CHIP-FIRING GAMES ON CONNECTED GRAPHS

ADANUR NAS

Abstract. In this paper, we introduced, analyzed, and detailed the *Chip-Firing Game* and its fundamentals and properties. Although there is not only one way to play the *Chip-Firing Game*, we still provided detailed descriptions and solutions to most known and utilized *Chip-Firing Game* variants and properties. We also provided visual representations to illustrate different scenarios with varying chip quantities. Since the topic has only been around for 40 years, there are a lack of accessible resources on it. Therefore, our aim is to consolidate available information on the *Chip-Firing Game* and present it to readers, while acknowledging existing constraints.

Contents

1. Introduction

Chip-Firing Game has emerged as a captivating and influential mathematical idea in various fields, bringing together experts from different fields such as mathematics, physics, or even economy. Although its precise date of invention is still being investigated and debated, the consensus among reputable resources claim that the *Chip-Firing Game*'s origins may be traced again to the pioneering paintings of mathematician *David Gale* in 1983. Since its inception, the *Chip-Firing Game* has garnered huge interest because of its connections with *graph theory* and its applications in various regions.

In the game, chips are placed on the *vertices* of a connected graph, and the researches use a strategic system of redistributing those *chips* based totally on the rules and conditions they set.

The game's mathematical fundamentals have been utilized to set deep and intricate standards related to areas such as electrical networks, equilibrium configurations, and even net

Date: July 16, 2023.

income calculations. Researchers have delved into its complexities, unveiling connections to topics inclusive of sandpile fashions, spanning trees, and group concept, among others.

Moreover, the Chip-Firing Game has been used as a effective tool for investigating essential questions in mathematics. It has provided insights into the structure and behavior of graphs, permitting researchers to address hard problems in a singular and innovative way.

In this paper, we aim to delve into the intricacies of the *Chip-Firing Game*, exploring its fundamentals, analyzing its properties, and uncovering the mathematical principles that govern its conduct. We also provide sufficient amount of graphical displays of the game to make this paper more readable and accessible. The accuracy of the information and calculations provided in this paper were checked through OEIS, research papers and talks from researchers from various different fields, and graphing calculators.

2. Fundamentals of Chip-Firing Game

2.1. **Chip-Firing Game.** *Chip-Firing Game* is a mathematical game that simulates the redistribution of resources/chips on a graph. That is, on a connected graph with *vertices* and *edges*, the game is played by placing a certain number of chips on the vertices of the graph and redistributing the chips among the neighboring vertices based on the number of chips the vertex possesses in relation to its *degree*. For example,

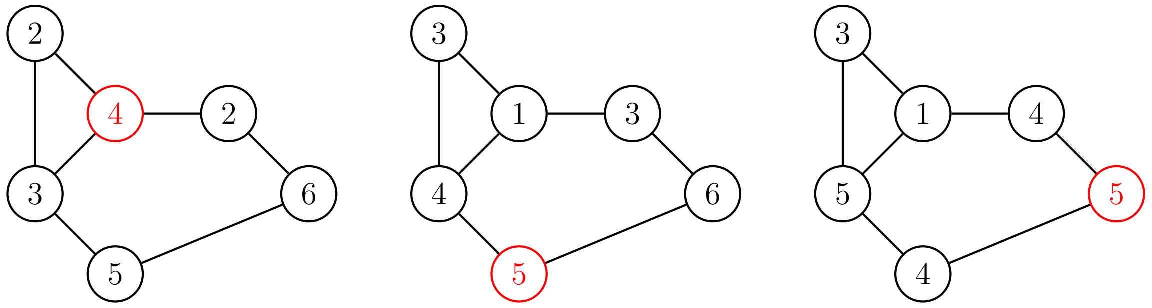


Figure 1. Chip-firing game with 6 vertices and 7 edges.

where red-colored chips represent the active vertices, the vertices that are being fired evenly to neighboring vertices. The game can be played potentially for infinite times until all the *vertices* have less chips than how many neighbors they have.

Definition 2.1. A *vertex* in a *Chip-Firing Game* refers to a node or a point in a graph, and it represents the location of chips being placed at or fired from. Most of the time, vertices on a graph are labeled or colored differently to distinguish them from each other.

Definition 2.2. An *edge* in a *Chip-Firing Game* refers to a connection or link between two vertices. Chips flow from one vertex to another vertex through *edges*, and those *edges* can be represented by lines or arcs.

Definition 2.3. The *degree* of a vertex represents the number of edges connected to that vertex.

The *Chip-Firing Game* involves redistributing chips among the vertices of a graph according to specific rules. The number of chips on a vertex represents a certain quantity, such as energy or resource in real life; thus, assigning negative values to vertices would not be possible since they do not represent a tangible quantity. Moreover, non-integer values are not used to assign values to vertices because the game's rules and dynamics are designed around the discrete nature of chip transfers, where whole numbers of chips are moved from one vertex to another.

Having a history of only 40 years, the *Chip-Firing Game* is a relatively young topic, falling under broader topics such as *algebraic geometry*, *graph theory*, *combinatorial games*, and *discrete mathematics* [8]. This multidisciplinary nature of the *Chip-Firing Game* helps theoretical physicists with measuring frequency in noise and sandpile shapes and theoretical computer scientists with sorting and latencies in networks and local balancing [7].

Theorem 2.4. *In a Chip-Firing Game labeled with G, if there are n number of vertices and m number of chips to redistribute, then there are*

$$v \quad = \quad \binom{m+n-1}{n-1}$$

ways to redistribute the chips [2, 10].

Proof. We will consider a *Chip-Firing Game* labeled with G, consisting of n number of *vertices* and m number of chips to redistribute. The chips will be considered as indistinguishable objects, meaning that all chips of the same quantity are identical. That is, to distribute the m number of chips among the n number of *vertices*, we will view this as placing *dividers*, which represent the chips, between the *vertices*. The number of *dividers* will be equal to the number of *edges*.

The *dividers* will divide the *vertices* into distinct regions, with each region representing the number of chips in that vertex. Thus, the total number of regions will be $n-1$ since we have $n-1$ *dividers* to place among the n number of *vertices*. Therefore, the problem can be formulated as determining the combinatorial possibilities associated with selecting the positions of $n-1$ dividers from the set of $m+n-1$ total possible positions, encompassing both *vertices* and *dividers*. This can be represented by the binomial coefficient

$$v \quad = \quad \binom{m+n-1}{n-1}$$

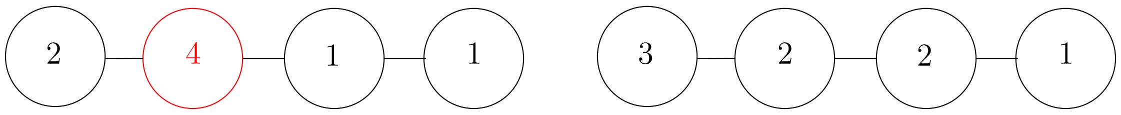
which counts the number of ways to choose $n-1$ positions out of the $m+n-1$ total positions. In conclusion, this formula accurately represents the number of ways to redistribute m number of chips among n number of *vertices* in any *Chip-Firing Game* labeled with G. ∎

Although there are several *configurations* to redistribute chips among neighboring vertices, the most common *configuration* is *diffusion*. In *diffusion*, the chips are redistributed evenly and simultaneously among neighboring vertices. At each step, each vertex sends a chip to each of its poorer neighbors. In order to maintain consistency, the values of vertices can be negative, although such a thing is not possible in real life.

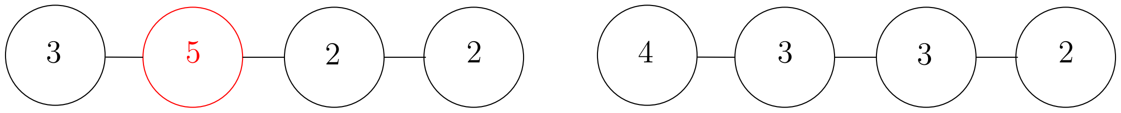
Definition 2.5. A *configuration* is the assignment of integer values to the vertices of a graph.

Proposition 2.6. *Two configuration sequences are isomorphic if, within the period, the corresponding steps only differ by an addition of some constant to each stack size.*

Proof. If there are no restrictions or additional rules, there are infinite ways to assign values to vertices. That is, we can assign any integral value to any vertices and have infinite number of different *configurations* for any *Chip-Firing Game*. To visualize, we will assume that we have a *Chip-Firing Game* with 4 vertices and 3 edges. When we fire it, we obtain this configuration sequence:



Now, we will add 1 chip to each vertices of our *Chip-Firing Game* and fire the same vertex.

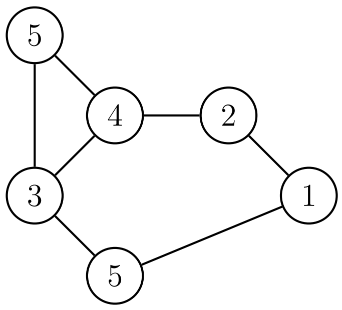


Therefore, this visualization shows that adding or subtracting a constant number of chips to each stack does not affect the relative differences between the chip counts on the *vertices* [11]. This is because adding or subtracting a constant value to all vertices simply shifts the entire configuration uniformly without altering the relative distribution of chips among the *vertices*.

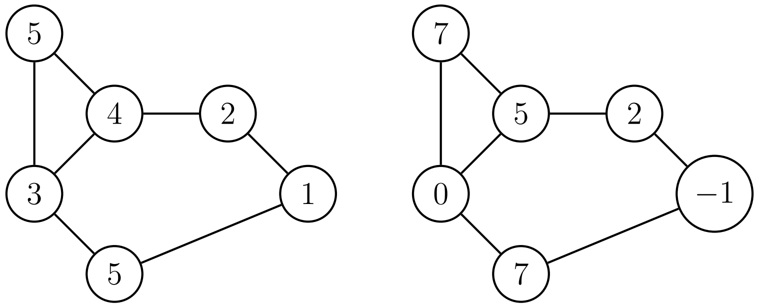
■

2.2. **Pre-Positions.** As much as it can be played by redistributing chips among neighboring vertices, the *Chip-Firing Game* can also be played backwards, resulting in obtaining the previous configuration sequences of that *Chip-Firing Game*. These previous configuration sequences are often called as the *pre-positions* of that *Chip-Firing Game*.

Example 2.7. *To visualize, we will assume that we have a Chip-Firing Game with 6 vertices and 7 edges.*



Instead of allowing vertices with more chips than their neighbors to send their chips, we will reverse the process and enable vertices with fewer chips than their neighbors to send their chips instead. Thus,

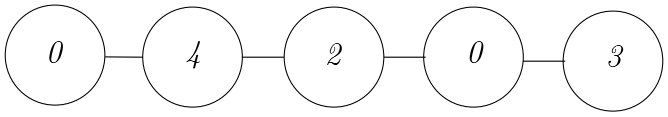


if we apply diffusion and allow vertices to obtain negative values to maintain the consistency.

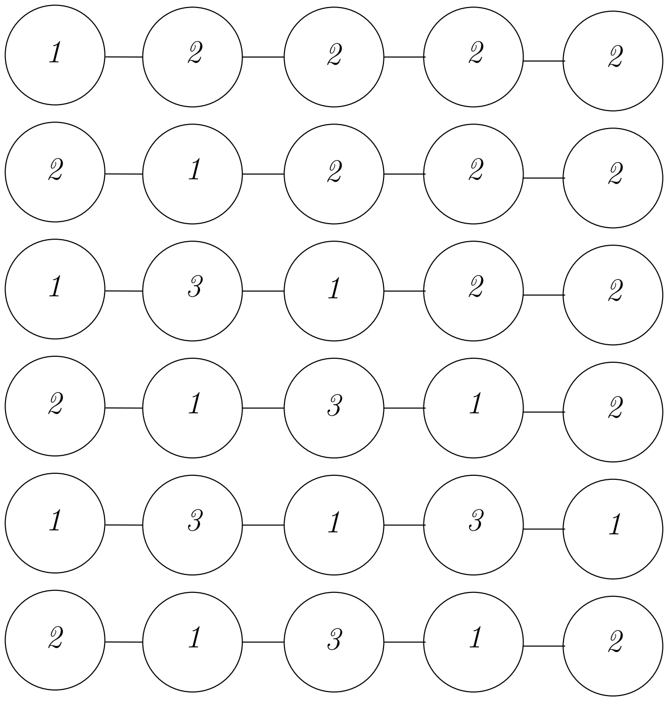
This example shows that *pre-positions* represent the state of the chip distribution on the graph before the game starts, and therefore, every *Chip-Firing Game* has at least one *pre-position*.

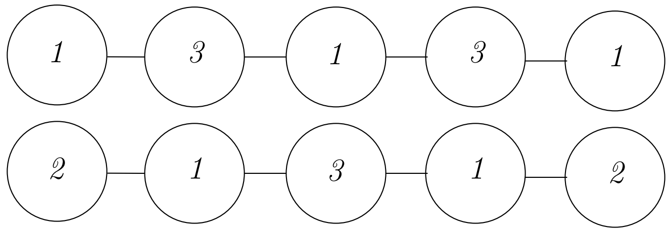
2.3. **Periods in Chip-Firing Games.** A *period* of a *Chip-Firing Game* is the property of the configuration sequence of chips on that graph. That is, the *period* of a configuration sequence represents the smallest positive integer such that after repeating the configuration sequence that many times, the configuration of chips returns to its original state. Thus, the *period* refers to the repetition of the configuration sequence, not the firing sequence itself.

Example 2.8. *We assume that we have a Chip-Firing Game with 5 vertices and 4 edges. Its initial configuration is*



Now, we will use diffusion and investigate how many unique configurations we need to identify before we find the period of our Chip-Firing Game.





Therefore, our Chip-Firing Game has a period of 2, which starts with the configuration 2-1-3-1-2, and 3 unique configurations to reach that period.

As this example depicts, how many unique *configurations*, or *positions*, we need to identify before we reach the *period* depends on the *initial configuration* of our *Chip-Firing Game.*

Moreover, we do not need to always go forward and only use *positions* to find the *period.* We can also use *pre-positions* to find the *period* because using *diffusion* while investigating the *pre-positions* of the *Chip-Firing Game* allows us to assign negative integer values to vertices. This flexibility in assigning values evidences that there are infinitely many unique *pre-positions* to any *Chip-Firing Game.* This finding is particularly essential for investigating *periods* in *Chip-Firing Game* since we will need to understand the properties of the initial *configuration* of the particular *Chip-Firing Game* we are investigating [4].

3. Properties of Chip-Firing Game

Over the course of 40 years of its history, *Chip-Firing Game* has been investigated and expanded by various mathematicians. Not only mathematicians, but also many physicists and economists frequently utilize the *Chip-Firing Game* since the concept of placing chips on *vertices* can be modified to investigate energy distributions, revenue calculations, and many more [1, 2, 5]. That is, experts can use chips to represent what they are analyzing, such as each chip equaling to $100 profit made by selling 10 hats. This is also where we can extend the rule of not assigning any negative integral values to *vertices* since a negative value would indicate debt in this context. However, this is a case-by-case situation and should not be used to generalize other *Chip-Firing Games.*

Thanks to this nature of the *Chip-Firing Game* and efforts of various experts, there are many essential properties of the *Chip-Firing Game* that had been discovered and studied on. In this section, we will analyze some of the properties that are highly essential.

3.1. **Stability of Chip-Firing Game.**

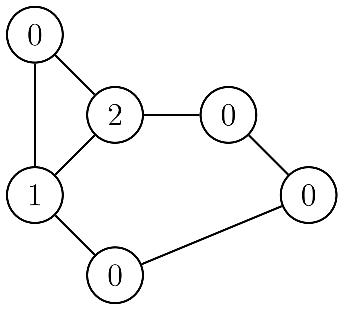
The stability of the *Chip-Firing Game* is a crucial topic to study since it provides insights into the dynamics and long-term behavior of various network structures and systems.

Moreover, understanding the stability of the *Chip-Firing Game* is essential in various fields such as *graph theory*, *network science*, and *distributed computing* [12]. This topic can help researchers analyze the stability and resilience of complex networks, such as social networks, communication networks, and transportation networks [9]. Studying the stability properties of the *Chip-Firing Game* enables researchers to obtain valuable insights into the propagation and spread of information, resources, or influence within a network by analyzing how the distribution of chips, or resources, evolves over time and whether it reaches a stable configuration or continues to fluctuate indefinitely.

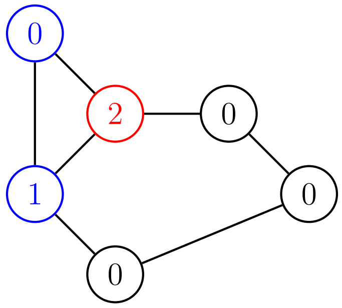
A stable configuration, or *stable state*, in the *Chip-Firing Game* occurs when no *vertex* in the network has a chip count that exceeds its outgoing degree. That is, each *vertex* possesses enough chips to redistribute to its neighboring *vertices*, maintaining a balance in the system.

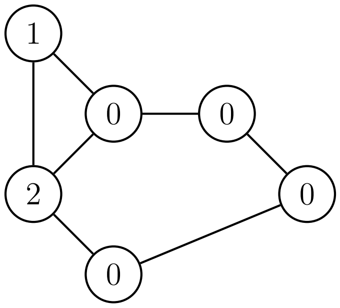
Definition 3.1. *Stable state* is the state in which no further firing is possible. This state is also often referred to as a *firing-squad* or *firing-free* configuration.

Example 3.2. *We will assume that we have a Chip-Firing Game with 6 vertices and 7 edges. The initial configuration for our Chip-Firing Game will be*

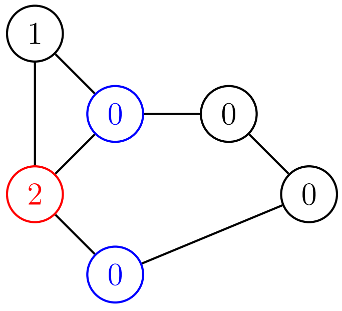


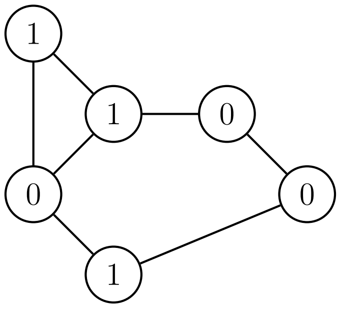
Unlike some of the previous examples, we will not use diffusion in this example. Instead, we will fire red-colored vertices to blue-colored vertices. Thus,





Lastly,

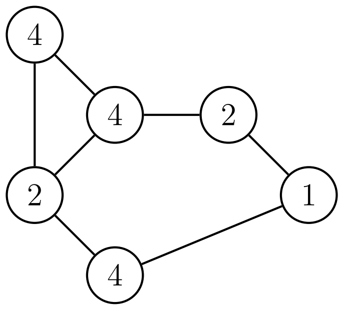




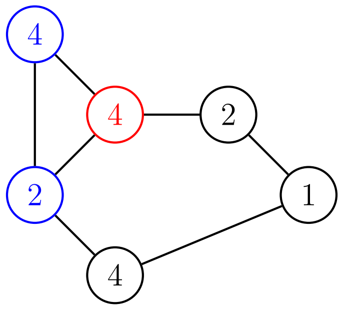
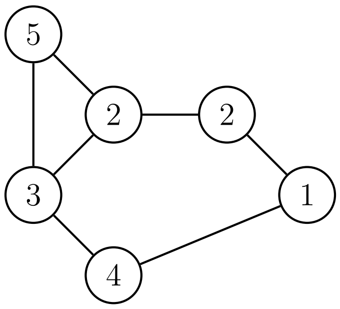
Since no other rules are stated nor implemented, such as vertices not being restricted to fire to all of their neighboring vertices, the firing has stopped, and the configuration has reached to a stable state. The reason for this is because no vertices has equal amount of or more chips than their degrees.

3.1.1. *Instability of Chip-Firing Game.* Even though many researchers focus on the concept of *stability* and *stable states*, the concept of *instability* is also a crucial topic to further analyze in *Chip-Firing Game*. Contrary to a *stable state*, an *unstable state* refers to a configuration, or state, in which further firing is possible and the chip distribution does not reach a balanced or stable state [6]. In an *unstable state*, certain *vertices* may accumulate an excessive number of chips, exceeding their degrees. *Unstable states* indicate a lack of equilibrium, and they can exhibit unpredictable chip dynamics, making it challenging to determine the long-term behavior of the system. *Unstable states* also have crucial properties to investigate.

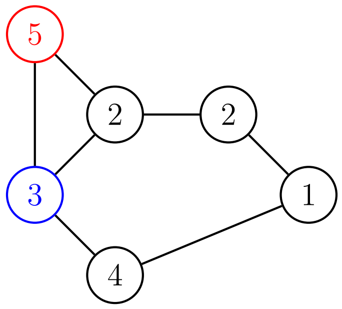
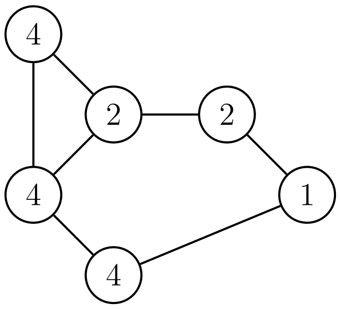
Example 3.3. *In this example, we will again assume that we have a Chip-Firing game with 6 vertices and 7 edges. The initial configuration for our Chip-Firing Game will be*



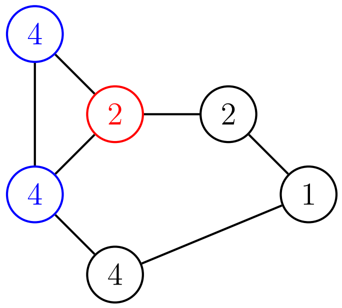
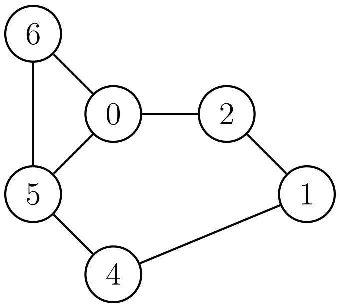
We will once again do not use diffusion, and red-colored vertices will fire to blue-colored vertices. Thus,

Then,

Lastly,

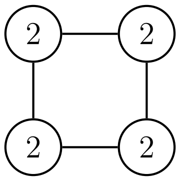



With the last firing, we reached an unstable state for this Chip-Firing Game. That is, after this state, this Chip-Firing Game will never reach a stable state with further firings because one vertex has a chip count of 0 with a degree of 2, and its two neighboring vertices at left have chip counts more than their respective degrees. This situation violates the stability condition, which asserts that no vertex can have more chips than its degree. Therefore, the chip configuration will continue to fluctuate indefinitely.

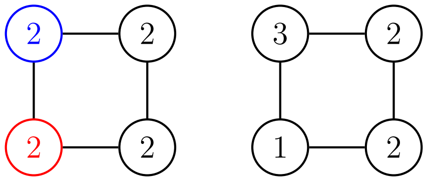
However, not all firing sequences for every *Chip-Firing Game* follow a random pattern since some of them, while still maintaining an *unstable state*, can have a *cyclic configuration.*

Definition 3.4. A *cyclic configuration* occurs when each *vertex* has at least the number of chips as their *degrees*, resulting in an infinite loop of firings and not allowing the game to reach a stable state [4].

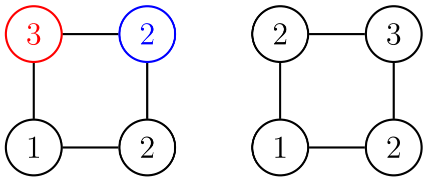
Example 3.5. *We will assume that we have a Chip-Firing Game with 4 vertices and 4 edges. The initial configuration will be*



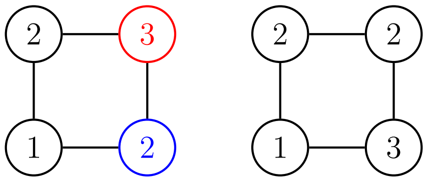
In this Chip-Firing Game, each vertex has 2 degrees and the same number of chips. Once again, red-colored vertices will fire to blue-colored vertices. Thus,



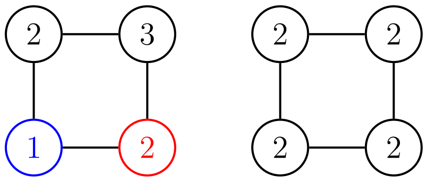
Then,



Then,



Lastly,



Although rather a simple example, this firing sequences demonstrate how after 4 firings, the initial configuration repeats itself. Therefore, this Chip-Firing Game has a cyclic configuration and will never reach a stable state.

3.1.2. *Super-stability of Configurations.* The term *super-stability* refers to the condition where configurations of a *Chip-Firing Game* is more stable than other stable configurations [13]. Under the section **Fundamentals of Chip-Firing Game**, we discussed that addition or subtraction of the same number of chips to each *vertices* do not change the relative differences between the chip counts on the *vertices*. However, if we add or subtract different number of chips from *vertices*, then we would disrupt its *stability state*, making it destabilized. Therefore, in comparing two *stable states* of the same *Chip-Firing Game*, if one of them requires more chips to be added or subtracted from it to become destabilized, then we would call that state the more *stable state*, or the *super-stable state* [13].

3.2. Infinite-Finite Structure of Chip-Firing Game.

The concept of *stability* is crucial to understanding and analyzing the *Infinite-Finite Structure of Chip-Firing Game*. In the previous subsection, we have already investigated this structure by making calculations to find out whether a *Chip-Firing Game* has a *stable state*. That is, we investigated whether a *Chip-Firing Game* terminates at some point or has additional firing sequences. Therefore, if it terminates at some point, we find out that the *Chip-Firing Game* has a *finite structure*. However, if it still has possible further firings, then we determine that the *Chip-Firing Game* has an *infinite structure*. To summarize,

(1) If each *vertex* possesses fewer chips than its *degree*, then the game has a *stable state*, and finite behavior is observed.
(2) If each *vertex* possesses equal number of or more chips than its *degree*, then the game has an *unstable state*, and infinite behavior is observed.

However, in the previous subsection, to determine the *stability* of the *Chip-Firing Game*, we focused on the relative number of chips possessed by *vertices* compared to their *degrees*. Instead, in this section, we will focus on the relative number of chips compared to the number of *edges* [3]. Therefore,

Theorem 3.6. *Insufficient chips relative to the number of edges guarantee finiteness, ensuring the game reaches a stable state.*

Proof. We will assume that we have a *Chip-Firing Game* played on a connected graph labeled with G with c number of *chips*, n number of *vertices*, and m number of *edges*. For our game, we will suppose the total number of *chips* in the game is less than the number of *edges*, that is,

$$c < m$$

Now, we need to consider the initial configuration of graph G. Since the number of *chips* is less than the number of *edges*, there will always be at least one *vertex* with fewer *chips* than its *degree* [3].

According to the stability conditions introduced earlier in this subsection, the structure of the game is infinite if vertices have chips equal to or more than their degree. However, in this game, we will not extend chip-firing rules. That is, each *vertex* will fire only it has chips at least the number of its *degree*. In this game, there exists a *vertex* with fewer chips than its *degree*; thus, this *vertex* cannot fire any chips. Consequently, the chip distribution remains unchanged, and the game reaches a *stable state*, resulting in a finite game.

■

Theorem 3.7. *If the number of chips is equal to or exceeds the number of edges, an appropriately chosen initial configuration can result in an infiniteness of the game. It never reaches a stable state under such circumstances.*

Proof. We will analyze two conditions:

Condition 1: We will assume that the number of *chips* is equal to the number of *edges*. Thus, the initial configuration will have the same number of *chips* as the number of *edges*, and we can distribute exactly one *chip* to each *vertex* in the graph.

Now, we will fire *chips* from *vertices* with at least as many *chips* as their *edges*. Since we assumed that each *vertex* initially has one chip, the firing process can continue indefinitely as long as each *vertex* has at least one *chip* to fire [3].

Moreover, if we analyze the contrary situation, the game terminates and reaches a state where no further *chips* can be fired. That is, every *vertex* will have fewer *chips* than its *edges*. However, this situation also contradicts the previous assumption that each *vertex* initially had one *chip*, and we distributed one *chip* to each *vertex*. Therefore, the firing process can continue indefinitely, leading to an infinite structure.

Condition 2: We will assume that the number of *chips* is greater than the number of *edges*. Thus, the initial configuration will have more *chips* than the number of *edges* in the graph. In this case, we cannot distribute exactly one chip to each *vertex*.

As we already discussed and demonstrated, in the *Chip-Firing Game*, firing *chips* from a *vertex* decreases the number of *chips* at that *vertex* and increases the number of *chips* at its *neighboring vertices*. By using a strategy to fire *chips*, we can move *chips* around the graph and eventually reach a state where the number of chips at each *vertex* is less than or equal to the number of its *edges*.

Once we reach this state, we will apply the argument from **Condition 1**. Since the total number of chips did not change and is still greater than the number of *edges*, we can distribute exactly one chip to each *vertex* and continue the firing process indefinitely, leading to an infinite structure [3].

In conclusion, in both conditions, if the number of *chips* is equal to or more than the number of *edges*, depending on the initial configuration, the Chip-Firing Game can have an infinite structure.

■

Theorem 3.8. *The game is always infinite when the number of chips exceeds twice the number of edges minus the number of vertices.*

Proof. We will assume that our *Chip-Firing Game* is played on a connected graph, and, at the beginning of the game, an arbitrary number of chips is placed on the *vertices* of the graph. Furthermore, we will set a couple of rules to distribute the chips among the vertices. The rules are

(1) If a *vertex* has at least as many chips as its *degree*, then that *vertex* is **active** [8].
(2) In each firing, an active vertex will fire one chip to its neighboring vertices on a connected graph.
(3) If a *vertex* has fewer *chips* than its *degree*, then that *vertex* is **inactive** [8].

After setting the rules, we will assume that our *Chip-Firing Game* have a configuration that satisfies the condition of our theorem. That is, the number of *chips* will exceed twice the number of *edges* minus the number of *vertices*. Now, we will analyze two conditions:

Condition 1: When there is an *active vertex* with more chips than its *degree*: In this condition, we will start by firing from the *active vertex* to its *neighboring vertices*. Since the number of *chips* exceeds twice the number of *edges* minus the number of *vertices*, at least one vertex will need to receive more *chips* than its *degree*. Therefore, this *vertex* will become active.

Condition 2: When all *active vertices* have exactly as many *chips* as their *degrees*: If all *active vertices* have the same number of *chips* as their *degrees*, firing a chip from any of these *vertices* will not change the game's configuration. Therefore, in this situation, we can fire chips from *inactive vertices* or choose any arbitrary *vertex* to fire *chips*.

After analyzing those two conditions and repeatedly firing *chips* according to the rules we already set, we will see that the number of *active vertices* will either increase or remain the same. At each firing, if we have an *active vertex* with more *chips* than its *degree*, the game will continue indefinitely without terminating.

In conclusion, we proved that when we play the game on a connected graph, there will always be at least one *active vertex* as long as the number of *chips* exceeds twice the number of *edges* minus the number of *vertices*. Therefore, the *Chip-Firing Game* will never terminate in this situation, resulting in an infinite sequence of moves.

■

References

[1] P. Bak and M. Paczuski. Complexity, contingency, and criticality, Jul 1995.
[2] N.L. Biggs. Chip-firing and the critical group of a graph - journal of algebraic combinatorics, Jan 1999.
[3] Anders Björner, László Lovász, and Peter W. Shor. Chip-firing games on graphs, Sep 2013.
[4] Anders Björner Björner and László Lovász. Chip-firing games on directed graphs, Jul 1992.
[5] Matthew Cho, Anton Dochtermann, Ryota Inagaki, Suho Oh, Dylan Snustad, and Bailee Zacovic. Chip-firing and critical groups of signed graphs, Jun 2023.
[6] James Dylan Douthitt. Chip-firing on graphs: Stability, the dollar game, and the tutte polynomial, May 2019.
[7] Kelley, Anne. 18.204: Chip firing games, Apr 2016.
[8] Caroline J. Klivans. The mathematics of chip-firing, Nov 2018.
[9] Bernhard Kroetz and Gestur Olafsson. The c-function for non-compactly causal symmetric spaces and its relations to harmonic analysis and representation theory, Apr 2003.
[10] Ken Levasseur. Wolfram demonstrations project, Jul 2018.
[11] Todd Mullen, Richard Nowakowski, and Danielle Cox. Counting path configurations in parallel diffusion, Oct 2020.

[12] M. Angeles Serrano, Dmitri Krioukov, and Marian Boguna. Self-similarity of complex networks and hidden metric spaces, Feb 2008.
[13] Haolin Tony Zhong. Introduction to the chip-firing game, Aug 2022.