CHIP-FIRING ON GRAPHS: STABILITY, THE DOLLAR GAME, AND THE TUTTE POLYNOMIAL

by

James Dylan Douthitt, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Mathematics
May 2019

Committee Members:

Anton Dochtermann, Chair

Eugene Curtin

Suho Oh

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

ACKNOWLEDGMENTS

Thank you to my advisor, Dr. Anton Dochtermann, for his assistance, guidance, and patience throughout this thesis process as well as my mathematical studies as a whole.

Thank you to Dr. Eugene Curtin for his instruction early in my mathematics career and honest guidance throughout.

Thank you to Dr. Suho Oh for his knowledge and assistance on many of the concepts we talk about here.

Thank you to the department of mathematics faculty involved in the algebra and combinatorics seminar for the time spent exploring the material on which I would eventually write this paper.

Thank you to Dr. Raymond Treinen for his encouragement, support, and mentorship toward my graduate studies.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure **Page**

I. INTRODUCTION

Chip-firing is a dynamical system defined by placing some number of 'chips' on the vertices of a graph and distributing them according to a simple rule. We will be looking at the use of chip-firing on graphs as a combinatorial structure and some developments since 1991. Versions of the game have appeared in a variety of contexts and we will look at some of the major results.

An early version of these games appeared in a couple of papers [14],[15] written in 1977 and 1986 respectively by Joel Spencer. In these papers he wrote of a two-person game played on vector spaces, where each person would take turns making moves and the goal of the game was to get the vector as close to zero as possible. He called this the 'balancing game.' In the latter paper he began thinking of the vectors as extended graphical paths. This gave way to the series of papers that we will explore.

We will begin the thesis by defining the basic structures that we will be using. First we will define the object we will be operating on, graphs, and discuss some basic properties of them such as the useful matrices and deletion/contraction. We then move to defining how chip-firing works and how exactly we will be referring to various aspects of our games. Then mixing the two we return to the graph to talk about the Laplacian and how it impacts our chip-firing game. We end the background material with one of the most notable and known results in graph theory, Kirchhoff's matrix-tree theorem. Throughout this paper much of the notation and language will follow that of the book [7] recently written by Scott Corry and David Perkinson and published in 2018. A reader with a strong knowledge of graph theory and chip-firing can safely skip this section.

In the next section, we look at the classical chip-firing game originally

introduced in 1991 by Björner, Lovász, and Shor [6]. Here we allow only a non-negative number of chips and using this they show that the long-term behavior of our games fall in to three possibilities dictated by the total number of chips in play. More precisely the main theorem is:

Theorem 6. (Björner, Lovász, Shor [6], 1991)
Let G be a graph with V vertices and E edges. Then legal games can be categorized as follows:

(a) If c is a configuration with $\deg(c) > 2E - V$ then the game is infinite, and each vertex is fired an infinite number of times.

(b) If $N \in \mathbb{Z}$ with $E \leq N \leq 2E - V$ then there exists an initial configuration of degree N guaranteeing finite termination and also a configuration of degree N guaranteeing an infinite game.

(c) If c is a configuration with $\deg(c) < E$ then the game is finite.

In the finite case there is an 'abelian' property guaranteeing that the game will terminate with the same final position after the same number of moves, regardless of what order the moves occur. Here we will spend some time setting language that is used in the context of this game. We will augment how we use this specific language throughout the paper as to satisfy the need of each game we play.

We will then move to the more algebraic aspects of chip-firing including a group structure present in the set of 'stable' and 'recurrent' configurations. Much of this work is originally due to Biggs [5]. Here we will consider a variation of the game discussed in the previous section where now we introduce the notion of a fixed 'root' vertex and explore what was originally called the 'dollar game.' This

allows us to look at the graph as an economic model of sorts with a 'bank.' An important result of [5] is the following:

Theorem 9. (Biggs [5], 1999)
The set $K(G)$ of critical configurations on a connected graph G is in bijective correspondence with the abelian group $\ker \sigma / \operatorname{Im} \mathcal{L}$.

Continuing in this general area we discuss some other notions of chip-firing including 'superstable' configurations associated with the non-root set of vertices. From these we will present various bijective type arguments including results relating to the number of spanning trees of the underlying graph. This opens the door to many other bijections between the underlying combinatorial objects that is well explored in other literature. One of the main tools in this context is known as Dhar's algorithm. This algorithm is useful to check if a configuration is superstable, as well as play the game more efficiently.

A related perspective of chip-firing, which we refer to as the dollar game, involves placing any integer value of chips on all vertices and passing chips attempting to get all players out of debt. We discuss various examples, including an in-depth analysis of how the game plays out on trees and cycles. This leads us to a generalization of 'winnable' that is made precise in the Riemann-Roch theorem for graphs, due to Baker and Norine [4]. Here we define the 'rank' of a configuration. The 'genus' of the graph plays an important role. The following is the main theorem from [4]:

Theorem 18. (Baker, Norine [4], 2007)

Let c be a configuration on a graph G, free of loops, with genus $g = E - V + 1$, and canonical configuration K. Then,

$$r(c) - r(K - c) = deg(c) + 1 - g.$$

We close the results of the paper by discussing a connection between chip-firing and the Tutte polynomial. The Tutte polynomial is a well-studied invariant of the graph characterized in many ways including a deletion-contraction property. We discuss a theorem due to Merino [12] which provides a connection between the superstable configurations of a graph and certain coefficients of the Tutte polynomial. This theorem relates states of our game to the activity of spanning trees. The version of Merino's theorem given in this paper is slightly different from the original, it in fact comes from the book [7]. An equivalent definition of the Tutte polynomial allows us to interpret the degree of the superstable configurations in terms of 'activity' of spanning trees.

We end with some final remarks about various topics not explored in this thesis as well as further research on these topics.

II. DEFINITIONS AND BASIC OBJECTS OF STUDY

Graphs

A *multigraph* is a pair $G = (V, E)$, where V is a set of vertices, often called nodes, and E is a set of edges between vertices, often thought of as unordered pairs of vertices. A multigraph differs from a simple graph by allowing for any pair of vertices $v_i, v_j \in V$, the edge between them may appear multiple times representing multiple edges connecting the same pair of vertices. We will often write $v_i v_j$ to denote an edge $\{v_i, v_j\}$ for convenience. Two vertices v_i and v_j are said to be *adjacent* if $\{v_i, v_j\} \in E$. A vertex v is said to be *incident* to all edges containing v. The *degree* of a vertex v, denoted $\deg(v)$, is equal to the total number of edges incident to v. Frequently we may denote $V = |V|$ and $E = |E|$ whenever the context is appropriate. A graph G is said to be *finite* if both $V < \infty$ and $E < \infty$. A graph is said to be *connected* if for any two vertices $v_i, v_j \in V$, there exists a sequence of edges joining v_i to v_j.

At some point we may want to differentiate certain edges in the graph by characteristics of the edges. A *bridge* or *isthmus* is an edge that, if removed, would disconnect the graph. A *loop* is an edge that connects a vertex to itself. Unless otherwise stated, graphs for the purpose of this paper will be considered to be finite, connected, undirected multigraphs, free of loops.

We have a few ways to encode graphs in terms of matrices, each with its own value to our system. There are three matrices we will use here, all derived from the graph. First is the degree matrix, $D(G)$. $D(G)$ is defined to be a $|V| \times |V|$ diagonal matrix in which each of the diagonal entries are the degrees of the vertex corresponding to that entry, $d_{ii} = \deg(v_i)$. The second is the adjacency matrix, $A(G)$, defined also as a $|V| \times |V|$ matrix where each entry a_{ij} counts the number

of edges between v_i and v_j. Third we define the incidence matrix, $B(G)$, to be a $|V| \times |E|$ matrix with each column representing an edge and the entries are 1 if the vertex is an endpoint of the edge and 0 if it is not. For a directed graph, the entries in this matrix would be -1 if the edge leaves the vertex, 1 if the edge enters the vertex, and 0 otherwise. For the purposes of this paper the incidence matrix will have an arbitrary orientation so that each of the columns sum to 0. Another way to think about the incidence matrix is as the boundary map of G where we consider G as an (oriented) 1-dimensional simplex. The following figure shows a graph and each of these corresponding matrices.

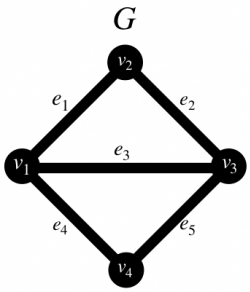


Figure 1: Basic Graph with Matrices

$$D(G) = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad A(G) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \quad B(G) = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix}$$

When applying mathematics to a graph one of the most used tools is the idea of edge *deletion* and *contraction* denoted $G\backslash e$ and G/e respectively. Deletion is just that, simply delete the edge from the graph. Contraction on the other hand is slightly more complicated in that when the edge is contracted both vertices are identified as the same. This can be seen in the figure below.

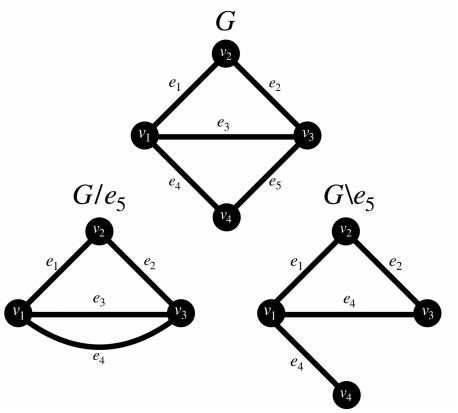


Figure 2: Deletion-Contraction of a Graph

Chip-firing

Now we turn to some of our main definitions. Throughout this let $G = (V, E)$ be a multigraph.

Definition 1. A *chip configuration*, also called divisor, c on G is an element of the free abelian group on the vertices:

$$\text{Config}(G) = \mathbb{Z}V = \{\sum_{v \in V} c(v)v : c(v) \in \mathbb{Z}\}.$$

Another way to think of this is that each configuration is placing some 'legal,' depending on the game in question, number of chips on each vertex of G, that is for each $v \in V(G)$, $c(v) \in \mathbb{Z}$. This can also be thought of as a vector in $\mathbb{Z}^V$ where the i^{th} entry is the number of chips on vertex v_i. Here we would like to mention that in much of the literature the term configuration is reserved to apply only with a fixed root in the graph, where the chips on that fixed root are not allowed, and a divisor refers to other cases. We choose to call them all configurations and define our games in such a way that it restricts the configurations to match the way they are played in the literature.

Definition 2. The *degree* of a configuration c is the sum of all chips, that is:

$$\deg(c) = \sum_{v \in V} c(v).$$

Now that we have defined the chips part of chip-firing we must discuss what it means to fire. If G is a graph and c a configuration on G, then informally firing a vertex v can be thought of as lending chips to all of its neighbors. When a vertex lends chips to its neighbors it must lend to all of them. That is to say that we decrease the number of chips on v by the degree of v and increase the number of chips on each adjacent vertex by the degree of their adjacency. An example of this can be seen below where v_1^* denotes the firing of v_1.

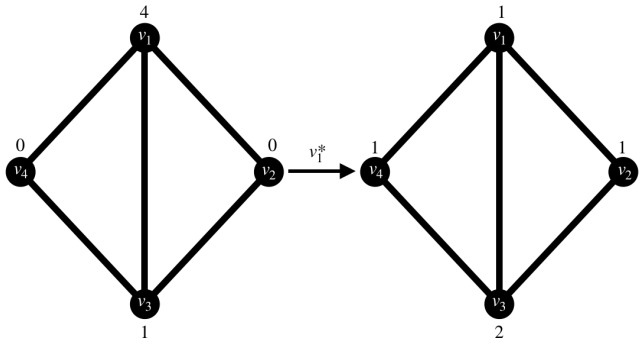


Figure 3: Example of a Vertex Fire

To define this more formally suppose c is a configuration on G, a new configuration c' is achieved by the following map:

$$c'(u) = \begin{cases} c(u) & ; u \notin N(v) \\ c(u) + k(uv) & ; u \in N(v) \\ c(u) - \deg(v) & ; u = v. \end{cases}$$

Here $k(uv)$ is the connectivity of the two vertices or said another way, the number of edges between u and v. While this method shows a vertex lending to its neighbors the idea of a vertex borrowing from its neighbors is also a legal move in some contexts. When it is legal, it is the same as every other vertex in the graph being fired so we will typically only refer to lending moves. One may notice that this type of mapping can be computationally expensive since we are checking each vertex one by one. This would lead us to another method of calculating the firing map.

To do this we will first need to explore another classic graph theory tool, the 'Laplacian matrix' of a graph. The *Laplacian matrix* of a graph G, denoted $\mathcal{L}(G)$ or just $\mathcal{L}$, is a $|V| \times |V|$ matrix that can be computed by taking the degree matrix D and subtracting the adjacency matrix A. That is

$$\mathcal{L}(G) = D - A$$

In our example above using $D(G)$ and $A(G)$ we see that

$$\mathcal{L}(G) = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

We will revisit this particular Laplacian momentarily.

Explicitly $\mathcal{L}(G)$ can be computed element wise as follows:

$$\ell_{i,j} = \begin{cases} \deg(v_i) & ; i = j \\ -(\#\text{ of edges between } v_i \text{ and } v_j) & ; i \neq j \end{cases}$$

Now to fire a vertex v_i, take the column of $\mathcal{L}(G)$ corresponding to v_i and subtract it from the configuration c. Naturally the column vector that was taken from $\mathcal{L}(G)$ was a vector in $\mathbb{Z}^V$ and as defined earlier c is also a vector in $\mathbb{Z}^V$ thus our subtraction in this way is well defined and in fact leads to a new configuration with the same degree. Once the Laplacian is in hand we can think of a sequence of firings as a vector in $\mathbb{Z}^V$ where the i^{th} entry is the number of times a given vertex say v_i is fired. That is to say if $x \in \mathbb{Z}^V$ is a vector encoding a firing sequence then a configuration c' can be obtained from c by

$$c' = c - \mathcal{L}(G)x.$$

Another equivalent way to compute the Laplacian of a graph is to use the (signed) incidence matrix, $B(G)$. Since we are working with undirected graphs, for the purposes of the incidence matrix we will need to artificially impose an orientation on the graph. The requirements for this orientation is only that the sum of each column of $B(G)$ be zero. That is to say that each edge either goes from vertex v to vertex w or the converse and in particular an edge between two vertices cannot be oriented out toward both its vertices or in from both of its vertices. The specific orientation is not important as the result we are seeking is independent of this choice. Using the incidence matrix B and its transpose B^t, we define the Laplacian to be:

$$\mathcal{L}(G) = BB^t$$

One can check that these two definitions result in the same matrix, however, we will not do that here.

Matrix-Tree Theorem

The matrix-tree theorem, also known as Kirchhoff's matrix-tree theorem, states that the total number of spanning trees of a graph G is equal to a specific evaluation of the Laplacian of G. To make this more precise we will first more clearly define what it means to be a spanning tree of a graph as well as talk about some basic ideas from linear algebra. A spanning tree T of a graph $G = (V, E)$ is a connected collection of edges which satisfy the following two conditions:

1. $T \subseteq E$ such that T contains no cycles (Tree)

2. All of the vertices in V are contained in T (Spanning)

Now that we have defined what a spanning tree is, we will proceed by delving into our linear algebra toolbox. We begin with the notion of determinant. The overall idea of a determinant is not difficult, but the precise definition is quite messy, and we choose to spare the reader and say only that $\det(A)$ will denote the determinant of a matrix A. We now define a cofactor of an entry a_{ij} to be

$$C_{ij} = (-1)^{i+j}\det(M_{ij})$$

where here M_{ij} is the reduced matrix obtained by deleting the i^{th} row and the j^{th} column from the full matrix M. We now have enough to formally write the matrix tree theorem.

Theorem 3. *(Matrix-Tree Theorem, [10])*
The number of spanning trees of a connected graph G is equal to the absolute value of any cofactor of the Laplacian of G.

Now we look at an example of applying this to the graph from above. Recall the

Laplacian of this graph as given below.

$$\mathcal{L}(G) = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}$$

For simplicity we will look at the cofactor of the entry in the first row, first column. We begin by reducing the full Laplacian to the matrix $M_{1,1}$, often in the literature this matrix is called the reduced Laplacian and denoted $\widetilde{\mathcal{L}}$:

$$\widetilde{\mathcal{L}} = M_{1,1} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Here we see that $\det(M_{1,1}) = 8$ so according to the Matrix-Tree Theorem there are 8 spanning trees of G and indeed, they are as follows:

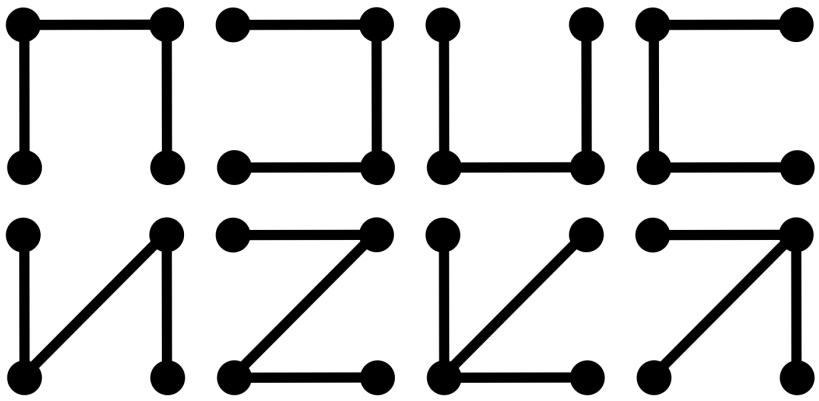


Figure 4: Spanning Trees of the Diamond Graph

One of the major tools used to prove the matrix-tree theorem came up numerous times during our exploration of chip-firing. While it will not be used in

this paper, we feel inclined to include it. The following formula is known as the Cauchy-Binet Formula. If A is an $m \times n$ matrix and B is an $n \times m$ matrix. Then,

$$\det(AB) = \sum_{S \in \binom{[n]}{m}} \det(A_{[m],S}) \det(B_{S,[m]})$$

Where $[n]$ denotes the set $\{1, \ldots, n\}$ and $A_{[m],S}$ denotes the $m \times m$ matrix with columns of A at the indices of S. In the case of the matrix-tree theorem, using the $\mathcal{L} = BB^t$ definition of the Laplacian, we see that $\widetilde{\mathcal{L}} = \widetilde{B}(\widetilde{B})^t$, where $\widetilde{B}$ is found by deleting a row from B. Applying the Cauchy-Binet formula to this gives

$$\det(\widetilde{\mathcal{L}}) = \sum_{S \in \binom{[n]}{m}} \det(\widetilde{B}_{[m],S}) \det((\widetilde{B}_{S,[m]})^t) = \sum_{S \in \binom{[n]}{m}} \det(\widetilde{B}_{[m],S})^2.$$

It is then shown that $\det(\widetilde{B}_{[m],S})$ is equal to 1 or -1 if and only if S induces a spanning tree, and 0 otherwise.

III. CLASSICAL CHIP-FIRING

We will now begin with our first game which we will refer to as classical chip-firing, in general this will be the game played and results from [6]. In this classical case we restrict the configurations to have entries only in $\mathbb{Z}_+$, however this will be weakened later. In the classical chip-firing game a vertex v is said to be *ready to fire* if $c(v) \geq \deg(v)$. Here when we *fire* a vertex v we get a new configuration c' as defined above and such a vertex fire is *legal* if the vertex was ready to fire in the first place, that is to say a firing would be illegal if the result of the firing would lead a vertex to have a negative number of chips. We take the following definition in the classical case under the pretense that it will change depending on the game at hand and for the purpose of this paper will change in the following chapters.

Definition 4. A configuration c is said to be *stable* if no vertex is ready to fire, that is for all $v \in V(G)$, $c(v) < \deg(v)$.

A *position* in the game is any distribution of chips on the graph such that $\sum_{v \in V(G)} c(v) = \deg(c_0)$, where $\deg(c_0)$ is the degree of the initial configuration on G. A *legal game* is any sequence of legal vertex firings.

A natural question at this point is what finite initial configurations c on G can be reduced to stable configurations? Furthermore, if it can be reduced to a stable configuration what are the possible stable configurations? The answers to these questions are major results of [6].

Theorem 5. *(Björner, Lovász, Shor [6], 1991)*
Given a connected graph and an initial configuration, either every legal game can be continued indefinitely, or every legal game terminates after the same number of moves with the same final position. The number of times a given vertex is fired is

the same in every legal game.

We omit the proof of the results in the finite case noting the proof is thorough in [6], using the order of firings as a language with special exchange properties.

It is also a result of [6] that if a game is infinite then each vertex is fired infinitely many times. To see this, observe that at least one vertex is fired infinitely many times. Since the graph is connected that vertex must have a set of neighbors. Each of the neighbors receive chips infinitely often but since the degree of the configuration was finite, each vertex cannot compile more than the degree of the configuration number of chips and therefore must be fired infinitely often itself. Again, the graph is connected so this process can be repeated indefinitely. This leads us to the main result of the Björner, Lovász, Shor paper.

Theorem 6. *Björner, Lovász, Shor [6], 1991*
Let G be a graph with V vertices and E edges. Then legal games can be categorized as follows:

(a) If c is a configuration with $deg(c) > 2E - V$ then the game is infinite, and each vertex is fired an infinite number of times.

(b) If $N \in \mathbb{Z}$ with $E \leq N \leq 2E - V$ then there exists an initial configuration of degree N guaranteeing finite termination and also a configuration of degree N guaranteeing an infinite game.

(c) If c is a configuration with $deg(c) < E$ then the game is finite.

The proof presented here will follow and expand on the proof presented in [6].

Proof. To prove (a), Let c be a configuration on G such that $\deg(c) > 2E - V$. Using the fact that $\sum_{v \in V(G)} \deg(v) = 2E$ we see that it must be the case that there exists

at least one vertex $v' \in V(G)$ with $c(v') > \deg(v')$ and therefore v' is ready to fire. After v' has been fired, since the total number of chips is unchanged it follows that there is another vertex that is ready to fire. Since this will always be the case after any number of firings, the game must be infinite.

To prove (b), we will be considering configurations c on G with $E \leq \deg(c) \leq 2E - V$. We will first show that that it is possible to have a finite game with $\deg(c) = 2E - V$. To do this place $\deg(v) - 1$ chips on each vertex. Obviously from this configuration $c(v) = \deg(v) - 1 < \deg(v)$ so no vertex can be fired, and the game terminates. Now,

$$\begin{aligned}\sum_{v \in V(G)} (\deg(v) - 1) &= \sum_{v \in V(G)} \deg(v) - \sum_{v \in V(G)} 1 \\ &= 2E - V\end{aligned}$$

To show the infinite case it is enough to prove we can construct a configuration c such that $\deg(c) = E$ that will lead to an infinite game. To do this we consider an acyclic orientation, $\mathcal{O}$ on G. Note that it is always possible to induce an acyclic orientation on any graph since we can order the vertices $\{v_1, v_2, \ldots, v_V\}$ and always direct each edge such that it is in the direction of the highest labeled vertex. One could easily check that this orientation is acyclic. For the purposes of this proof we will be using this specific orientation but the method is the same for any acyclic orientation. Now denote $\text{outdeg}(v)$ as the out degree of v and place $\text{outdeg}(v)$ chips on each vertex v. Since each edge is directed then

$$\sum_{v \in V(G)} \text{outdeg}(v) = E$$

Thus, our configuration has degree E.

Claim: This initial configuration will lead to an infinite game.

First consider v_1. Since there does not exist a vertex in our ordering such that $v_i \to v_1$ we have that $\deg(v_1) = \text{outdeg}(v_1)$ and thus it is ready to fire. Fire this vertex and reverse the orientation of all edges incident to it. The resulting configuration and orientation could have been achieved by a different labeling. That is let G' be a graph isomorphic to G such that $v_1 = v'_V$ and $v_i = v'_{i-1}$ for all $1 < i \leq V$. Once more our graph G' has a vertex that can be fired. This process is clearly non-terminating and thus our game is infinite.

For (c), Let c be a configuration on G such that $\deg(c) < E$ and fix an acyclic orientation, $\mathcal{O}$, on G. Consider the following quantity:

$$T = \sum_{v \in V(G)} \max\{0, c(v) - \text{outdeg}(v)\}$$

A vertex v with $c(v) < \text{outdeg}(v)$ is said to be *deficient*, and since $\deg(c) < E$ it must be the case that there is at least one deficient vertex. Our goal then is to show that the orientation can be modified mid-game so that T does not increase and in fact if our set of deficient vertices increases then T will actually decrease. If the game is infinite, then it must be the case that every vertex gets fired infinitely often and since a deficient vertex cannot be legally fired then the set of deficient vertices must also be changing. Thus, if T cannot increase and cannot decrease infinitely often, the game must terminate at some point.

First consider the case where the configuration is already stable. Then the game is clearly finite.

Now suppose the configuration is not already stable. Therefore, there must be a vertex $v \in V(G)$ with $c(v) \geq \deg(v)$. So, fire v and reverse the orientation of all edges leaving v. Here the firing is still happening in the undirected sense.

The orientation is just giving us a way to count what is going on in the game. As before, this does not create a cycle. In fact, T does not increase. To see this, denote T' to be the new quantity after the vertex is fired. Then,

$$c'(u) = \begin{cases} c(u) & ; u \notin N(v) \\ c(u) + k(uv) & ; u \in N(v) \\ c(u) - \deg(v) & ; u = v \end{cases}$$

and

$$\text{outdeg}'(u) = \begin{cases} \text{outdeg}(u) & ; u \notin N^+(v) \\ \text{outdeg}(u) + k^+(uv) & ; u \in N^+(v) \\ 0 & ; u = v \end{cases}$$

Here $N^+(v)$ represents the set of vertices that are adjacent to v by a directed edge out of v. Consider the term corresponding to v in T. The difference from T to T' is

$$\begin{aligned} c(v) - c'(v) - (\text{outdeg}(v) - \text{outdeg}'(v)) &= c(v) - (c(v) - \deg(v)) - (\text{outdeg}(v) - 0) \\ &= \deg(v) - \text{outdeg}(v) \end{aligned}$$

So, this term can only decrease or stay constant if $\deg(v) = \text{outdeg}(v)$.

Now consider a vertex w with w adjacent to v. If $w \in N^+(v)$ then the term in T corresponding to w would be unchanged which can be shown using an equation similar to the one above resulting in a difference of zero. However, if $w \notin N^+(v)$ the term is increased by 1, if it was not deficient before v was fired. Since there are exactly $\deg(v) - \text{outdeg}(v)$ such vertices that may possibly increase by 1, we can see that T cannot increase. If w was deficient then $c(w) < \text{outdeg}(w)$ and thus $c'(w) = c(w) + k(wv) \leq \text{outdeg}(w) = \text{outdeg}'(w)$ implying that $c'(w) - \text{outdeg}'(w) \leq 0$. This

shows that the term corresponding to w in fact actually decreases if w was deficient before v was fired. If none of the vertices adjacent to v were deficient before the firing, T is unchanged, if there were deficient vertices then T is decreasing. This is enough to prove our claim. □

IV. RECURRENCE AND THE CRITICAL GROUP

Configuration Classes

Another useful way to think about the configurations is to consider them as equivalence classes. To do this we must allow a vertices to have negative amounts of chips. We then identify two configurations $c \sim c'$ if there exists a sequence of 'legal' firings to get from c to c', where the term legal is dependent on the game being played. Using the Laplacian definition of firing using sequences, showing this is an equivalence class is not difficult. We need to show that for all configurations c, c', c'' on G we have:

1. $c \sim c$, (reflexive)
2. If $c \sim c'$ then $c' \sim c$, (symmetric)
3. If $c \sim c'$ and $c' \sim c''$, then $c \sim c''$ (transitive).

First let c be a configuration on G. Clearly $c \sim c$ where the sequence could be realized by the vector $x = 0 \in \mathbb{Z}^V$.

Now suppose c and c' are configurations with $c \sim c'$ then there exists a firing sequence $x \in \mathbb{Z}^V$ such that

$$c' = c - \mathcal{L}x$$

it then follows that

$$c = c' + \mathcal{L}x = c' - \mathcal{L}y$$

where here $y = -x \in \mathbb{Z}^V$. To negatively fire a vertex is the same as borrowing from its neighbors or firing every other vertex in the graph. So, this is in fact a well-defined firing sequence. Thus $c' \sim c$.

Finally let c, c', c'' be configurations such that $c \sim c'$ and $c' \sim c''$. Then we have that there exists $x, y \in \mathbb{Z}^V$ with

$$c' = c - \mathcal{L}x \quad \text{and} \quad c'' = c' - \mathcal{L}y$$

It then follows that

$$\begin{aligned} c'' &= c - \mathcal{L}x - \mathcal{L}y \\ &= c - (\mathcal{L}x + \mathcal{L}y) \\ &= c - \mathcal{L}(x + y) \end{aligned}$$

As $x, y \in \mathbb{Z}^V$ and $\mathbb{Z}^V$ is a vector space, $(x + y) \in \mathbb{Z}^V$ and therefore $c \sim c''$.

With this equivalence relation in hand we will define the *configuration class* determined by $c \in \text{config}(G)$ as

$$[c] = \{c' \in \text{config}(G) : c' \sim c\}$$

Rooted, Stable, and Recurrent

The following game has been referred to in many different ways, including avalanche, snowfall, and dollar game to name a few. To differentiate this from the dollar game we will work with later we shall refer to this game as the rooted dollar game. We begin by identifying a root vertex q that is allowed to go into debt. We will in fact keep this vertex in debt at all times and set the chips on q equal to the negative of the sum of the chips on all of the other vertices combined. That is, a

configuration c in this game would be defined as follows:

$$c(w) = \begin{cases} c(w) \geq 0 & ; w \neq q \\ -\sum_{v \in V} c(v) & ; w = q. \end{cases}$$

If the game were being played with money, we would consider such a vertex q to be a bank of sorts that can in some sense pump money into the 'economy' if and only if it gets stuck. In the context of chip-firing, q can be fired if and only if no other vertex in the graph can be fired. Since the debt of q is equal to the opposite of the rest of the vertices, we can always assume the total number of chips is zero. This leads us to another definition of 'stable.'

Definition 7. Given a fixed vertex q a configuration c is said to be *stable* if

$$0 \leq c(v) < \deg(v) \quad (v \neq q).$$

Furthermore, a sequence of firings is said to be *q-legal* if and only if for every time a vertex v is fired the prior configuration satisfied $c(v) \geq \deg(v)$ when $v \neq q$ and q is only fired if the configuration was stable before firing q. The only difference between our classical game and this game is the allowance of debt on a specific vertex and the inability to fire q until the configuration is stable away from q.

We now define the notion of 'recurrence.' A configuration c is said to be *recurrent* if there exists a q-legal sequence of firings leading from a specific configuration and returning to that same configuration. Furthermore, a configuration is said to be *critical* if it both stable and recurrent. This definition can be slightly misleading because we need q-legal firings for recurrence. In general, we want to start from a stable position, fire q, and return to the same stable

position. It is important to note that not all stable configurations are critical, and it is in fact very rare for a trivial, all 0, configuration to be critical. In the following figure we see an example of the all zero configuration being stable but not recurrent. After firing the root, we see our configuration becomes a critical configuration.

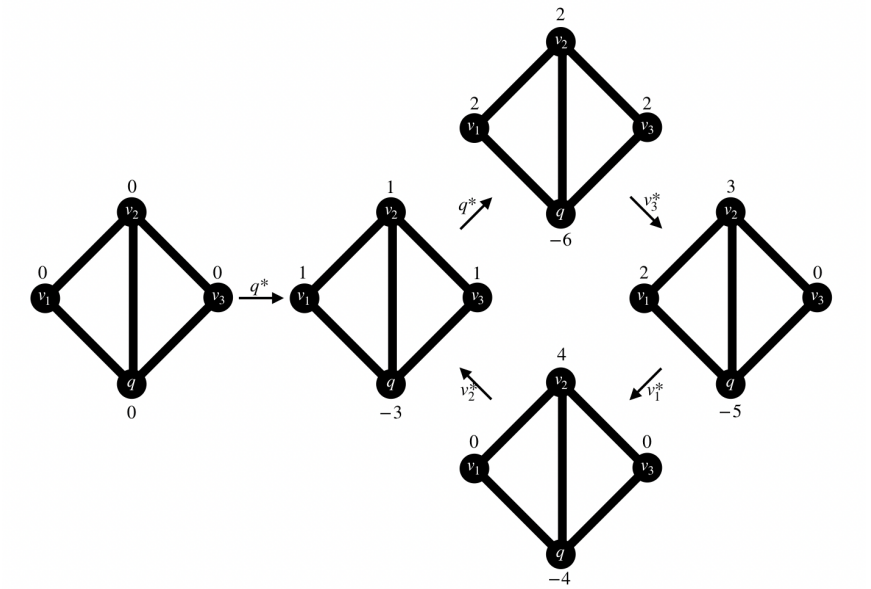


Figure 5: Not Critical and Critical Configuration

Let $K(G)$ be the set of all critical configurations on a graph G and it is known the order of $K(G)$ denoted κ, is equal to the number of spanning trees of G. The following is one of the main results of [5].

Theorem 8. *(Biggs [5],1999)*

Let c be a configuration of the rooted dollar game on a connected graph G. Then there is a unique critical configuration which can be reached by a q-legal sequence of firings beginning with c.

The Critical Group

As we have said before we can consider the configurations and firing sequences as vectors in $\mathbb{Z}^V$ and we have implicitly stated that this is a group under vector addition. Using this we can define a group structure on the set of critical configurations. To allow for this we will again need to relax slightly our restriction that $c(v) \geq 0$ for all $v \neq q$. Otherwise it is fairly clear that the way we have defined our games and their relation to $\mathbb{Z}^V$ is a group where we can add and subtract chip configurations from one another.

Next, $\mathcal{L}(G)$, the Laplacian matrix of our graph, is a homomorphism from $\mathbb{Z}^V \rightarrow \mathbb{Z}^V$ since it is a $\mathbb{Z}$ valued matrix. Let σ denote a function that takes a configuration in the form of a vector, $c \in \mathbb{Z}^V$ and outputs the degree of the configuration by:

$$\sigma(c) = \sum_{v \in V} c(v)$$

Algebraically this is the same thing as left multiplying a $1 \times |V|$ all 1's vector by the configuration and from that it follows that $\sigma : \mathbb{Z}^V \rightarrow \mathbb{Z}$ is a homomorphism.

Our hope now is to show that the set of critical configurations $K(G)$ is in bijection with the abelian group $\ker \sigma / \operatorname{Im} \mathcal{L}$, with a compatible group structure. Before we can proceed with this, we need to show that $\operatorname{Im} \mathcal{L}$ forms a normal subgroup of $\ker \sigma$. Since $\ker \sigma$ is itself an abelian group it suffices to only show that $\operatorname{Im} \mathcal{L} \subseteq \ker \sigma$. To see this let $x \in \operatorname{Im} \mathcal{L}$. Then there exists a vector y with $x = \mathcal{L}y$. Using the definition $\mathcal{L} = BB^t$ we can write $x = BB^t(y)$. Applying σ to both sides gives $\sigma(x) = \sigma(BB^t y)$. Now recall that the sum of each column of B is zero with exactly two non-zero entries, 1 and -1. This gives that $\sigma(B) = 0$. Now, making use of the associative property and the fact that σ is a homomorphism we have the

following:

$$\begin{aligned}\sigma(x) &= \sigma(BB^t c)\\ &= \sigma B(B^t c)\\ &= 0(B^t c)\\ &= 0.\end{aligned}$$

Therefore, we have $x \in \ker\sigma$ and $\operatorname{Im}\mathcal{L} \subseteq \ker\sigma$ and therefore $\operatorname{Im}\mathcal{L} \lhd \ker\sigma$ and it makes sense to even consider $\ker\sigma/\operatorname{Im}\mathcal{L}$.

Theorem 9. *(Biggs [5], 1999)*
The set $K(G)$ of critical configurations on a connected graph G is in bijective correspondence with the abelian group $\ker\sigma/\operatorname{Im}\mathcal{L}$.

The proof presented here closely follows the original given by Biggs in [5].

Proof. To begin we will first establish that every coset of $\ker\sigma/\operatorname{Im}\mathcal{L}$ contains a (legal) configuration. Given $f \in \ker\sigma$ define a configuration c on all vertices $v \neq q$ by

$$c(v) = \begin{cases}\deg(v) - 1 & ; f(v) \geq 0\\ \deg(v) - 1 - f(v) & ; f(v) < 0\end{cases}$$

and $c(q) = -\sum_{v\in\widetilde{V}} c(v)$. Then there is a stable configuration $\bar{c}$ that is reachable by some sequence of q-legal firings. Take x to be the vector representing this sequence of firings and it follows that $\bar{c} = c - \mathcal{L}x$. Finally define a vector $\bar{f} = f + c - \bar{c}$, that is $\bar{f} = f + \mathcal{L}x$. It is clear then that $\bar{f}$ is in the same coset as f and

$$\bar{f}(v) = f(v) + c(v) - \bar{c}(v) \geq \deg(v) - 1 - \bar{c} \geq 0.$$

Thus $\bar{f}$ is a legal configuration representing the coset containing f.

Next, we must define a function that is well defined and in bijection from $\ker\sigma/\operatorname{Im}\mathcal{L} \to K(G)$. To do this define

$$h : \ker\sigma/\operatorname{Im}\mathcal{L} \to K(G)$$

by $h(\alpha) = \gamma(c)$, where c is any legal configuration in the coset α and $\gamma(c)$ is the unique critical configuration ensured by Theorem 8. To show that this is well defined suppose c_1 and c_2 are both configurations with $[c_1] = [c_2] = \alpha$. It must be the case that $c_1 - c_2 = \mathcal{L}x$ for some $x \in \mathbb{Z}^V$. Thus, we can write $x = x_1 - x_2$ as the difference of two legal firing sequences x_1 and x_2 so that $x_1(v)$ and $x_2(v)$ are non-negative for all v. Then

$$\begin{aligned} c_1 - c_2 &= \mathcal{L}x \\ &= \mathcal{L}(x_1 - x_2) \\ &= \mathcal{L}x_1 - \mathcal{L}x_2 \end{aligned}$$

With this, let $c_0 = c_1 - \mathcal{L}x_1 = c_2 - \mathcal{L}x_2$.

Now suppose $\gamma(c_1) = \bar{c_1}$. Then there is a sequence of q-legal firings F_1 starting from c_1 and resulting in $\bar{c_1}$. Since $\bar{c_1}$ is recurrent we can ensure that F_1 fires each vertex v a minimum of $x_1(v)$ times. Now take the sequence F_1 and remove $x_1(v)$ occurrences of v resulting in $F_1^{x_1}$, a q-legal sequence of firings on c_0. It then follows that $F_1^{x_1}$ applied to c_0 yields the same result as F_1 applied to c_1, that is to say $\gamma(c_1) = \bar{c_1} = \gamma(c_0)$. Following the same process for c_2 shows that in fact $\gamma(c_1) = \gamma(c_0) = \gamma(c_2)$, and thus our map is well defined.

It remains then only to show that there is a bijective correspondence

between our two sets. For surjectivity let $c \in K(G)$. Clearly, c is a configuration on G and therefore is itself in $[c] \in \ker\sigma/\operatorname{Im}\mathcal{L}$. Thus $h([c]) = \gamma(c) = c$.

To show injectivity let $[c_1], [c_2] \in \ker\sigma/\operatorname{Im}\mathcal{L}$ with $h([c_1]) = h([c_2])$. Then we have that $\gamma(c_1) = \gamma(c_2) = c$. Thus, there exists some q-legal sequences x_1 and x_2 such that $c = c_1 - \mathcal{L}x_1$ and $c = c_2 - \mathcal{L}x_2$. We have then that $c_1 - c_2 = \mathcal{L}x_1 - \mathcal{L}x_2 = \mathcal{L}(x_1 - x_2)$ and thus c_1 and c_2 are in the same coset of $\ker\sigma/\operatorname{Im}\mathcal{L}$ and thus $[c_1] = [c_2]$. □

The abelian structure on $\ker\sigma/\operatorname{Im}\mathcal{L}$ defined by $[c_1] + [c_2] = [c_1 + c_2]$ would imply that $K(G)$ also has an abelian structure. Thus, using a binary operator $\bullet$ with $h([c_1]) \bullet h([c_2]) = h([c_1 + c_2])$ then $\gamma(c_1) \bullet \gamma(c_2) = \gamma(c_1 + c_2)$ and for any two critical configurations $\bar{c_1}$ and $\bar{c_2}$:

$$\bar{c_1} \bullet \bar{c_2} = \gamma(\bar{c_1} + \bar{c_2}).$$

The set $K(G)$ is called the *critical group* of G. We would like to describe slightly more explicitly the structure of $K(G)$. Suppose the operation in $K(G)$ is denoted by $\bigoplus$. Now given two configurations $c_1, c_2 \in K(G)$, then $c_1 \bigoplus c_2$ is computed first by adding the chips on each vertex to achieve a new configuration $c'(v) = c_1(v) + c_2(v)$. As above, we have that there exists a $\bar{c} \in K(G)$ with $\bar{c} \in [c']$. That is, we stabilize c' and the stable configuration $\bar{c}$ is recurrent and therefore critical.

In [5] it is shown that the map $h : \ker\sigma/\operatorname{Im}\mathcal{L} \to K(G)$ is in fact a group isomorphism.

q-reduced Configurations

While we are on the topic of these rooted configurations, we find it to be a good time to introduce the idea of 'q-reduced configurations'. To do this we will first need to introduce a new notion of firing involving firing sets of vertices. Before

stating the formal definition, we need to set the following notation. If $S \subseteq V$, then denote $\text{outdeg}_S(v)$ to be the number of edges $\{v, w\}$ with $w \notin S$.

Definition 10. Let c be a configuration on G and let $S \subseteq V$. Suppose c' is reached by starting at c and firing all of the vertices in S at the same time. The impact on a single vertex can be calculated by $c'(v) = c(v) - \text{outdeg}_S(v)$ and such a map $c \xrightarrow{S} c'$ is said to be a *legal set-fire* if $c'(v) \geq 0$ for all $v \in V$.

We use this definition of set firing to create an analogous definition to stability.

Definition 11. Given a configuration $c \in \mathbb{Z}V$, fix $q \in V$ and denote $\widetilde{V} = V \backslash \{q\}$. The configuration c is said to be *q-reduced* if

1. $c(v) \geq 0$ for all vertices $v \in \widetilde{V}$
2. For all non-empty $S \subseteq \widetilde{V}$, it is not legal to fire S. That is, for all $S \subseteq \widetilde{V}$ there is a $v \in S$ with

$$c(v) < \text{outdeg}_S(v).$$

.

The following figure is an example of a configuration and graph that is stable but not q-reduced.

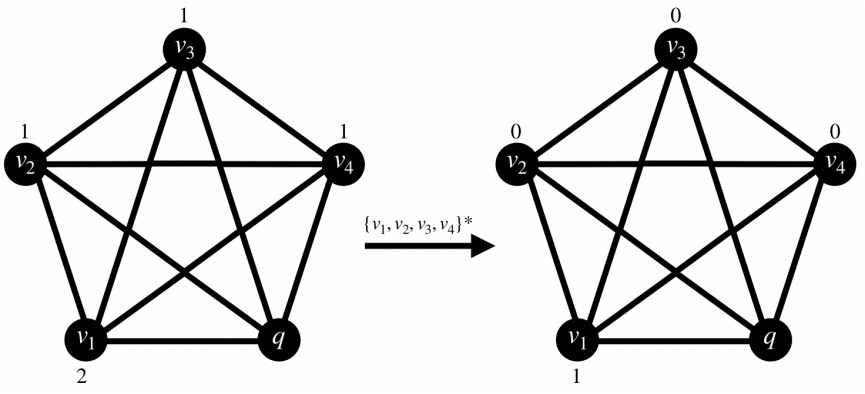


Figure 6: Stable not q-Reduced

We can see that no single vertex can be fired since each has degree 4. However, if we were to fire every non-root vertex this is a legal set fire since each has $c(v) \geq 1$ and $\text{outdeg}_S(v) = 1$. Where here $S = \{v_1, v_2, v_3, v_4\}$.

We would now like to show that there is also a bijection from these q-reduced configurations to the critical group we defined above, originally shown in [4]. To do this we will first need to establish some more notation. To do this define $c^* = K^+ - c$ for some configuration c on G. Here K^+ is defined by:

$$K^+ = \sum_{v \in V} (\deg(v) - 1)(v)$$

The definition of q-critical in [4] is slightly different from the definition of critical in [5] but as stated in the prior, Lemma 2.6 of the latter shows that the two definitions are in fact equivalent. Using this we then have the following result from [4], which gives the desired bijection.

Theorem 12. *(Baker, Norine [4], 2007) A configuration is q-reduced if and only if the configuration $c^* = K^+ - c$ is critical with respect to $\widetilde{V} = V \backslash \{q\}$.*

Superstables in Brief

With the definition of q-reduced in hand we would like to introduce the idea of a superstable configuration though it will not come up again until later in this paper. Thus far we have always considered a configuration c to be an element of $\mathbb{Z}V$ where each of the vertices receives some number of chips. In order to define what it means for a configuration to be superstable we will need to change slightly what we mean by configuration. When referring to these superstables we first fix a sink, say q from above or maybe s for sink. With this vertex s fixed we define a configuration c now only on the non-sink vertices of the graph. That is to say that $c \in \mathbb{Z}\widetilde{V} = \mathbb{Z}(V\backslash\{s\})$. It can be slightly ambiguous as to what kind of configuration we are working with, but we shall identify the type of configuration by the context it is used in. This combined with the definition of set-firing above yields the following notion of superstability.

Definition 13. Fix $s \in V$ and denote $\widetilde{V} = V\backslash\{s\}$. A configuration $c \in \mathbb{Z}\widetilde{V}$ is said to be *superstable* if

1. $c(v) \geq 0$ for all vertices $v \in \widetilde{V}$
2. For all non-empty $S \subseteq \widetilde{V}$, it is not legal to fire S. That is, for all $S \subseteq \widetilde{V}$ there is a $v \in S$ with
$$c(v) < \text{outdeg}_S(v).$$
where here we write $\text{outdeg}_S(v)$ to be the number of edges $\{v, w\}$ with $w \notin S$.

We then observe the fact that a configuration $c \in \mathbb{Z}V$ is q-reduced if and only if c restricted to $\widetilde{V}$ is superstable. This is quite clear from the relaxation given to the chips allowed on q in the first place.

Dhar's Burning Algorithm

Dhar's burning algorithm (after Dhar [8]) is an extremely useful tool to determine if a given configuration is superstable. While it has many other applications the direct result of the algorithm is given a configuration it outputs a set that is legal to fire. If the output is empty, the configuration was superstable. Let us first explain in words what the algorithm does and then give a psuedoalgorithm detailing the same.

To begin fix a vertex q and let c be a configuration on G. Our goal is to determine if a configuration is superstable, but since the algorithm is the same regardless whether the configuration is on V or $\widetilde{V}$, we do not need to differentiate between q-reduced and superstable here. Now imagine that our edges are wooden bridges, our vertices are towns, and our chips are firefighters. We begin the algorithm by setting fire to the root vertex. Once a town is on fire all of its connecting bridges will also catch fire. Now a town that has a connecting bridge on fire can only hold the fire off if it has enough firefighters to battle each bridge fire. That is to say if 4 bridges are on fire and the town only has 3 firefighters then the town will also go up in flames. Repeat this process until either no towns are unburned, or all remaining unburned towns have enough fire fighters to constantly fight the fires. If the entire graph is burned, then there were no sets that could be chip-fired. If there is an unburned set then this set can in fact be chip-fired and the algorithm can be run again. It is important to note that no firefighters were harmed in the process of this algorithm.

Now we will attempt to formalize Dhar's algorithm. For this suppose we have a connected graph G, a fixed sink q, denote $\widetilde{V} = V \backslash \{q\}$ and $\text{outdeg}_S(v)$ is the number of edges connecting v to vertices outside of $S \subseteq \widetilde{V}$.

Data: a non-negative configuration c

Result: a legal firing set $S \subset \widetilde{V}$, empty if and only if c is superstable

initialization: $S = \widetilde{V}$;

while $S \neq \varnothing$ **do**

  if $c(v) < outdeg_S(v)$ *for some* $v \in S$ **then**

    $S = S \backslash \{v\}$;

  else

    return S

  end

  return S

end

Algorithm 1: Dhar's Algorithm [7]

The following figures show an iteration of Dhar's burning algorithm where the configuration was not superstable and then the result of the algorithm after firing the unburned set from the first figure.

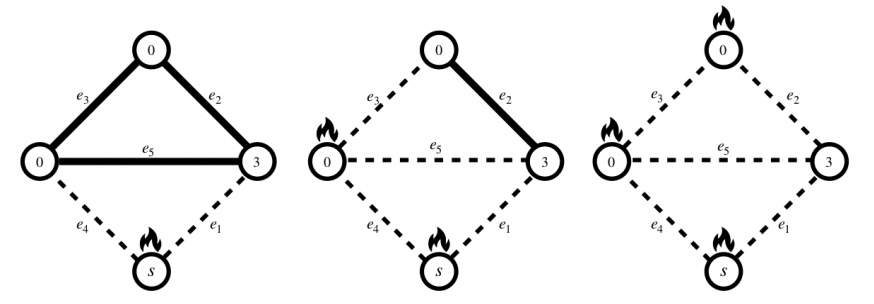


Figure 7: Dhar's Burning Algorithm on a Non-Superstable Configuration

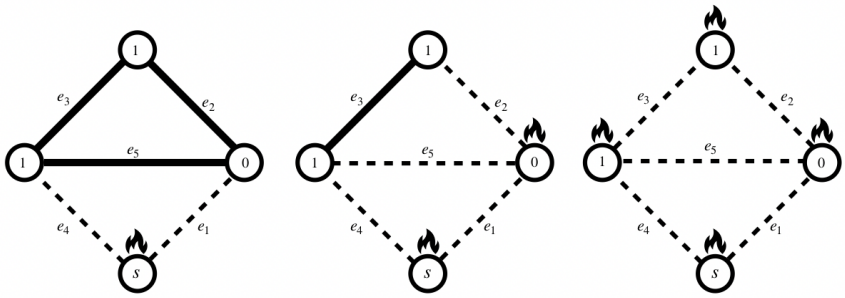


Figure 8: Dhar's Burning Algorithm on a Superstable configuration

Now, we would like to consider what happens if we fix an ordering on the edges. If we look at specifically the figure with the configuration that was superstable we could keep track of which edges caused the vertex to burn. Here we see that $\{e_1, e_5, e_3\}$ are the edges that first lit each vertex. We also notice that this set of edges is a spanning tree of G. This is not a coincidence. In fact, any time Dhar's algorithm is applied to a graph with a superstable configuration, a spanning tree can be recovered. We have also passively placed an ordering on the edges. The ordering does not appear to be useful in this case but if perhaps there was a tie, two edges simultaneously causing a vertex to burn, convention is to record the lower labeled edge as a member of our spanning tree. This establishes a bijection from the superstables of a graph to the spanning trees.

V. DOLLAR GAME AND RIEMANN-ROCH

Dollar Game

As mentioned above, the next variation that we will consider is also called the dollar game and has appeared in many places. A paper by Baker and Norine [4], and the book by Corry and Perkinson [7] are both literature with this variation of the game, as well as [1], a popular youtube channel "Numberphile." The rules here are very similar to the previous game but in this case, we allow all vertices to be assigned any integer number of chips, allowing negative values. The goal of this game is to get all vertices 'out of debt,' by making legal chip-firing moves as above. A game is said to be *winnable*, or a configuration is said to be *effective*, if there is a sequence of legal firings leading to a configuration such that $c(v) \geq 0$ for all $v \in V$.

Before trying to characterize which configurations are winnable, we will first make a few observations. Our first observation is that if the total number of chips is negative then obviously the game is not winnable. Our second observation is that if having a total of n chips guarantees a configuration is winnable then having a total greater than n will also guarantees a configuration is winnable.

Proposition 14. *Suppose T is a tree with a configuration c such that $deg(c) = 0$. Then the dollar game played on this configuration is winnable.*

Proof. Let T be a tree with configuration c of degree 0. Identify one degree 1 vertex as v_1 and its lone neighbor v_2. Note: Deleting a vertex of degree 1 will not disconnect the graph nor will it create any cycles.

Case 1: ($c(v_1) \geq 0$) Fire v_1, $c(v_1)$ times to result in v_1 having no chips. Now consider $T' = T \backslash \{v_1\}$ with configuration c', the result of zeroing out v_1. Since T' is still a tree suppose inductively that c' is a winnable configuration on T' with

some sequence of firings resulting in configuration $\bar{c}$. Perform this same sequence of firings on T after zeroing v_1, since c' was winnable in T' then it must be the case that $\bar{c}(v_i) = 0$ for all $i \geq 3$. Now recall $\deg(c) = 0$, so it must be the case that $\bar{c}(v_1) + \bar{c}(v_2) = 0$. Since v_1 was not fired again after being reduced to zero it must be the case that either $\bar{c}(v_2) = 0$ or $\bar{c}(v_1) = -\bar{c}(v_2)$. In the former the game is won, in the latter simply fire v_1 to zero again which will also force v_2 to zero.

Case 2: $(c(v_1) < 0)$ Borrow chips to get v_1 out of debt thus reducing to case 1, this completes the proof. □

Thus, there are graphs with the property that any configurations with degree 0 leads to a winnable dollar game and we will see that trees are precisely the graphs with this property. Consider a cycle $G = C_n$ with a configuration of degree zero. Obviously there does exist winning configurations of this type, namely a trivial configuration such that $c(v) = 0$ for all v. While this is not the only example on a cycle of a configuration with degree zero that is winnable it is an example that in any graph there exists a configuration of degree zero that is winnable. The question then is what the minimum configuration degree requirement should be for a game to always be winnable. Continuing our exploration into cycle graphs, let us go back for a moment to see if we can cook up an example of a cycle graph with a configuration of degree zero that is not winnable. Our claim is that this is possible, and the following figure is an example of just that. In the figure below, all of the configurations have degree 0. We will fire all vertices with positive chips and see we return to the same place. Since any number of these firings will lead us to one of these configurations, this is enough to say that the game is unwinnable.

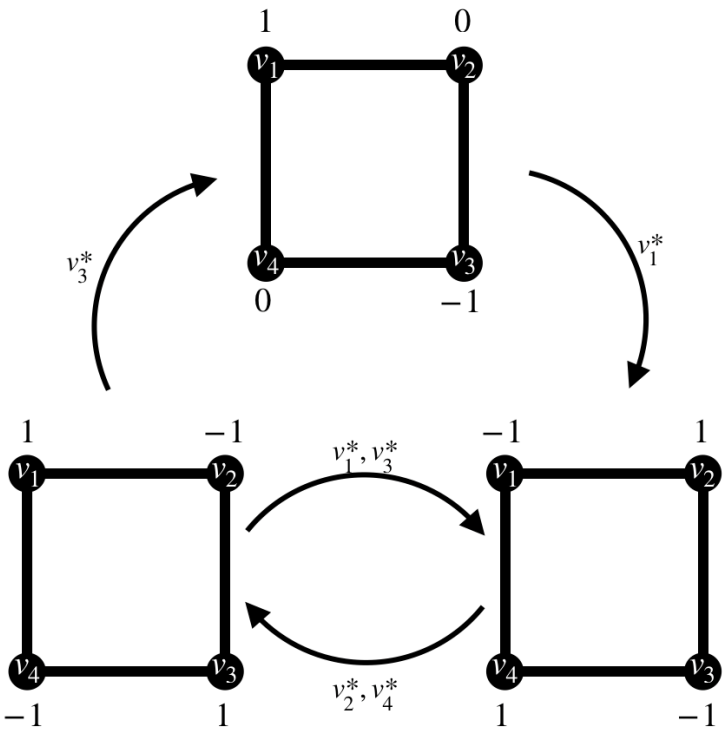


Figure 9: Non-Winnable Configuration with Degree 0

Hence, we must increase our minimum required number of chips to guarantee that the configuration on a cycle is a winnable dollar game. Let $G = C_n$ be a cycle graph with n vertices. We have established a lower bound of 1 for the minimum degree of our configuration to be winnable on a cycle graph. We will now prove that it is in fact an upper bound as well.

Proposition 15. *If c is a configuration on $G = C_n$ with $deg(c) > 0$. Then, c is a winnable configuration.*

It suffices to show that this is true when $\deg(c) = 1$.

First note if two adjacent vertices v and w with values n and $-(n-1)$ respectively and zeros elsewhere then this game is winnable. To win this game, fire v $n-1$ times to clear the debt on w. However, v fired $2(n-1)$ chips leaving

$n - 2n + 2 = -(n-2)$ on v and the adjacent vertex other than w now has $n-1$ chips. Repeating this process $n-1$ more times will result in a winning configuration. Our goal in this proof is to show that a cycle on n vertices with a configuration of degree 1 can always be reduced to a cycle with all of the chips on a vertex and all of the debt on a vertex adjacent to it.

Proof. Given an arbitrary cycle graph C_n with any initial configuration c such that $\deg(c) = 1$. Identify a vertex v_1 with $c(v_1)$ odd, this is always possible since if all of the vertices have an even number of chips just fire one of them and its neighbors will have an odd number of chips.
Now observe the following algorithm to win the game.

1. Identify all of the vertices going clockwise around the cycle starting at v_1 and labeling the remaining vertices v_i for $i = 2, \ldots, n$.

2. Fire or borrow v_1 until it has 1 chip. Now identify the number of chips on each vertex such that v_i has $\alpha_{i,1}$ chips. Here i identifies the vertex and 1 identifies that it is our first iteration.

3. Fire or borrow the set $\{v_n, v_1, v_2\}$ until v_2 has zero chips. Notice that the value of v_1 does not change while the change in v_n is equal to the change in v_2, that is:

$$\alpha_{i,2} = \begin{cases} 1 & ; i = 1 \\ 0 & ; i = 2 \\ \alpha_{i,1} + \alpha_{2,1} & ; i = 3, n-1 \\ \alpha_{i,1} - \alpha_{2,1} & ; i = n \\ \alpha_{i,1} & ; i \neq 2, 3, n-1, n \end{cases}$$

4. Continue this recursively resulting in the following equation:

$$\alpha_{i,n-2} = \begin{cases} 1 & ; i = 1 \\ 0 & ; i = 2, \ldots, n-2 \\ \alpha_{i,1} + \sum_{j=2}^{n-2}(n-2-j+1)\alpha_{j,1} + \sum_{j=2}^{n-2}\alpha_{j,1} & ; i = n-1 \\ \alpha_{i,1} - \left(\sum_{j=2}^{n-2}(n-2-j+1)\alpha_{j,1} \right) & ; i = n \end{cases}$$

5. Fire or borrow using v_n until v_{n-1} is left with zero chips. The result of this move would be that:

$$\alpha_{i,n-1} = \begin{cases} 1 - \left(\sum_{j=2}^{n-2}(n-2-j+1)\alpha_{j,1} + \sum_{j=2}^{n-1}\alpha_{j,1} \right) & ; i = 1 \\ 0 & ; i = 2, \ldots, n-1 \\ \alpha_{i,1} + \sum_{j=2}^{n-2}(n-2-j+1)\alpha_{j,1} + 2\sum_{j=2}^{n-1}\alpha_{j,1} & ; i = n \end{cases}$$

Adding up the chips on the two vertices that are holding non-zero chips.

$$\begin{aligned} \alpha_{1,n-1} + \alpha_{n,n-1} &= 1 + \sum_{i=2}^{n} \alpha_{i,1} \\ &= 1 + 0 \\ &= 1 \end{aligned}$$

Here we applied the fact that in our initial configuration $\alpha_{1,1} = 1$ and therefore the remainder of our configuration must sum to zero. Now we have that either v_1 has one more positive chip than v_n does negative or v_n has one more positive chip than v_1 does negative. In either case this proves that c was a winnable configuration on G. □

Therefore, we have established for a game to be winnable, the minimum degree required for a tree is 0, whereas the minimum degree required for a cycle is 1. Recall, the *genus*, also called the *cycle rank*, of a graph is equal to $g = E - V + 1$. Note that, in the case of a tree the genus is 0 and in the case of a cycle the genus is 1. This would lead us to believe that the minimum configuration degree required to win any dollar game is equal to the genus of the graph. In fact, this is the case and was one of the main results of [4].

Theorem 16. *(Baker, Norine [4], 2007)*
Let G be a graph with genus g. Then,

1. *For any configuration c with $deg(c) \geq g$ the game is always winnable by some sequence.*

2. *For each $n < g$ there exists a configuration c with $deg(c) = n$ that is unwinnable.*

There is actually a greedy algorithm to decide if a game is winnable. The following algorithm, from [7], reads in a configuration and outputs true if the configuration is winnable and false if it is not winnable. The greedy approach to win a game that is used here is to borrow at every vertex that is in debt until it is not in debt any longer. With a slight variation, the algorithm could be modified to also output exact sequences for winning the game also. Using this we could also show that any winnable configuration has a unique winning state after the same number of moves. Stronger than that it shows that every vertex is in fact fired the same number of times and that any two firing scripts starting from the same initial configuration are the same up to rearrangement of the order of firing.

```
1  Data: c is a configuration on G
2  Result: TRUE if c is winnable; FALSE if not
3  initialization: M = ∅ ⊂ V, the set of marked vertices;
4  while c is not effective do
5      if M ≠ V then
6          choose any vertex in debt: v ∈ V such that c(v) < 0;
7          augment c by borrowing at v;
8          if v ∉ M then
9              add v to M;
       else
10         return FALSE
       end
11     return TRUE
   end
```

Algorithm 2: Greedy algorithm for the dollar game [7]

While this algorithm shows that every configuration c with $\deg(c) \geq g$ is winnable this can also be seen as a corollary of a stronger theorem that we will discuss next.

Riemann-Roch

Before discussing Riemann-Roch for graphs we need to review some more background. Given a configuration c is winnable, we would like to in some sense quantify how winnable a game is. That is to say, how many chips can be removed, and the configuration remain winnable. This inquiry is the motivation for the following definition.

Definition 17. The *rank* or *dimension* of a configuration, denoted $r(c)$, to be equal to the maximum number of chips that can be removed from a configuration before it becomes ineffective, that is not winnable.

Obviously this definition only makes sense if the configuration was winnable in the first place. To remedy this, define $r(c) = -1$ if the configuration is not winnable. Further, we define the canonical configuration K such that $K(v) = \deg(v) - 2$ and that

$$\deg(K) = \sum(\deg(v) - 2),$$

we see that the degree of the canonical configuration is $\deg(K) = 2E - 2V = 2g - 2$. This now gives us all of the tools and definitions we need to make the statement of the graphical analogue to Riemann-Roch.

Theorem 18. *(Baker, Norine [4], 2007)*
Let c be a configuration on a graph G, free of loops, with genus $g = E - V + 1$, and canonical configuration K. Then,

$$r(c) - r(K - c) = deg(c) + 1 - g.$$

In particular,

$$r(c) \geq \deg(c) - g,$$

which provides a generalization of the theorem from above.

VI. APPLICATIONS TO CO-GRAPHIC MATROIDS

The Tutte Polynomial

We next describe a surprising connection between chip-firing and the Tutte polynomial. The Tutte polynomial of a graph has many applications and many forms. One of the most generally used versions is given a graph G, the Tutte polynomial is as follows:

$$T(G;x,y) = \sum_{A \subseteq E} x^{k(E)-k(a)} y^{k(A)+|A|-|E|}$$

Here $k(A)$ denotes the number of connected components of A. While this one may be the most familiar, we choose to consider an equivalent version, which is as follows:

$$T(G;x,y) = \sum_{i,j} t_{ij} x^i y^j$$

Here t_{ij} is the number of spanning trees with internal activity i and external activity j. For the exact definitions of internal and external activity see below. We note that the concept of internal and external activity requires an ordering on the edges, although their value has been proven to be independent of the ordering. For now, we will consider an evaluation of this polynomial when $x = 1$. This reduces our formula to:

$$T(G;1,y) = \sum_{j} t_j y^j,$$

where now we need only consider external activity. Another useful characterization of the Tutte polynomial involves recursive use of deletion-contraction of a graph.

This is done as follows:

(1) If G is a single vertex then: $T(G; x, y) = 1$

(2) If e is a loop, then: $T(G; x, y) = yT(G\backslash e; x, y)$

(3) If e is a bridge, then: $T(G; x, y) = xT(G/e; x, y)$

(4) If e is neither a loop nor a bridge, then: $T(G; x, y) = T(G/e; x, y) + T(G\backslash e; x, y)$

This deletion-contraction definition along with the form $T(G; 1, y)$, considering external activity, will motivate where we go next.

We choose to define internal activity in what may seem to be backwards. We fix an ordering on the edges, although the ordering does not affect the result of the count on the total number of these objects. Suppose T is a spanning tree of G, an edge e_i is said to be *internally passive* in T if:

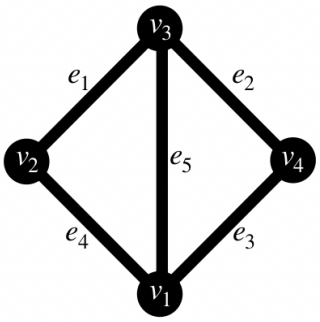
1. $e_i \in T$,
2. There exists an edge e_j not in T such that $j < i$ and $(T\backslash e_i) \cup \{e_j\}$ is again, a spanning tree of G.

Using this we say an edge e_i is *internally active* if $e_i \in T$ and is not internally passive. While we have reduced the Tutte polynomial in such a way that we need not consider internal activity we find it useful to have this definition first.
For simplicity we will denote $T^c = G\backslash T$.

Definition 19. An edge e_i is said to be *externally passive* for a spanning tree T if:

1. $e_i \in T^c$,
2. There exists an edge e_j not in T^c such that $j < i$ and $(T^c\backslash e_i) \cup \{e_j\}$ is again the complement of a spanning tree of G.

Once more we use this definition to say that an edge e_i is *externally active* if $e_i \in T^c$ and is not externally passive. Using our favorite graph and a labeling on its edges, the figure below shows all of the spanning trees of this graph. In addition to the spanning trees it also details whether the edges of the tree are internally active or passive and if the edges of its complement are externally active or passive.



T	$G\backslash T$	$EP(T)$	$EA(T)$	$IP(T)$	$IA(T)$
123	45	45	∅	∅	123
124	35	35	∅	4	12
125	34	34	∅	5	12
134	25	25	∅	34	1
135	24	4	2	35	1
234	15	5	1	234	∅
245	13	3	1	45	2
345	12	∅	12	345	∅

Figure 10: External and Internal Activity of a Graph

Note: This graph and a similar table is presented in [2] with a different labeling on the edges, with a slight error. This further supports that the specific labeling does not affect the total count of the internal and external activity.

Now consider the following graph G with a sink s and corresponding superstables.

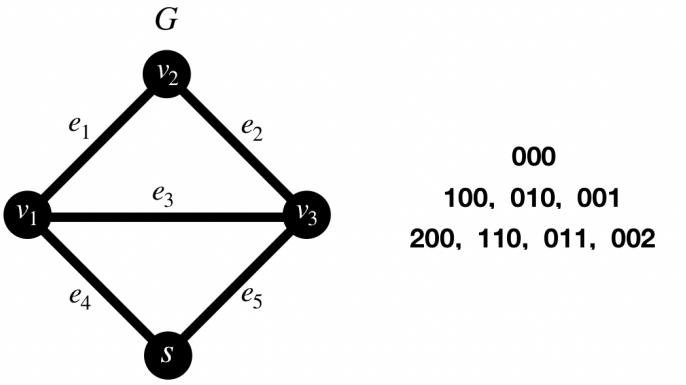


Figure 11: Superstables of the Diamond Graph

Using the internal-external definition of the Tutte polynomial we see that this graph has the following Tutte polynomial.

$$T(G;x,y) = x + 2x^2 + x^3 + 2xy + y + y^2 = x + 2x^2 + x^3 + (1+2x)y + y^2$$

Evaluating this polynomial when $x = 1$ gives,

$$T(G;1,y) = 4 + 3y + y^2$$

We notice the coefficients are the same as the number of superstables of a given degree in the graph above but in reverse order. This is part of the motivation behind Merino's theorem.

To formally state the theorem, we will need some facts regarding the degree of superstable configurations. In particular, we note that for any graph G the maximal superstables have degree $g = E - V + 1$, the genus that we have already used

above. For this, note that any orientation $\mathcal{O}$ on G induces a configuration $c(\mathcal{O})$ by taking $c(\mathcal{O})(v) = \text{indeg}_{\mathcal{O}}(v) - 1$. It is a result of [7] that this defines a bijection between the acyclic orientations of G with unique source, given by the root, and maximal superstables of G. Hence, the degree of such a configuration is as follows:

$$\begin{aligned}
\deg(c) &= \deg(c(\mathcal{O})) \\
&= \sum_{v \in \widetilde{V}} (\text{indeg}_{\mathcal{O}}(v) - 1) \\
&= \sum_{v \in \widetilde{V}} \text{indeg}_{\mathcal{O}}(v) - \sum_{v \in \widetilde{V}} 1 \\
&= E - (V - 1) \\
&= g
\end{aligned}$$

It then follows that given a configuration c is superstable, it is maximal if and only if $\deg(c) = g$. As we stated above, we are only considering loop free graphs. If by chance our graph contained loops then each loop would increase the genus by 1 but would not affect the number of superstables. That is to say, if there are exactly ℓ loops in our graph, the maximal degree of a superstable would be $g - \ell \leq g$.

For $i = 0, \ldots, g$ let h_i be the number of superstables of degree i and define

$$h = h(G) = (h_0, h_1, h_2, \ldots, h_g)$$

to be the *h-vector* of a graph G. If we revisit the Tutte polynomial evaluation of our graph above, we see that the following substitutions can occur.

$$T(G; 1, y) = 4 + 3y + y^2 = h_2 y^0 + h_1 y^1 + h_0 y^2$$

This is precisely the case that we will prove now with Merino's theorem.

Theorem 20. *(Merino [12], 1997) Let G be an undirected multigraph, possibly with loops and let $T(x,y) = T(G;x,y)$ be the Tutte polynomial. Then*

$$T(1,y) = \sum_{i=0}^{g} h_{g-i} y^i.$$

The proof presented here will follow and expand on the proof found in [7].

Proof. Let g be the genus of G and fix s as the sink vertex for all graphs appearing below. The proof goes by induction on the number of edges. For these purposes we define $h_0 = 1$ when G has only the single vertex s (possibly with some loop edges). Thus, in the case where G is a single vertex, we have $T(1,y) = 1 = h_0$, and the result holds.

Suppose that e is an edge of G incident to s. There are three cases to consider.

Case 1. Suppose that e is a loop, then let $G' = G\backslash e$.

Claim: $T(G;x,y) = yT(G\backslash e;x,y)$, the cycle rank of G' is $g' = g-1$, and the superstables on G' are the same as on G. $T(G;x,y) = yT(G\backslash e;x,y)$ is direct from the definition of $T(G;x,y)$. The fact that the cycle rank of G' is $g' = g-1$ is fairly clear since $V(G') = V(G)$, $E(G') = E(G) - 1$, and $g = |E(G)| - |V(G)| + 1$ which implies $g' = |E(G')| - |V(G')| + 1 = |E(G)| - 1 + |V(G)| + 1 = g - 1$. Finally, since the sink cannot be a part of a firing set, we see that removing the loop edge does not change the firing on the non-sink vertices and therefore the superstables are unchanged. Therefore, $h' = h(G') = h$. It then follows by induction and that $h_g = 0$:

$$T(G;x,y) = yT(G\backslash e;x,y) = y\sum_{i=0}^{g'} h_{g-i} y^i = \sum_{i=0}^{g} h_{g-i} y^i.$$

Case 2. Suppose that e is a bridge, and let $G' = G/e$. Claim: The circuit rank of G' is $g' = g$, and $h' = h(G') = h$. The fact that the circuit rank does not change is clear from the fact that contracting a bridge does not change the genus of a graph. To see that $h' = h(G') = h$, consider the endpoints of the bridge. Clearly, s is one end of the bridge, if v is the other vertex of the bridge and $c(v) \neq 0$ then v was a legal set-fire. Therefore, the $h-$vector is unchanged when contracting the edge. Again, by induction and applying the definition of the Tutte polynomial we get the following:

$$T(G; x, y) = xT(G/e; x, y) = x \sum_{i=0}^{g'} h_{g-i} y^i = \sum_{i=0}^{g} h_{g-i} y^i.$$

Case 3. Suppose $e = \{s, v\}$ is neither a loop nor a bridge. We divide the superstables of G as follows:

$$\mathcal{A} = \{c : c(v) = 0\}$$

$$\mathcal{B} = \{c : c(v) > 0\}$$

Our goal is to show that the elements of $\mathcal{A}$ are in bijection with the superstables of G/e and the elements of $\mathcal{B}$ are in bijection with the elements of $G\backslash e$. In our set $\mathcal{A}$, we first contract e and identify the resulting vertex in G/e as s. We denote the vertices of G/e as $V' = V\backslash\{v\}$, the non-sink vertices of G as $\widetilde{V} = V\backslash\{s\}$, and the non-sink vertices of G/e as $\widetilde{V}' = \widetilde{V}\backslash\{s\}$. Since we are considering only the configurations c of G with $c(v) = 0$, it is obvious that they would still be configurations on G/e. Now, let c be any configuration on G with $c(v) = 0$. If there is a non-empty set $W \subseteq \widetilde{V}$ that can be legally fired then since v is adjacent to s, with $c(v) = 0$, then it must be the case that $v \notin W$. Thus, c is superstable if and only if there does not exist a non-empty set $W \subseteq \widetilde{V}'$ that is legal to fire. Now since

the degree of any vertex of W is unchanged when considering it in G or G/e. It is clear from this that c is superstable in G if and only if it is superstable in G/e. Thus, the bijection holds from $\mathcal{A}$ to the superstables of G/e. Furthermore, under this bijection, since $c(v) = 0$, the degree of the superstable is preserved.

Now we consider $\mathcal{B}$ and given a configuration c of G we define the following

$$c^-(u) = \begin{cases} c(u) & ; u \neq v \\ c(v) - 1 & ; u = v \end{cases}$$

Claim: c is superstable in G if and only if c^- is superstable in $G\backslash e$.

To see this, let c be a configuration on G. If it is legal to fire any set $W \subseteq \widetilde{V}$, in G, it is the case that for all $w \in W$,

$$c(w) > \text{outdeg}_W^G(w).$$

Now for $w \neq v$ then it follows that $\text{outdeg}_W^G(w) = \text{outdeg}_W^{G\backslash e}(w)$ and by our definition of c^- we have that $c^-(w) = c(w)$ and therefore, $c^-(w) > \text{outdeg}_W^{G\backslash e}(w)$. If $w = v$ then we have that $\text{outdeg}_W^G(v) = \text{outdeg}_W^{G\backslash e}(v) + 1$. Therefore,

$$c(v) > \text{outdeg}_W^G(v) = \text{outdeg}_W^{G\backslash e}(v) + 1.$$

It then follows that as $c(v) - 1 = c^-(v)$ we have

$$c^-(v) > \text{outdeg}_W^{G\backslash e}(v).$$

Hence, if W is a legal set fire on G with configuration c then W is a legal set fire on $G\backslash e$ with configuration c^-.

Now suppose c^- is a configuration in $G\backslash e$. If it is legal to fire any set $W \subseteq \widetilde{V}$, in $G\backslash e$, it is the case that for all $w \in W$,

$$c^-(w) > \text{outdeg}_W^{G\backslash e}(w).$$

Now for $w \neq v$ then it follows that $\text{outdeg}_W^{G\backslash e}(w) = \text{outdeg}_W^G(w)$ and by our definition of c^- we have that $c^-(w) = c(w)$ and therefore $c(w) > \text{outdeg}_W^G(w)$. If $w = v$ then we have that $\text{outdeg}_W^{G\backslash e}(v) = \text{outdeg}_W^G(v) - 1$. Therefore,

$$c^-(v) > \text{outdeg}_W^{G\backslash e}(v) = \text{outdeg}_W^G(v) - 1.$$

It then follows that as $c(v) = c^-(v) + 1$ we have

$$c(v) > \text{outdeg}_W^G(v).$$

Again, we see that if W is a legal set fire on $G\backslash e$ with configuration c^- then W is a legal set fire on G with configuration c. Hence, c is superstable in G if and only if c^- is superstable in $G\backslash e$. Furthermore, the degree of each superstable in c^- is one less than the degree of each corresponding superstable in c.

Let h' and h'' denote the h-vector of G/e and $G\backslash e$ respectively, then

$$h_i = h'_i + h''_{i-1}$$

for all i. All of these are well defined except $i = 0$. To resolve this, we set $h''_{-1} = 0$. Using the fact that the cycle rank of G/e is the same as that of G and the cycle

rank of $G\backslash e$ is one less than that of G, it follows by induction that:

$$\begin{aligned} T(G;1,y) &= T(G/e;1,y) + T(G\backslash e;1,y) \\ &= \sum_{i=0}^{g} h'_{g-i}y^i + \sum_{i=0}^{g} h''_{g-i-1}y^i \\ &= \sum_{i=0}^{g} (h'_{g-i} + h''_{g-i-1})y^i \\ &= \sum_{i=0}^{g} h_{g-i}y^i. \end{aligned}$$

□

This result leads to a proof of Stanley's conjecture regarding the h-vector of matroids. In particular, it proves the conjecture for the class of 'co-graphic' matroids.

VII. FINAL REMARKS AND FUTURE DIRECTIONS

We have come to the end of the paper and while we did detail many of the concepts and strong results that we found there is still more that can be done. One possibility for work in the future would be to look at the idea of taking these chip-firing concepts to higher dimensions. As we have seen, using the 1-dimensional boundary map of a graph defines the Laplacian, which determines the firing rules for our chip-firing games on graphs. One can mimic this construction with a 2-dimensional boundary map on a 2-dimensional simplicial complex, but in this case the 'chip-firing rule' depends on how one picks the orientations on the 2-faces.

In another direction, one way to generalize chip-firing is through the theory of 'M-matrices,' which mimic certain properties of the (reduced) Laplacian of a graph [9]. An M-matrix is a square matrix with the property that a properly defined notion of chip-firing is 'avalanche-finite' and many other characterizations exits. M-matrices came up a few times while we made our way through different papers and we think there are more interesting connections to be made there. As an example of an M-matrix, the notion of an arithmetical graph [13] arises in the study of degenerating curves in algebraic geometry. More recently, this approach was taken in [3] in defining a 'tropical Laplacian' and the study of the Hodge conjecture. This is certainly something we want to look at in the future.

Changing directions slightly, in the Corry and Perkinson book, they speak on the geometry of sandpiles. For instance, the result of firing large numbers of chips on the origin of the planar lattice creates a number of beautiful images known as 'Apollonian circle packing' [11]. If we take the origin as our root this amounts to finding the identity in the underlying critical group. In general, the identity is far from trivial and itself can create some great images.

REFERENCES

[1] The Dollar Game - Numberphile. https://youtu.be/U33dsEcKgeQ. Accessed: 2019-02-21.

[2] F. Ardila, F. Castillo, and J. Samper. The topology of the external activity complex of a matroid. *Electronic Journal of Combinatorics*, 23:8, 10 2014.

[3] F. Babaee and J. Huh. A tropical approach to a generalized hodge conjecture for positive currents. *Duke Mathematical Journal*, 166(14):2749–2813, 10 2017.

[4] M. Baker and S. Norine. Riemann–roch and abel–jacobi theory on a finite graph. *Advances in Mathematics*, 215:766–788, 11 2007.

[5] N. L. Biggs. Chip-firing and the critical group of a graph. *Journal of Algebraic Combinatorics*, 9(1):25–45, January 1999.

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

[7] S. Corry and D. Perkinson. *Divisors and Sandpiles : An Introduction to Chip-Firing.* Providence, Rhode Island : American Mathematical Society, [2018], 2018.

[8] D. Dhar. Self-organized critical state of sandpile automaton models. *Physical Review Letters*, 64:1613–1616, Apr 1990.

[9] J. Guzmán and C. Klivans. Chip-firing and energy minimization on m-matrices. *Journal of Combinatorial Theory, Series A*, 132:14 – 31, 2015.

[10] G. Kirchhoff. Über die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148(12):497–508, 1847.

[11] L. Levine, W. Pegden, and C.K. Smart. Apollonian structure in the abelian sandpile. *Geometric and Functional Analysis*, 26:306–336, 1 2016.

[12] C. Merino López. Chip firing and the tutte polynomial. *Annals of Combinatorics*, 1(1):253–259, Dec 1997.

[13] D.J. Lorenzini. Arithmetical graphs. *Mathematische Annalen*, 285:481–501, 1989.

[14] J. Spencer. Balancing games. *Journal of Combinatorial Theory, Series B*, 23(1):68 – 74, 1977.

[15] J. Spencer. Balancing vectors in the max norm. *Combinatorica*, 6(1):55–65, Mar 1986.