(n^\omega)$ of [17]. For repeated generation of spanning trees, it is not clear how our running time $O(\sqrt{n}m/\lambda_2)$ compares with $O(n^\omega)$. We are optimistic that there is room for improvement in Algorithm 3.2 and/or its running time analysis. Once again we emphasize that the techniques used for bounding the running time of Algorithm 3.2 is blind to the algorithm itself, and in practice the algorithm seems to run much faster than what is predicted by the upper bound.

A.2. **The group law on reduced divisors or critical configurations.** Fix a vertex q. $\mathrm{Jac}(G)$ induces a group structure on the set of q-reduced divisors (G-parking functions) or q-critical divisors (or sandpile predictions) of G. The latter is called *critical group* (or sandpile group) of G. The group law requires to first add the two given divisors as elements of $\mathrm{Div}(G)$, and then find the unique q-reduced or q-critical divisor equivalent to the addition result. Clearly our algorithm can be used to perform the group law. A different approach for performing the group operation is given in [34]. Our algorithm seem to be faster, and our proof is significantly simpler and more algebraic.

A.3. **Algorithm for the chip-firing game of Baker and Norine.** The following Proposition is implicit in the proof of Theorem 3.3 in [5]. Proof of part (a) is, of course, trivial.

Proposition A.1. *Assume the chip-firing game of Baker and Norine starts with a configuration $D \in \mathrm{Div}(G)$, and let $D' \sim D$ be the corresponding q-reduced divisor.*

(a) *If $D'(q) \geq 0$ then there exists a wining strategy, and D' is a wining configuration.*
(b) *If $D'(q) < 0$ then no winning strategy exists.*

As a corollary, if one computes the q-reduced divisor associated to a given configuration, then one can efficiently decide whether there exists a wining strategy or not. Moreover, it is a simple linear algebra exercise to find the moves for the winning strategy, once the initial configuration and the wining configuration is known.

A.4. **Checking whether $r(D) \geq 0$.** Baker and Norine prove a Riemann-Roch theorem for finite graphs in [5]. To any divisor $D \in \mathrm{Div}(G)$ they associate an integer, called rank, $r(D) \geq -1$. We refer to their paper for details. The following lemma follows from the proof of Theorem 3.3 in [5].

Lemma A.2. *$r(D) \geq 0$ if and only if the unique q-reduced divisor equivalent to D has non-negative number on q.*

In fact, we can use this lemma and our algorithm to check whether $r(D) \geq c$ for any "constant" c. Question of finding reduced divisors was first posed by Henrik Lenstra to Baker and Norine through personal communication. This connection with Riemann-Roch theory was the original motivation of the author to work on this problem.

References

[1] D. J. Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM J. Discrete Math.*, 3(4):450–465, 1990.

[2] L. Babai. Lecture notes on sandpile model. *REU 2005, Discrete Mathematics*, 2005. Notes available at http://people.cs.uchicago.edu/ laci/REU05/notes/Jul11/10.pdf.

[3] R. Bacher, P. de la Harpe, and T. Nagnibeda. The lattice of integral flows and the lattice of integral cuts on a finite graph. *Bull. Soc. Math. France*, 125(2):167–198, 1997.

[4] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality. *Phys. Rev. A (3)*, 38(1):364–374, 1988.

[5] M. Baker and S. Norine. Riemann-Roch and Abel-Jacobi theory on a finite graph. *Adv. Math.*, 215(2):766–788, 2007.

[6] A. Ben-Israel and T. N. E. Greville. *Generalized inverses*. CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC, 15. Springer-Verlag, New York, second edition, 2003. Theory and applications.

[7] B. A. Benson and P. Tetali. Parking functions and acyclic orientations of graphs. 2008. Preprint available at `arXiv:0801.1114`.

[8] N. Biggs. *Algebraic graph theory*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, second edition, 1993.

[9] N. Biggs. Algebraic potential theory on graphs. *Bull. London Math. Soc.*, 29(6):641–682, 1997.

[10] N. Biggs. Chip-firing and the critical group of a graph. *J. Algebraic Combin.*, 9(1):25–45, 1999.

[11] N. Biggs. The Tutte polynomial as a growth function. *J. Algebraic Combin.*, 10(2):115–133, 1999.

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

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

[14] B. Bollobás. *Modern graph theory*, volume 184 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1998.

[15] A. Broder. Generating random spanning trees. pages 442–447, Oct-1 Nov 1989.

[16] D. Chebikin and P. Pylyavskyy. A family of bijections between G-parking functions and spanning trees. *J. Combin. Theory Ser. A*, 110(1):31–41, 2005.

[17] C. J. Colbourn, W. J. Myrvold, and E. Neufeld. Two algorithms for unranking arborescences. *J. Algorithms*, 20(2):268–281, 1996.

[18] R. Cori and Y. Le Borgne. The sand-pile model and Tutte polynomials. *Adv. in Appl. Math.*, 30(1-2):44–52, 2003. Formal power series and algebraic combinatorics (Scottsdale, AZ, 2001).

[19] R. Cori, D. Rossin, and B. Salvy. Polynomial ideals for sandpiles and their Gröbner bases. *Theoret. Comput. Sci.*, 276(1-2):1–15, 2002.

[20] D. Dhar. Self-organized critical state of sandpile automaton models. *Phys. Rev. Lett.*, 64(14):1613–1616, Apr 1990.

[21] A. Gabrielov. Abelian avalanches and Tutte polynomials. *Phys. A*, 195(1-2):253–274, 1993.

[22] J. D. Gilbey and L. H. Kalikow. Parking functions, valet functions and priority queues. *Discrete Math.*, 197/198:351–373, 1999. 16th British Combinatorial Conference (London, 1997).

[23] A. Guénoche. Random spanning tree. *J. Algorithms*, 4(3):214–220, 1983.

[24] M. D. Haiman. Conjectures on the quotient ring by diagonal invariants. *J. Algebraic Combin.*, 3(1):17–76, 1994.

[25] A. E. Holroyd, L. Levine, K. Mészáros, Y. Peres, J. Propp, and D. B. Wilson. Chip-firing and rotor-routing on directed graphs. In *In and out of equilibrium. 2*, volume 60 of *Progr. Probab.*, pages 331–364. Birkhäuser, Basel, 2008.

[26] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Comput. Complexity*, 13(3-4):91–130, 2004.

[27] G. Kirchhoff. Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt wird. *Ann. Phys. Chem.*, (72):497–508, 1847.

[28] A. G. Konheim and B. Weiss. An occupancy discipline and applications. *SIAM Journal on Applied Mathematics*, 14(6):1266–1274, 1966.

[29] V. G. Kulkarni. Generating random combinatorial objects. *J. Algorithms*, 11(2):185–207, 1990.

[30] D. Lorenzini. Arithmetical graphs. *Math. Ann.*, 285(3):481–501, 1989.

[31] A. Postnikov and B. Shapiro. Trees, parking functions, syzygies, and deformations of monomial ideals. *Trans. Amer. Math. Soc.*, 356(8):3109–3142 (electronic), 2004.

[32] R. P. Stanley. Parking functions and noncrossing partitions. *Electron. J. Combin.*, 4(2):Research Paper 20, approx. 14 pp. (electronic), 1997. The Wilf Festschrift (Philadelphia, PA, 1996).

[33] G. Tardos. Polynomial bound for a chip firing game on graphs. *SIAM J. Discrete Math.*, 1(3):397–398, 1988.
[34] J. van den Heuvel. Algorithmic aspects of a chip-firing game. *Combin. Probab. Comput.*, 10(6):505–529, 2001.
[35] D. B. Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 296–303, New York, 1996. ACM.

Georgia Institute of Technology, Atlanta, Georgia 30332-0160, USA
E-mail address: shokrieh@math.gatech.edu - farbod@ece.gatech.edu