BROWN UNIVERSITY

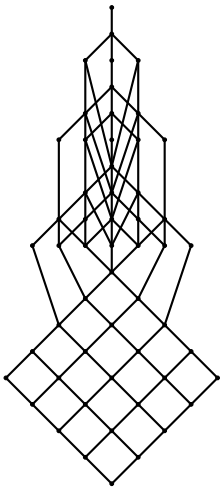
DOCTORAL THESIS

Confluence in Chip-Firing and Related Combinatorial Processes

---

Author:
Patrick Liscio

Supervisor:
Dr. Caroline Klivans



A thesis submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy

in the

Division of Applied Mathematics

May 2022

This dissertation by Patrick Liscio is accepted in its present form
by The Division of Applied Mathematics as satisfying the
dissertation requirement for the degree of Doctor of Philosophy.

Date_____ _____

Caroline Klivans, Ph.D., Advisor

Recommended to the Graduate Council

Date_____ _____

Melody Chan, Ph.D., Reader

Date_____ _____

Johnny Guzmán, Ph.D., Reader

Approved by the Graduate Council

Date_____ _____

Andrew Campbell, Dean of the Graduate School

CV

Education

Brown University *August 2017 - Expected May 2022*

PhD in Applied Mathematics

Masters in Applied Mathematics, 2019

Massachusetts Institute of Technology *August 2013 - June 2017*

Bachelor of Science, Mathematics

Minors in Computer Science, Physics

Publications

C. Klivans and P. Liscio. Confluence in Labeled Chip-Firing. *Journal of Combinatorial Theory, Series A*, 186, 2022.

Klivans, C., Liscio, P., Results in Labeled Chip-Firing, *Accepted to FPSAC 2020*

Liscio, P., Lattices in Chip-Firing, *Preprint at https://arxiv.org/abs/2010.15650*

Klivans, C., Liscio, P., Move and Configuration Posets, *In preparation*

Teaching

Brown University

Instructor, Statistical Inference I *Summer 2020*

Instructor, The Mathematics of Rubik's Cubes and Related Puzzles *Summer 2021*

Teaching Assistant, Computational Probability and Statistics *Fall 2018, Fall 2019*

Teaching Assistant, Recent Apps. of Prob. and Stat. *Spring 2019, Spring 2020*

Teaching Assistant, Statistical Inference I *Fall 2020, Spring 2021*

Massachusetts Institute of Technology

Teaching Assistant, Mathematics for Computer Science *Fall 2016, Spring 2017*

Lab Assistant, Mathematics for Computer Science *Spring 2016*

Academic Mentor, Mathroots Summer Program *Summer 2015, 2016, 2017*

Outreach and Service

Treasurer, Brown University SIAM Chapter	*Fall 2019 - Present*
Brown University Math CoOp	*Fall 2017 - Present*
Brown University Math Circle Organizing Committee	*Spring 2019 - Fall 2021*
Tutor, Math Resource Center	*Fall 2017 - Spring 2020*
Directed Reading Program Mentor	*Fall 2018, Spring 2020*
Graduate Student Mentor	*Fall 2018 - Spring 2019*
Secretary, Brown University SIAM Chapter	*Fall 2017 - Spring 2019*

Acknowledgements

Thank you first to my advisor, Caroline Klivans, for always providing ideas, insight, and inspiration. She wished in her own PhD thesis to be as good an advisor as her own, Richard Stanley. This is a standard that she has surely exceeded.

Thank you to Melody Chan and Johnny Guzmán for taking the time out of their busy schedules to read this thesis. Thank you to Johnny especially for helping me to find my way at a time when the world of academia seemed especially daunting.

This experience would not have been possible without the support of my cohort. Thank you to Charlie, Emily, Jenna, Justin, Rebecca, Xinyue, and Yin-Ting for making the department a fun place to work every day. Thank you to Becky for filling three different offices with much-needed diversions and insightful discussions.

On the subject of the department, I greatly appreciate the help of everyone who keeps it running every day. Thank you especially to Candy, Jean, Rosanna, and Stephanie for making my time here run smoothly, and for making this place feel like home.

Thank you to my parents and sister for all of your love and support, and to Cheerio, for giving me a reason to get up every morning.

To Anna, thank you for making these last few years better, both inside the office and out.

Abstract of *Confluence in Chip-Firing and Related Combinatorial Processes*,
by Patrick Liscio, Ph.D., Brown University, May 2022

Confluence is an important property in many combinatorial processes. A globally confluent process is one in which a fixed initial state always leads to a fixed final state, even if different choices of moves are made throughout the process. In many processes, including chip-firing, global confluence is proven using local confluence: if two moves are available at the same time, they may be performed in either order.

We consider processes that exhibit global confluence without local confluence. We begin with labeled chip-firing, a modified chip-firing process in which chips with numerical labels are fired according to certain rules. Under some conditions, the chips are guaranteed to end the process in sorted order. We provide a new proof that labeled chip-firing from an initial configuration with $2m$ chips at the origin always leads to a sorted final configuration. The proof involves analyzing a poset of firing moves, which contains a diamond of locally confluent moves at the end of the process.

We use similar methods to prove confluence in related processes, including other initial configurations, variations of the line graph, and classical root system types. We also analyze the class of move posets. We show that locally confluent diamonds are generated in move posets on a much larger class of bipartite graphs. We also show that the move poset is isomorphic to the poest of join-irreducible chip configurations. We prove this by introducing edge colorings, called S-colorings, of lower locally distributive lattices. We show that S-colorings describe moves in many other combinatorial processes, and we use S-colorings to provide a new proof of a theorem stating that all shortest paths between flip-connected domino tilings use the same flip moves in a possibly different order.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

We study the concept of confluence in combinatorial processes. We consider processes that start from some initial configuration, and reach other configurations through non-deterministic sequences of moves, either terminating in some final configuration, or running forever. A confluent process is one that not only terminates, but that terminates in a fixed final configuration, regardless of the moves chosen or the order in which they are performed. In a confluent process, two people can choose sequences of moves completely independently, and they will always be assured of reaching the same configuration at the end of the process. Confluence is a frequently studied concept in computer science, where it pertains to areas such as abstract rewriting systems [13] [41]. It is also important in many combinatorial processes, several of which are studied in this thesis.

We consider the dynamics of chip-firing and related combinatorial processes. A chip-firing process is a dynamical system on a graph. A natural number of chips is placed at each node in the graph. If a site has at least as many chips as it has outgoing edges, the site can fire by sending one chip along each edge to the corresponding neighbor. Chip-firing either terminates at a stable configuration in which no site can fire, or it continues forever.

Confluence is extremely important to the study of chip-firing. Given a graph and a configuration from which the chip-firing process terminates, the process is known to be globally confluent: it must always terminate in the same final configuration after the

same number of moves. Global confluence is proven using local confluence: if at any time, sites a and b are both ready to fire, then both sites can fire in either order [8].

However, there are many combinatorial processes that exhibit global confluence without local confluence, including several that are closely related to chip-firing. These processes include labeled chip-firing [27] [31], root system chip-firing [22] [21], and flow-firing [19].

The primary goal of this thesis is to find new methods for establishing global confluence in systems that are not locally confluent. We show that, even in some processes that are not locally confluent, there can sometimes be local confluence hidden in the end of the process. A small collection of locally confluent moves near the end of the process can often be enough to ensure global confluence of the entire system.

The first problem considered in this study of confluence is labeled chip-firing. In the original labeled chip-firing problem presented by Hopkins, McConville, and Propp [27], chip-firing is performed on a 1-dimensional grid graph, beginning with n chips at the origin. The major change is that the chips are labeled from 1 to n. As in unlabeled chip-firing, each move consists of two chips at the same site being chosen, with one sent to the left and the other sent to the right. However, we add the rule that the chip with the smaller label must be sent to the left, while the chip with the larger label must be sent to the right.

Hopkins et al. [27] show that if the number of chips n is even, the process ends with the n chips in sorted order, regardless of the order in which the firing moves are performed, and the choices of chips involved in each move. This is a prominent example of a proof of global confluence in a process that lacks local confluence.

Liscio and Klivans [31] provide another proof of this sorting result, which is reproduced in Chapter 3 of this thesis. The authors study a partial ordering of the firing moves of the process, and we find that a locally confluent diamond of firing moves at the end of the process is able to correct any inversions that are created throughout the process. The existence and utility of these diamonds form the foundation for this thesis. Similar diamonds are found in many combinatorial processes, and they are used to

prove confluence results in many of the proceesses that we study.

We are able to prove confluence in many problems related to labeled chip-firing. While the original problem only dealt with initial configurations consisting of n chips at the origin, we are able to classify all weakly sorted initial configurations as either "confluent" or "non-confluent." We also prove confluence results on several variations of the 1-dimensional grid graph, in which self-loops and varied edge multiplicities can change the firing behavior, and thus make confluence either "easier" or "harder" to achieve. Our methods are used to reduce the run-time of labeled chip-firing by a factor of n, and by relating labeled chip-firing to sorting networks, we are able to turn labeled chip-firing into an $O(n \log(n))$ sorting algorithm.

Even in processes that are not confluent, our methods can still provide some insight. In labeled chip-firing with odd $n = 2m + 1$, it is known that the chips do not generally end in sorted order, but there are two major open problems on the behavior of these processes. We analyze the poset of firing moves and prove results about the number of inversions created throughout the process, providing insight into the "m inversion conjecture." We also prove probabilistic results about random firing sequences, reducing the "$\frac{1}{3}$ conjecture" from a statement about 3 different probability distributions to a statement about just 1.

Our methods can even be used to prove confluence in combinatorial processes beyond labeled chip-firing. Galashin et al. view labeled chip-firing as a Type A special case of a more general problem of labeled chip-firing on root systems. We use our methods to resolve all of their conjectures on Type A and B root systems, and the grid structures we develop shed some light on the remaining open problems for Types C and D. We also show that the diamond structures found in chip-firing on a line actually extend to a much larger class of bipartite graphs, including all leafless infinite trees and all higher dimensional grid graphs. These diamonds are locally confluent for any choice of labeling, although it is not yet clear if there is a natural way to perform labeled chip-firing on any of these graphs.

Our study eventually takes us further away from labeled chip-firing, ultimately

leaving graphical chip-firing altogether. We prove results about the convexity of the class of M-matrices, which provides us with information about the best choices of M-matrices in M-matrix chip-firing. We also convert our objects of study into the language of poset theory. Chip-firing is one of many combinatorial processes whose configurations form LLD lattices, and the move posets studied throughout the paper correspond to posets of join-irreducible configurations. We interpret the moves of the move poset as colors of an edge coloring on the poset of configurations. This interpretation allows us to prove further results on labeled chip-firing and root system chip-firing, as well as results about shortest paths between domino tilings.

In Chapter 2, we provide some preliminaries on chip-firing and lattices. Most of our results are focused on variants of chip-firing involving different types of graphs, configurations, and firing rules, along with distinguishable chips. Posets and lattices are used later on, when chip-firing and related processes are viewed as posets with special properties.

In Chapter 3, we present the main subject of the thesis: labeled chip-firing. We provide a more illuminating and generalizable proof that labeled chip-firing with an even number of chips produces a sorted final configuration. We introduce the poset of firing moves for unlabeled chip-firing and establish the "diamond lemma:" proving the existence of a small, locally confluent diamond of moves near the end of the process. This diamond, combined with some weak bounds on the position of each chip throughout the process, is enough to prove sorting. We use similar methods to prove sorting on several variants of the 1-dimensional grid with self-loops and varying edge multiplicities. Proving sorting on a graph with a self-loop at each node shows the limits of how effectively a diamond can achieve sorting.

The methods from chapter 3 are extended in three main directions throughout this thesis. In chapters 3, 5, and 7, we use move posets to prove sorting on other variations of the line graph, initial configurations, and Coxeter types. In chapters 4 and 8, we examine properties of move posets. Chapter 4 views move posets in the context of lattice theory, and chapter 8 studies which graphs and initial configurations produce

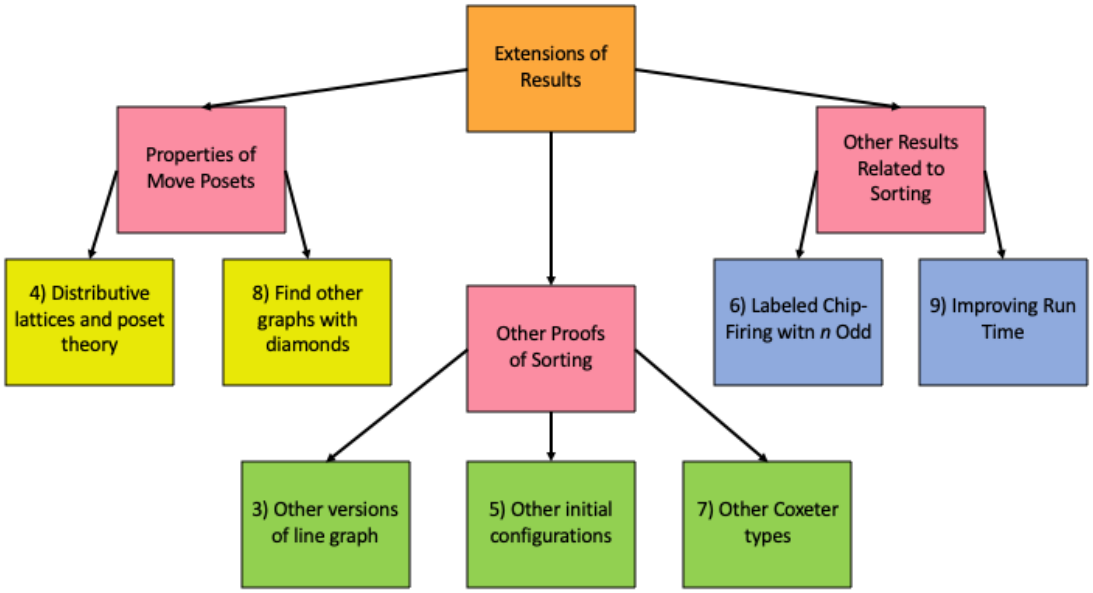


FIGURE 1.1: An outline of Chapters 3-9

move posets with diamonds. Chapters 6 and 9 consider other results related to sorting. Chapter 6 studies a prominent problem in labeled chip-firing in which sorting does not occur, while Chapter 9 uses methods related to move posets to improve the asymptotic run time of labeled chip-firing. These different research directions are outlined in Figure 1.1.

In Chapter 4, we view chip-firing through the lens of poset theory. The configurations of a chip-firing process form a lower locally distributive (LLD) lattice, and the elements of the move poset form a coloring of this lattice, which we call the S-coloring. We show that the move poset is isomorphic to the poset of join-irreducibles of the configuration poset, which we use in many results later in the thesis. We prove properties of S-colorings, which we use to prove results about labeled chip-firing. We also show that these S-colorings exist in many other processes of interest, including processes involving graph orientations, domino tilings, spanning trees, and the sand pile model. We provide a new proof of a result of Saldanha et al. [46] about shortest paths between flip connected domino tilings.

In Chapter 5, we combine the methods of Chapters 3 and 4 to classify labeled chip-configurations on the line as "sorting" or "non-sorting." We show that any connected configuration whose stabilization has a hole, must produce a chip-firing process with a locally confluent grid of moves at the end. If each site has enough firing moves, this grid forms a rectangular diamond, whose exact shape depends on the location of the hole in the final configuration. We use this result to prove sorting for two classes of configurations originally conjectured to sort by Hopkins et al. [27] and Galashin et al. [21], as well as much larger classes of firing-equivalent configurations. We classify every weakly sorted configuration on the line as either sorting or non-sorting, and we provide some results on classifying configurations that are not weakly sorted.

In Chapter 6, we turn to a class of configurations known not to sort: configurations consisting of $2m + 1$ chips at the origin. While an odd number of chips at one site present a "worst-case scenario" for sorting, there are two main conjectures on their behavior: that the probability of sorting converges to $\frac{1}{3}$ as n goes to ∞ (for certain natural probability distributions over firing sequences), and that the maximum number of inversions in the final configuration is at most m [27]. We provide insight on both of these conjectures. We find a way to unify all of the probability distributions used in the $\frac{1}{3}$ conjecture, which provides a clearer path to a proof. We also prove results on the number of inversions that can be created throughout the process and on the structure of the poset of firing moves, which provide insight into both conjectures.

In Chapter 7, we apply our methods to root system central-firing, originally introduced by Galashin et al. [22] [21]. Labeled chip-firing is viewed as a Type A special case of root system central-firing, in which Types B, C, and D allow for more types of firing moves. Galashin et al. provide a conjecture about which cases sort for all Types of central-firing. We use lattice theoretic methods from Chapter 4 to resolve the last remaining open problem on Type B central-firing (the remaining Type A open problems are resolved in Chapter 5). We disprove and modify a conjecture on Type D central-firing, and we use move posets to establish a visual distinction between sorting and non-sorting cases on Types C and D. We also present a rare instance in which we use

labeled chip-firing to prove a result about unlabeled chip-firing.

In Chapter 8, we try to find the most general class of graphs that can produce the diamonds found in Chapter 3. We find that the key to producing such a diamond lies in the number of edges directed "toward" or "away" from the origin at each node. If, for each i, all vertices at a distance i from the origin have the same integer ratio of outgoing to incoming edges, then an initial configuration with a specific number of chips at the origin will lead to a diamond. If we are willing to manipulate multiplicity of each edge in the graph, then many large classes of bipartite graphs can produce diamonds, including a grid graph in any number of dimensions.

In Chapter 9, we view labeled chip-firing from a computer science perspective. Labeled chip-firing is an interesting sorting algorithm with useful applications to questions of confluence, but it is not efficient (its run time is $O(n^3)$). We answer a question of Hopkins [26] about whether it is possible to improve this run time by performing labeled chip-firing on a different graph. We use the diamond from Chapter 3 to show that the time can be reduced to $O(n^2)$ by choosing a different initial configuration and performing some light preprocessing. To further reduce the time down to $O(n \log(n))$, we introduce the general labeled chip-firing process, allowing labeled chip-firing to be performed on a graph that is not on a line. By showing that any sorting network can be simulated by a general labeled chip-firing process, we are able to use a result of [1] to perform labeled chip-firing in $O(n \log(n))$ time.

In Chapter 10, we examine M-matrices. Guzmán and Klivans show that any invertible matrix can be paired with an M-matrix to produce a chip-firing like process involving the subtraction of vectors from a configuration. The search for a natural M-matrix to pair with a given invertible matrix leads to questions about the convexity of the class of M-matrices. We show that the set of $n \times n$ M-matrices is not convex, but that it is star convex with respect to the diagonal matrices, and that it has many large convex subsets.

CHAPTER 2

Preliminaries

2.1 Processes

We define a **process** R as $R = (P, E)$, where P is a set of **configurations** and E is a set of **moves** between the configurations of P. Each move is represented using an ordered pair of configurations. If $(u, v) \in E$ for some $u, v \in P$, we say that there is a move from u to v, or that it is possible to get from u to v in one move. We say that u is the **start point** of move (u, v), and v is the **end point**.

A **cycle** is a sequence of moves $(u_0, u_1), (u_1, u_2), \ldots (u_{k-1}, u_k)$, where the start point of each move is equal to the end point of each previous move, and such that $u_0 = u_k$. If R contains no cycles of length greater than 0, we say that R is **acyclic**.

Define an **instance** $(u_0, u_1), (u_1, u_2), \ldots (u_{k-1}, u_k)$ of an acyclic process R is a sequence of moves in which the start point of each move (except for the first move) is equal to the end point of the previous move. If there exists an instance of process R that begins at configuration u and ends at configuration v, then v is **reachable** from u. A configuration v is a **final configuration** if there is no edge in E containing v as a start point. R is **terminating** if it is acyclic, and if there exists an N such that all instances of R have length less than N.

This thesis deals heavily with processes and their instances, especially chip-firing processes.

2.1.1 Confluence

A process $R = (P, E)$ is **locally confluent** if the following condition is met. For all $u, v, w \in P$ such that $(u, v) \in E$ and $(u, w) \in E$, there exists a $z \in P$ such that $(v, z) \in E$ and $(w, z) \in E$. R is **globally confluent** if for each $u \in P$, there is a unique final configuration v reachable from u.

Newman's Lemma on abstract rewriting systems states that any locally confluent, terminating process must also be globally confluent [41].

2.2 Chip-Firing

Here, we introduce chip-firing processes. For more on chip-firing, see [30].

A **chip-firing process**, or chip-firing game, is a dynamical system on a graph $G = (V, E)$. G may be directed or undirected. Each vertex (or **site**) $v \in V$ is assigned a number of chips, which is generally a non-negative integer. Unless otherwise specified, chips are considered to be indistinguishable, so we are only concerned with the number of chips at each site. A **chip configuration** is a state of the system, specified by the number of chips at each site. A chip configuration is often expressed as a function $c : V \to \mathbb{N}$ such that $c(v)$ represents the number of chips at vertex v.

A site v is **ready to fire** if $c(v) \geq deg(v)$ (if G is undirected) or $c(v) > outdeg(v)$ (if G is directed). A site v that is ready to fire may **fire** by sending one chip along each outgoing edge. This increases the chip count of each neighbor v' of v by the number of edges from v to v', while decreasing the number of chips at v by its degree (resp. outdegree). In the language of Section 2.1, chip-firing on a graph G is a process where the configurations are chip-configurations and the moves are firing moves. An example of a chip-firing process on a graph with 5 vertices is shown in Figure 2.1.

Given a subset $U \subseteq V$, we define U^c ("U complement") to be $V \backslash U$. Given a configuration c and a collection of vertices $U \subseteq V$, U can **cluster fire** if for each vertex $v \in U$, $c(v)$ is greater than or equal to the number of edges from v to U^c. In this case, a cluster fire at U sends one chip along each edge from U to U^c.

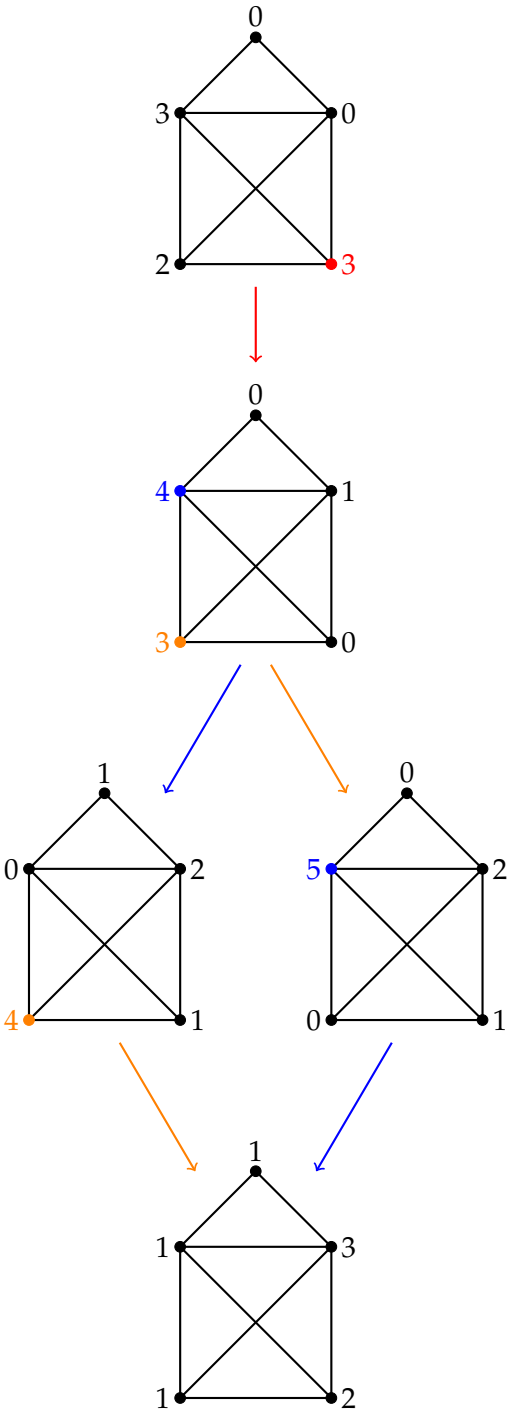


FIGURE 2.1: A chip firing process on a graph with 5 vertices and 8 chips. The process takes 3 moves to terminate: one at the bottom left vertex, one at the top left vertex, and one at the bottom right vertex. The top left and bottom left vertices may fire in either order. The process terminates in the configuration shown at the bottom, in which no vertex is ready to fire.

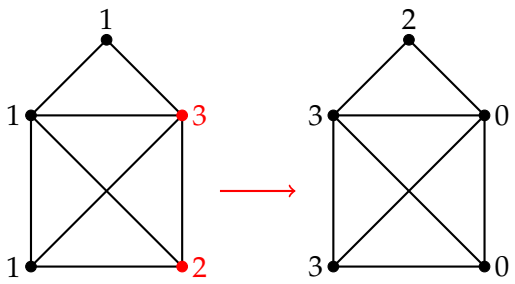


FIGURE 2.2: An example of a cluster fire on a configuration where no individual vertex is ready to fire. The top right and bottom right vertices fire simultaneously, so neither firing site needs to send a chip to the other.

In an undirected graph, cluster firing a set of vertices U is the same as firing every vertex $v \in U$. For every edge e between two vertices $v_1, v_2 \in U$, the cluster fire sends no chips along e, while firing every vertex in U sends one chip from v_1 to v_2, and one chip from v_2 back to v_1, with no net effect. However, it is sometimes possible to cluster fire a set U even if there is no way to legally fire the vertices of U one at a time. For example, the final configuration in Figure 2.1 has no vertices that are ready to fire, but there are multiple cluster fires that can be performed. One example of a cluster fire from that final configuration is shown in Figure 2.2.

2.2.1 Confluence in Chip-Firing

A chip-firing process either terminates in a **stable** configuration in which no site is ready to fire, or it runs forever. Several key results are known about terminating chip-firing processes. First, chip-firing is a locally confluent process: if two sites, a and b, are both ready to fire, then it is possible to fire both vertices in either order. This is because firing at one site cannot decrease the number of chips at any other site, so if site b is ready to fire before a is fired, then it must still be ready to fire after a is fired.

Furthermore, any terminating chip-firing process is globally confluent. This implies that for any configuration c, if there exists a sequence of firing moves starting from c, that terminates at a stable configuration c', then all sequences of firing moves from c must terminate at c'. Furthermore, for any two sequences s_1 and s_2 of firing moves from c to c', and any site v, the number of firing moves at site v in s_1 must be the same as

the number of firing moves at site v in s_2 [8]. Given a chip-configuration c such that the chip-firing process beginning with c terminates, define the **stabilization** $stab(c)$ as the unique stable configuration at which the process terminates.

Confluence is a rather surprising property, and it proves to be very useful in analyzing chip-firing processes. Given an initial configuration c, the process beginning with c is either guaranteed to run forever, or it is guaranteed to terminate in a fixed final configuration $stab(c)$, which depends only on c and not on the potentially large number of possible choices of firing moves to be performed throughout the process. Furthermore, every sequence of moves from c to $stab(c)$ "looks pretty much the same." The order of the firing moves may change, but the same sites must fire the same number of times on any stabilizing sequence of moves starting from c.

Confluence will be a central part of this thesis. While global confluence is proven in chip-firing via a straightforward local confluence property, there are many similar processes that exhibit global confluence without local confluence. These processes include labeled chip-firing [27] [31], root system chip-firing [22] [21], and flow-firing [19]. One primary goal of this thesis is to develop methods to prove global confluence when local confluence is not present.

2.2.2 Chip-Firing on the Line

Much of this thesis is concerned with labeled chip-firing, a variant of chip-firing with distinguishable chips that happens on a 1-dimensional grid graph. Many of our results make use of results from traditional, unlabeled chip-firing on the 1-dimensional grid.

Chip-firing on the 1-dimensional grid, or chip-firing on the line, as we'll often call it, was first studied by Anderson et al. in 1989 [2], before the concept of chip-firing on more general graphs was invented. n chips are placed at position 0, and all sites are allowed to fire until completion.

Anderson et al. prove that the process must terminate, and that the final configuration does not depend on the order of the firing moves. Furthermore, they prove the form of the final configuration by choosing firing moves that lead to a certain pattern.

Theorem 2.2.1 ([2] Theorem 2)**.** *A chip-firing process on the line beginning with* $2m+1$ *chips at site 0, terminates with 1 chip at each site from* $-m$ *through* m*.*

We sketch the proof here. Because chip-firing is globally confluent, we need to find a single sequence of firing moves that leads to the desired final configuration. They consider a sequence that leads to the firing intermediate configurations:

$$
\begin{array}{ccccccccc}
 & & & 2m+1 & & & & & \\
 & & 1 & 2m-1 & 1 & & & & \\
 & 1 & 1 & 2m-3 & 1 & 1 & & & \\
1 & 1 & 1 & 2m-5 & 1 & 1 & 1 & & \\
 & & & \vdots & & & & &
\end{array}
$$

To get from one of the above configurations to the next, the following pattern is followed (replacing the 3 in the first row with any odd number).

$$
\begin{array}{ccccccccccc}
 & 1 & 1 & 1 & 1 & 3 & 1 & 1 & 1 & 1 & \\
 & 1 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 & \\
 & 1 & 1 & 2 & 0 & 3 & 0 & 2 & 1 & 1 & \\
 & 1 & 2 & 0 & 2 & 1 & 2 & 0 & 2 & 1 & \\
 & 2 & 0 & 2 & 0 & 3 & 0 & 2 & 0 & 2 & \\
1 & 0 & 2 & 0 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \\
1 & 1 & 0 & 2 & 0 & 3 & 0 & 2 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 2 & 1 & 2 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
$$

Because there is always at least 1 chip at the origin in the odd case, the same procedure can be done with $2m$ chips, leading to the same final configuration, but with 0 chips at the origin.

Anderson et al. also show that the number of firing moves needed for the process to terminate is $m(m+1)(2m+1)/6$. Furthermore, it is possible to compute the number of firing moves at each site, which is proved in [30].

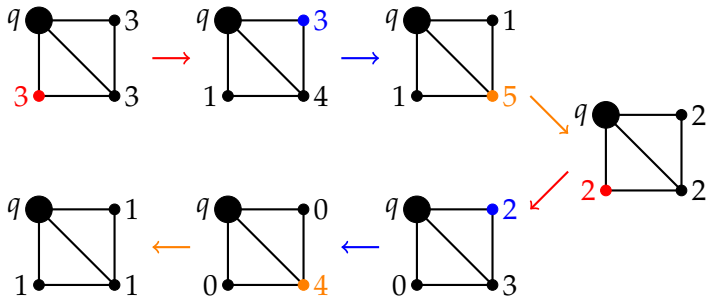


FIGURE 2.3: An instance of a chip-firing process on a graph with a sink q. All vertices except for the sink fire twice each, and the process terminates in a critical configuration with 1 chip at each non-sink site.

Theorem 2.2.2. *In a chip-firing process on the line beginning with* $2m$ *or* $2m+1$ *chips at site 0, the number of firing moves at each site* i $(-m \leq i \leq m)$ *is equal to* $\binom{m-|i|+1}{2}$.

Both Theorems 2.2.1 and 2.2.2 are relevant to labeled chip-firing, in which the final configuration and number of moves at each site are used to impose an order on the moves of the chip-firing process.

2.2.3 Sinks and Critical Configurations

Any chip-firing process on a finite graph can be turned in to a terminating process by including a **sink vertex**. A sink is a vertex q that can receive chips from its neighbors, but does not fire. Any chip-firing process on a connected graph with a sink must terminate, as the number of chips at all vertices other than the sink is permanently reduced every time a neighbor of the sink is fired [30]. An instance of a chip-firing process on a graph with a sink is shown in Figure 2.3.

While every chip-firing process on a graph with a sink eventually leads to a stable configuration, not all stable configurations can be reached from arbitrarily large configurations. In particular, a configuration c is known as **critical** if:

(1) c is stable.

(2) c reachable from a configuration b in which every non-sink site is ready to fire.

In general, not all stable configurations are critical. Specifically, the number of critical configurations on a connected graph G with a sink q, is equal to the number of spanning trees of G (see, for example, [39]). Note that this equality holds for any choice of the sink vertex q.

While critical configurations are mentioned only briefly in this thesis (see the end of Section 5.3), they are crucial to the study of chip-firing. In addition to the spanning trees of G, critical configurations are also in bijection with the **superstable configurations** on G. Superstable configurations are configurations from which no firing moves or cluster firing moves are available. Critical configurations also form the elements of the **sandpile group**, in which two critical configurations c and d can be added with the operation $c \oplus d = stab(c + d)$.

The elements of the sandpile group are representative elements of equivalence classes of an equivalence relation. For every configuration c, there is a unique critical configuration that can be reached from c through some sequence of firing moves, possibly including firing the sink. We say that $c \equiv d$ if the same critical configuration can be reached from both c and d in this way. In this case, we say that c is **firing equivalent** to d. The sandpile group has been analyzed in a variety of contexts, see [30] and [36] for more details.

In Chapter 5, we examine certain classes of firing-equivalent configurations on a 1-dimensional grid graph. The stable configurations in these classes are then compared to critical configurations of a cycle graph.

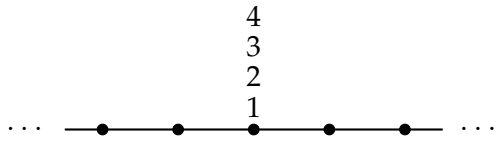
2.3 Labeled Chip-Firing

Much of this thesis is based on labeled chip-firing, a variant of chip-firing on a 1-dimensional grid graph in which the chips are distinguishable. Labeled chip-firing was first introduced by Hopkins, McConville, and Propp [27].

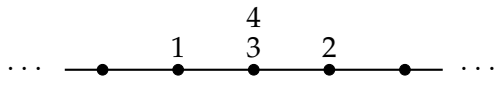
Labeled chip-firing is defined formally as follows. Consider the infinite path graph (or 1-dimensional grid) on $\mathbb{Z}$, where each integer i is connected by a single undirected edge to both $i - 1$ and $i + 1$. Place n chips, labeled from 1 to n, at site 0. A firing move

consists of choosing two chips labeled a and b ($a < b$) at a common site i. Chip a is sent to site $i - 1$, while chip b is sent to site $i + 1$. The process is repeated until all chips are at distinct sites, and thus no further firing moves may be performed. Example 2.3.1 shows a complete firing sequence for $n = 4$.

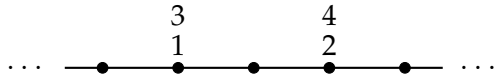
Example 2.3.1. We begin with chips labeled 1 through 4 at the origin.



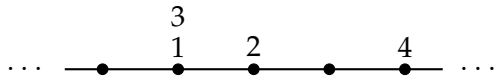
We can then choose to fire any pair of chips at the origin. We choose chips 1 and 2, sending 1 to the left and 2 to the right.



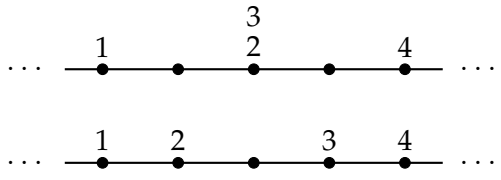
We now fire chips 3 and 4:



There are now two sites that can fire. We choose to fire 2 and 4.



This leaves two more firing moves:

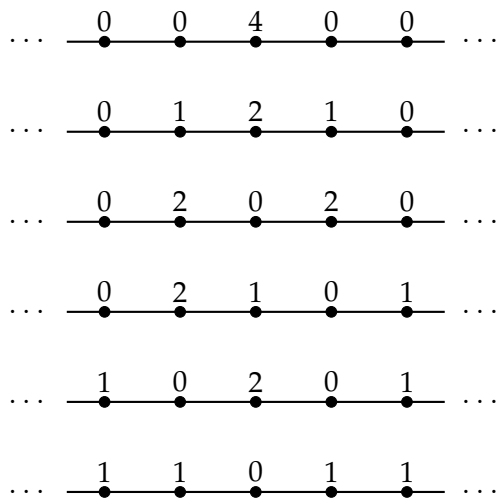


There are no more available firing moves, so this is a final configuration.

Note that in Example 2.3.1, the chips are in sorted order from left to right. In fact, this system was shown in [27] and [31] to always terminate in a unique final configuration, in which all chips end in sorted order, as long as the number of chips n is even.

In Chapter 3, we reproduce the proof from [31] that labeled chip-firing with n even always leads to a sorted configuration. Our proof of the sorting result involves relating the labeled chip-firing process to the unlabeled chip-firing process on the line. The corresponding unlabeled chip-firing process on the line is precisely the chip-firing process considered by Anderson et al. [2]. Place n (indistinguishable) chips at site 0. A firing move consists of choosing two chips at a common site i. One chip is sent to site $i-1$, while the other chip is sent to site $i+1$. The process is repeated until all chips are at distinct sites, and thus no further firing moves may be performed.

For any instance of the labeled chip-firing process, we obtain an instance of the unlabeled chip-firing process by forgetting the labels of the chips. The "label-free" version of Example 2.3.1 looks as follows, where the integer value at each vertex indicates the number of chips at that vertex.



Performing any labeled chip-firing move corresponds to a single firing move on the underlying unlabeled configuration.

The unlabeled representation is particularly useful to us because the unlabeled process satisfies the properties of ordinary chip-firing. In particular, unlabeled chip-firing

is locally confluent. Thus, for any instance of labeled chip-firing, the corresponding unlabeled chip-firing process is globally confluent, and the total number of chips at each site in the final configuration will always be the same, see Theorem 2.2.1. Moreover, the total number of firing moves at each site will always be the same, see Theorem 2.2.2.

2.4 Posets and Lattices

In Chapter 4, we view chip configurations in a chip-firing process as elements of a partially ordered set.

A partially ordered set, or **poset**, is a set P with a relation $\leq$. The relation must satisfy the following properties:

(1) Reflexivity: For all $u \in P$, $u \leq u$.

(2) Antisymmetry: If $u \leq v$ and $v \leq u$, then $u = v$.

(3) Transitivity: If $u \leq v$ and $v \leq w$, then $u \leq w$.

If $u \leq v$, then we also say $v \geq u$. If $u \leq v$ and $u \neq v$, we say $u < v$ and $v > u$. If $u < v$, and in addition, there does not exist a w such that $u < w < v$, we say that v **covers** u or v is an **upper cover** of u. We similarly say that u is a **lower cover** of v, and denote this by $u \lessdot v$ or $v \gtrdot u$.

Given a poset P, a poset P' is a **subposet** of P if $P' \subseteq P$ and for all $u, v \in P'$, $u \leq v$ in P' whenever $u \leq v$ in P. A **chain** is a sequence of elements $U = (u_0, u_1, \ldots u_k)$ of P such that for each $1 \leq i \leq k$, $u_{i-1} \leq u_i$. Note that for any chain U and elements $u, v \in U$ such that $u \neq v$, either $u \leq v$ or $v \leq u$. A **maximal chain** is a chain $U = (u_0, u_1, \ldots u_k)$ such that u_0 is a minimal element, u_k is a maximal element, and for each $1 \leq i \leq k$, $u_{i-1} \lessdot u_i$.

P is **ranked** if every maximal chain has the same length. Define the **height** of a ranked poset to be the length of each maximal chain.

Given a poset P, we can represent P with a graph G, which is embedded in the plane in a diagram known as a **Hasse diagram**. Each element $u \in P$ is represented by a vertex in G. Given two elements $u, v \in P$, if $u \gtrdot v$, then there is an edge from u to v in G, and u

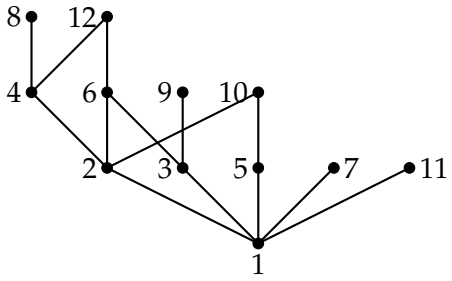


FIGURE 2.4: Let $P = [12]$, with the relation $u \leq v$ if u divides v. The Hasse diagram of P is shown here.

is drawn above v in the Hasse diagram. An example of a poset Hasse diagram is shown in Figure 2.4. Given an integer n, define $[n] = \{1, 2, \ldots n\}$. The poset in Figure 2.4 is defined on the set $[12]$.

Consider an element u in poset P. If there does not exist a v such that $v \geq u$, then u is a **maximal** element of P. If u is the only maximal element of P, then u is the **maximum** element of P. A maximum element $u \in P$ satisfies $u \geq v$ for all v in P. If P has a maximum element, the maximum element is denoted by $\mathbf{1}$.

Similarly define **minimal** and **minimum** elements. A minimum element $u \in P$ satisfies $u \leq v$ for all v in P. If P has a minimum element, the minimum element is denoted by $\mathbf{0}$.

2.4.1 Lattices

Given elements u, v, w of a poset P, if $w \geq u$ and $w \geq v$, then w is an **upper bound** of u and v. If, in addition, for any upper bound z of u and v, we have $z \geq w$, then w is the **least upper bound**, or **join** of u and v. This is denoted by $u \vee v = w$. Similarly define **lower bound** and **greatest lower bound**. The greatest lower bound of u and v is also known as the **meet** of u and v, and is denoted $u \wedge v$.

We can similarly define the join or meet of a subset $P' \subseteq P$. The join of P' (denoted $\bigwedge P'$) is the least element u such that for all $v \in P'$, $u \geq v$, if a unique least element exists. Similarly, the meet of P' (denoted $\bigvee P'$) is the greatest element u such that for all $v \in P'$, $u \leq v$, if a unique greatest element exists.

If, for all pairs $u, v \in P$, the bounds $u \vee v$ and $u \wedge v$ both exist, then P is a **lattice**. Every finite lattice has a maximum and a minimum element. If P is a lattice, then an element $u \in P$ is a join-irreducible element, or simply a **join-irreducible**, of P, if there do not exist elements $v \neq u$ and $w \neq u$ such that $u = v \vee w$. Similarly, u is a meet-irreducible element, or **meet-irreducible**, if there do not exist elements $v \neq u$ and $w \neq u$ such that $u = v \wedge w$. In a finite lattice, an element u is a join-irreducible if and only if it covers exactly one other element, and u is a meet-irreducible if and only if it is covered by exactly one other element. Given a poset P, define $Irr(P)$ to be the subposet of join-irreducibles of P.

A poset P is a **distributive lattice** if P is a lattice, and for all $u, v, w \in P$, the following is true:

$$u \wedge (v \vee w) = (u \wedge v) \vee (u \wedge w)$$

Equivalently, a lattice P is a distributive lattice if for all $u, v, w \in P$, the above statement holds, but with the $\vee$ and $\wedge$ symbols swapped. In a distributive lattice, each element has a unique representation as a join of join-irreducibles, and also a unique representation as a meet of meet-irreducibles.

Furthermore, every distributive lattice is ranked, with height equal to the number of join-irreducibles plus one (which is equal to the number of meet irreducibles plus one). If $u \gtrdot v$, and P_u and P_v are the sets of join-irreducibles such that $u = \bigvee P_u$ and $v = \bigvee P_v$, then $P_v \subseteq P_u$ and $|P_u \backslash P_v| = 1$. Again, a comparable result holds for meet representations. Hasse diagrams for several types of posets and lattices are shown in Figure 2.5.

While every distributive lattice P has a corresponding poset of join-irreducibles P', it is also possible to convert from any poset P' to a corresponding distributive lattice P, such that P' is the poset of join-irreducibles of P.

Given a poset P, a subposet P' is an **order ideal** of P if for all $u \in P'$ and all $v \leq u$ in P, v is also in P'. Define an ordering of the order ideals of P. Given order ideals P' and P'', we say $P' \leq P''$ if $P' \subseteq P''$. Define $J(P)$ to be the poset of order ideals of P.

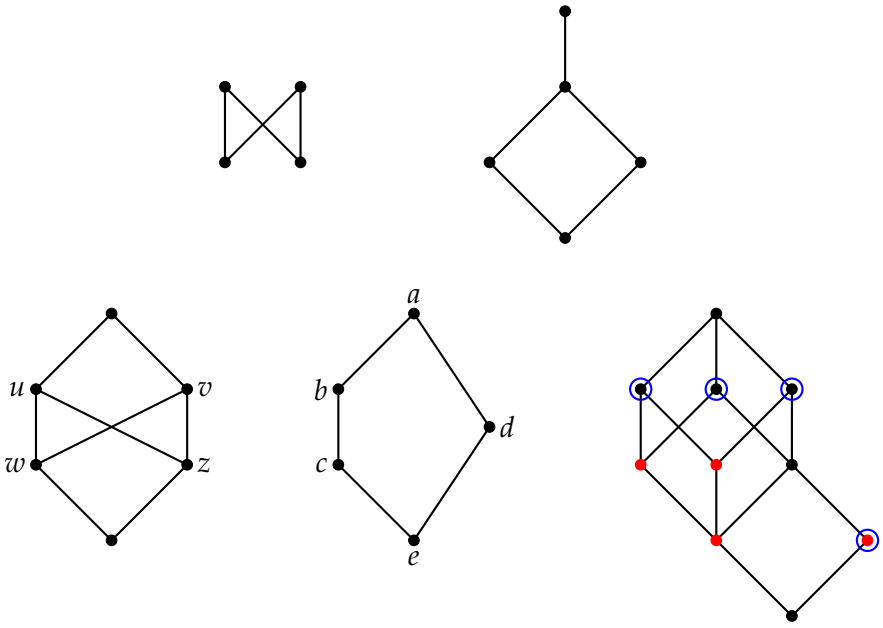


FIGURE 2.5: Hasse diagrams for 5 posets are shown. Top Left: a poset with no minimum or maximum element. Top Right: the poset of configurations for the chip-firing process from Figure 2.1. For configurations u and v, $u \geq v$ if it is possible to get from u to v through some sequence of firing moves. For more examples of posets generated by chip-firing processes, see Example 4.7.3, Example 4.7.4, and Figure 4.10. Bottom Left: a poset that is not a lattice. Elements w and z are both lower bounds for u and v, but there is no greatest lower bound. Bottom Center: a lattice that is not distributive. $b \wedge (c \vee d) = b$, but $(b \wedge c) \vee (b \wedge d) = c$. Bottom Right: a distributive lattice. The join-irreducibles are drawn in red, and the meet-irreducibles are circled in blue.

Birchoff's Theorem, also known as the Fundamental Theorem of Finite Distributive Lattices, shows that $J(\cdot)$ and $Irr(\cdot)$ are, in some sense, inverses of each other.

Theorem 2.4.1 (Birkhoff [6]). *Up to isomorphism, a finite lattice is distributive if and only if it can be expressed as $J(P)$ for some finite poset P. Moreover, $L \cong J(Irr(L))$ for every distributive lattice L and $P \cong Irr(J(P))$ for every poset P.*

Here, $\cong$ denotes an isomorphism. Two posets P and P' are **isomorphic** if there exists a bijection $f : P \to P'$ such that for all $u, v \in P$, $u \leq v$ in P if and only if $f(u) \leq f(v)$ in P'.

2.4.2 Connection to Chip-Firing

Beginning in Chapter 4, we view the set of chip-firing configurations on a graph G (in addition to configurations of other processes) as a poset. Given two chip configurations c and c', we say that $c \geq c'$ if it is possible to get from c to c' through a sequence of firing moves. Posets generated by chip-firing processes in this way are known to be lower locally distributive (LLD) lattices, meaning that the posets have a diamond property closely related to local confluence. LLD lattices maintain the property of distributive lattices that each element has a unique representation as a join of join-irreducibles, although an element may or may not have a unique representation as a meet of meet irreducibles.

We define LLD lattices in more detail in Chapter 4, where we use properties of LLD lattices to provide more insight into chip-firing processes.

2.5 Root Systems

A labeled chip configuration with n chips labeled from 1 to n can be expressed as a vector as follows. Let $v = (v_1, v_2, \ldots v_n)$, where for each i, v_i denotes the position of chip i. If $v_i = v_j$ for some $i < j$, then we can fire by increasing v_j by 1 and decreasing v_i by 1. In other words, if the vector $e_j - e_i$ is orthogonal to the vector v, then we can add

$e_j - e_i$ to v. Here, e_i is the unit vector where the i^{th} component is equal to 1, and all other components are equal to 0.

Galashin et al. [22] observe that vectors of the form $e_j - e_i$, where $1 \leq i < j \leq n$, are all positive roots of a root system of Type A_{n-1}. Furthermore, the standard initial configuration in labeled chip-firing corresponds to v as the all zeros vector, which is a fundamental weight of a Type A_{n-1} root system. This leads to the question of whether other firing moves based on positive roots of root systems lead to confluent processes.

We begin by defining root systems, and from there, we can find the positive roots and fundamental weights for each type of root system, and show how they correspond to firing moves and initial configurations.

2.5.1 Defining Root Systems

Consider an n-dimensional vector space V with inner product $\langle \cdot, \cdot \rangle$. Throughout this thesis, we always let $V = \mathbb{R}^n$. Given a nonzero vector $\alpha \in V \backslash \{0\}$, define its **covector** to be $\alpha^\vee = \frac{2\alpha}{\langle \alpha, \alpha \rangle}$. Define the **reflection** of v across the hyperplane orthoganal to α as $s_\alpha(v) = v - \langle v, \alpha^\vee \rangle \alpha$.

Definition 2.5.1 ([28])**.** A **root system** is a finite collection $\Phi \subseteq V \backslash \{0\}$ of nonzero vectors such that:

(1) Φ is finite and spans V.

(2) If $\alpha \in \Phi$, then the only multiples of α in Φ are $\pm\alpha$

(3) If $\alpha \in \Phi$, the reflection s_α leaves Φ invariant.

(4) If $\alpha, \beta \in \Phi$, then $\langle \beta, \alpha \rangle \in \mathbb{Z}$.

Now, if we fix a root system Φ in V, the vectors $\alpha \in \Phi$ are called **roots**. The **root lattice** Q is the set of all integer combinations of vectors in Φ. The **Weyl group** of Φ, denoted by W, is the group generated by the reflections s_α for each root α.

Fix a linearly independent set $\Delta = \{\alpha_1, \alpha_2, \ldots \alpha_n\} \subseteq \Phi$ of roots. $\alpha_1, \alpha_2, \ldots \alpha_n$ are **simple roots** if every $\alpha \in \Phi$ can be expressed as an integer combination of $\alpha_1, \alpha_2, \ldots \alpha_n$ such

that either all coefficients are nonnegative or all coefficients are nonpositive. Simple roots form a basis of V.

Any root α that can be expressed as a nonnegative integer combination of simple roots is known as a **positive root**, and any other root is a **negative root**. Since positive roots correspond to firing moves in a central-firing process, we choose the set Δ of simple roots so that certain properties, such as larger chips being fired to the right, are maintained.

The weight lattice of a root system Φ is another collection of vectors with properties closely tied to Φ. Given a root system Φ, the **weight lattice** P is defined by

$$P = \{v \in V : \langle v, \alpha^\vee \rangle \in \mathbb{Z} \text{ for all } \alpha \in \Phi\}.$$

For each $i \in [n]$, define the **fundamental weight** $\omega_i \in V$ such that $\langle \omega_i, \alpha_j^\vee \rangle = \delta_{i,j}$, where $\delta_{i,j}$ denotes the Kronecker delta. Denote the set of fundamental weights by $\Omega = \{\omega_1, \omega_2, \ldots \omega_n\}$. The **Weyl vector** is defined as $\rho = \sum_{i=1}^{n} \omega_i$.

2.5.2 Types A, B, C, and D

There are four main classical types of root systems, known as Type A, B, C, and D. Each type corresponds to a specific Φ in n dimensions for each value of n. The realizations of Φ are as follows.

- A_n: The roots of Φ are the vectors in $\mathbb{R}^{n+1}$ of the form $e_j - e_i$ for $i, j \in [n+1]$, $i \neq j$. We choose simple roots $\alpha_i = e_{i+1} - e_i$ for $i \in [n]$, so that the positive roots are the roots $e_j - e_i$ for $i < j$. The weight lattice P is the set of all integer vectors in $\mathbb{R}^{n+1}$ whose coordinates sum to 0. Note that the span of Φ (as well as the span of P) is the n-dimensional subspace of R^{n+1} of vectors whose coordinates sum to 0.

- B_n: The roots of Φ are the vectors in $\mathbb{R}^n$ of the form $\pm e_i \pm e_j$ for $i, j \in [n]$, $i < j$, and $\pm e_i$ for $i \in [n]$. We choose simple roots $\alpha_1 = e_1$ and $\alpha_i = e_i - e_{i-1}$ for $2 \leq i \leq n$, so that the positive roots are the roots $e_j \pm e_i$ for $i < j$, and e_i for $i \in [n]$. The weight lattice $P = \mathbb{Z}^n \cup (\mathbb{Z} + \frac{1}{2})^n$, where $(\mathbb{Z} + \frac{1}{2})$ is the set of half-integers.

- C_n: The roots of Φ are the vectors in $\mathbb{R}^n$ of the form $\pm e_i \pm e_j$ for $i, j \in [n]$, $i < j$, and $\pm 2e_i$ for $i \in [n]$. We choose simple roots $\alpha_1 = 2e_1$ and $\alpha_i = e_i - e_{i-1}$ for $2 \leq i \leq n$, so that the positive roots are the roots $e_j \pm e_i$ for $i < j$, and $2e_i$ for $i \in [n]$. The weight lattice $P = \mathbb{Z}^n$.

- D_n: The roots of Φ are the vectors in $\mathbb{R}^n$ of the form $\pm e_i \pm e_j$ for $i, j \in [n]$, $i > j$. We choose simple roots $\alpha_1 = e_1 + e_2$ and $\alpha_i = e_i - e_{i-1}$ for $2 \leq i \leq n$, so that the positive roots are the roots $e_j \pm e_i$ for all $i < j$. The weight lattice $P = \mathbb{Z}^n \cup (\mathbb{Z} + \frac{1}{2})^n$, where $(\mathbb{Z} + \frac{1}{2})$ is the set of half-integers.

Note that our choices of simple and positive roots are different from the choices made in [21]. We choose our simple and positive roots in this way to remain consistent with our convention that firing moves of two chips at the same site send larger chips to the right and smaller chips to the left. This convention was reversed in [21], which resulted in chips being sorted with the smallest chips on the right and largest chips on the left.

2.5.3 Central-Firing

Now that we have defined each classical type of root system, we can define the initial configurations and firing moves for central-firing on each root system.

A weight $\lambda = (\lambda_1, \lambda_2, \ldots \lambda_n)$ defines a labeled chip configuration with n chips. This allows us to express each fundamental weight as a chip configuration.

- For Φ of Type A_{n-1}, the fundamental weight ω_i, $1 \leq i \leq n-1$ corresponds to a configuration with the smallest $n - i$ chips at the origin, and the largest i chips at site 1.

- For Φ of Type B_n, the fundamental weight ω_i, $1 \leq i \leq n-1$ corresponds to a configuration with the smallest $n - i$ chips at the origin, and the largest i chips at site 1. ω_n corresponds to all chips at site $\frac{1}{2}$.

- For Φ of Type C_n, the fundamental weight ω_i, $1 \leq i \leq n$ corresponds to a configuration with the smallest $n-i$ chips at the origin, and the largest i chips at site 1.

- For Φ of Type $z = D_n$, the fundamental weight ω_i, $1 \leq i \leq n-2$ corresponds to a configuration with the smallest $n-i$ chips at the origin, and the largest i chips at site 1. ω_{n-1} corresponds to the smallest chip at site $-\frac{1}{2}$, and all other chips at site $\frac{1}{2}$. ω_n corresponds to all chips at site $\frac{1}{2}$.

Given a root system Φ with weight lattice P, and two weights $\lambda, \lambda' \in P$, we say that λ' is **strongly reachable** from λ if $\lambda' = \lambda + \alpha$ for some positive weight α such that $\langle \lambda, \alpha^\vee \rangle = 0$. λ' is **reachable** from λ if there exists some sequence of weights $(\lambda = \lambda^0, \lambda^1, \ldots \lambda^k = \lambda')$ such that for each $1 \leq i \leq k$, λ^i is strongly reachable from λ^{i-1}.

Given an initial weight ω, define the **configuration poset** P_ω to be the set of weights reachable from ω, ordered by reachability (so that $\lambda \geq \lambda'$ if λ' is reachable from λ). Define the **central-firing process** on Φ with initial weight ω to be $R(\Phi, \omega) = \Big(P_\omega, E(P_\omega)\Big)$, where $E(P_\omega)$ is the cover relation of P_ω. Galashin et al. show that P_ω is finite for any Φ and ω [21].

The **moves** of a central-firing process are pairs of weights (λ, λ') such that λ covers λ' in P_ω. These moves correspond to four types of operations on the chips:

(a) For $i < j$, if chips i and j are in the same position, move chip i one step to the left, and chip j one step to the right.

(b) For $i \in [n]$, if chip i is at the origin, move it one step to the right.

(c) For $i \in [n]$, if chip i is at the origin, move it two steps to the right.

(d) For $i \neq j$, if chips i and j are in opposite positions ($v_i = -v_j$), move both chips one step to the right.

Galashin et al. [21] show that for any weights λ and λ' such that $\lambda \geq \lambda'$, it is possible to get from λ to λ' by applying

- (a) operations, if Φ is of Type A_{n-1}.
- (a), (b), and (d) operations, if Φ is of Type B_n.
- (a), (c), and (d) operations, if Φ is of Type C_n.
- (a) and (d) operations, if Φ is of Type D_n.

Because every move can be expressed as one of these four types of operations, we will use the term "move" instead of "operation" throughout this chapter.

Note that (a) moves are the same moves used in traditional labeled chip-firing. Thus, labeled chip-firing is a Type A special case of central-firing.

For more information on root systems, see [28], [9], or [7].

CHAPTER 3

Labeled Chip-Firing

3.1 Introduction

This chapter is based heavily on Klivans and Liscio's "Confluence in Labeled Chip-Firing" [31].

This chapter is concerned with a labeled variant of the chip-firing process as defined by Hopkins, McConville, and Propp [27]. For background on unlabeled chip-firing, see Chapter 2.

The labeled chip-firing process is a variation of the chip-firing process on the infinite path graph, where the chips are made to be distinguishable. Each chip is given an integer label, and additional rules govern which chip moves in each direction during a firing move. If a node has at least as many chips as it has neighbors, it can fire. The difference in the labeled case is that two distinct chips at the node are chosen and the chip of larger label moves to the right while the chip of smaller label moves to the left, see Example 2.3.1.

In [27], it was shown that, for an even number of chips, labeled chip-firing terminates in a unique configuration regardless of the order in which nodes fire and regardless of the choice of chips made at each node. Moreover, in the unique terminal configuration, the chips are in sorted order. The sorting result for labeled chip-firing from [27] is particularly notable for proving global confluence even though local confluence, and thus

Newman's Lemma, does not apply. Without this tool, the labeled case proved significantly more challenging to establish.

In [31], we give a novel, more general and more illuminating proof of global confluence for labeled chip-firing and related systems, which is reproduced here. Our proof is based on the analysis of a poset of firing moves. Figures 3.1 and 3.2 visually demonstrate the structure of confluent versus non-confluent labeled chip-firing processes that arise if the initial configuration has an odd number of chips. The existence of the diamond shape at the bottom of the Haase diagram in the even case is crucial for confluence.

There have been attempts to generalize the results of [27] including several conjectures from [26] and [27] in which labeled chip-firing is extended to modified versions of the 1-dimensional grid graph. We provide proofs of some of those conjectures in [31], which we reproduce in this chapter. In Chapter 5 of this thesis, we use the methods developed in [31] and this chapter to prove sorting for other classes of initial configurations on a line. Hopkins et al. [27] establish conjectures on what can happen with non-sorting initial configurations, which are studied further by Klivans and Liscio [31] and in Chapter 6 of this thesis. Hopkins [26] and Chapter 9 of this thesis use alternative versions of the line graph, alternative initial configurations, and graphs inspired by sorting networks to produce labeled chip-firing processes with more efficient run times.

Additionally, Galashin et al. [21, 22] treat the labeled chip-firing problem as chip-firing on a Type A root system, and then generalize the problem to apply to other types of root systems and more general classes of firing moves. For more on root system chip-firing, see Chapter 7. In Chapter 8, we examine the properties of the move poset that lead to confluence in labeled chip-firing, and we find a much larger class of chip-firing processes that produce these properties. In [19], global confluence is proved without local confluence for higher dimensional forms of chip-firing. Our methods allow us to solve many of the above problems via a unified methodology.

In Section 3.2, we present the proof that labeled chip-firing sorts. In Section 3.3, we apply these methods to prove a series of related conjectures on sorting via chip-firing on modified versions of the one-dimensional grid graph.

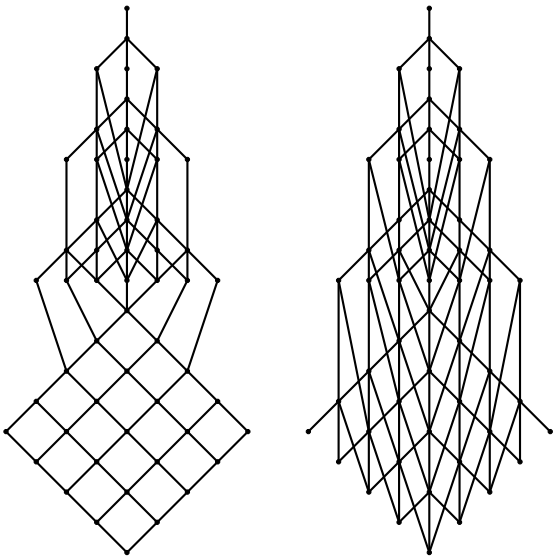


FIGURE 3.1: Move poset for $n = 10$ (left) and $n = 11$ (right).

3.2 Sorting

For background on labeled chip-firing, see Chapter 2. To prove sorting for the labeled chip-firing process, we first reduce it to its underlying unlabeled chip-firing process. Given a labeled chip-firing process beginning with n labeled chips at the origin, the corresponding unlabeled chip-firing process takes place on the same line graph beginning with n unlabeled chips at the origin.

The following results appear in [2] and [8], see also [30, Chapter 5], for the unlabeled chip-firing process. Our discussion shows that the results hold for the labeled chip-firing process as well.

Theorem 3.2.1. *The chip-firing process with $2m$ chips at the origin terminates at the final configuration with a single chip at every position from $-m$ to -1, and from 1 to m.*

Theorem 3.2.2. *Over the course of the chip-firing process with $2m$ chips at the origin, the number of firing moves at site k is $\binom{m-|k|+1}{2}$ for $-m \leq k \leq m$.*

For the rest of this section, our emphasis will be on the labeled chip-firing process beginning with $2m$ chips at the origin. We can now state our main result, which is a

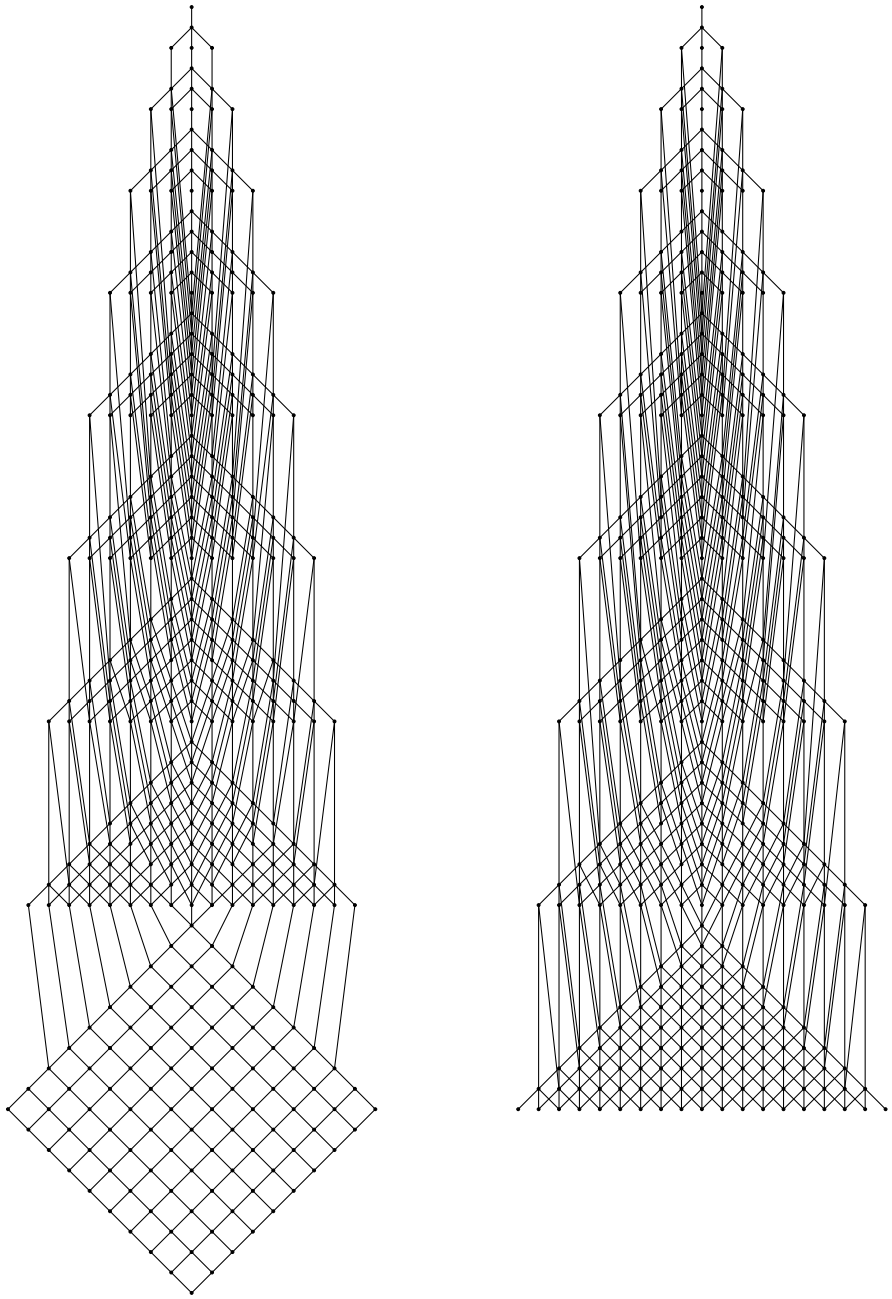


FIGURE 3.2: Move poset for $n = 20$ (left) and $n = 21$ (right)

new proof of the sorting property. Let the total number of chips be $n = 2m$ and label the chips $-m, -m+1, \cdots, -1, 1, 2, \cdots, m$.

Theorem 3.2.3 (**Labeled chip-firing sorts** [27, Theorem 14])**.** *The labeled chip-firing process with $2m$ chips at the origin terminates at the final configuration with a single chip at every position from $-m$ to -1, and from 1 to m. Furthermore, the final position of each chip is equal to its label.*

The novelty of Theorem 3.2.3 in comparison to Theorem 3.2.1 is that each labeled chip must end at the position equal to its label, which is equivalent to the statement that the chips end up in sorted order. We provide a new proof of Theorem 3.2.3 that exploits the relationship between the labeled and unlabeled chip-firing processes on the line.

In light of Theorem 3.2.2, it is well defined to consider "the j^{th} to last firing move at site k," because the number of firing moves at site k is an invariant of the process. We denote

$$k_j = \text{ the } j^{th} \text{ to last firing move at site } k$$

Further, even though some choices can be made about the order in which the moves are performed, there are certain orderings to the moves that must exist for any possible sequence of choices. For example, when $n \geq 5$, the first two moves at site 0 must occur before the first move at site 1, because site 1 begins the process with no chips and must get its first two chips from site 0. However, the third move at site 0 and the first move at site 1 may occur in either order. If we let $j_0 = \binom{m+1}{2}$ and $j_1 = \binom{m}{2}$, then this says that moves 0_{j_0} and 0_{j_0-1} must occur before move 1_{j_1}, but 0_{j_0-2} and 1_{j_1} can happen in either order.

We are now ready to define the main object of study of the chapter (and indeed, of much of the thesis). For the chip-firing process with n chips at the origin of the one-dimensional line, the **move poset** P is the poset on moves k_j where

$$(k_1)_{j_1} \geq (k_2)_{j_2} \text{ if move } (k_1)_{j_1} \text{ must occur before move } (k_2)_{j_2}$$

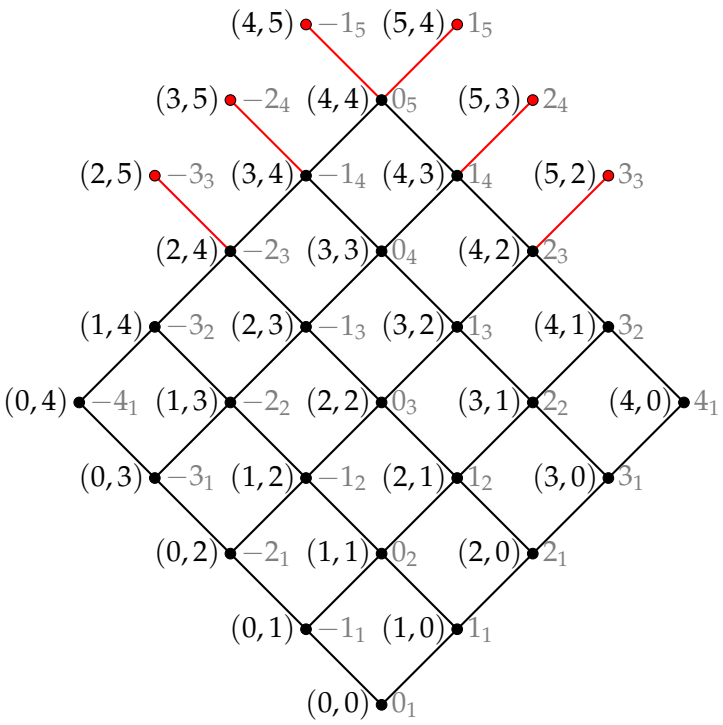


FIGURE 3.3: Bottom of the firing move poset, $n = 10$

It is worth noting that P is only dependent on the unlabeled chip-firing process. The inclusion of labels does not affect the relative order in which firing moves can occur at different sites.

Example 3.2.4. Let $n = 10$. Figure 3.3 shows a bottom portion of the Haase diagram of P.

The proof of Theorem 3.2.3 consists of three main steps:

(1) Prove the last moves in the process follow a locally confluent grid structure.

(2) Bound the positions that chips can reach throughout the firing process.

(3) Combine (1) and (2) to uniquely constrain a chip's final location.

Our first goal is to show that the bottom portion of the Hasse diagram of P always has the grid structure shown in Figure 3.3. For general n, the diagram will consist of an m by m black diamond. At all but the two leftmost and two rightmost vertices at

the top of the diamond, there is an additional edge oriented away from the diamond, drawn in red in Figure 3.3. The coordinates $(0,0)$ are assigned to the bottom vertex in the diamond. A move of one unit up and to the right corresponds to an increase of 1 in the first coordinate, while a move of one unit up and to the left corresponds to an increase of 1 in the second coordinate.

Each column corresponds to a site, with the last move at that site appearing on the bottom border of the diamond, and with earlier moves moving up the diamond. We will often use "firing move (x,y)" or simply "(x,y)" to refer to the firing move that aligns with coordinates (x,y) in the diagram.

The proof of the main theorem begins with the following lemma:

Lemma 3.2.5 (Diamond Lemma, Proof 1)**.** *Let (x,y) be a firing move such that $0 \leq x,y \leq m-1$. Then the following conditions must hold:*

1) $(x,y) \leq (x+1,y)$ and $(x,y) \leq (x,y+1)$, i.e. the firing move (x,y) must take place after the moves $(x+1,y)$ and $(x,y+1)$, if such firing moves exist.

2) When the firing move (x,y) occurs, there are exactly 2 chips present at the site that is firing.

Proof. We first show that move $(0,0)$ (the last move at site 0) must take place after moves $(1,0)$ and $(0,1)$ (the last moves at sites ± 1). By Theorem 3.2.2, site 0 fires $\frac{m(m+1)}{2}$ times in total, while sites 1 and -1 each fire $\frac{(m-1)m}{2}$ times in total. Prior to the last fire at site 0, it has already lost $2 \cdot (\frac{m(m+1)}{2} - 1)$ chips due to firing. In order to fire again, it needs to have 2 chips available, and since it started with $2m$ chips, this means that it must have gained at least $2 + 2 \cdot (\frac{m(m+1)}{2} - 1) - 2m = (m-1)m$ chips from its neighbors. However, this is equal to the total number of firing moves at sites 1 and -1, so all firing moves at those sites must occur prior to move $(0,0)$. In addition, move $(0,0)$ must take place with only 2 chips present.

Assume that the move $(0,x)$ (the last move at position x) must occur before move $(0,x-1)$ (the last move at $x-1$), for some $x > 0$. We want to show that $(0,x)$ takes place

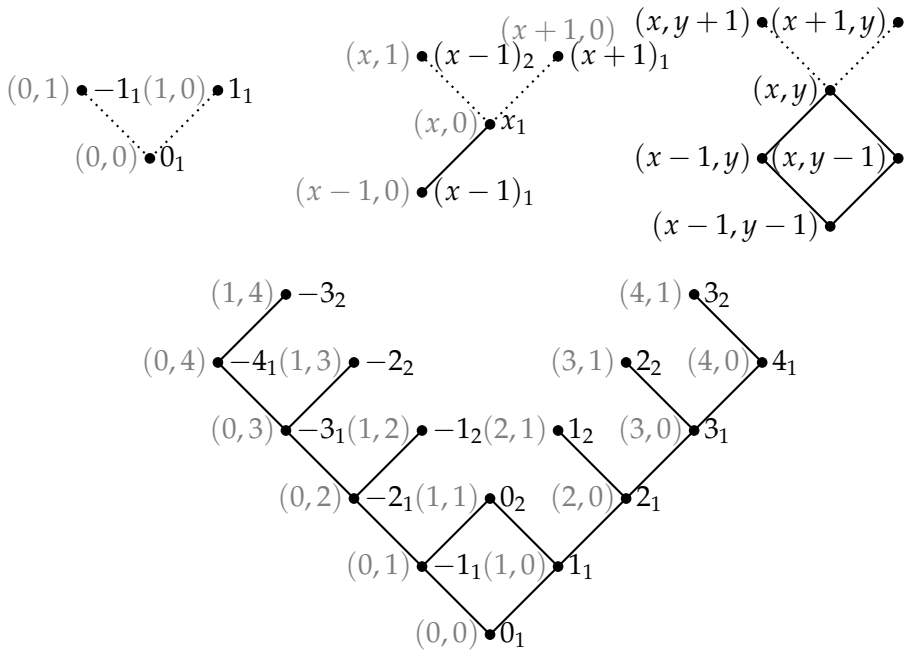


FIGURE 3.4: Top left: The base case. The last move at site 0 must take place after the last moves at sites 1 and -1. Top middle: Inductive step for the last firing moves at each site. Bottom: The edges that we have shown to exist after the completion of the above step for the case $n = 10$. Note the completed 1 by 1 diamond at the bottom, which we use for the main inductive step. Top right: The main inductive step. We assume the presence of the four edges directly below (x, y) in the diagram and prove the existence of the two edges directly above those coordinates.

after $(0, x+1)$ (the last move at $x+1$, if it exists) and $(1, x)$ (the second to last move at $x-1$). Site x fires $\frac{(m-x)(m-x+1)}{2}$ times, so it has lost $(m-x)(m-x+1)-2$ chips due to firing before the last move occurs. Its two neighbors fire a total of $\frac{(m-x+1)(m-x+2)}{2} + \frac{(m-x-1)(m-x)}{2}$ times, of which one is known to happen after the last move at site x. As a result, site x may have received at most $\frac{(m-x+1)(m-x+2)}{2} + \frac{(m-x-1)(m-x)}{2} - 1 = (m-x+1)(m-x)$ chips from its neighbors prior to its last firing move. Thus, all firing moves at its neighbors need to occur before x can fire for the final time with exactly 2 chips present.

We now move onto the main inductive step. Induct on $x+y$, or equivalently, on the rows of Figure 3.3. For fixed k and all $x, y > 0$ with $x+y = k$, assume that move (x, y) must take place before move $(x-1, y)$ and $(x, y-1)$, which both take place before move $(x-1, y-1)$. Also assume that moves $(x-1, y)$, $(x, y-1)$, and $(x-1, y-1)$ all occur with 2 chips present. This is illustrated in Figure 3.4.

Between the time that move (x, y) is about to occur and the time that move $(x-1, y-1)$ is about to occur, site $x-y$ must lose 2 chips due to firing move (x, y) and gain at least 2 chips due to moves $(x-1, y)$ and $(x, y-1)$. Since there are 2 chips present when move $(x-1, y-1)$ occurs, this means that there are at most (and thus exactly) 2 chips present when move (x, y) occurs. Furthermore, no other moves may take place at neighboring sites $x-y-1$ and $x-y+1$ between the moves (x, y) and $(x-1, y-1)$. In particular, the moves $(x+1, y)$ and $(x, y+1)$ must take place before (x, y) in order for this to be possible. □

Remark 3.2.6. It is not immediately clear why this same argument cannot be extended to show that all firing moves in the entire chip firing process must follow this grid structure. The reason is that our inductive step breaks down at the left and right corners of the diamond. There are no firing moves at sites $\pm m$, and there is only one each at $\pm(m-1)$.

We see that because there is one fewer firing move that must occur directly after, say, the move $(3, m)$, this gives more freedom to what can happen at that firing move. It allows the move to take place either with 3 chips present, or before all other firing

moves at neighboring sites have occurred. This effect cascades upward, and ultimately means that all other firing moves in the process (other than the first two at site 0), can have a wider range of numbers of chips present, preventing this structure from applying elsewhere. For the rest of the Hasse diagram for $n = 10$, as well as for the odd $n = 11$, see Figure 3.1.

The Diamond Lemma is crucial to the study of labeled chip-firing and several problems related to it. There are actually 5 versions of this proof throughout this thesis. In Chapter 4, we provide a proof that takes advantage of the fact that the move poset is isomorphic to the poset of join-irreducibles of the poset of unlabeled chip-configurations. We show that the structure holds by showing that corresponding join-irreducibles are reachable from one another.

In Chapter 5, we prove a more general version, in which we show that a similar diamond structure exists for several other initial configurations for labeled chip-firing. In fact, some form of this grid structure exists for very large classes of chip-firing processes on the line, and we show how far this grid structure extends upward in a more general case.

Chapter 8 attempts to generalize the Diamond Lemma as far as it can go. We obtain a version that applies not only to the line, but to a much larger class of undirected bipartite graphs. We see that in some sense, the "key ingredient" necessary for the diamond lemma is that each non-origin vertex has as many edges directed toward the origin as away from it.

We provide the second version of the proof here. Proof 1 is the version that appeared in [31], but it requires us to first calculate the exact number of firing moves occurring at each site in order to prove the ordering on the last moves at each site. An earlier version of the proof prior to publication actually used the number of firing moves at each site in order to prove the ordering for every single move in the diamond.

We provide a proof that does not require the exact numbers of moves at each site. This logic helps in related proofs, where these move counts may be difficult to calculate. In some cases in root system chip-firing, the number of moves at each site is not even an

invariant of the process, so we cannot rely on methods that require exact calculations of move numbers at each site. In Proof 2, we make use of the final unlabeled configuration, without directly using the number of firing moves at each site.

Lemma 3.2.7 (Diamond Lemma, Proof 2)**.** *Let (x, y) be a firing move such that $0 \leq x, y \leq m-1$. Then the following conditions must hold:*

1) $(x, y) \leq (x+1, y)$ and $(x, y) \leq (x, y+1)$, i.e. the firing move (x, y) must take place after the moves $(x+1, y)$ and $(x, y+1)$, if such firing moves exist.

2) When the firing move (x, y) occurs, there are exactly 2 chips present at the site that is firing.

Proof. We first show that move $(0, 0)$ (the last move at site 0) must take place after moves $(1, 0)$ and $(0, 1)$ (the last moves at sites ± 1). As site 0 ends the process with 0 chips, it cannot receive any chips from its neighbors after it fires for the last time. Thus, the last move at site 0 must occur after the last moves at sites ± 1. Furthermore, as site 0 has 0 chips after its last firing move, move $(0, 0)$ must take place with only 2 chips present.

Assume that the move $(0, x)$ (the last move at position x) must occur before move $(0, x-1)$ (the last move at $x-1$), for some $x > 0$. We want to show that $(0, x)$ takes place after $(0, x+1)$ (the last move at $x+1$, if it exists) and $(1, x)$ (the second to last move at $x-1$). Site x receives 1 chip from move $(0, x-1)$ after site x has finished firing. Since site x ends the process with 1 chip, no other moves at neighboring sites may occur after the move $(0, x)$. In particular, $(0, x)$ must occur after $(0, x+1)$ and $(1, x-1)$. Furthermore, site x must have 0 chips after its last firing move, so move $(0, x)$ must occur with exactly 2 chips present.

The rest of the proof proceeds exactly as in Proof 1. □

Next, we consider bounds on the locations of chips with given labels. The following is a weaker bound than the one provided in [27]. Having proved the existence of the grid structure, we do not need as tight bounds as in [27].

Lemma 3.2.8 (Position Bounds)**.** *The position of chip k $(k < 0)$ must never exceed $k + m$ at any point in the chip firing process. Similarly, the position of chip k $(k > 0)$ must never be less than $k - m$ at any point in the chip firing process.*

Proof. We proceed by strong induction on $k < 0$. $k = -m$ is the smallest label that a chip can have. Thus, at no point may the position of this chip increase from its initial position of 0. As a result, the position of $-m$ may never exceed $-m + m = 0$.

Suppose that for all chips with labels less than k ($k < -1$), we have that the positions of these chips may not exceed $k + m$. Consider the chip with label $k + 1$. If this chip ever reaches position $k + 1 + m$, then by our inductive assumption, it must be the smallest chip to reach this position. As a result, it cannot be fired to the right and increase its position from $k + 1 + m$. Thus, the position of chip $k + 1$ can never exceed $k + 1 + m$, and the result for negative k follows by induction. The result for positive k is analogous. □

Visually, the next Lemma proves that for $k < 0$, chip k must always stay at or to the left of the line $y = -k - 1$ in the grid, while for $k > 0$, chip k must stay at or to the right of the line $x = k - 1$, see Figure 3.5.

Lemma 3.2.9. *For each chip k with $-m \leq k < 0$, and each x with $0 \leq x \leq m - 1$, the position of chip k may not exceed $x + k + 1$ immediately preceding firing move $(x, -k - 1)$. Similarly, for each $0 < k \leq m$ and $0 \leq y \leq m - 1$, the position of chip k must be at least $k - 1 - y$ immediately preceding firing move $(k - 1, y)$.*

Proof. We proceed by induction, first on coordinate x, and then on chip k. For the base case, let $k = -m$ and $x = m - 1$; x will decrease to 0, while k increases to -1.

We must first show that chip $-m$ has position at most 0 prior to firing move $(m - 1, m - 1)$. But by Lemma 3.2.8, the position of chip $-m$ may never exceed 0.

Next, fix $k = -m$ and induct on x. Suppose that chip $-m$ has position at most $x - m + 1$ prior to firing move $(x, m - 1)$. This leaves us with two cases: either the position of chip $-m$ is at most $x - m$ prior to move $(x, m - 1)$, in which case its position will not change as a result of move $(x, m - 1)$, or its position is exactly equal to $x - m + 1$

when move $(x, m-1)$ occurs. From Lemma 3.2.5,there are only 2 chips present when move $(x, m-1)$ occurs, so chip $-m$ must be fired by this move. Because $-m$ is the smallest chip, it must be fired to the left, to position $x-m$. Since no moves may occur at site $x-m$ between moves $(x, m-1)$ and $(x-1, m-1)$, chip $-m$ must still be at site $x-m$ prior to firing move $(x-1, m-1)$.

Now, induct on k. Assume that for all $k' \leq k$ and $0 \leq x \leq m-1$, the position of chip k' may not exceed $x+k'+1$ immediately preceding firing move $(x, -k'-1)$. Fixing chip $k+1$ and setting $x = m-1$, we need the position of chip $k+1$ to not exceed $k+1+m$ prior to firing move $(m-1, -k-2)$. Again by Lemma 3.2.8, chip $k+1$ may never exceed that position.

Finally, assume that chip $k+1$ has position at most $x+k+2$ prior to firing move $(x, -k-2)$. There are again two cases: either the position of chip $k+1$ is at most $x+k+1$ prior to move $(x, -k-2)$, in which case its position will not change as a result of move $(x, -k-2)$, or its position is exactly equal to $x+k+2$ when move $(x, -k-2)$ occurs. From Lemma 3.2.5, there are only 2 chips present when move $(x, -k-2)$ occurs, so chip $k+1$ must be fired by this move. Since move (x, j) has already taken place for all $j < k-2$, all chips with values less than or equal to k must have position at most $x+k$. As a result, chip $k+1$ must be the smallest chip at site $x+k+2$ when move $(x, -k-2)$ occurs, so it is fired to the left, to position $x+k+1$. Therefore chip $k+1$ has position at most $x+k+1$ after move $(x, -k-2)$, and since no moves may take place at site $x+k+1$ between moves $(x, -k-2)$ and $(x-1, -k-2)$, chip $k+1$ must still have position at most $x+k+1$ immediately prior to move $(x-1, -k-2)$ as desired. The positive case follows similarly.

□

We are now ready for our main result.

Theorem 3.2.10. *When the labeled chip-firing process terminates, each chip k is at position k.*

Proof. By Lemma 3.2.9, chip k ($k < 0$) must have position less than or equal to $k+1$ immediately prior to firing move $(0, -k-1)$. Using the same argument as in Lemma 3.2.9,

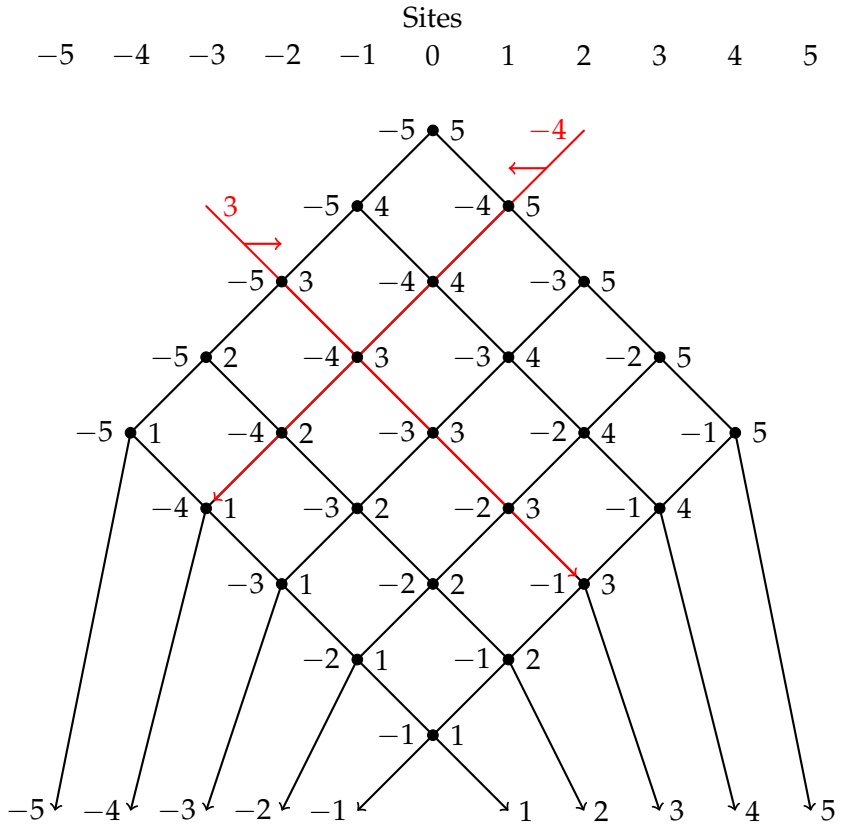


FIGURE 3.5: Diagram for Lemma 3.2.9 and Lemma 3.2.10. For each firing move shown, the chip written to the left of the vertex must be at or to the left of that site when the firing move occurs, while the chip written to the right must be at or to the right of that site. The arrows at the bottom show how the last firing moves at each site send each of the chips to their final, sorted positions. As examples, the red arrows shown are the respective left and right bounds for the positions of chips 3 and -4 when given firing moves occur.

it must have position at most k immediately after that firing move. Since no firing moves may occur at positions less than or equal to k after firing move $(0, -k-1)$, the final position of chip k must be less than or equal to k. The only way to satisfy this condition simultaneously for all $k < 0$ is if each chip k is at position k. The case for $k > 0$ follows similarly.

□

3.3 Related Results

Using the methods above, we are able to prove confluence for a number of similar cases. All of the results essentially come down to the same principles:

(1) There is a diamond of moves at the end of the process that satisfies local confluence.

(2) The initial portion of the chip-firing process permutes the chips by only a "small amount."

(3) The final portion of the chip-firing process can always "fix" the resulting errors.

In general, larger diamonds lead to more straightforward proofs, while smaller diamonds require stricter bounds on chip positions.

3.3.1 Multiple Edges

Consider the 1 dimensional grid in which each edge is replaced with r edges. A firing move sends r chips to the left and r chips to the right. In the labeled setting, each firing move consists of choosing $2r$ chips at the same site and sending the r chips with the smallest labels to the left, and the r chips with the largest labels to the right.

Starting with a multiple of $2r$ chips, the final configuration has r chips at each nonempty site, so the notion of sorting does not directly apply. The next Theorem, resolving Conjecture 25 in [27], shows the strongest possible notion of sorting does hold. We say that a labeled chip configuration is **weakly sorted** if for any two chips a and b with $a < b$, the position of chip a is at or to the left of the position of chip b.

Theorem 3.3.1. *[27, Conjecture 25] Consider the labeled chip-firing process on the line with r copies of each edge and $2rm$ chips initially at the origin. Then the final configuration is weakly sorted.*

Proof. The proof is similar to that of Theorem 3.2.10. Consider an initial unlabeled configuration with $2rm$ chips and then fire to completion. If, at each step, the number of chips at each site is divided by r, then the resulting process begins with $2m$ chips, and each firing move sends one to the left and one to the right. Since this can be reversed by multiplying all chip configurations in the original process by r, there is a bijection between chip configurations in the r edge case and the 1 edge case, with the same available firing moves, so the moves must follow the same partial order P as in the case with single copies of each edge.

Using the same argument as in Lemma 3.2.8, we get that the r smallest chips cannot move past position 0, the next r smallest chips cannot move past position 1, and so on, simply because the r smallest chips at any given site cannot be fired to the right from that site. Then, using the arguments from Lemma 3.2.9 and Theorem 3.2.10, we can show that those groups of r chips satisfy the same position bounds in the diamond as did the original single chips. Thus the r smallest chips end up at position $-m$, the next r smallest chips end at position $-m+1$, and so on, giving the analogous results in the r edge case.

□

3.3.2 Self-Loops at the Origin

Next consider the 1 dimensional grid with self-loops at the origin. Firing moves at all sites but the origin are the same as in the original problem. If there are s self-loops at the origin, then a firing move at the origin consists of choosing $s+2$ chips at the origin and firing the smallest to the left and the largest to the right. Here, we reprove Theorem 21 from [27]:

Theorem 3.3.2. *Consider the labeled chip-firing problem with s self-loops at the origin. For n chips with $n \geq s$ and $n \equiv s \bmod 2$, the final configuration is weakly sorted with s chips at the origin and one chip at every other site from $\frac{s-n}{2}$ to $\frac{n-s}{2}$.*

Proof. As in the multiple edge case, there is a bijection between unlabeled states in this problem and in the original problem. All unlabeled states in this chip-firing process must have at least s chips at the origin. Consider an initial unlabeled configuration with n chips. If, at each step, s is subtracted from the number of the chips at the origin, then the resulting process begins with $n - s$ chips, and each firing move sends one to the left and one to the right. This can be reversed by adding s chips at the origin at each step of the original chip-firing process. We again get the same available firing moves, and since $n - s$ must be even, we get the same partial order P as in the case without self-loops at the origin.

The same bounds on chip positions exist as before, which in turn give the same bounds on chip positions that occur at various firing moves in the diamond. While these bounds are only ever applied to the m smallest chips and the m largest chips, the remaining chips are eventually forced to be at the origin by the fact that specific other chips are required to occupy every other site. This results in each of the m smallest and m largest chips being placed in sorted order at every position but the origin, which results in all of the chips being weakly sorted.

□

3.3.3 Exponentially Many Edges

We now consider a new problem with more structure in the move poset than the original problem. This results in certain parts of the proof becoming simpler, as the chips' movements are more constrained.

We now vary the number of edges between each pair of adjacent nodes. Choose a t, and then for every k from 0 to t, place 2^{t-k} edges between nodes k and $k+1$, as well as between nodes $-k$ and $-k-1$. Past nodes $t+1$ and $-t-1$, all other pairs of adjacent nodes are connected by a single edge. If a site has a leftward edges and b rightward

edges, we can fire at that site by choosing $a + b$ chips, and then sending the smallest a to the left and the largest b to the right. We will show that a process beginning with 2^{t+2} chips must end in a weakly sorted configuration. We first establish a relationship between the numbers of firing moves at each site:

Lemma 3.3.3. *For $0 \leq k \leq t$, site k fires exactly 2 more times than site $k+1$. Similarly, for $-t \leq k \leq 0$, site k fires exactly 2 more times than site $k-1$.*

Proof. We begin with the $k \geq 0$ case. The negative case follows by symmetry.

Proceed by induction. Define $f(k)$ to be the number of firing moves at site k. By counting the number of chips lost or gained due to firing, the number of chips at site 0 at the end of the process is equal to $2^{t+2} + 2^t(f(1) + f(-1) - 2f(0))$. By symmetry, this is equal to $2^t(2f(1) - 2f(0))$ The number of chips cannot be negative, and if there are at least 2^{t+1} chips left at the end, site 0 can fire. As a result, we must have $f(0) = 2 + f(1) = 2 + f(-1)$ in order to have a valid final configuration. This results in a final configuration with 0 chips at the origin.

Now, suppose that $f(k-1) = 2 + f(k)$ for some k from 1 to $t-1$. The final number of chips at k is equal to $2^{t-|k|+1}f(k-1) - (2^{t-|k|+1} + 2^{t-|k|})f(k) + 2^{t-|k|}f(k+1)$. By our inductive assumption, we get that $f(k-1) = f(k) + 2$, so this simplifies to $2^{t-|k|+2} - 2^{t-|k|}f(k) + 2^{t-|k|}f(k+1)$. Again, in order to have a stable final configuration, we need $f(k) = 2 + f(k+1)$, with $2^{t-|k|+1}$ chips at site k in the final configuration. The result follows by induction. □

In addition to obtaining properties about the numbers of firing moves, we also obtain almost the entire final configuration of the process. There must be 1 chip each at sites greater than $t+1$ and less than $-t-1$. Since 1 chip is not enough to fire at any site, this means that these chips must occupy sites $t+2$ and $-t-2$ in the final configuration. Since no firing moves occur at these sites, site $t+1$ must fire exactly once, meaning that for each $-t-1 \leq k \leq t+1$, site k must fire $2(t-|k|)+3$ times.

Next consider the structure of the move poset P when performing chip-firing on this graph. We prove a result analogous to showing the existence of the diamond in Section

2. In this case, the grid structure actually extends to the entire move poset, rather than just a small collection of moves at the end, see Figure 3.8.

Lemma 3.3.4. *For $1 \leq k \leq t+1$ and $1 \leq j \leq 2(t-|k|)+3$, the j^{th} move at site k must occur between moves $j+1$ and $j+2$ at site $k-1$. Similarly, for $-t-1 \leq k \leq -1$ and $1 \leq j \leq 2(t-|k|)+3$, the j^{th} move at site k must occur between moves $j+1$ and $j+2$ at site $k+1$. Furthermore, all firing moves in the entire process, with the exceptions of the first firing move at each site from sites $-t$ to t, leave 0 chips behind at their respective sites after firing.*

The proof is virtually identical to the proof of Lemma 3.2.5, but the different numbers of chips being fired in each direction enable the grid structure to extend throughout the rest of the poset. If the second move at site k is known to occur before the first move at site $k+1$ and the third move at site $k-1$, then it must take place with exactly $3(2^{t-k+1}) - 2^{t-k+1} - 2^{t-k} = 2^{t-k+1} + 2^{t-k}$ chips present. This is exactly the number needed for the site to fire, so it leaves 0 chips at that site after firing, and the move must take place after the second move at site $k-1$. The rest of the proof uses the same induction argument as Lemma 3.2.5.

We then obtain the following result. The smallest and largest chips must repeatedly return to within one site of their desired final position.

Lemma 3.3.5. *For $0 \leq k \leq t$, and $2 \leq j \leq 2(t-k)+3$, the following holds. After the j^{th} move at site k, the largest 2^{t-k} chips are at position greater than k. Similarly, for $-t \leq k \leq 0$ and $2 \leq j \leq 2(t-k)+3$, we have that after the j^{th} move at site k, the smallest 2^{t+k} chips are at position less than k.*

Proof. We will prove the $k \geq 0$ case, and the negative case follows similarly. We induct on k. For $k = 0$, site 0 fires 2^t chips to the right, so the largest 2^t chips can never be to the left of site 0. Since all firing moves starting with the second must fire all of their chips, this means that the largest 2^t chips must either already be at a site $k > 0$ when such a move happens, or they must be fired to the right by any such move.

Now, suppose that this result holds for position $k-1$. By our inductive assumption, the largest 2^{t-k+1} chips must be at position at least k immediately after the j^{th} firing

move at $k-1$, for any $j \geq 2$. Since chips can only go back to site $k-1$ through a firing move at site k, and since only one such move may occur between any two consecutive moves at site $k-1$, this means that whenever a move occurs at site k, the largest 2^{t-k+1} chips must be at position at least k. Since any such move fires 2^{t-k} chips to the right, the largest 2^{t-k} chips must be fired to the right by this move if they are not already at positions greater than k. This completes the induction. □

This gives us enough information to complete our result.

Theorem 3.3.6. *In the exponential edge problem beginning with 2^{t+2} chips at the origin, the final configuration has all chips in weakly sorted order.*

Proof. We prove this by considering the last time that each chip is fired, beginning at position $t+1$ and inducting downward toward 0. We again show the positive case, with the negative case following by symmetry.

By Lemma 3.3.5, the largest chip must be at position $t+1$ after the second firing move at position t. Thus, it must be included in the lone firing move at position $t+1$, and since it is the largest chip, it must be fired to the right. This places the largest chip at a final position of $t+2$.

Now, assume that the largest 2^{t-k} chips reach their correct final positions in order to be weakly sorted. We have that after the second to last move at site $k-1$, the largest 2^{t-k+1} chips must be at position at least k. The largest 2^{t-k} chips are assumed to be in their correct final positions when the last move at site k occurs, which means that the next 2^{t-k} chips must be the largest 2^{t-k} chips at site k. They are thus fired to the right by the firing move at site k, reaching the desired final position at site $k+1$.

□

The larger grid structure helps to simplify the proof from that of Theorem 3.2.10. Instead of tracking chip positions throughout the process and then bounding their positions throughout the diamond, we can show that chips tend to be near where they are supposed to be for almost the entire process, finally slotting into their desired positions at the end. The Hasse diagram for the firing move poset appears in Figure 3.8.

3.3.4 One Self-Loop at Every Site

We now turn to a case that is further from the original problem. Consider the 1 dimensional grid graph with a self-loop at every vertex. Every firing move requires 3 chips in order to fire, with the smallest of the three moving to the left, and the largest of the three moving to the right.

The goal of this section is to provide a proof of a conjecture from [27] (which the present authors originally proved in [31]): for an initial configuration on this graph with $n \equiv 3 \bmod 4$ chips at the origin, the final configuration is weakly sorted. We first prove some analogous results in this setting:

Lemma 3.3.7. *The chip-firing process on the self-loop graph, beginning with $4m-1$ chips at the origin, terminates with 1 chip each at positions $-m$, 0, and m; 2 chips each at positions $-m+1$ through -1 and 1 through $m-1$; and 0 chips elsewhere.*

Proof. We modify the proof of Theorem 2 from [2]. Since there are finitely many chips on an infinite, connected graph, the process must terminate in a unique final configuration. Thus, it suffices to show a single sequence of firing moves that leads to this configuration. Consider a sequence that leads to the following intermediate configurations:

$$
\begin{array}{ccccccccc}
 & & & 4n-1 & & & & \\
 & & 1 & 4n-3 & 1 & & & \\
 & & 2 & 4n-5 & 2 & & & \\
 & 1 & 2 & 4n-7 & 2 & 1 & & \\
 & 2 & 2 & 4n-9 & 2 & 2 & & \\
1 & 2 & 2 & 4n-11 & 2 & 2 & 1 & \\
2 & 2 & 2 & 4n-13 & 2 & 2 & 2 & \\
 & & & \vdots & & & &
\end{array}
$$

In each step, we remove 2 chips from the origin and place 1 chip at each the two closest sites to the origin that do not yet have 2 chips. Each step only requires the use

of 3 chips at the origin, so we treat each case as if only 3 chips are present there initially. In case 1 (2 chips initially at the farthest sites), we make all possible firing moves simultaneously and obtain the following pattern:

$$
\begin{array}{ccccccccccc}
 & & & & & 2 & 2 & 2 & 2 & & \\
 & & & & 2 & 2 & 2 & 3 & 1 & 3 & 2 & 2 & 2 & & \\
 & & & 2 & 2 & 3 & 1 & 3 & 1 & 3 & 2 & 2 & & & \\
 & & 2 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 2 & & & & \\
 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & & & & & \\
1 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 1 & & & & \\
1 & 2 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 2 & 1 & & & & \\
1 & 2 & 2 & 1 & 3 & 1 & 3 & 1 & 2 & 2 & 1 & & & & \\
1 & 2 & 2 & 2 & 1 & 3 & 1 & 2 & 2 & 2 & 1 & & & & \\
1 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 1 & & & &
\end{array}
$$

And in case 2, with 1 chip at the farthest sites initially:

$$
\begin{array}{ccccccccccc}
1 & 2 & 2 & 2 & 2 & 3 & 2 & 2 & 2 & 2 & 1 \\
1 & 2 & 2 & 2 & 3 & 1 & 3 & 2 & 2 & 2 & 1 \\
1 & 2 & 2 & 3 & 1 & 3 & 1 & 3 & 2 & 2 & 1 \\
1 & 2 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 2 & 1 \\
1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 \\
2 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 2 \\
2 & 2 & 1 & 3 & 1 & 3 & 1 & 3 & 1 & 2 & 2 \\
2 & 2 & 2 & 1 & 3 & 1 & 3 & 1 & 2 & 2 & 2 \\
2 & 2 & 2 & 2 & 1 & 3 & 1 & 2 & 2 & 2 & 2 \\
2 & 2 & 2 & 2 & 2 & 1 & 2 & 2 & 2 & 2 & 2
\end{array}
$$

In both cases, there is a line of alternating 3's and 1's that expands until extra chips are deposited at the farthest points, and then contracts again. Starting with $4m - 1$ chips and then alternating between configurations of the two above forms until no more firing moves may be performed yields a configuration of the desired form.

□

Lemma 3.3.8. *In the chip-firing process on the self-loop graph, beginning with* $4m-1$ *chips at the origin, and running to completion, the number of firing moves at site* k *for* $-m \leq k \leq m$ *is equal to* $(m-|k|)^2$.

Proof. We prove the results for nonnegative k, and negative k will follow by symmetry. Since the final configuration has no chips past position m, there must be no firing moves at position m throughout the process. For positions $1 \leq k \leq m-1$, the number of chips sent right by site k must be equal to the number of chips sent to the left by $k+1$, plus the number of chips remaining at sites greater than k. Let $f(k)$ be the number of firing moves at site k. This gives the recurrence

$$f(k) = f(k+1) + 2(m-k) - 1 \text{ for } 1 \leq k \leq m-1$$

A similar argument gives the following recurrence relation for $k = 0$:

$$f(0) = \frac{1}{2}(f(1) + f(-1) + 4m - 2)$$

If we define the first recurrence analogously for negative k, these recurrences have a unique solution with each site k firing $(m-|k|)^2$ times.

□

We again consider the poset P of firing moves in the underlying unlabeled process. These lemmas allow us to prove the existence of an identical diamond to the one from Lemma 3.2.5.

We again assign the coordinates $(0,0)$ to the bottom vertex in the diamond, representing the final move at the origin. A move of one unit up and to the right corresponds to an increase of 1 in the first coordinate, while a move of one unit up and to the left corresponds to an increase of 1 in the second coordinate. An example of a diamond for $n = 19$ is shown in Figure 3.6

Lemma 3.3.9. *Let* (x,y) *be a firing move such that* $0 \leq x,y \leq m-1$. *Then the following conditions must hold:*

1) $(x,y) \leq (x+1,y)$ *and* $(x,y) \leq (x,y+1)$*, i.e. the firing move* (x,y) *must take place after the moves* $(x+1,y)$ *and* $(x,y+1)$*, if such firing moves exist.*

2) When the firing move (x,y) *occurs, there are exactly* 3 *chips present at the site that is firing.*

Proof. The proof is virtually identical to that of Lemma 3.2.5, with the change that firing moves now take place with 3 chips instead of 2. □

Now, Lemma 3.3.7 suggests a new chip-numbering scheme for this problem. Perform this chip-firing process beginning with $4m-1$ chips at the origin, with the chips labeled as follows. Assign the labels $-m$, 0, and m to one chip each, and assign the labels $1-m$ through -1 and 1 through $m-1$ to two chips each. If we perform a firing move involving two chips with the same label, arbitrarily treat one of them as the smaller chip. The weak sorting result is now similar to Theorem 3.2.10; we prove that the final position of each chip is equal to its label.

We then need bounds analogous to Lemma 10 from [27]. Given labeled configuration ℓ, define $u(\ell)$ to be the underlying unlabeled configuration of ℓ. Define $\ell|_{\geq k}$ as the restriction of ℓ to chips with labels greater than or equal to k. Given labeled configurations ℓ_1 and ℓ_2, we say that ℓ_2 is **reachable** from ℓ_1 if it is possible to get from ℓ_1 to ℓ_2 through some sequence of firing moves. Given unlabeled configurations u_1 and u_2, we say that u_2 is **right reachable** from u_1 if it is possible to get from u_1 to u_2 through some sequence of moves of the following two forms:

- Perform a legal chip-firing move.
- Send one chip one unit to the right.

This gives us the following lemma for the self-loop graph, analogous to Proposition 7 from [27].

Lemma 3.3.10. *Given two labeled chip configurations* ℓ_1 *and* ℓ_2 *on the self-loop graph, if* ℓ_2 *is reachable from* ℓ_1*, then* $u(\ell_2|_{\geq k})$ *is right reachable from* $u(\ell_1|_{\geq k})$*.*

Proof. The proof is virtually identical to the proof of Proposition 7 from [27]. Suppose we fire three chips (a), (b), and (c) in ℓ_1. If all 3 chips have values less than k, then the firing move does not affect $\ell_1|_{\geq k}$. If one or two of the chips have values greater than or equal to k, then the firing move sends one chip to the right from $\ell_1|_{\geq k}$. If all three of the chips have values greater than or equal to k, then the firing move is equivalent to a firing move from $\ell_1|_{\geq k}$. □

We define (as in [27]) a partial ordering $\leq_r$ on unlabeled chip configurations, such that $u_1 \leq_r u_2$ if u_2 can be obtained from u_1 by moving zero or move chips to the right. We say $u_1 \lessdot_r u_2$ if u_2 can be obtained from u_1 by moving one chip one unit to the right. This gives us the following lemma, analogous to Lemma 10 from [27].

Lemma 3.3.11. *Throughout the labeled chip firing process, the position of a chip numbered k ($k > 0$) must always be between $\lfloor \frac{k-m}{2} \rfloor$ and $\lfloor \frac{k+m}{2} \rfloor$ inclusive. The position of a chip numbered k ($k < 0$) must always be between $\lceil \frac{k-m}{2} \rceil$ and $\lceil \frac{k+m}{2} \rceil$ inclusive. Chip 0 must remain between position $\lceil \frac{-m}{2} \rceil$ and $\lfloor \frac{m}{2} \rfloor$ inclusive. Furthermore, if $k+m$ is even (for the lower bounds) or $m-k$ is even (for the upper bounds), at most one chip with a given label may satisfy the equality cases of the above bounds at any given time.*

Proof. The proof is very similar to the proof of Lemma 10 from [27]. Let ℓ_0 be the initial labeled configuration, and ℓ_1 to be any other configuration reachable from ℓ_0. Allowing $\ell_{\geq k}$ to stabilize (for $k > 0$) results in one chip at position $\lfloor \frac{k-m}{2} \rfloor$ and all other chips at position greater than $\lfloor \frac{k-m}{2} \rfloor$. By Lemma 3.3.10, $u(\ell_1|_{\geq k})$ is right reachable from $u\big(stab(\ell_0|_{\geq k})\big)$. In particular, the leftmost chip must be at position at least $\lfloor \frac{k-m}{2} \rfloor$ and the second chip from the left must be at position greater than $\lfloor \frac{k-m}{2} \rfloor$. This proves the lower bound on the position of a chip with value greater than or equal to k. The rest of the bounds follow similarly. □

The idea behind Lemma 3.3.11, as well as the corresponding lemma in [27], is that chips with values less than k cannot "help" chip k move any further to the left. Thus, to find the leftmost possible position of chip k, we can first ignore all chips with values less than k, which makes the leftmost possible position much easier to determine.

We then want to show how many chips with labels in certain ranges can appear in certain regions. Such bounds will often result from showing that a violation of these conditions would force certain chips to violate their bounds from Lemma 3.3.11.

We say that a move k_j is a **diamond move** if $j \leq m - |k|$, so that move k_j appears in the diamond at the bottom of the move poset. Given an instance of the labeled chip-firing process s and a chip d, define $f_s(d)$ to be the first of the diamond firing moves of s for which chip d is present. Then define $g_s(d)$ to be the position at which that firing move occurs. Note that g_s maps each chip to a site, and thus defines a chip configuration. We refer to this as the *diamond configuration* of the process s.

We then have the following lemma:

Lemma 3.3.12. *For any k with $-m-1 \leq k \leq 0$, and any h with $0 \leq h \leq k+m-1$, there must be at most $k+m-h-1$ chips d such that the value of chip d is less than k and such that $g_s(d) > h$.*

Proof. We proceed by contradiction. In particular, we will show that if the diamond configuration violates these bounds, then there must be a reachable configuration violating Lemma 3.3.11. In particular, the diamond configuration must be such a configuration.

First, we must show that the diamond configuration is reachable. Given an instance s of the labeled chip-firing process, define instance s' as follows: complete all firing moves in s in the same order, but skip all diamond firing moves. Then perform all diamond firing moves in the same order in which they occurred in s. Since all diamond firing moves must occur after all non-diamond moves at neighboring sites, all of the moves in s' are legal, and performing all of them in this order produces a configuration g' after all of the non-diamond moves have been performed. The number of firing moves that has occurred at any site k is equal to $(m-|k|)^2 - (m-|k|)$, so the resulting configuration g' has 3 chips at the origin and 2 chips each at every other site from $-m+1$ to $m-1$.

We will show that this configuration g' is actually equal to the diamond configuration g_s. Any chip's position in g' must also be the location of the first diamond move that it is involved in in s', since the next firing move involving that chip must be a diamond move, and since all sites with chips must fire at least one more time. Furthermore,

since any diamond move in s' must still occur after the same firing moves at the corresponding site and neighboring sites as in s, all diamond moves in s' must contain the same chips as in s. As a result, we get that $s = g_s = g_{s'}$, and since $g_{s'}$ is a reachable configuration, then g_s must be as well.

Now, we can prove the desired chip bounds by contradiction. Suppose that the number of chips with values less than k at positions greater than h is at least $k + m - h$. Then the diamond configuration must have a chip at position at least $h + \lceil \frac{k+m-h}{2} \rceil = \lceil \frac{k+m+h}{2} \rceil$ with value less than k. If $h > 0$, then a chip with value at most $k-1$ would have to reach position $\lceil \frac{k+m+2}{2} \rceil$, and if $h = 0$, then two chips with values at most $k-1$ would have to reach position $\lceil \frac{k+m+1}{2} \rceil$. Both of these are prohibited by Lemma 3.3.11. Since the diamond configuration is a reachable configuration, it must satisfy Lemma 3.3.11, so this is a contradiction. Thus, the diamond configuration may not have more than $k + m - h - 1$ chips with values less than k at positions greater than h, as desired. □

We can now move on to the main lemma for the self-loop labeled chip firing problem:

Lemma 3.3.13. *Let $-m + 1 \leq k \leq 0$ and $1 \leq j \leq m - |k|$. After the j^{th} diamond firing move at site k, there are at least $j + k + m - 1$ chips with values less than k at positions less than k.*

Proof. We induct first on j, and then on k. For the case $k = -m + 1$, there is only one firing move at that site, and there are no chips at smaller positions prior to that firing move. Since the chip labeled $-m$ can never reach a position greater than 0, it must occupy a non-positive position in the diamond configuration, so it must be present for the first diamond move at some site with position at most 0 (corresponding to moves with y values of m in our labeling from 3.2.9. Since it is the smallest chip, and since all diamond moves take place with exactly 3 chips present, it must then be fired to the left at every site it reaches until it is fired to the left at site $-m + 1$ to its final position of $-m$. Thus, after the 1 firing move at site $-m + 1$, there is 1 chip with value less than $-m + 1$ at site $-m$, satisfying the necessary conditions for $k = 1$.

We then fix $k > -m + 1$ and induct on j. Beginning with the degenerate $j = 0$, we have that before the first diamond move at site k, there are at most $k + m - 1$ chips

with values greater than or equal to k at positions less than k by 3.3.12. Since there are $2(k+m)-2$ total chips at positions less than k prior to that move, at least $k+m-1$ of them must have values less than k.

Now, suppose that our bounds hold for a particular value of j. Consider diamond move $j+1$ at site k. After move j at this site, there are at least $j+k+m-1$ chips with values less than k at positions less than k. It is possible for this number to decrease by 1 after the j^{th} firing move at site $k-1$, but this will only occur if all three of the chips at that site have values less than k. Since there are at least $j+(k-1)+m-1$ chips with values less than $k-1$ at positions less than $k-1$, along with the chip that remains at position k with value less than k after that move occurs, this means that after the j^{th} diamond move at site $k-1$, there will still be at least $j+k+m-1$ chips with values less than k at positions less than k.

We have from 3.3.13 that when the first diamond move at site j occurs, there are at most $k+m-j-1$ chips with values less than k and positions greater than j. Since there are $2(k+m)-1$ total chips with values less than k, there must be at least $k+m+j$ chips with values less than k at positions less than or equal to j. Thus, if there are not already $k+m+j$ chips with such values at positions less than k, then there must be at least one more at position less than or equal to j when its first diamond move occurs. Using the labeling scheme from 3.2.9, this implies that some chip with value less than k will be involved in some firing move with $y=j-1$ and $x-y\geq k$. If we consider the smallest label of all such chips, then that chip will be fired to the left by firing moves with $y=j-1$ until it is fired to the left by move $(k+j-1,j-1)$, which is the j^{th} move at site k. Since this chip has a label less than k, the number of chips with labels less than k at positions less than k will increase by 1 if that number was previously equal to its minimum possible value of $k+m+j-1$. As a result, this number is at least $k+m+j$ after the firing move occurs, and the induction is complete.

□

This brings us to our main result of this section, which proves part of Conjecture 22 from [27]:

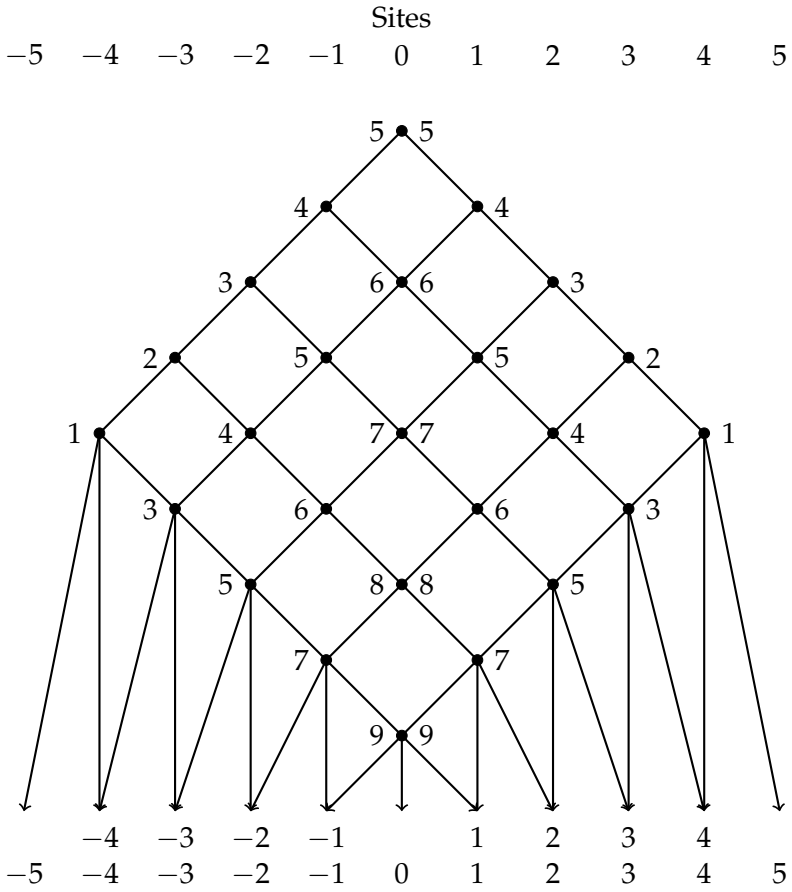


FIGURE 3.6: Diagram for Lemma 3.3.13 and Lemma 3.3.14. A number written to the left of a node is a lower bound on the number of chips with values less than the site number that must be to the left of the corresponding site when the firing move occurs. The initial value at the top of the diamond is equal to $m - |k|$, where k is the site number, and this minimum increases by 1 with each firing move at that site. The arrows at the bottom show how the last firing moves at each site send each of the chips to their final, sorted positions.

Theorem 3.3.14. *[27, Conjecture 22] In labeled chip firing with self-loops, if the initial number of chips at the origin is congruent to 3 mod 4, then the chips end up in weakly sorted order.*

Proof. By Lemma 3.3.13, after the final firing move at position k ($k < 0$), there are $2(k + m) - 1$ chips with values less than k at positions less than k, with an analogous result holding for $k > 0$. Since the only way to satisfy this condition is to have all chips appear at positions equal to their labels, this means that the chips end up in weakly sorted order, as desired.

□

Non-Sorting Example: Self-Loop Graph, $n \equiv 1$ mod 4

In Chapter 6, we study the dynamics of labeled chip-firing on the line for n odd, which fails to sort largely due to a lack of a locally confluent diamond. It turns out that there can be distinctions that are even more subtle that can still prevent sorting from occurring. If we consider the one dimensional grid with a single self-loop at each node, then for similar reasons to the odd case of traditional labeled chip-firing, the chips might not sort when n is even. The last firing move may take place with 4 chips present, so that move may not sort those 4 chips if the wrong 3 are chosen to fire.

In the case where $n \equiv 1$ mod 4, this issue does not arise. In fact, the cases $n = 4m - 1$ and $n = 4m + 1$ both have identical diamonds at the end of their respective firing move posets, although the latter case does have additional moves that take place before then. However, aside from $n = 1$, sorting is never guaranteed when $n \equiv 1$ mod 4. The Hasse diagrams for the full firing-order posets for $n = 15$ through 18 are shown in Figure 3.7.

Because this is so similar to the $n \equiv 3$ mod 4 case, which does sort, this is in some sense a cutoff at which the diamond becomes "too small" to fix all of the out-of-order chips that it would need to fix. There still appear to be relatively few unsorted configurations that can be reached, but such configurations nonetheless always exist for $n > 1$.

To see this, first consider the final unlabeled configuration in the 1 mod 4 case. This will consist of 2 chips each at every site from $-m$ to -1 and 1 to m (where $n = 4m + 1$), and 1 chip at site 0. This differs from the 1 mod 4 case in that the outermost sites now

have 2 chips instead of 1. If 1 mod 4 is the case where 1 extra chip "spills over" into each of two new positions, the 3 mod 4 case is where a second chip spills over to those positions.

Now, to reach an unsorted final configuration, do the following. Letting $n = 4m + 1$, take both chips labeled $-m$ (since there are now 2 such chips instead of 1) and leave them at the origin as long as possible. If we perform all other possible firing moves, then this is equivalent to the labeled chip-firing problem with only $4m - 1$ chips, so all of these will end up in sorted order. In particular, a chip with label $-m - 1$ will end up at position $-m$. We then allow the two chips labeled $-m$ to fire and run the process to completion.

Since site $-m$ never fires, this means that the chip labeled $-m - 1$ will remain at site $-m$ until the end of the process, resulting in an unsorted configuration. This can be done for any m, so it is always possible to avoid sorting if we choose our moves carefully for this case.

The slight differences between the firing move posets for $n \equiv 0, 1, 2, 3 \bmod 4$ are shown in Figure 3.7.

3.3.5 Self-Loops and Multiple Edges

We now combine the cases of self-loops and multiple edges. In this problem, each pair of adjacent nodes is connected by r edges, and each node also has r self-loops. Each firing move now consists of choosing $3r$ chips at a given site, and then sending the r smallest of those chips to the left and the r largest to the right. Conjecture 26 of [27] states that all chips must end in weakly sorted order if the number of chips is congruent to $3r \bmod 4r$.

This is essentially the same as the previous case in which everything has been scaled up by a factor of r. In the same way that the multiple edges case extends the original problem, we can extend the self-loop problem here without too much extra work. In particular, the firing move poset is the same, and the bounds that we were previously able to place on individual chips now apply to groups of r chips. With those changes

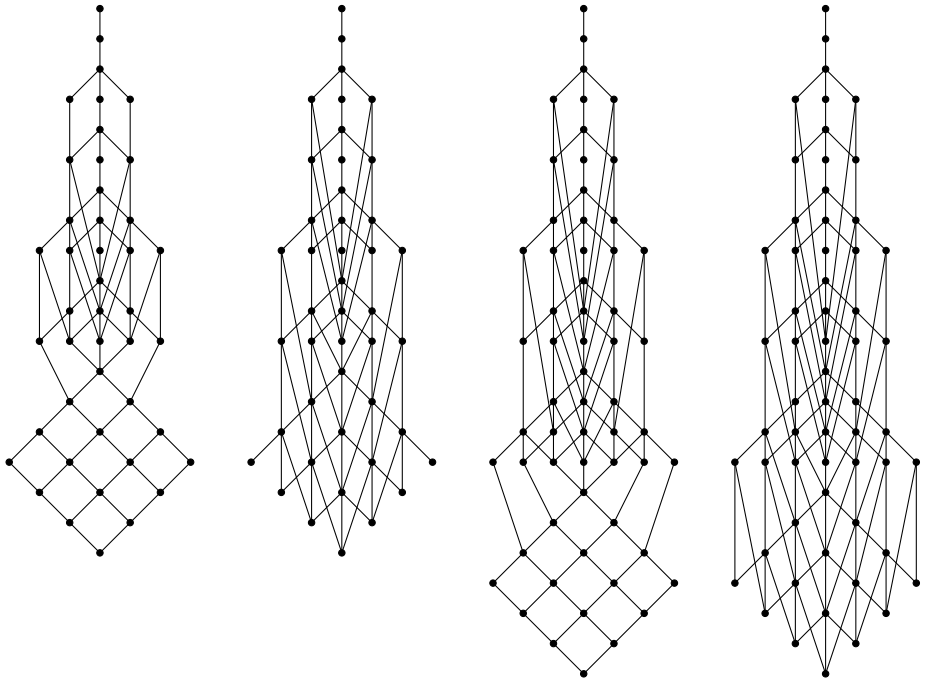


FIGURE 3.7: The full move poset with one self-loop at each node for $n =$ 15, 16, 17, and 18. Both the $n \equiv 1 \bmod 4$ and $n \equiv 3 \bmod 4$ cases have a diamond grid at the bottom, but the $n \equiv 1 \bmod 4$ case has another equally large triangle above the diamond that can send chips past their sorted positions.

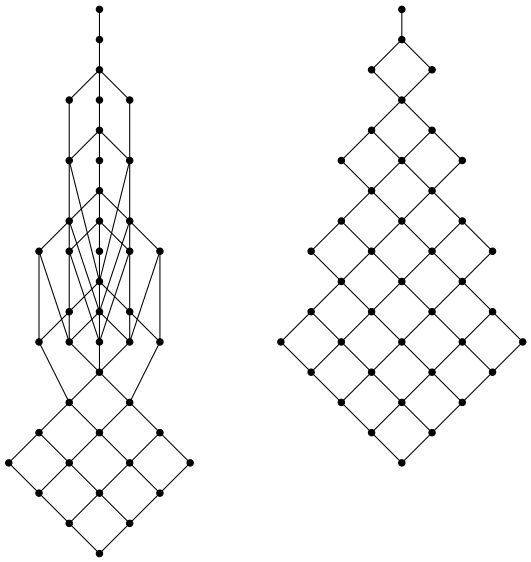


FIGURE 3.8: Left: the full poset P for $n = 15$ in the self-loop problem. Right: the poset P for the exponential edge case, with $t = 3$ and $n = 32$.

in mind, the proofs of all relevant lemmas remain the same, and Conjecture 26 of [27] is established to be true.

3.4 Conclusion

The results in this section form the basis for much of the rest of this thesis. In Chapter 4, we show that the move poset is actually isomorphic to the poset of join-irreducibles of the poset of unlabeled chip configurations. This allows us to approach labeled chip-firing from the direction of lattice theory.

In Chapter 5, we see how we can extend the methods used here to prove other results in labeled chip-firing on a line. These include sorting proofs for certain classes of configurations conjectured to sort in [27] and [21], but we develop a framework for dealing with any initial configuration. In particular, we classify every weakly sorted labeled configuration as either "sorting" or "non-sorting." We do this by studying the final unlabeled chip configurations corresponding to each initial configuration, and proving a more general diamond lemma that can show us how far the grid structure extends for any initial configuration.

Chapter 6 deals with configurations that do not sort, especially labeled chip-firing with n odd. This odd case lacks any of the grid structure that makes the even case work, but our work on move posets and unlabeled configurations still allows us to say a lot about the odd case, including progress on the "$\frac{1}{3}$ conjecture".

In Chapter 7, we then apply our methods to root system chip-firing, in which traditional labeled chip-firing is seen as a Type A special case. Finding move posets and grid structures allows us to resolve some conjectures of [21]. For conjectures that we do not prove, move posets still provide a clear distinction between sorting cases and non-sorting cases.

In Chapter 8, we step away from the line, and try to "mine for diamonds" anywhere we can find them. It turns out that a grid structure exists a much larger class of bipartite graphs, allowing us to impose our grid structure on trees and d-dimensional grids.

CHAPTER 4

Lower Locally Distributive Lattices and Move Posets

4.1 Introduction

This chapter is based heavily on Liscio's "Lattices in Chip-Firing," [35], a version of which will soon be submitted for publication by Klivans and Liscio.

In chapter 3, we developed the notion of "move posets" in chip-firing processes. A terminating chip-firing process has a fixed number of firing moves at each site, so it is well defined to consider whether "the second move at site k_1" occurs before "the fifth move at site k_2" in a particular instance of the process. The move poset captures whether one particular type of move is *guaranteed* to occur before another type in *every* instance of the process.

The move poset is not the only poset that can be defined on a terminating chip-firing process. The chip configurations can also be ordered by reachability [8] [18]. For configurations u_1 and u_2, let $u_1 \geq u_2$ if it is possible to get from u_1 to u_2 through a sequence of legal firing moves. The purpose of this chapter is to study the relationships between configuration posets and move posets on various combinatorial processes.

Define a combinatorial process represented by a poset P as follows. The process consists of **configurations**, represented by elements in P, and **moves**, represented by edges

in the Hasse diagram of P. We usually consider processes with a fixed initial configuration, represented by a maximum element in P. Final configurations correspond to minimal elements in P. A single instance of the process is represented by a downward path through the Hasse diagram of P.

Conversely, any finite poset P with a maximum element has a corresponding process, in which elements in P correspond to configurations, and edges in the Hasse diagram of P correspond to moves. Again, any downward path through the Hasse diagram corresponds to one possible instance of the nondeterministic process.

As one might expect, certain properties of a given poset P translate into properties of the corresponding process. We are particularly concerned with properties related to confluence. If a poset corresponding to a confluent process has a unique maximum element, then it also has a unique minimum element.

Local confluence has the following interpretation in terms of configuration poset P: for any configurations u, v, w such that $u \succcurlyeq v$ and $u \succcurlyeq w$, there exists a configuration z such that $v \succcurlyeq z$ and $w \succcurlyeq z$. Configurations u, v, w, and z form a diamond in the Hasse diagram of P (see Figure 4.1). Recall that Newman's Lemma states that any locally confluent, terminating process with a unique initial configuration must also have a unique final configuration [41] (see Section 2.2.1). Furthermore, all sequences of moves from the initial configuration to the final one must be of the same length.

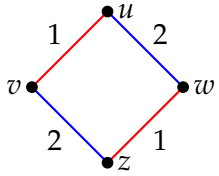


FIGURE 4.1: In the Hasse diagram above, u, v, w, and z represent configurations in a process. From u, it is possible to get to v or w in one move (labeled 1 and 2 respectively). Local confluence, or similarly, the diamond property of LLD lattices, require that there exists a state z reachable from both v and w in one move. The $w - z$ edge and the $u - v$ edge are given the same color, and the $v - z$ edge and the $u - w$ edge are given the same color, to represent that we think of the two paths from u to z as consisting of the same moves in a different order.

In many processes, the two paths through the diamond from u to z come from performing the same two moves in two different orders. We say that a poset generated by a locally confluent process has the **diamond property**.

This chapter deals with a specific class of posets with the diamond property, known as lower locally distributive (LLD) lattices, as well as the behavior of processes corresponding to LLD lattices. Processes exhibiting LLD lattice structure include chip-firing [38] [18], root system central-firing from [21] (shown in [35] and Chapter 7 of this thesis), and the sand pile model from [3] (shown here). All distributive lattices are LLD, so processes involving graph orientations, bipartite matchings, domino tilings, and graph spanning trees [45] also have LLD configuration posets.

We use properties of LLD lattices to prove results on several locally confluent processes. In particular, we present a simpler, third proof of the Diamond Lemma from Chapter 3. We then turn to domino tilings, where we provide a new proof of a result of Saldanha et al. [46] that any "shortest path" between flip-connected tilings must consist of the same flip moves, although possibly in a different order. In Chapter 5, we use the methods developed in this chapter to prove sorting in labeled chip-firing on more general classes of initial configurations. In Chapter 7, we extend these methods to prove conjectures of Galashin et al. [21] on sorting in type A, B, and D root system chip-firing.

We prove results on the above processes by considering the poset of configurations for each process, and then coloring the edges of the resulting Hasse diagrams according to certain rules. These colorings are based on the idea that locally confluent diamonds represent two moves occurring in two different orders. Felsner and Knauer [18] formalize this idea on LLD lattices, by defining a coloring, called an L-coloring, that is compatible with this diamond property.

The first contribution of this chapter is to provide an equivalent characterization of LLD lattices that uses a more refined coloring to more directly address the relationship between LLD lattices and their underlying processes. A saturated L-coloring, or S-coloring, gives two edges in the Hasse diagram the same color if they appear across from each other in some diamond in the Hasse diagram of P. Subject to that constraint,

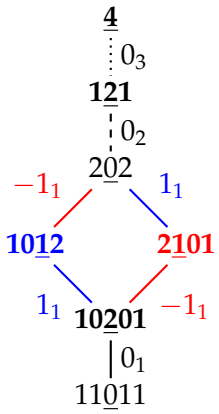


FIGURE 4.2: The poset of configurations for unlabeled chip-firing on the line beginning with $n = 4$ chips at the origin. The number of chips at the origin is underlined, and the join-irreducible configurations are in bold. There is an L-coloring with 3 colors in which each color corresponds to which site is fired. Here, red represents firing moves at -1, black represents firing moves at 0, and blue represents firing moves at 1. The L-coloring can is subdivided into an S-coloring, in which each color corresponds to a specific move k_j. The complete black edge corresponds to move 0_1, the dashed black edge corresponds to move 0_2, and the dotted black edge corresponds to move 0_3.

the S-coloring uses as many colors as possible, ensuring that any instance of the process contains exactly one edge of each color.

Given an LLD lattice P, the unique S-coloring of P is the most refined L-coloring possible, in which each color corresponds to a single join-irreducible of P, while colors of legal L-colorings correspond to chains of join-irreducibles. We show that these chains provide us with useful interpretations of colors in an S-coloring, in terms of the orderings of moves in a process. Each color, and thus each join-irreducible, corresponds to a single element of the move poset of P. See figure 4.2.

S-colorings also allow us to compare LLD lattices to other types of lattices that are also defined by edge labelings, such as supersolvable lattices, with their corresponding S_n EL-labelings.

In section 4.2, we provide preliminaries on LLD lattices and L-colorings, as well as define an S-coloring. In section 4.3, we prove the main results on S-colorings. We show that S-colorings are precisely saturated L-colorings, and that each color corresponds to a single join-irreducible of the configuration poset. We introduce the color poset and

show that it is isomorphic to the poset of join-irreducibles. Finally, we show that all shortest paths between two configurations must consist of edges of the same colors.

In section 4.4, we apply our results on S-colorings and LLD lattices to chip-firing and use them to prove sorting results on labeled chip-firing. In section 4.5, we interpret the S-colorings of distributive lattices generated by graph orientations, bipartite matchings, and spanning trees. The S-coloring of the bipartite matching lattice is used to provide a new proof of a result on shortest paths between flip-connected domino tilings. In section 4.6, we show that the sand pile model of [3] fits into our framework. In section 4.7, we relate LLD lattices and S-colorings to supersolvable lattices and their corresponding S_n EL-labelings.

4.2 Preliminaries

Consider a poset P with Hasse diagram $H = (V, E)$, where V is the set of elements of P, or equivalently, the vertices of the Hasse diagram, and E is the set of edges of the Hasse diagram of P, such that $(u, v) \in E$ if $u \gtrdot v$ in P. Define E' to be the inverse of E, so that $(v, u) \in E'$ if $(u, v) \in E$. If $u \gtrdot v$, then there is a **downward edge** from u to v in E, and there is an **upward edge** from v to u in E'.

Define a **downward path** through the Hasse diagram H as a sequence downward edges $(v_0, v_1), (v_1, v_2), \ldots (v_{k-1}, v_k)$. If P has a maximum element $\mathbf{1}$ and a minimum element $\mathbf{0}$, define a **complete downward path** to be a downward path in which $v_0 = \mathbf{1}$ and $v_k = \mathbf{0}$.

We are concerned primarily with a class of posets known as **lower locally distributive (LLD) lattices**. LLD lattices are characterized by two equivalent definitions.

Definition 4.2.1. A poset P is a **lower locally distributive (LLD) lattice** if either of the following equivalent conditions is met:

(1) P is a lattice, and each element $u \in P$ has a unique minimal representation as a join of join-irreducibles.

(2) For each element $u \in P$, the interval between u and the meet of its lower covers is a boolean lattice.

The inverse of an LLD lattice is called upper locally distributive (ULD). ULD lattices can similarly be defined in terms of meet-irreducibles, or in terms of the join of upper covers of each element. A poset is a distributive lattice if and only if it is LLD and ULD.

More can be said about the representations in definition (1). Define J_u as the set of all join-irreducibles less than or equal to the element u. If $u \gtrdot v$, then $J_v \subseteq J_u$, and $|J_u \backslash J_v| = 1$. As a consequence, every LLD lattice is ranked, with height equal to the number of join-irreducibles. Furthermore, each downward edge (u, v) in E can be associated with a single join-irreducible j such that $\{j\} = J_u \backslash J_v$. Define the **join change** of downward edge (u, v) to be the single join-irreducible in $J_u \backslash J_v$. As **1** is the join of all join-irreducibles, and **0** is an empty join of join-irreducibles, every complete downward path through H must contain one edge with each join change.

Define a **coloring** c of a poset P with Hasse diagram $H = (V, E)$ to be a function $c : E \cup E' \to [n]$ such that $c\big((v, u)\big) = c\big((u, v)\big)$ for all $(u, v) \in E$. c induces a partition of $E \cup E'$ into n subsets of the form $c^i = \{e \in E \cup E' : c(e) = i\}$. Each of these n subsets is a **color** ("c^i is a color from the coloring c"). Thus, coloring c has n colors.

Given two colorings c and c', c' is a **refinement** of c if each color in c' is a subset of a color in c. c' is a **proper refinement** of c if c' is a refinement of c and c' has more nonempty colors than c. This chapter deals with two colorings of LLD lattices.

Definition 4.2.2. [18] Given a poset P, an L**-coloring** of P is a coloring c such that:

(1) For all $u, v, w \in P$ with $(u, v) \in E$, $(u, w) \in E$, it must be the case that $c\big((u, v)\big) \neq c\big((u, w)\big)$.

(2) For all $u, v, w \in P$ with $(u, v) \in E$, $(u, w) \in E$, there exists a $z \in P$ such that $(v, z) \in E$, $(w, z) \in E$, $c\big((u, v)\big) = c\big((w, z)\big)$, and $c\big((u, w)\big) = c\big((v, z)\big)$.

A poset P that admits an L-coloring is known as an L**-poset**. An example of an LLD Lattice with an L-coloring is shown in Figure 4.3.

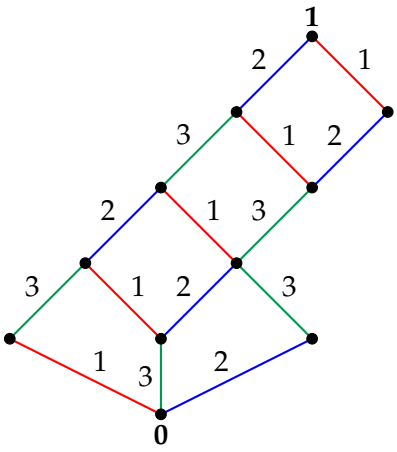


FIGURE 4.3: The Hasse diagram of an LLD Lattice P, along with an L-coloring for that Hasse diagram. The L-coloring consists of 3 colors, labeled 1, 2, and 3 and corresponding to the physical colors red, blue and green. Each downward path from **1** to **0** consists of 1 red edge, 2 blue edges, and 2 green edges.

Theorem 4.2.3 ([18], Theorem 1)**.** *A poset is an LLD lattice if and only if it is an L-poset with a global maximum.*

We introduce a closely related coloring.

Definition 4.2.4. An S**-coloring** of P is a coloring c such that

(1) For any u, v, w, z such that $(u, v), (w, z) \in E$ and $v \geq w$, $c\big((u, v)\big) \neq c\big((w, z)\big)$.

(2) For all $u, v, w \in P$ with $(u, v), (u, w) \in E$, there exists a $z \in P$ such that $(v, z) \in E$, $(w, z) \in E$, $c\big((u, v)\big) = c\big((w, z)\big)$, and $c\big((u, w)\big) = c\big((v, z)\big)$.

A poset is S**-colorable** if it admits an S-coloring. We refer to condition (1) as the "downward path condition," as it states that any downward path through the Hasse diagram cannot contain multiple edges from the same color. Condition (2) of the S-coloring definition is precisely the same as the corresponding L-coloring condition. We will use S-colorings to provide a characterization of LLD lattices, by showing that a poset is an LLD lattice if and only if it admits an S-coloring and has a global maximum (see Lemma 4.3.8). An S-coloring of the lattice from Figure 4.3 is shown in Figure 4.4:

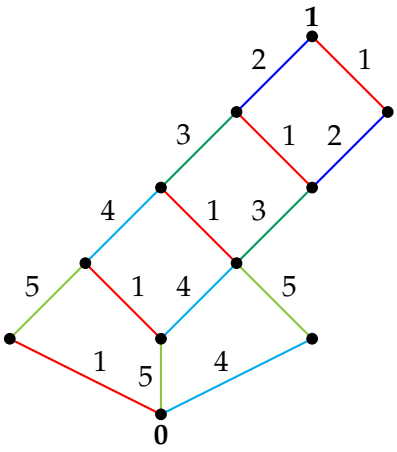


FIGURE 4.4: The S-coloring corresponding to the L-coloring of the LLD Lattice P in Figure 4.3. Color 2 has been subdivided into two colors (2 and 4, represented by light blue), and color 3 has been subdivided into two colors (3 and 5, represented by light green). Each downward path from **1** to **0** contains one edge of each color.

We provide the characterization of LLD lattices by relating S-colorings to a specific type of L-coloring. Define a **saturated L-coloring** as an L-coloring that does not have any proper refinement that is also an L-coloring. We show that every S-coloring is an L-coloring, and every L-coloring has a refinement that is an S-coloring (which is unique up to permutation of the colors). The S in S-coloring stands for "saturated."

An S-coloring is of particular interest because of its relationship with the join-irreducibles of the lattice. In particular, each color in an S-coloring contains a single edge that has a join-irreducible as its upper endpoint. This join-irreducible is equal to the join change of each edge in the coloring. We show that coloring each edge according to the join change of the edge forms the unique S-coloring:

Theorem 4.3.11. *Each LLD lattice P has a unique S-coloring up to isomorphism. In this S-coloring, each edge is colored according to its join change.*

One consequence of Theorem 4.3.11 is that all downward paths between any elements u and v must contain the same colors, and any complete downward path must contain exactly one edge of each color. When viewing a poset as representing a process, we can view the colors of the S-coloring as defining the types of moves that must occur throughout the process.

This allows us to define a partial ordering on the colors in an S-coloring, which we call the **color poset**. Given an S-coloring c with colors c^1 and c^2, we say that $c^1 \geq c^2$ if every complete downward path contains its edge of color c^1 before its edge of color c^2. Our next result relates the color poset to the poset of join-irreducibles:

Theorem 4.3.12. *Consider a finite LLD lattice P, and join-irreducibles u and w of P, which are upper endpoints of the edges (u, v) and (w, z), respectively. Let c be the S-coloring of P. Then $c\big((u, v)\big) \geq c\big((w, z)\big)$ in the color poset if and only if $u \geq w$. Equivalently, the color poset is isomorphic to the poset of join-irreducibles.*

In many processes, the colors of an S-coloring have a more natural interpretation: each color corresponds to "the j^{th} move of a certain type." This is precisely how moves in the move poset were defined in Chapter 3. We provide a general way to prove correspondences between colors and elements of move posets, by providing conditions on when colors form a chain in the color poset.

Theorem 4.3.16. *Consider a finite LLD lattice P with an L-coloring c' and an S-coloring c. Then all of the colors of c corresponding to a single color of c' form a chain in the color poset.*

Since a color in an L-coloring often corresponds to a certain type t of move in a process, the refined colors of the S-coloring allow us to refer to "the n colors of type t." Theorem 4.3.16 extends this idea further, showing that the colors of the S-coloring represent "the $1^{st}, 2^{nd}, \ldots n^{th}$ moves of type t" rather than just "the n moves of type t."

Many of our results deal with downward paths through LLD lattices. However, certain things can be said about any path, and especially about shortest paths. In Section 4.3.2, we prove the following result about shortest paths through LLD lattices and the colors that appear along those paths.

Theorem 4.3.21. *Given a finite LLD lattice P, and elements $u, v \in P$, any shortest path from u to v must contain at most one edge of each color in the S-coloring of P. Furthermore, all shortest paths must include the same set of edge colors, traversed in the same respective directions. There exists a shortest path in which all downward edges precede all upward edges.*

We also prove further results on more general paths in Section 4.3.2. We show that for any two elements u and v in finite LLD lattice P, any path between u and v must contain the same colors as the shortest path, in addition to other colors that must be traversed in both directions. These results on paths in LLD lattices are important in areas such as domino tilings. We prove that "shortest paths" between flip-connected domino tilings, using as few flip moves as possible, must involve the same flip moves in a possibly different order.

4.2.1 Locally Confluent Processes

L- and S-colorings are used in the study of locally confluent processes. Define a **process** $R = (P, E)$ as follows. Let P be a poset and E be the set of downward edges in the Hasse diagram of P. We say that P is the poset **generated** by process R, and R is the **underlying process** of poset P. Each element of P is a **configuration**, and each edge in E is a **move**. If u and v are configurations in P such that $u \gtrdot v$, then we say that "there is a move from u to v" or "it is possible to get from u to v in one move." If P admits an S-coloring then each color in the S-coloring of E defines a "move type."

Note from this construction that if a process R generates a poset P, then there cannot be an instance of the process that reaches the same configuration multiple times. We say that a process is **acyclic** if it satisfies this property.

Recall that a process is locally confluent if for every $u, v, w \in P$ with $(u, v), (u, w) \in E$, there exists a $z \in P$ such that $(v, z), (w, z) \in E$. Local confluence is often studied in part because it implies global confluence: any finite process with a unique initial configuration (corresponding to a maximum element in P), that satisfies local confluence, must have a unique final configuration (minimum element in P). See Section 4.1.

Furthermore, local confluence is important to us because it is closely related to the diamond property of L- and S-colorings. In fact, the diamond property on a Hasse diagram is precisely the same as local confluence on the underlying process, with additional rules about how the edges must be colored when a diamond is formed.

There do exist locally confluent processes that do not generate LLD lattices (see Figure 4.5). However, many of the processes that concern us are locally confluent processes that generate LLD lattices. It is often possible to prove that a locally confluent process generates an LLD lattice by showing that certain types of moves correspond to colors in an L- or S-coloring.

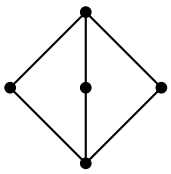


FIGURE 4.5: An example of a poset representing a locally confluent process, but that is not L- or S-colorable. Any coloring that satisfies the diamond property must color every edge the same color (see Figure 4.15). However, any coloring with just one color violates condition (1) of both L- and S-coloring.

4.3 S-colorings

We begin by showing that a finite poset is LLD if and only if it has a maximum element and is S-colorable. We do this by relating S-colorings to L-colorings. By Theorem 4.2.3, a post is an LLD lattice if and only if it is L-colorable and has a global maximum.

Lemma 4.3.1. *Every S-coloring is an L-coloring.*

Proof. Condition (2) is identical for both colorings. Thus, we must show that an S-coloring satisfies condition (1) of L-coloring. Suppose that a poset P has a coloring c that satisfies condition (2) of L-coloring, but not condition (1). Thus, there exist $u, v, w \in P$ and edges $(u,v), (u,w) \in E$ such that $c\big((u,v)\big) = c\big((u,w)\big)$. By condition (2), there exists a $z \in P$ such that $(v,z), (w,z) \in E$, $c\big((v,z)\big) = c\big((u,w)\big)$, and $c\big((w,z)\big) = c\big((u,v)\big)$. This implies that $c\big((u,v)\big) = c\big((u,w)\big) = c\big((v,z)\big)$, so there is a downward path from u to z containing two edges of the same color. As a result, c violates condition (1) of S-coloring. □

Next we show the other direction, that every L-coloring can be refined into an S-coloring. We begin by showing that each L-coloring can be refined into a single saturated L-coloring, which is unique up to permutation of the colors.

Lemma 4.3.2. *Every L-coloring can be refined into a saturated L-coloring. Furthermore, if c_1 and c_2 are saturated L-colorings, then each color of c_1 is equal to a color of c_2.*

Proof. We show this by providing a construction of the unique saturated L-coloring for an arbitrary L-poset P. Given two edges (u, v) and (w, z) of the Hasse diagram of P, we say that $(u, v) \sim (w, z)$ if $(u, w) \in E$ and $(v, z) \in E$, or if $(w, u) \in E$ and $(z, v) \in E$. In other words, two edges are related by $\sim$ if they appear opposite each other in a diamond in the Hasse diagram. Let $\approx$ be the reflexive, transitive closure of $\sim$ (thus making it an equivalence relation), and define a coloring c_1 in which one color is assigned to each equivalence class of $\approx$. By definition, c_1 must meet condition (2) of L-coloring. Furthermore, if $c_1((u, v)) = i$, then under any valid L-coloring, we must have $c_1((u', v')) = i$ for all $(u', v') \in E$ such that $(u', v') \approx (u, v)$.

Thus, for any L-coloring c of P and any color c^i of c, color c^i must be a union of equivalence classes of $\approx$, or equivalently, a union of colors of c_1. If c satisfies condition (1) of L-coloring, then c_1 must also satisfy condition (1) of L-coloring, and is thus an L-coloring. As a result, c_1 is a refinement of every L-coloring, so every L-coloring can be refined into a saturated L-coloring.

Suppose that there exists some other saturated L-coloring $c_2 \neq c_1$. We have shown that c_1 must be a refinement of c_2. However, if any color of c_2 is a union of 2 or more colors of c_1, then c_2 cannot be a saturated L-coloring. Thus, each color of c_2 must also be a color of c_1, so any two saturated L-colorings of P must include the same colors. □

Example 4.3.3. The L-coloring from Figure 4.3 is refined into the saturated L-coloring in Figure 4.4. It can be confirmed that any saturated L-coloring of P must include the same 5 colors. Note that every complete downward path through the Hasse diagram contains exactly 1 edge of each color.

Next, we provide a few lemmas to build up to the proof that any saturated L-coloring is an S-coloring. We show a correspondence between the colors in a saturated L-coloring and the join-irreducibles of the poset P. In particular, each color in a saturated L-coloring corresponds to the set of edges in E with a particular join change.

Lemma 4.3.4. *Let P be a finite LLD lattice with saturated L-coloring c. Then for each color $i \in [n]$, there exists an edge $(u, v) \in E$ such that u is a join-irreducible of P, and $c\big((u, v)\big) = i$.*

Proof. Consider a color c^i of coloring c. Consider the set $U = \{u : (u, v) \in c^i\}$, the set of upper endpoints of edges in c^i. Since P is a finite poset, U, when viewed as a subposet of P, must have some minimal element u, which is the upper endpoint of edge $(u, v) \in c^i$.

Suppose for contradiction that u is not a join-irreducible. Then there must exist some $w \neq v$ such that $(u, w) \in E$. By property (2) of L-coloring, there must exist a z such that $(v, z) \in E$, $(w, z) \in E$, and $c\big((w, z)\big) = c\big((u, v)\big) = i$. Thus, we must have $w \in U$. However, $w < u$, which contradicts that u is a minimal element of U. Thus, u must be a join-irreducible of P. As a join-irreducible u can be found for any color c^i, every color must contain an edge with an upper endpoint that is a join-irreducible, as desired. □

Lemma 4.3.5. *Let P be a finite LLD lattice with saturated L-coloring c. If $c\big((u, v)\big) = c\big((w, z)\big)$ for two edges $(u, v), (w, z) \in E$, then $J_u \backslash J_v = J_w \backslash J_z$.*

Proof. We will show that the result holds if $(u, v) \sim (w, z)$, and the result follows by transitivity. Consider edges (u, v) and (w, z) such that $(u, w), (v, z) \in E$, and $c\big((u, v)\big) = c\big((w, z)\big)$. Suppose that $J_u \backslash J_v = \{j_1\}$ and $J_v \backslash J_z = \{j_2\}$. Thus, $J_u \backslash J_z = \{j_1, j_2\}$. It is not possible to have $J_u \backslash J_w = \{j_1\}$ and $J_w \backslash J_z = \{j_2\}$, since this would imply $J_v = J_w$, which would violate the uniqueness of join-irreducible representations in LLD lattices. Thus, we have $J_u \backslash J_w = \{j_2\}$ and $J_w \backslash J_z = \{j_1\}$, which implies that $J_u \backslash J_v = J_w \backslash J_z$, as desired. Since the result holds when $(u, v) \sim (w, z)$, and since $c\big((u, v)\big)$ is an equivalence class of the transitive closure of $\sim$, the result must hold for any two edges of the same color. □

Theorem 4.3.6. *Consider a finite LLD lattice P with saturated L-coloring c with n colors. Then for all $i \in [n]$, there exists a join-irreducible j of P such that $c^i = \big\{(u, v) : J_u \backslash J_v = \{j\}\big\}$.*

Proof. Given a join-irreducible j, let $E_j = \{(u,v) \in E : J_u \backslash J_v = \{j\}\}$.

Consider a color c^i from saturated L-coloring c. By Lemma 4.3.4, c^i contains an edge (j,k) such that j is a join-irreducible of P. The join change of edge (j,k) is j, so by Lemma 4.3.5, all edges of c^i have join change j, so $c^i \subseteq E_j$.

Similarly, for any color $c^{i'} \neq c^i$, $c^{i'}$ must contain an edge (j',k') such that j' is a join-irreducible, and $j' \neq j$. All edges of $c^{i'}$ must have join change $j' \neq j$, so $E_j \subseteq c^i$.

Thus, for any color c^i from a saturated L-coloring c, we must have $c^i = E_j$ for some join-irreducible j, as desired. □

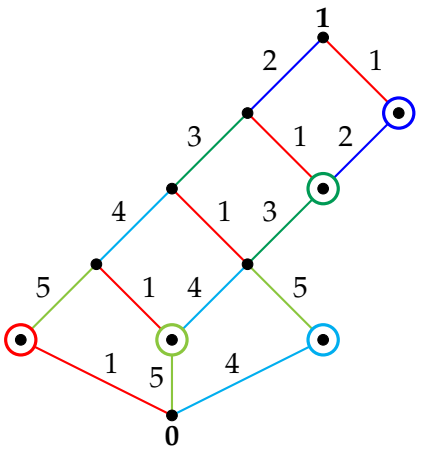


FIGURE 4.6: In the LLD lattice from Figure 4.3, each join-irreducible is circled using the color of its lone downward edge. Note that there is exactly one join-irreducible of each color. For each color, all edges of that color have a join change equal to the corresponding join-irreducible.

In addition to being useful in its own right, Theorem 4.3.6 allows us to complete our correspondence between L-colorings and S-colorings.

Lemma 4.3.7. *Every saturated L-coloring is an S-coloring.*

Proof. As condition (2) of L-colorings and S-colorings are the same, we must prove that every saturated L-coloring satisfies condition (1) of S-coloring.

Consider a finite LLD lattice P with saturated L-coloring c, and consider a complete downward path $S = \big((v_0,v_1),(v_1,v_2),\ldots(v_{n-1},v_n)\big)$, where $v_0 = \mathbf{1}$ and $v_n = \mathbf{0}$. The join changes of the n edges in this path must be distinct, so by Theorem 4.3.6, the n edges

must have n distinct colors under coloring c. This must be true for every complete downward path, so condition (1) of an S-coloring is satisfied, as desired. □

Lemma 4.3.8. *A finite poset is an LLD lattice if and only if it has a maximum element and admits an S-coloring.*

Proof. By lemmas 4.3.1 and 4.3.7, a finite poset with a maximum element is S-colorable if and only if it is L-colorable. Since a finite poset with a maximum element is L-colorable if and only if it is LLD, a finite poset with a maximum element is S-colorable if and only if it is LLD. □

Lemma 4.3.9. *Every S-coloring is a saturated L-coloring.*

Proof. Consider a finite LLD lattice P with S-coloring c and (unique) saturated L-coloring c'. By Lemma 4.3.1, c is an L-coloring. Thus, by Lemma 4.3.2, each color in c is a union of colors in c'.

Suppose for contradiction that c is not a saturated L-coloring. Thus, there must be some color c^i of c that is a union of two or more colors in c'. Suppose that c'^{i_1} and c'^{i_2} are two colors of c' such that $c'^{i_1} \cup c'^{i_2} \subseteq c^i$. By 4.3.6, there exist join-irreducibles j_1 and j_2 such that c'^{i_1} consists of all edges with join change j_1, and c'^{i_2} consists of all edges with join change j_2.

Now, consider a complete downward path $S = \big((v_0, v_1), (v_1, v_2), \ldots (v_{n-1}, v_n)\big)$, where $v_0 = \mathbf{1}$ and $v_n = \mathbf{0}$. As $\mathbf{1}$ is the join of all join-irreducibles of P, and $\mathbf{0}$ is the empty join of join-irreducibles, there must exist some edge $e \in S$ with join change j_1, and some edge in $e' \in S$ with join change j_2. However, e and e' are both of color c^i in the S-coloring c. This violates condition (1) of S-coloring, so every S-coloring is a saturated L-coloring, as desired. □

Lemmas 4.3.7 and 4.3.9 combine to give the following result.

Theorem 4.3.10. *Consider a finite LLD lattice P and a coloring c. Then c is an S-coloring if and only if it is a saturated L-coloring.*

We thus have a correspondence between saturated L-colorings and S-colorings, in which each color corresponds to the join change of all of the edges of that color. This is summarized in the following result.

Theorem 4.3.11. *Each LLD lattice P has a unique S-coloring up to isomorphism. In this S-coloring, each edge is colored according to its join change.*

Proof. By Theorem 4.3.10, a coloring is an S-coloring if and only if it is a saturated L-coloring, and by Theorem 4.3.6, P has a unique saturated L-coloring up to isomorphism. In this coloring, each edge is colored according to its join change. □

4.3.1 The Color Poset

A consequence of Theorem 4.3.6 is that for any S-coloring c and any complete downward path S in a finite LLD lattice, S must contain one edge of each color in c. Furthermore, for any elements $u > v$, any two downward paths from u to v must contain edges of the same colors, in a possibly different order.

Thus, it is well-defined to consider whether one color "occurs before" another color on a path from the top to the bottom. As a result, the color poset is well-defined: given a finite LLD lattice P and S-coloring c with colors c^1 and c^2, we say that $c^1 \geq c^2$ in the color poset if there is no complete downward path in which the edge of color c^2 occurs before the edge of color c^1. Our main result of this subsection is that the colors of the S-coloring of P induce the same partial ordering as join-irreducibles corresponding to each color.

Theorem 4.3.12. *Consider a finite LLD lattice P, with join-irreducibles u and w, which are upper endpoints of the edges (u, v) and (w, z) of the Hasse diagram of P. Then $c\big((u, v)\big) \geq c\big((w, z)\big)$ in the color poset, if and only if $u \geq w$. Equivalently, the color poset is isomorphic to the poset of join-irreducibles.*

Proof. First suppose $c\big((u, v)\big) \geq c\big((w, z)\big)$. No downward path from $\mathbf{1}$ to u may contain an edge of color $c\big((w, z)\big)$. Thus, every downward path from u to $\mathbf{0}$ contains an edge of color $c\big((w, z)\big)$. Thus, starting at u, we can follow downward edges of colors other

than $c\big((w,z)\big)$ until no edges of colors other than $c\big((w,z)\big)$ are available. The resulting element of the poset must be w, the join-irreducible from which the only downward edge is of color $c\big((w,z)\big)$. Since there is a downward path from u to w, we have $u \geq w$ as desired.

Instead, suppose that $c\big((u,v)\big) \not\geq c\big((w,z)\big)$. Consider some complete downward path $s = (e_1, e_2, \ldots e_n)$ in which the edge of color $c\big((w,z)\big)$ occurs before the edge of color $c\big((u,v)\big)$. Thus, $c(e_j) = c\big((w,z)\big)$ and $c(e_k) = c\big((u,v)\big)$ for some $k > j$. Create a new complete downward path s' as follows: Let the first $k-1$ edges in s' be the same as the first $k-1$ edges in s. From there, instead of following edge e_k, follow edges of colors other than $c\big((u,v)\big)$ until no edges of colors other than $c\big((u,v)\big)$ are available. The resulting vertex must be u, from which the only downward edge is (u,v). From there, follow edge (u,v) and then follow any downward path from v to $\mathbf{0}$.

Thus, there is a downward path from $\mathbf{1}$ to u containing an edge of color $c\big((w,z)\big)$. Since an element of the poset uniquely corresponds to the colors of downward edges needed to reach it from $\mathbf{1}$, every downward path from $\mathbf{1}$ to u must contain an edge of color $c\big((w,z)\big)$, and no downward path from u to $\mathbf{0}$ may contain an edge of color $c\big((w,z)\big)$. In particular, no downward path from u to $\mathbf{0}$ may contain edge (w,z). Since (w,z) is the only downward edge from w, there can be no downward path from u to w, so $u \not\geq w$, as desired. □

Example 4.3.13. In the poset P from Figure 4.6, the dark blue join-irreducible is greater than the light green join-irreducible. This is consistent with the fact that every complete downward path must go through a dark blue edge before it goes through a light green edge, which implies that the color dark blue (number 2) is greater than the color light green (number 5) in the color poset.

We will apply Theorem 4.3.12 to a variety of processes and other types of lattices, where the colors represent certain types of moves. The following lemmas help to analyze what the colors represent in these processes. First, we show that, given an upper and lower endpoint in an LLD lattice, it is possible to transform one downward path into another by "switching pairs of consecutive edge colors" one at a time.

Given two downward paths $s = (e^1, e^2, \dots e^n)$ and $s' = (e'^1, e'^2, \dots e'^n)$, we say that s and s' are related by an **edge pair swap** if there exists an index i such that the following holds:

- $e^h = e'^h$ for all $h < i$
- $c(e^i) = c(e'^{i+1})$
- $c(e^{i+1}) = c(e'^i)$
- $e^h = e'^h$ for all $h > i + 1$

Using this notation, we obtain the following result:

Lemma 4.3.14. *Consider a finite LLD lattice P with S-coloring c. Consider vertices $u, v \in P$ where $u \geq v$, and two downward paths $s_0 = (e_0^1, e_0^2, \dots e_0^n)$ and $s_f = (e_f^1, e_f^2, \dots e_f^n)$ from u to v. Then there exists a sequence of downward paths from u to v $(s_0, s_1, s_2, \dots s_k = s_f)$ $\big($where for each j, sequence $s_j = (e_j^1, e_j^2, \dots e_j^n)\big)$, such that for each $1 \leq j \leq k$, s_{j-1} and s_j are related by an edge pair swap.*

Proof. Induct on n. If v can be reached from u in one edge, then there is only one downward path from u to v, so the result holds for $n = 1$.

Now, assume that the result holds for all downward paths of length $n - 1$. Consider edge e_f^1, the first edge of s_f, and consider the edge e_0^k of s_0 such that $c(e_0^k) = c(e_f^1)$. If $k = 1$, then s_0 and s_f both contain a path from the lower endpoint of e_f^1 to v of length $n - 1$. Thus, by our inductive assumption, s_0 can be transformed into s_f through a series of edge pair swaps.

Otherwise, assume that $k \neq 1$. There is a downward edge of color $c(e_f^1)$ starting at u. Thus, by condition 2 of S-coloring, any downward path starting from u that does not contain an edge of color $c(e_f^1)$, must lead to a vertex that is the start point of an edge of color $c(e_f^1)$. In particular, an edge of color $c(e_f^1)$ must be available after following the downward path $(e_0^1, e_0^2, \dots e_0^{k-2})$ from u. Thus, edges of colors $c(e_0^k)$ and $c(e_0^{k-1})$ must both be available from the end of this path, so by the L-coloring property, the element of P reachable by following the edge of color $c(e_0^{k-1})$ and then the edge of color $c(e_0^k)$,

is also reachable by following the edge of color $c(e_0^k)$ and then the edge of color $c(e_0^{k-1})$. By performing an edge color swap that reverses the order of colors $c(e_0^k)$ and $c(e_0^{k-1})$, we create a new downward path s_1 in which the edge of color e_f^1 appears earlier in the sequence than it does in path s_0.

Performing $k-1$ edge pair swaps involving edges of color e_f^1 results in a sequence s_{k-1} such that $c(e_{k-1}^1) = c(e_f^1)$. Again from our inductive assumption, s_{k-1} can be transformed into s_f through a series of edge pair swaps, so s_0 can be transformed into s_f through edge pair swaps. □

Lemma 4.3.15. *Given a finite LLD lattice P with S-coloring c, colors c^1 and c^2 are incomparable in the color poset of c if and only if there exist vertices $u, v, w \in P$ such that:*

- $(u, v), (u, w) \in E$.
- $c\big((u, v)\big) = c^1$
- $c\big((u, w)\big) = c^2$.

Proof. If such vertices u, v, and w exist, then by condition (2) of S-coloring, there exists a vertex z and edges (v, z) and (w, z) such that $c\big((v, z)\big) = c^2$ and $c\big((w, z)\big) = c^1$. As a result, there are downward paths from u to z (and thus from **1** to **0**) where colors c^1 and c^2 occur in either order. Thus, c^1 and c^2 must be unrelated.

Now, suppose that c^1 and c^2 are unrelated, so that there exist two different complete downward paths s_1 and s_2, where s_1 contains its edge of color c^1 before its edge of color c^2, but s_2 contains its edge of color c^2 before its edge of color c^1. By Lemma 4.3.14, it is possible to transform s_1 into s_2 through successive operations of replacing two edges with two edges with the same edge colors in the opposite order. At some point, we must reach intermediate states s_1' and s_2', where the edge of color c^1 appears immediately before the edge of color c^2 in s_1', the edge of color c^2 appears immediately before the edge of color c^1 in s_2', and all other edges are the same in s_1' and s_2'. Thus, there exists some k such that $c((s_1')_k) = 1$, $c((s_1')_{k+1}) = 2$, $c((s_2')_k) = 2$, and $c((s_2')_{k+1}) = 1$. Thus, starting from **1** and following the first $k-1$ edges of s_1' results in a vertex from which downward edges of colors c^1 and c^2 are both present, as desired. □

The following theorem allows us to evaluate individual L-colorings in terms of the corresponding S-coloring.

Theorem 4.3.16. *Consider a finite LLD lattice P with an L-coloring c_1 and S-coloring c. Consider a single color c_1^i of c_1. All of the colors of c that are subsets of c_1^i form a chain in the color poset.*

Proof. Since c_1 is an L-coloring, there are no two edges of color c_1^i with the same top vertex. By Lemma 4.3.15, any two colors of c that are subsets of c_1^i must be comparable in the color poset, so they form a chain. □

Corollary 4.3.17. *Consider a finite LLD lattice P with an L-coloring c_1 and S-coloring c. Consider a single color c^{i_1} of c, which is a subset of color $c_1^{i_2}$ of c_1. Then there exist nonnegative integers j_1 and j_2 such that for every complete downward path through the Hasse diagram of P, the edge of color c^{i_1} occurs after j_1 other edges of color $c_1^{i_2}$, and before j_2 other edges of color $c_1^{i_2}$*

Example 4.3.18. In poset P from Figure 4.4, the dark blue color is greater than the light blue color, and the dark green color is greater than the light green color in the color poset. The two blue colors (and, respectively, the two green colors) are guaranteed to be comparable in the color poset by Theorem 4.3.16, because both blue colors were part of the original, single blue color from Figure 4.3, while both green colors were part of the original green color.

Theorem 4.3.16 and Corollary 4.3.17 are relevant to many applications of LLD lattices, in which LLD lattices are used to represent processes. There is often a natural L-coloring in which each color corresponds to a certain type of move, and Corollary 4.3.17 tells us that each color in the S-coloring corresponds to the "j^{th} move of a given type."

4.3.2 Shortest Paths and S-Colorings

A consequence of Theorem 4.3.12 is that for any $u \geq v$, any sequence of downward edges from u to v must consist of the same collection of distinct colors in a possibly different order. We can now generalize this downward path result to a result on *any*

sequence of (upward or downward) edges between any two elements u and v. This idea has applications to many of the processes we care about, most notably domino tilings and transformations between them.

Consider a finite LLD lattice P with maximum element **1**, minimum element **0**, and S-coloring c with n distinct colors $c^1, \dots c^n$. Given an element u, define the **color vector** of u, $V(y) = \big(V_1(u), V_2(u), \dots V_n(u)\big)$ as follows. For each i from 1 to n, let $V_i(u)$ be 1 if an edge of color c^i is included in any downward path from **1** to u, and 0 otherwise. Note that as a consequence of Theorem 4.3.12, this color vector is uniquely defined for each element of P.

Given two elements u and v and a path $s = (e_1, e_2, \dots e_k)$ from u to v in the Hasse diagram of P, define the **color vector** $V(s) = \big(V_1(s), V_2(s), \dots V_n(s)\big)$ as follows. For each i from 1 to n, $V_i(s)$ is equal to the number of downward edges of color c^i minus the number of upward edges of color c^i in s. We show that this color vector is the same for all paths from u to v.

Lemma 4.3.19. *Consider a finite LLD lattice P with S-coloring C, elements u and v, and a path s from u to v. If $V(s)$ is the color vector of s defined using coloring C, then $V(s) = V(v) - V(u)$.*

Proof. We show that $V(v) = V(u) + V(s)$ by inducting on the length of s. If there are no edges in s, then $v = u$, so $V(v) = V(u)$. Now, suppose that the equality holds for all sequences of length n. Consider a sequence $s = (s_1, s_2, \dots s_{n+1})$ of length $n+1$ from u to v, with subsequence $s' = (s_1, s_2, \dots s_n)$ of length n from u to v'. By our inductive assumption, we have $V(v') = V(u) + V(s')$. Then consider edge s_{n+1}. If s_{n+1} is a downward edge of color c^i, then $V(s) = V(s') + e_i$, where e_i is the unit vector consisting of all 0's, except for a 1 in the i^{th} coordinate. Furthermore, configuration v is reachable from **1** by first reaching v', and then following an additional downward edge of color c^i, so $V(v) = V(v') + e_i$. As a result, we get $V(v) = V(v') + e_i = V(u) + V(s') + e_i = V(u) + V(s)$ as desired.

If s_{n+1} is an upward edge of color c^i, then $V(s) = V(s') - e_i$. Furthermore, v' is reachable from **1** by first reaching v, and then following an additional downward edge

of color c^i, so $V(v') = V(v) + e_i$, or equivalently, $V(v) = V(v') - e_i$. As a result, we get $V(v) = V(v') - e_i = V(u) + V(s') - e_i = V(u) + V(s)$ as desired. □

The fact that the color vector is the same for every sequence s connecting u to v allows us to define the **color vector** of a pair of elements $V(u, v)$ as $V(s)$ for any path s from u to v.

Lemma 4.3.20. *Consider a finite LLD lattice P with S-coloring c, and elements u and v. If $V(u, v)$ is the color vector of u, v defined using coloring c, then all entries of $V(u, v)$ are in $\{-1, 0, 1\}$.*

Proof. By Theorem 4.3.12, any downward path from **1** to any element u contains at most one edge of each color. As a result, all entries of $V(u)$ and $V(v)$ are in $\{0, 1\}$. Subtracting two vectors with entries in $\{0, 1\}$ yields a vector with entries in $\{-1, 0, 1\}$. □

Theorem 4.3.21. *Consider a finite LLD lattice P with S-coloring c, and elements u and v. If $V(u, v)$ is the color vector of u, v defined using coloring c, then the shortest path from u to v consists of $||V(u, v)||_{L^1}$ edges. Furthermore, there exists a path of length $||V(u, v)||_{L^1}$ in which all downward edges precede all upward edges.*

Proof. Consider the vector V_{max}, the pointwise maximum of $V(u)$ and $V(v)$. We will show that there exists an element $w \in P$ such that $u \geq w$, $v \geq w$, and such that $V(w) = V_{max}$. Thus, we can achieve the desired bound by moving from u to w through a sequence of downward edges, and then to v through a sequence of upward edges.

We show that there exists an element w such that $u \geq w$ and $V(w) = V_{max}$. We prove that from u, there exists a downward edge of some color c^i such that $V(v)_i = 1$ and $V(u)_i = 0$. Following this edge, and then repeatedly applying the argument to the resulting element will eventually yield w. Suppose that there exists a path from **1** to u consisting of edges $e_1, e_2, \ldots e_k$ with corresponding colors $c^1, c^2, \ldots c^k$, and there exists a path from **1** to v consisting of edges $e'_1, e'_2, \ldots e'_l$ with corresponding colors $(c^1)', (c^2)', \ldots (c^l)'$. Define s_1 to be the sequence $(c^1, c^2, \ldots c^k)$ and s_2 to be the sequence $\big((c^1)', (c^2)', \ldots (c^l)'\big)$. Let $(c^n)'$ be the first color in s_2 that does not appear in s_1. Thus, $(c^1)', (c^2)', \ldots (c^{n-1})'$ all appear in s_1. By condition (2) of S-coloring, s_1 can be rearranged into a sequence s'_1 that

still starts at **1**, ends at u, and contains the same edge colors as s_1, but which begins with edge colors $(c^1)', (c^2)', \ldots (c^{n-1})'$. A downward path starting at **1** and following edges of colors $(c^1)', (c^2)', \ldots (c^{n-1})'$ in order gives some configuration u_n such that:

- There exists a downward path from u_n to u.
- There is an edge of color $(c^n)'$ with upper endpoint u_n

Since $(c^n)'$ does not appear in s_1, by the L-coloring property, there is also an edge of color $(c^n)'$ with upper endpoint u.

Starting at u and following the downward edge of color $(c^n)'$ reaches a configuration u', reachable from u through one downward edge, such that $V(u')$ is equal to $V(u)$, except for a single 0 replaced with a 1 in the place corresponding to color $(c^n)'$. As $(c^n)'$ is in s_2 but not s_1, the corresponding entry in $V(v)$ (and thus V_{max}), is equal to 1, while the corresponding entry in $V(u)$ is equal to 0. Thus, $||V(u) - V_{max}||_{L^1} - ||V(u') - V_{max}||_{L^1} = 1$. Since the initial distance between $V(u)$ and V_{max} must be finite, we can show by induction that by repeatedly following downward edges in this manner, an element w whose color vector is V_{max} is eventually reached. We can show that the same is possible from v, and since a configuration is uniquely determined by its color vector, this same configuration w must be a lower bound for u and v.

Thus, it is possible to go from u to w through a sequence of downward edges, and then to v through a sequence of upward edges with the total number of moves equal to $||V(u, v)||_{L^1}$.

Since following one edge changes a single entry in the color vector by exactly 1, $||V(u, v)||_{L^1}$ is the smallest number of edges possible to get from u to v. □

Example 4.3.22. In Figure 4.7, vertices u and v are labeled in the LLD poset from Figure 4.4. Every shortest path from u to v contains one red edge, one light blue edge, and one dark green edge.

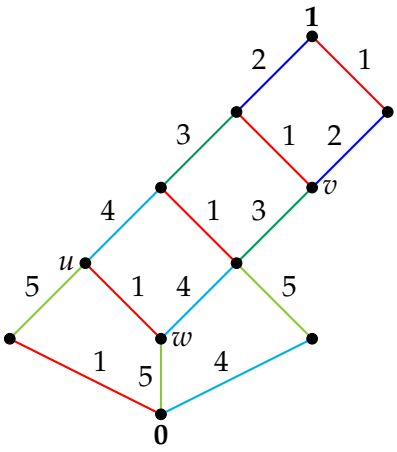


FIGURE 4.7: Every shortest path from u to v contains one red edge, one light blue edge, and one dark green edge. In every shortest path, the red edge is traversed downward, while the light blue and dark green edges are traversed upward. There exists a path through vertex w in which the downward red edge is traversed before the two upward edges.

This framework allows us to deal with many types of processes that generate LLD lattices, in which downward edges represent moves between configurations in a given state space, and colors represent certain classes of moves. We can prove that a process forms an LLD lattice by showing that it meets the L-coloring property, usually by associating each color with a certain class of moves (e.g. a color corresponding to all firing moves at a given site in chip-firing). We can then refine this L-coloring into a saturated L-coloring, or equivalently, an S-coloring. If it is never possible to have two moves of the same class available from the same configuration (e.g. there can be at most one firing move available at any given site from a given configuration in chip-firing), then there is a further interpretation of the colors in the S-coloring. Each corresponds to "the j^{th} move of a given class" (the first firing move at site i, the second firing move at site i, etc.) by Corollary 4.3.17.

In some settings, including domino tilings, there is little distinction between which moves are considered "forward" (corresponding to downward edges in P), and which moves are considered "backward" (corresponding to upward edges). In contexts without that distinction, our results on shortest paths are still useful, as those results do not require us to only consider paths going in a single direction.

4.4 Chip-Firing

Now that we have the framework and main results on using colorings in LLD lattices, we can apply the results of Section 4.3 to processes that generate LLD lattices. Consider an acyclic process R generating a poset P. Each vertex in P corresponds to a configuration of the process, and each edge in the Hasse diagram of P corresponds to a possible move between two configurations in the process. A color in the color poset of P corresponds to some type of move, which often has a natural interpretation in the underlying process.

We first consider chip-firing. Define a **chip configuration** on a graph $G = (V, E)$ to be a $\mathbb{N}$-valued function on V, designating the number of chips at each $v \in V$. On a terminating chip-firing process with initial configuration u_0, the **configuration poset** P is defined so that configuration $u_1 \geq u_2$ in P if it is possible to get from u_1 to u_2 through some sequence of firing moves. An edge in the Hasse diagram of P corresponds to a single firing of exactly $deg(v)$ chips at a single vertex v.

For any terminating chip-firing process, there is a natural coloring for the edges: each color corresponds to all firing moves at a single vertex. Since there can be at most one available move at each site from a given configuration, this coloring must meet condition (1) for L-coloring. As chip-firing is locally confluent (any two available firing moves can be performed in either order), this coloring also meets condition (2). Thus, this coloring is an L-coloring.

Consider a vertex v. By Theorem 4.3.16 and Corollary 4.3.17, the colors in the S-coloring corresponding to firing moves at v form a chain in the color poset. We are thus able to interpret each color in the S-coloring as "the j^{th} move at site k" for some fixed j. Furthermore, given colors c^1 and c^2 in the color poset, c^1 appears before c^2 in every complete downward path if and only if the move corresponding to c^1 must occur before the move corresponding to c^2. Thus, the color poset in chip-firing corresponds precisely to the move poset introduced in Chapter 3.

In particular, we have the following result relating the move poset in chip-firing to the join-irreducibles of the configuration poset:

Theorem 4.4.1. *Consider a terminating chip-firing process on a graph G with initial configuration u. Then the move poset of the resulting process is isomorphic to the poset of join-irreducibles of the poset of configurations reachable from c.*

Proof. By Theorem 4.3.16 and Corollary 4.3.17, each color c^i contains all edges corresponding to a move of the form "the j^{th} move at site k" for some j and k. Thus, each color c^i corresponds precisely to a move in the move poset. Furthermore, color $c^i \geq c^{i'}$ in the color poset if and only if the move corresponding to c^i must occur before the move corresponding to $c^{i'}$. Thus, the color poset is isomorphic to the move poset. By Theorem 4.3.12, the color poset is isomorphic to the poset of join-irreducible configurations in the configuration poset, so the move poset is isomorphic to the poset of join-irreducibles of the configuration poset. □

By viewing the move poset as the poset of join-irreducibles, or equivalently, as the color poset on the S-coloring of the poset of chip configurations, we can provide a simpler proof of the Diamond Lemma in labeled chip-firing. Proving the lemma using join-irreducibles also makes the proof more adaptable to other related problems.

As in the second proof in Chapter 3, we ignore the exact number of firing moves at each site, and instead use only the final configuration:

Theorem 4.4.2 ([2])**.** *The chip-firing process with $2m$ chips at the origin terminates at the final configuration with a single chip at every position from $-m$ to -1, and from 1 to m.*

Now, we present this thesis' third proof the Diamond Lemma from Chapter 3. By using the fact that the move poset is equivalent to the poset of join-irreducibles of the configuration poset, we can remove two layers of induction from the proof. Given a firing move k_j, define $J(k_j)$ to be the join-irreducible corresponding to move k_j. This is the minimum element in the poset of configurations at which a move of type k_j can occur, or equivalently, the unique configuration from which a move of type k_j is the only one available.

Lemma 4.4.3. *For all $0 \leq j \leq m$, move 0_j must take place before moves 1_{j-1} and -1_{j-1}. For $1 \leq k \leq m$ and $0 \leq j \leq m-k$, move k_j must take place before moves $(k+1)_{j-1}$ and $(k-1)_j$. For $-m \leq k \leq -1$ and $0 \leq j \leq m-|k|$, move k_j must take place before moves $(k-1)_{j-1}$ and $(k+1)_j$.*

Proof. By Theorem 4.3.12, a move $(k_1)_{j_1}$ being required to take place before $(k_2)_{j_2}$ is equivalent to $J\big((k_2)_{j_2}\big)$ being reachable from $J\big((k_1)_{j_1}\big)$. As a result, we will show that $J(1_{j-1})$ is reachable from $J(0_j)$, $J(-1_{j-1})$ is reachable from $J(0_j)$, etc.

We will show that for $0 \leq j \leq m-|k|$, $J(k_j)$ consists of 2 chips at site k, 0 chips at site $k+j$ ($k \geq 0$) or $k-j$ ($k \leq 0$), 0 chips at site $-j$ ($k \geq 0$) or j ($k \leq 0$), and 1 chip at all other sites from $-m$ to m.

Assume that $k \geq 0$. The proof for $k \leq 0$ is identical, but with the signs of the sites changed.

A configuration of the specified form is reachable through the following firing moves at each site:

- all moves at sites $k+j$ through m and $-m$ through $-j$
- all but the last j moves at sites 0 through k
- all but the last $k+j-x$ moves at sites $x = k+1$ through $x = k+j-1$
- all but the last $j+x$ moves at sites $x = -j+1$ through $x = -1$

We show that these firing moves produce the desired configuration by comparing the number of firing moves at each site and its neighbors to the total number of firing moves throughout the process.

- At sites $x > k+j$ and $x < -j$, all moves have occurred at site x and its neighbors, so the number of chips at site x in configuration $J(k_j)$ is equal to the number of chips (1 chip) at site x in the final configuration.
- At sites $x = k+j$ and $x = -j$, all moves have occurred at site x and one of its neighbors, but one move has not yet occurred at its other neighbor. As a result,

site x has one fewer chip (0 chips) in configuration $J(k_j)$ than site x has in the final configuration.

- If $k \neq 0$, the number of firing moves yet to occur at the origin exceeds the average number of remaining firing moves at its neighbors by $\frac{1}{2}$, so the origin has one more chip (1 chip) in configuration $J(k_j)$ than the origin has in the final configuration. If $k = 0$, then the number of firing moves yet to occur at the origin exceeds the average number of remaining firing moves at its neighbors by 1, so the origin has 2 more chips (2 chips) in configuration $J(k_j)$ than the origin has in the final configuration.

- if $k \neq 0$, the number of firing moves yet to occur at site k exceeds the average number of remaining firing moves at its neighbors by $\frac{1}{2}$, so site k has one more chip (2) in configuration $J(k_j)$ than site k has in the final configuration.

- At all other sites x, the number of firing moves yet to occur at site x is equal to the average number of remaining firing moves at its neighbors, so site x has as many chips (1) in configuration $J(k_j)$ as site x has in the final configuration.

As a result, each join-irreducible takes the form specified above. Now consider moves k_j and $(k+1)_{j-1}$. $J(k_j)$ consists of 2 chips at site k, 0 chips at site $k+j$, 0 chips at site $-j$, and 1 chip at all other sites from $-m$ to m. $J\big((k+1)_{j-1}\big)$ consists of 2 chips at site $k+1$, 0 chips at site $k+j$, 0 chips at site $-j+1$, and 1 chip at all other sites from $-m$ to m. It is possible to get from $J(k_j)$ to $J\big((k+1)_{j-1}\big)$ by firing successively at every site from k down to $-j+1$, so $J\big((k+1)_{j-1}\big)$ is reachable from $J(k_j)$. Similar arguments apply for showing comparable cases for each join-irreducible and its descendants.

Because the corresponding join-irreducibles satisfy the desired reachability constraints, the moves of the poset satisfy the corresponding order, as desired. □

Corollary 4.4.4. *Every move k_j referenced in Lemma 4.4.3 must take place with exactly 2 chips present at site k.*

Proof. For each k_j referenced in the lemma, $J(k_j)$ has 2 chips at site k. Because all firing moves at neighboring sites that can occur before $J(k_j)$ must do so to reach configuration $J(k_j)$, the number of chips at site k in configuration $J(k_j)$ must be the maximum number of chips that can be available for move k_j. Since k_j cannot take place with fewer than 2 chips at site k, it must take place with exactly 2 chips. □

This provides a few improvements over the first proof of Lemma 3.2.5. First, this shares the property of the second proof in Lemma 3.2.7 that the exact number of firing moves at each site is not needed in order to prove the existence of the diamond. Only the final configuration is needed.

Second, unlike the proof of Lemma 3.2.7, this proof does not need two layers of induction. In order to prove that move $(k_2)_{j_2}$ occurs after move $(k_1)_{j_1}$, we only need to show that $J\big((k_2)_{j_2}\big)$ is reachable from $J\big((k_1)_{j_1}\big)$, which does not depend on any other join-irreducibles. Furthermore, move k_j occurring with at most 2 chips present follows directly from the fact that $J(k_j)$ occurs with 2 chips present.

Third, this proof generalizes well to other problems. Join-irreducibles often take on nice forms in many chip-firing processes. Furthermore, it is often easier to prove that one chip configuration *can* be reached from another than to prove that one move *must* take place after another. Thus, it is often easier to use join-irreducibles to prove the existence of diamonds in move posets of chip-firing processes.

4.5 Distributive Lattices

As every distributive lattice is LLD, the results in Section 4.3 apply to all distributive lattices. Here, we consider some classes of distributive lattices previously studied by Propp [45]. Specifically, we analyze distributive lattices generated by graph orientations, bipartite matchings and tilings, and spanning trees.

4.5.1 Graph Orientations

Given a graph G, an orientation R of the edges of G, and a directed cycle C, define the **circulation** of R around C as the number of edges in R oriented in the same direction as C, minus the number of edges of R oriented in the opposite direction. Define the **circulation** of R to be a function r that maps each cycle C to the circulation $r(C)$ of r around C. We say that R is an **r-orientation**.

Define an **accessibility class** of R as a set of vertices that can be reached from one another through the oriented edges in R. An accessibility class is **maximal** if all directed edges in R between A and A^c point toward A, and it is **minimal** if they all point toward A^c. If A is maximal, then the operation of reversing all edges between A and A^c (so that they now point toward A^c) is called **pushing down** A.

Push-down operations allow us to define a distributive lattice on the set of orientations with a fixed circulation r. Designating a special accessibility class A^*, Propp shows that we can define a partial ordering on the r-orientations of G, in which r-orientation R covers r-orientation S when S can be obtained from R by pushing down an accessibility class other than A^*. Propp proves that the poset generated by this cover relation is a distributive lattice. This poset further defines a process, in which each orientation is a configuration, and each push-down operation is a move.

We show how to interpret the results of Section 4.3 on this lattice of orientations. First, we show that coloring each move based on which accessibility class is pushed down is a valid L-coloring.

Lemma 4.5.1. *Given a graph G and a circulation r, then color the poset of r-orientations such that each color corresponds to an accessibility class A being pushed down. Then this coloring is a valid L-coloring.*

Proof. First, given an orientation R and an accessibility class A, there is at most one way to push down A from R. Thus, this coloring satisfies condition (1) of L-coloring.

Now, consider an orientation R and two accessibility classes A_1 and A_2 that can be pushed down from R. If both can be pushed down, then all directed edges between

A_1 and A_1^c are directed toward A_1, and the same is true for A_2. Thus, there can be no edges between A_1 and A_2, so pushing one down will not affect the other. As a result, the push-down operations can be performed in either order to reach the same resulting configuration, so the coloring satisfies condition (2), and is thus an L-coloring. □

Theorem 4.5.2. *Given a graph G and circulation r, consider the distributive lattice of r-orientations on G. Then each color in the S-coloring of this lattice consists of all edges corresponding to moves of the form "the j^{th} push-down operation at class A" for a fixed accessibility class A and positive integer j.*

Proof. By Lemma 4.5.1, there exists a valid L-coloring in which each color corresponds to a single accessibility class being pushed down. By Lemma 4.3.15, each color in this L-coloring is a union of colors of the S-coloring which form a chain in the color poset. As a result, given an accessibility class A, all colors of the S-coloring corresponding to push-down operations at A must occur in the same order in any complete downward path in the lattice of r-orientations. Thus, each color in the S-coloring corresponding to class A must consist of all moves of the form "the j^{th} move at A" for some fixed j. □

4.5.2 Tilings

Propp shows that bipartite matchings on planar graphs also form distributive lattices by relating them to processes involving r-orientations [45].

Given a planar bipartite graph $G = (V, E)$ on the sphere, and a function $d : V \to \mathbb{N}$, define a **d-factor** of G as a subgraph of G (viewed as a set of edges) in which each vertex v has degree $d(v)$. The example we are most concerned with is a 1-factor, in which $d(v) = 1$ for all $v \in G$, corresponding to a perfect matching of G.

Color the vertices of one part of the bipartite graph white, and the vertices of the other part black. Define an **elementary cycle** as a cycle encircling a single face of the planar graph. Given a d-factor M, an **alternating cycle** in G relative to M is an elementary cycle in G in which the edges alternatively belong to M and M^c. Call the cycle **positive** if the edges in M, oriented clockwise around the cycle, go from black vertices to white vertices, and negative if they go from white vertices to black vertices. Given a

d-factor M, we can perform a **twisting down** operation by taking a positive alternating cycle, and replacing the edges of M in that cycle with the edges of M^c in that cycle, thus turning the cycle from a positive to a negative alternating cycle. Define **twisting up** to be the inverse of twisting down.

Designate a special face f^* of G. Since G is on a sphere, we usually let f^* be the "outside" face when G is viewed on a plane. Propp shows that the set of d-factors can be partially ordered such that a d-factor M_1 covers a d-factor M_2 if it is possible to get from M_1 to M_2 by twisting down a face other than f^*. He further shows that this partial ordering forms a distributive lattice. We can show that coloring the edges of this lattice according to which face is twisted down, forms an L-coloring.

Lemma 4.5.3. *Given a bipartite planar graph G and an $\mathbb{N}$-valued function d on the vertices, along with special face f^*, consider the lattice of d-factors of G with special face f^*. Then the edge coloring of the lattice such that each color corresponds to the face being twisted down, is a valid L-coloring.*

Proof. First, given a d-factor M and a face f, there is at most one way to twist down f from M. Thus, this coloring satisfies condition (1) of L-coloring.

Now, consider a d-factor M and two faces f_1 and f_2 that can be twisted down from M. If both can be twisted down down, then all black-white edges in f_1 and f_2 must be oriented clockwise. Since an edge is oriented clockwise in f_1 if and only if it is oriented counterclockwise in the face that borders f_1 along that edge, f_1 and f_2 must not be adjacent. Thus, twisting down f_1 does not affect whether f_2 can be twisted down and vice versa, so the operations can be performed in either order. As a result, this coloring satisfies condition (2), and thus is an L-coloring. □

We now proof that the moves of the S-coloring of a d-factor lattice correspond to move types of the form "j^{th} twisting down operation at face f." This proof is nearly identical to the proof of Theorem 4.5.2.

Theorem 4.5.4. *Given a bipartite planar graph G and and an $\mathbb{N}$-valued function d on the vertices, along with special face f^*, consider the distributive lattice of d-factors on G with respect*

to f^*. Then each color in the S-coloring of the lattice consists of all edges corresponding to moves of the form "the j^{th} twist-down operation at face f" for a fixed face f and positive integer j.

Proof. By Lemma 4.5.3, there exists a valid L-coloring in which each color corresponds to a single face being twisted down. By Lemma 4.3.15, each color in this L-coloring is a union of colors of the S-coloring which form a chain in the color poset. As a result, given a face f, all colors of the S-coloring corresponding to twist-down operations at f must occur in the same order in any complete downward path in the lattice of d-factors. Thus, each color in the S-coloring corresponding to class face f must consist of all moves of the form "the j^{th} move at f" for some fixed j. □

As a result of Theorem 4.5.4, the distributive lattices generated by bipartite matchings (and as a consequence, domino and lozenge tilings of simply connected regions in 2 dimensions) fall under our framework. The framework provides us with some nice results on paths between tilings. In particular, Theorem 4.3.19 and Lemma 4.3.21 provide a method for finding shortest paths between tilings using flip moves, as well as showing which flip moves must be performed on a shortest path between any two flip-connected tilings. Previous approaches to this result considered the graph of flip-connected tilings as a CW-complex [46], and then showed that the complex was contractible. Here, the distributive lattice structure allows us to show these results on shortest paths using colorings.

Given a bipartite planar graph G with function d on the vertices and special face f, consider the distributive lattice generated by the twist down relation, and its maximum element $\mathbf{1}$. Suppose that the S-coloring of the lattice has colors $c^1, c^2, \ldots c^n$. Define the color vector of d-factor u_1 as $V(u_1) = (v_1(u_1), v_2(u_1), \ldots v_n(u_1))$ such that for each i from 1 to n, v_i is 1 if any shortest path from $\mathbf{1}$ to u_1 contains color c^i, and 0 otherwise. Similarly define a color vector of a path s as $V(s) = (v_1(s), v_2(s), \ldots v_n(s))$ such that for each i from 1 to n, $v_i(s)$ is the number of downward edges of color c^i in s, minus the number of upward edges of color c^i in s. The following results follow directly from their counterparts in Section 4.3:

Theorem 4.5.5. *Consider a path s from tiling u_1 to tiling u_2 on a simply connected region G. Then $V(s) = V(u_2) - V(u_1)$. Thus, all paths from u_1 to u_2 contain the same flip moves, up to moves being done and then undone.*

Theorem 4.5.5 allows us to uniquely define a color vector $V(u_1, u_2)$, equal to the color vector of any path s from u_1 to u_2.

Corollary 4.5.6. *For any two tilings u_1 and u_2 on simply connected region G, all entries of $V(u_1, u_2)$ are in $\{-1, 0, 1\}$.*

Lemma 4.5.7. *Given tilings u_1 and u_2 on simply connected region G, there exists a path from u_1 to u_2 with length $|V(u_1, u_2)|_{L^1}$. This is the shortest possible length of any path from u_1 to u_2. Furthermore, there exists a path of length $|V(u_1, u_2)|_{L^1}$ from u_1 to u_2 in which all downward edges precede all upward edges, and there also exists a path of length $|V(u_1, u_2)|_{L^1}$ from u_1 to u_2 in which all upward edges precede all downward edges.*

Proof. This result follows from Lemma 4.3.21. The fact that a shortest path exists with upward moves followed by downward moves is due to the fact that the dual of a distributive lattice is still distributive, and hence LLD. □

We conclude this section by providing a lattice-based proof of the following result of Saldanha et al. (equivalent to Theorem 3.4 of [46]).

Theorem 4.5.8. *Consider tilings u_1 and u_2 on simply connected region G, and two shortest paths s_0 and s_f from u_1 to u_2. Then it is possible to get from s_0 to s_f through a sequence of edge pair swaps.*

Proof. Suppose that the tilings of G have configuration poset P with S-coloring c. We first show that it is possible to get from s_0 to some path s_0' in which all downward edges precede all upward edges. Let $s_0 = (e^1, e^2, \ldots e^n)$. Suppose that there exists some index i such that e_i is a downward edge, but e_{i+1} is an upward edge. By property (2) of L coloring, there must be a downward edge of color $c(e_{i+1})$ followed by an upward edge of color $c(e_i)$ from the start point of edge e_i. Thus, we can perform an edge pair swap to

reverse the order of colors $c(e_i)$ and $c(e_{i+1})$, resulting in a downward edge moving before an upward edge in the sequence, without changing the numbers of downward and upward edges. This operation reduces the sum of the indices of the downward edges by 1. Since that sum can never be lower than $\binom{\text{number of downward edges}+1}{2}$, this process must terminate with a path s'_0 in which all downward edges preced all upward edges. Similarly, it is possible to get from s_f to a path s'_f in which all downward edges precede all upward edges.

By Lemma 4.3.19, the downward edges (and the upward edges respectively) of s'_0 and s'_f must be the same. By Theorem 3.2, color vectors are unique, so s'_0 and s'_f must follow i downward edges from u to a common configuration w, and then follow $n-i$ upward edges to v. By Lemma 4.3.14, it is possible to transform the first i edges in s'_0 to the first i edges in s'_f via edge pair swaps, and it is possible to transform the last $n-i$ edges in s'_0 to the last $n-i$ edges in s'_f via edge pair swaps.

Thus, it is possible to transform from s_0 to s'_0 to s'_f to s_f via edge pair swaps, as desired. □

A domino tiling example on a 2x6 grid is shown in Figure 4.8.

Tilings as Chip-Firing

One interesting consequence of Propp's correspondence between graph orientations and bipartite matchings is that it also provides us with a correspondence between chip-firing and bipartite matchings.

Propp [45] provides a method for converting any connected bipartite planar graph G into an orientation of its dual graph $G^\perp$. To do this conversion, color all of the vertices in one part of the bipartite graph black, and color all of the vertices in the other part white. Define the standard orientation on $G^\perp$ as the orientation in which each edge e' in $G^\perp$ corresponding to edge e in G is oriented so that the white endpoint of e is to its left and the black endpoint of e is to its right. Given a d-factor u on G, create a corresponding orientation on $G^\perp$ such that all edges in the standard orientation corresponding to edges in u are reversed, and all edges in the standard orientation not corresponding to edges

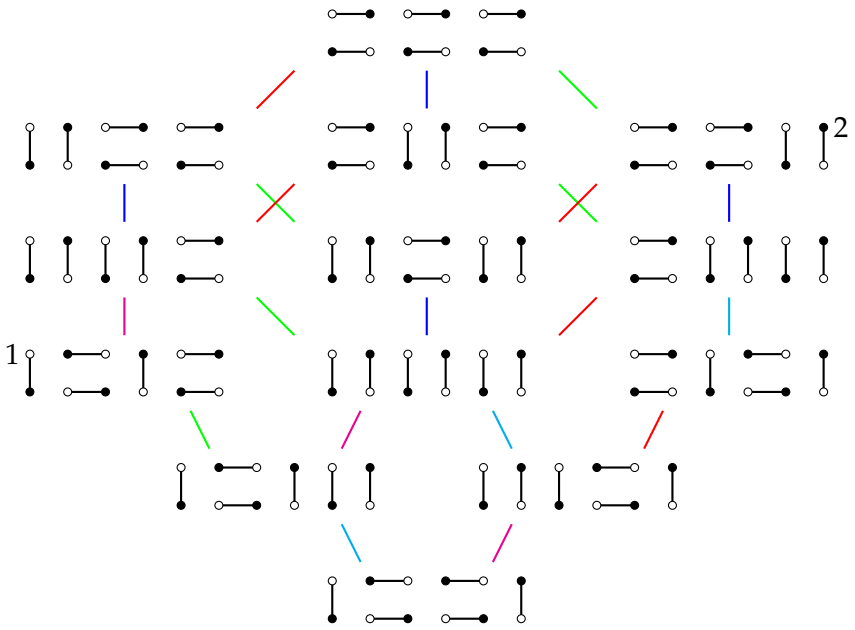


FIGURE 4.8: The partial ordering of bipartite matchings (or equivalently, domino tilings) on a 2x6 grid. As each face is only twisted down once to get from the top to the bottom, each edge color corresponds to twisting down a different face. The distance between the configurations labeled 1 and 2 in this poset is 4, and each shortest path from 1 to 2 must contain upward edges colored red, blue, and magenta, and a downward edge colored green.

in u have their directions maintained. Propp shows that a pushing down operation in the dual orientation corresponds to a twisting down operation in G, and that the pushing down operations generate the same distributive lattices as the twisting down operations. The set of d factors corresponding to a particular function d corresponds to the set of orientations with a particular circulation on $G^{\perp}$.

Now, any graph orientation R can be converted to a chip-configuration by assigning to each vertex a number of chips equal to its number of incoming edges in R. As noted in Theorem 3.3 of [8], a vertex can fire if and only if all of its incident edges in R are directed inward, in which case firing corresponds to reversing the directions of all incident edges in R, or, in the language of Propp, pushing down that vertex.

We further note that pushing down any accessibility class A corresponds to cluster-firing all of the vertices in that class in the same manner. A cluster fire of cluster A can occur if and only if each vertex v in A has at least as many chips as it has edges to A^c, in which case a cluster fire reduces the number of chips at v by the number of edges between v and A^c, sending one chip along each edge to A^c. Cluster firing A corresponds to reversing the direction of every edge between A and A^c in the corresponding orientation.

Thus, we can create a correspondence between d-factors (or, more specifically, bipartite matchings or domino tilings) and chip-firing processes. Take a connected bipartite planar graph G with a distinguished region f^*, which corresponds to a sink vertex in the corresponding chip-firing process. Then use Propp's methods to convert this d-factor to an orientation of the graph $G^{\perp}$, and then convert from that graph orientation to a chip configuration.

Each push-down operation at a face f in G corresponds to a firing move at the corresponding vertex in $G^{\perp}$. In particular, flip moves in a domino tiling correspond to firing moves (or un-firing moves) in the corresponding chip-firing process. This implies that the distributive lattice generated by push-down operations is the same as the lattice generated by a chip-firing process with a particular initial configuration on the dual graph. The graph orientations and chip configurations generated by the domino tiling system

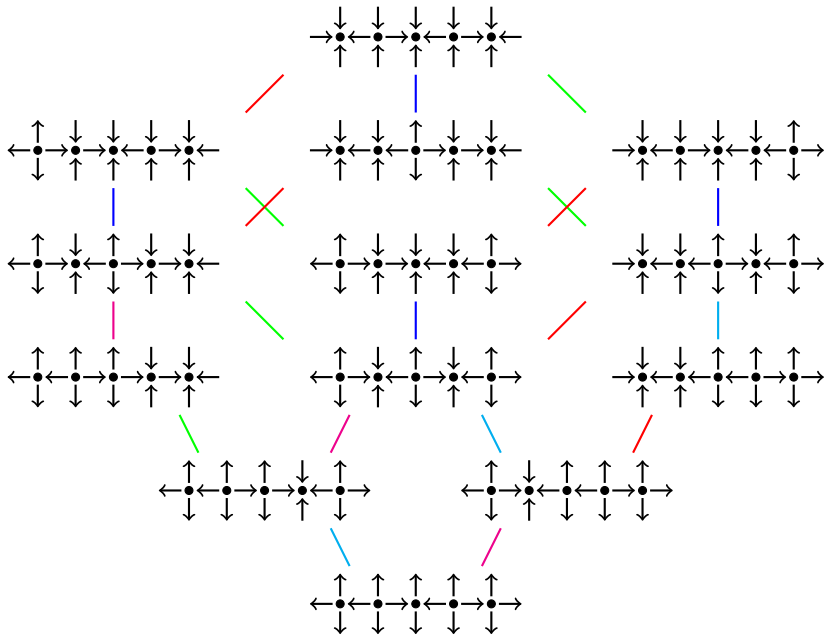


FIGURE 4.9: The graph orientations corresponding to each bipartite matching in Figure 4.8. Edges without a second endpoint represent edges in which one endpoint is the "outside" of $G^{\perp}$

in Figure 4.8 are shown in Figure 4.9 and Figure 4.10.

4.5.3 Spanning Trees

Propp also defines a distributive lattice on the set of spanning trees of a planar graph. Trees are related by a successor relation involving a "swinging down" move that transforms one spanning tree to another through rotating one edge clockwise about a vertex, while meeting certain other conditions, as described below.

Given a planar graph G, designate a special face f^*, with a special vertex v^* adjacent to f^*. Then consider a vertex v and an edge e containing v. Define the **clockwise successor** e' of e with respect to v to be the next edge coming out of v if we move clockwise from e through face f. Given v, e, f, and clockwise successor e', and spanning tree T, angle eve' is **positively pivotal** if all of the following are met:

(1) $e \in T$ and $e' \notin T$.

(2) $T' = T \backslash \{e\} \cup \{e'\}$ is a spanning tree of G.

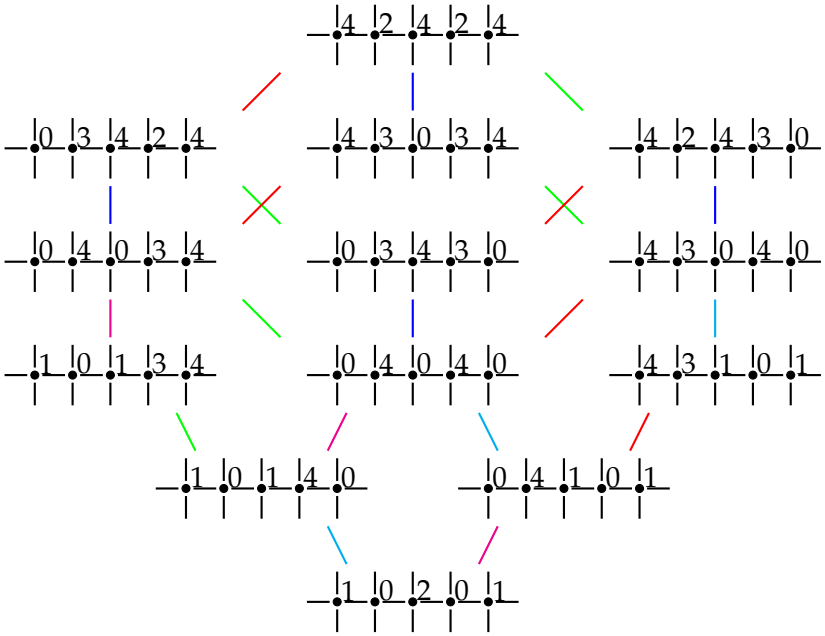


FIGURE 4.10: The chip configurations corresponding to the graph orientations in Figure 4.8. Edges without a second endpoint correspond to edges to the sink.

(3) The simple path from v to v^* in T contains e.

(4) The simple path from v to v^* in T' contains e'.

(5) The (unique) simple cycle in $T \cup T'$ separates f from f^*.

If eve' is positively pivotal, then the operation transforming T to T' is called **swinging down** at v through angle eve'. Transforming T' to T is called swinging up. Propp shows that every choice of f^* and v^* generates a partial ordering of spanning trees on G with a cover relation $T \gtrdot T'$ if it is possible to get from T to T' through one swing-down operation. He further shows that the poset generated by this cover relation is a distributive lattice. The proof that this poset is a distributive lattice involves showing that a swinging down operation through a specific angle eve' corresponds to a twisting down operation along a specific face of a graph $H(G)$. See [45] for more details.

We can show that coloring the edges of this lattice of spanning trees according to the angle of the corresponding swinging down operation, is an L-coloring.

Lemma 4.5.9. *Given a planar graph G with special face f^* and special vertex v^* on f^*, consider the distributive lattice P of spanning trees generated by swinging down operations. Let c be an edge coloring of P such that each color of c includes all edges that correspond to swinging down through a particular angle $A = eve'$. Then c is a valid L-coloring.*

Proof. Let $H(G)$ be the Hasse diagram of the face-poset of G, viewed as a graph. $H(G)$ is a graph with a vertex $\bar{v}$ corresponding to each vertex v of G, a vertex $\bar{e}$ corresponding to each edge e of G, and a vertex $\bar{f}$ corresponding to each face f of G. For each edge e of G, and each endpoint v of e, there is an edge in $H(G)$ between $\bar{e}$ and $\bar{v}$. For each face f of G and each edge e adjacent to f, there is an edge in $H(G)$ between $\bar{f}$ and $\bar{e}$. This idea was introduced by Temperly [49], discussed by Lovasz [37], and generalized by Burton and Pemantle [12] to infinite planar graphs. Propp shows that the lattice of spanning trees of G is isomorphic to the lattice of matchings of $H(G)$, such that swinging down through an angle eve' in G corresponds to twisting down a unique face of $H(G)$ [45].

As swinging down through eve' corresponds to twisting down a unique face of $H(G)$, coloring P according to the swinging angle is equivalent to coloring a lattice of d-factors on $H(G)$ with a coloring c', in which each color corresponds to the face of $H(G)$ that is twisted down. Since each angle in G corresponds to a face in $H(G)$, each color of c on the lattice of spanning trees of G must correspond to a color of c' on the isomorphic lattice of d-factors on $H(G)$. Since c' must be an L-coloring, coloring c must also be an L-coloring. □

Lemma 4.5.9 again allows us to apply our results from Section 4.3.

Theorem 4.5.10. *Given a planar graph G with special vertex v^* and special face f^*, consider the distributive lattice of spanning trees on G with special vertex v^* and face f^*. Then each color in S-coloring of the lattice consists of all edges corresponding to moves of the form "the j^{th} swing-down operation at angle eve'" for a fixed angle eve' and positive integer j.*

Proof. By Lemma 4.5.9, there exists a valid L-coloring in which each color corresponds to a single angle being swung down. By Lemma 4.3.15, each color in this L-coloring is a union of colors of the S-coloring. Thus, for any angle eve', all colors in the S-coloring

corresponding to swing-down operations through eve' must occur in the same order in any complete downward path in the lattice of spanning trees. Thus, each color in the S-coloring corresponding to angle eve' must consist of all moves of the form "the j^{th} move at eve'" for some fixed j. □

By Theorem 4.5.10, all of the coloring results on S-colorings from Section 4.3 apply to the lattice of spanning trees. In particular, for any two spanning trees T_1 and T_2 such that $T_1 \geq T_2$, all sequences of swing-down operations must involve swinging down through the same multiset of angles by Theorem 4.3.11. For any two spanning trees T_1 and T_2, any shortest path from T_1 to T_2 involving swing-up and swing-down operations must involve swinging up and down through the same multiset of angles by Theorem 4.3.21.

Because of Theorem 4.5.10, all of the coloring results from Section 4.3 apply to the poset of spanning trees related by swinging down operations. In particular, we have the following two results:

Corollary 4.5.11. *Consider a planar graph G with spanning trees T_1 and T_2 such that $T_1 > T_2$ in the poset of spanning trees. Then all sequences of swing-down operations from T_1 to T_2 involve swinging down through the same multiset of angles.*

Proof. The poset of spanning trees is an LLD lattice with an S-coloring as defined in Theorem 4.5.10. As a consequence of Theorem 4.3.6, for any S-coloring c, all downward paths from T_1 to T_2 must contain edges of the same colors, with at most one edge of each color. Since each color corresponds to a swinging down move through a particular angle, all sequences of swing-down moves from T_1 to T_2 must involve swinging down through the same multiset of angles. □

Corollary 4.5.12. *Consider a planar graph G with spanning trees T_1 and T_2. Then any two shortest paths from T_1 to T_2 involving swinging up and swinging down operations must include the same number of swinging up operations through each angle eve', and the same number of swinging down operations through each angle eve'.*

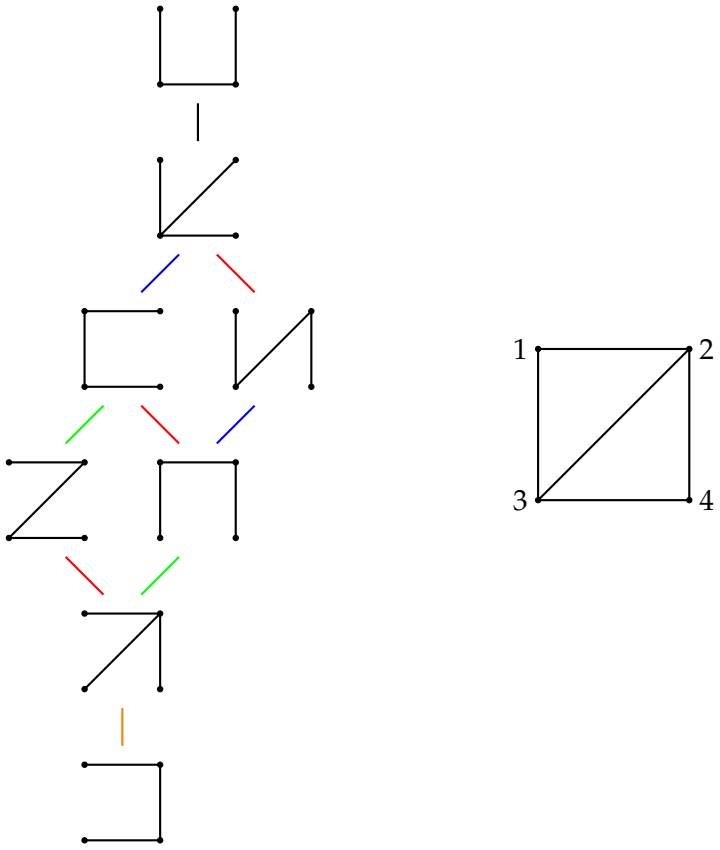


FIGURE 4.11: The partial ordering of spanning trees on the diamond graph (K_4 with an edge removed), as shown on the right. Each edge color in the Hasse diagram corresponds to swinging down through a particular angle (the induced process never requires swinging down multiple times through the same angle). Black corresponds to the angle 423, blue corresponds to 321, red corresponds to 342, green corresponds to 132, and orange corresponds to 234.

Proof. As the poset of spanning trees is an LLD lattice with an S-coloring as defined in Theorem 4.5.10, this follows from Lemma 4.3.19. □

4.6 The Sand Pile Model

We now turn to another application of these results. Bak, Tang, and Wiesenfeld introduce a model called the **sand pile model** [3]. While their work is much better known for introducing the concept of self-organized criticality and its impact on stabilization times, the model that they introduce is interesting in its own right. The model was

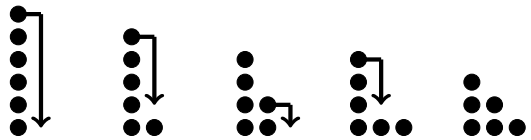


FIGURE 4.12: An example of a stabilization for a configuration with 6 grains of sand at the origin. Note that in the third step, we could have chosen to fire site 1 or site 2. Either would have led to the same final configuration.

studied in more detail by Goles and Kiwi [23], who compared the sand pile model to chip-firing processes and studied its lattice-theoretic properties. The sand pile model is not to be confused with the abelian sandpile model, which is much more well known despite being a far inferior model for the behavior of actual piles of sand.

In the sand pile model, begin with n grains of sand at site 1 on the positive integer lattice (although the results here hold for any non-increasing sequence of nonnegative integers that is eventually 0). A **sand configuration** is an ℓ-tuple of natural numbers $s = (s_1, s_2, \ldots s_\ell)$, which denotes the number of grains of sand at each site. If, at any point, we have $s_k \geq s_{k+1} + 2$, then site k can "fire" by sending one grain of sand from k to $k+1$ (increasing s_{k+1} by 1 and decreasing s_k by 1). An example of a sand pile model which fires to completion is shown in Figure 4.12.

A configuration corresponds to one of these ℓ-tuples, with configurations ordered so that configuration $s^1 \geq s^2$ if s^2 is reachable from s^1, and a move consists of a single grain of sand going from one site to the next when allowed by the rules. It has been shown that for every process generated by a sand pile model, there exists a chip-firing process that generates an isomorphic poset of configurations [23] [33], but our methods can be used to analyze this system directly.

We first show that this process is locally confluent.

Lemma 4.6.1. *In the sand pile model, if two sites can fire from the same configuration, then they can fire in either order.*

Proof. Suppose that in some configuration s_1, sites k_1 and k_2 ($k_1 \neq k_2$) are both able to fire. Firing k_1 cannot decrease the number of grains of sand at k_2 or increase the number

of grains of sand at $k_2 + 1$. As a result, k_2 can fire even after k_1 has done so. The same is true in reverse, so k_1 and k_2 can fire in either order. □

Further, there is at most one move available at a given site from any configuration. We obtain the following results.

Lemma 4.6.2. *Given a sand pile model R, consider the distributive lattice P of sand configurations of R. Let c be an edge coloring of P such that each color of c includes all edges that correspond to firing moves at a particular site k. Then c is a valid L-coloring.*

Proof. First, given a sand configuration s and a site k, there is at most one way to fire site k from configuration s. Thus, c satisfies condition (1) of L-coloring.

Consider a sand configuration s and sites k_1 and k_2. By Lemma 4.6.1, if it is possible to fire sites k_1 or k_2 from configuration s, then it is possible to fire from those sites in either order. Thus, c satisfies condition (2) of L-coloring, so it is a valid L-coloring. □

Lemma 4.6.2 allows us to apply our results from Section 4.3.

Theorem 4.6.3. *Given a sand pile model R, consider the distributive lattice P of sand configurations of R. Then each color in the S-coloring of P consists of all edges corresponding to moves of the form "the j^{th} firing move at site k" for a fixed site k and positive integer j.*

Proof. By Lemma 4.6.2, there exists a valid L-coloring in which each color corresponds to a single site being fired. By Lemma 4.3.15, each color in this L-coloring is a union of colors of the S-coloring. Thus, for any site k, all colors in the S-coloring corresponding to firing moves at k must occur in the same order in any complete downward path in the lattice of sand configurations. Thus, each color in the S-coloring corresponding to firing at site k must consist of all moves of the form "the j^{th} firing move at site k" for some fixed j. □

Thus, the colors of the S-coloring of poset P correspond to the moves of the move poset in the sand pile model. All results from Section 4.3 apply to the sand pile model. An example of an S-coloring of a configuration poset, along with the corresponding poset of join-irreducibles, is shown in Figure 4.13.

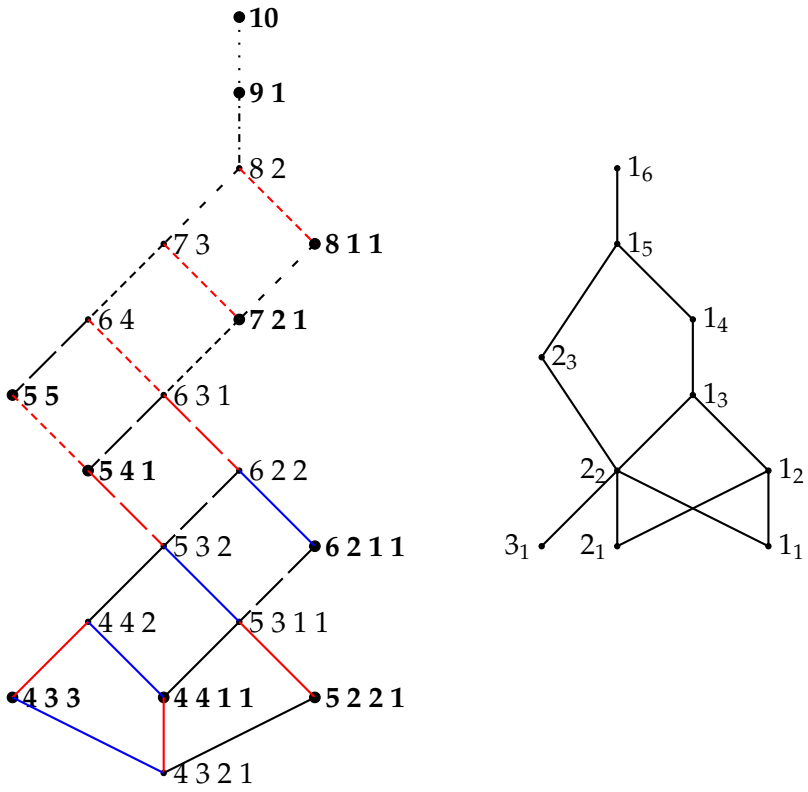


FIGURE 4.13: Left: the poset of configurations for the sand pile model with $n = 10$ grains of sand. The black, red, and blue edges correspond to firing moves at site 1, 2, and 3 respectively, thus forming an L-coloring. The various dashings correspond to a further subdivision into an S-coloring. Join-irreducibles are in bold. Right: the color poset of the 10 firing moves in this process.

4.7 S-Colorings and S_n EL-Labelings

Now, we compare LLD lattices and their corresponding S-colorings to another class of lattices that are similarly determined by an edge labeling: supersolvable lattices and their S_n EL-Labelings.

A finite lattice L is supersolvable if it contains a maximal chain M, which together with any other chain in L generates a distributive sublattice [48].

Given a finite graded lattice L of rank n, define an edge labeling λ to be a function from the edges of the Hasse diagram of L to $[n]$. Given a labeling λ, consider a maximal chain $m = (s = s_0 < s_1 < \cdots < s_k = t)$ on an interval $[s, t]$. m is **increasing** under labeling λ if $\lambda\big((s_0, s_1)\big) \leq \lambda\big((s_1, s_2)\big) \leq \cdots \leq \lambda\big((s_{k-1}, s_k)\big)$. Let $\leq_L$ denote the lexicographic ordering on maximal chains on $[s, t]$. Given two maximal chains $m = (s = s_0 < s_1 < \cdots < s_k = t)$ and $m' = (s = s'_0 < s'_1 < \cdots < s'_k = t)$, we have $m <_L m'$ if there exists some i such that $\lambda\big((s_i, s_{i+1})\big) < \lambda\big((s'_i, s'_{i+1})\big)$ and for all $j < i$, $\lambda\big((s_j, s_{j+1})\big) = \lambda\big((s'_j, s'_{j+1})\big)$. We provide the following definition:

Definition 4.7.1 ([40])**.** Let P be a finite graded poset of rank n with a $\mathbf{0}$ and a $\mathbf{1}$. An edge labeling $\lambda : E(P) \to [n]$ is an S_n EL-labeling if

(1) Every interval $[s, t]$ has exactly one increasing maximal chain m.

(2) Any other maximal chain m' of $[s, t]$ satisfies $\lambda(m') >_L \lambda(m)$.

(3) The labelings of any maximal chain on $[\mathbf{0}, \mathbf{1}]$ form a permutation of $[n]$.

McNamara shows that that a finite graded lattice of rank n is supersolvable if and only if it admits an S_n EL-labeling [40]. Thus, it is natural to compare the lattices admitting S_n EL-labelings to those admitting S-colorings, as both assign labelings to edges in a similar manner.

We will show that neither class of lattices contains the other. We can construct counterexamples to each inclusion relation.

Example 4.7.2. There exists an S_n EL-labelable lattice that does not admit an S-coloring. A Hasse diagram for such a lattice is shown in Figure 4.14, along with an S_n EL-labeling.

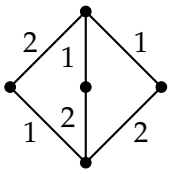


FIGURE 4.14: A poset P with an S_n EL-labeling, but no S-coloring

Every upward path from **0** to **1** in poset P contains exactly one 1 and one 2, including one such path that is increasing. All smaller intervals contain at most one edge, and so must satisfy the properties of S_n EL-labelings. Thus, the labeling in Figure 4.14 is an S_n EL-labeling.

However, poset P does not admit an S-coloring. Any coloring of the Hasse diagram of P that satisfies property (2) of S-coloring must have exactly one color (see Figure 4.15). However, this would result in every path from **1** to **0** using the same color twice, thus violating property (1) of S-coloring. Thus, P has no S-coloring, and the set of LLD lattices does not contain the set of supersolvable lattices.

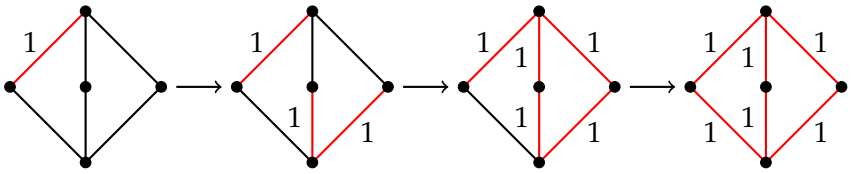
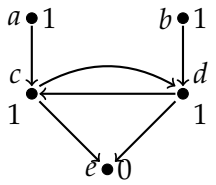


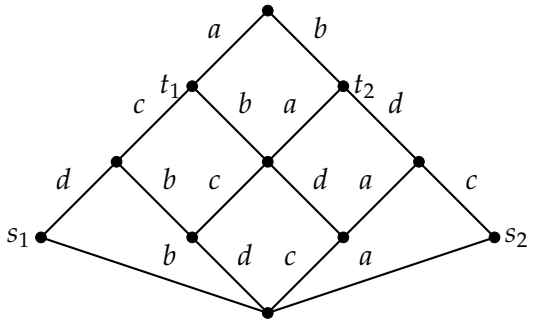
FIGURE 4.15: Suppose there exists an S-coloring, in which the color of the top-left edge is 1 (represented by red). Then by property (2) of S-coloring, the color of the bottom-middle and bottom right edges must be 1. This implies that the color of the top-middle and top-right edges is 1, which implies that the color of the bottom-left edge is 1. Thus, the color of all edges must be the same, but this contradicts property (1) of S-coloring.

Example 4.7.3. There exists an S-colorable lattice that does not admit an S_n EL-labeling. Consider the chip-firing process on the following graph and initial configuration.



Here, each vertex is labeled with a letter, as well as a number representing the number of chips at that site in the initial configuration. This chip-firing process produces

the following configuration poset P. Each edge is labeled with the corresponding vertex being fired:

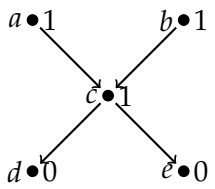


P is S-colorable because it is the poset of configurations for a chip-firing process. Since each site fires at most once, each color in the S-coloring of P corresponds to a specific vertex being fired. Suppose that an S_n EL-labeling exists for poset P. As every interval of length 2 is either a single path or a diamond, any S_n EL-labeling of P must satisfy the diamond property. Since there are 4 labels and the poset is of rank 4, any S_n EL-labeling must have exactly one label corresponding to each of the sites a, b, c, and d.

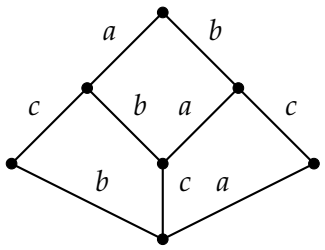
The interval $[s_1, t_1]$ consists of a single path with one edge corresponding to site d, and another edge corresponding to site c. Any S_n EL-labeling must have one increasing path from s_1 to t_1, so the label corresponding to d must be less than the label corresponding to c. However, the interval $[s_2, t_2]$ has the same edge labels in the opposite order, so any S_n EL-labeling must have a smaller label for c than for d. This is a contradiction, so no S_n EL-labeling exists for poset P.

As our goal is to produce a Venn diagram relating LLD, supersolvable, and distributive lattices, we include an additional example for completeness. It is known that all distributive lattices are both LLD and supersolvable [40], but we will show that this is a strict inclusion: namely, that there is an LLD, supersolvable lattice that is not distributive.

Example 4.7.4. Consider the chip-firing process with the following graph and initial configuration.

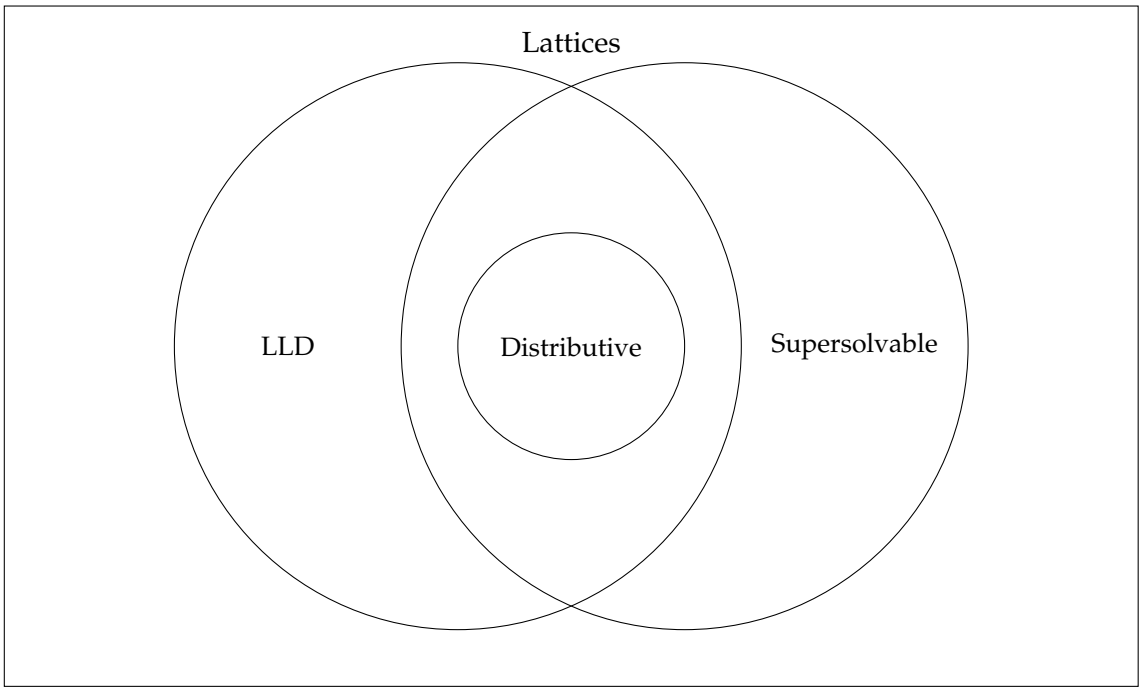


This chip-firing process produces the following poset P of configurations, in which each edge is labeled with the corresponding vertex being fired:



As P is the configuration poset of a finite chip-firing process, it is LLD. Furthermore, the labeling that assigns label 1 to all of the edges corresponding to firing moves at c, label 2 to firing moves at b, and label 3 to firing moves at a, is an S_n EL-labeling. However, P is not a distributive lattice (its dual does not satisfy property (2) of S-colorings, so P is not ULD).

We thus have the following relationship between our relevant classes of lattices: LLD and supersolvable lattices overlap, but neither class includes the other. Furthermore, distributive lattices are included in this overlap, and this is a strict inclusion. These relationships are summarized in the following Venn diagram:


Lattices
LLD
Distributive
Supersolvable

CHAPTER 5

Other Sorting Configurations on the Line

5.1 Introduction

In Chapter 3, we proved that labeled chip-firing on the line sorts from an initial position with $2m$ labeled chips at the origin. Using similar methods, we were able to prove sorting on several variations of the line. We changed the number of edges at each site, either by making copies of edges, by adding self-loops, or by placing different numbers of edges at different sites.

In each case, however, the initial configuration remained relatively consistent. A collection of labeled chips were placed at the origin. Certain numbers of chips were determined to lead to sorting, because those numbers of chips led to a special grid structure in the move poset, and that grid was large enough to sort the chips.

However, by only considering one initial configuration per graph, we are ignoring a lot of the variation that can appear in labeled chip-firing processes, as well as their underlying unlabeled chip-firing processes. The goal of this chapter is to examine the dynamics of labeled chip-firing on the line from other initial configurations, and to try to determine which configurations sort, and which ones do not.

The first new class of initial configurations studied in labeled chip-firing came from the notion of root system chip-firing. Hopkins, McConville, and Propp first introduce this idea in [27], and Galashin et al. study the idea further in [22] and [21]. They view labeled chip-firing as a type A special case of a more general chip-firing problem on root systems, in which types B, C, and D allow for additional types of firing moves. We study root system chip-firing in Chapter 7.

In the work of Galashin et al. [21], each root system has associated with it certain fundamental weights, which correspond to initial labeled chip-configurations. Some fundamental weights are conjectured to sort, while others do not. On type A root systems, 3 classes of fundamental weights are conjectured to sort. These classes are:

(1) $2m$ chips at the origin

(2) $2m$ chips, with the largest chip at site 1 and the rest at the origin

(3) $2m+1$ chips, with the largest m chips at site 1 and the rest at the origin

Class (1) is proven to sort in [27] and again in [31] (see Chapter 3 of this thesis). Sorting for classes (2) and (3) are mentioned by Klivans and Liscio in [31] and [35], but the proofs are not provided. We prove sorting for classes (2) and (3) in this chapter, by showing that their move posets exhibit a very similar diamond grid structure to class (1).

However, there are many sorting configurations that do not fall into the 3 classes listed above. These other sorting configurations are either firing-equivalent to classes (1)-(3), or the initial configurations are close enough to sorting from the beginning that an entire diamond of locally confluent moves is not needed to guarantee sorting. We address these cases by studying classes of firing equivalent unlabeled configurations. Each equivalence class produces a differently shaped rectangular diamond, and different initial configurations produce move posets containing subsets of the corresponding diamond.

Our study of confluent configurations reduces largely to answering two questions:

(1) Which unlabeled chip configurations result in sorted final configurations if the chips in the initial configuration are labeled in weakly sorted order?

(2) Given a sorting configuration that answers question (1), in what ways can the labels be reassigned (without changing the underlying unlabeled configuration), while still maintaining the "sorting" property?

We provide a complete answer to question (1), and we also address question (2) in a few special cases. Note that chip-firing with $2m$ chips at the origin does not allow for question 2 to be considered. There is no meaningful way to rearrange the labels of the chips when all chips are at the same site.

In Section 5.2, we study classes of firing equivalent chip configurations on the line. Each class has a representative stable element, as well as a representative "maximally unstable" element. These classes turn out to be relevant to labeled chip-firing, as all configurations in a given equivalence class produce chip-firing processes that end in similar ways.

In Section 5.3, we prove a more general form of the Diamond Lemma from Chapter 3. For every connected configuration whose stabilization contains a hole (a location with no chips), there exists a grid of firing moves near the end of the process. If the number of firing moves at each site exceeds certain bounds, this grid forms a diamond of firing moves. These more general results allow us to prove sorting for the remaining open problems in Type A root system chip-firing, and they also provide a method for proving sorting on other labeled chip configurations.

In Section 5.4, we provide a full classification of which weakly sorted configurations sort, and which do not. We accomplish this by proving a partial converse to the generalized Diamond Lemma, which states that every move not covered by the generalized Diamond Lemma may take place with at least 3 chips present. Using both directions of the Diamond Lemma and some chip bounds modified from Hopkins, McConville, and Propp, we provide necessary and sufficient conditions for sorting.

5.2 Firing Equivalence Classes on the Line

Our results about firing equivalent configurations and their stabilizations are relevant to the study of labeled chip-firing. We say that a labeled configuration ℓ **sorts** if a labeled chip-firing process beginning with ℓ always ends with the chips in sorted order, for any stabilizing sequence of legal moves starting from ℓ.

We say that a labeled chip-configuration is **weakly sorted** if for any two chips $i < j$, either chips i and j are at the same site, or chip i is to the left of chip j. Given an unlabeled configuration u with n chips, we define $ws(u)$ to be the unique labeled configuration consisting of n labeled chips in weakly sorted order with unlabeled configuration u. We say that an unlabeled configuration u with n chips **sorts** if $ws(u)$ sorts.

Given an unlabeled chip configuration u, define the **support** $supp(u)$ to be the set of vertices that have at least one chip in configuration u. Define $max(u)$ to be the largest (rightmost) site in $supp(u)$, and $min(u)$ to be the smallest (leftmost) site in $supp(u)$.

Given an unlabeled configuration u, define the vector $v(u)$ such that each component $v_i(u)$ represents the number of chips at site i in u.

Given two unlabeled configurations u_1 and u_2, we say that $v(u_1) \leq v(u_2)$ if each component of $v(u_1)$ is less than or equal to the corresponding component of $v(u_2)$. Unlabeled configurations $\{u_1, u_2, \ldots u_k\}$ form a **partition** of configuration u if $v(u_1) + v(u_2) + \cdots + v(u_k) = v(u)$.

Given an unlabeled chip configuration u with n chips, define the mean chip position $mean(u)$ to be

$$mean(u) = \frac{1}{n} \sum_{i=min(u)}^{max(u)} i \cdot v_i(u)$$

An unlabeled configuration u is **disconnected** if u can be partitioned into configurations u_1 and u_2 such that $max(stab(u_1)) < min(stab(u_2))$. If u is not disconnected, it is **connected**.

An unlabeled configuration u_1 is a **connected component** of configuration u if

- $v(u_1) \leq v(u)$
- u_1 is connected.
- There does not exist a connected u_2 such that $v(u_1) \leq v(u_2) \leq v(u)$, $u_1 \neq u_2$.

Note that, due to the definition of connectedness, we can partition any configuration u into connected components $u_1 \dots u_k$ such that the following hold:

- For each $1 \leq i \leq k-1$, $max(u_i) < min(u_{i+1})$
- For each $1 \leq i \leq k-1$, $max(stab(u_i)) < min(stab(u_{i+1}))$

We can show that we only need to focus on connected configurations when studying the sorting of unlabeled configurations.

Lemma 5.2.1. *An unlabeled configuration u sorts if and only if each of its connected components sorts.*

Proof. If u sorts, then any subset of $ws(u)$'s chips must end up in sorted order. In particular, the chips in a particular connected component must end up in sorted order, so each connected component sorts.

Suppose that each of u's connected components sorts. Consider a labeled chip-firing process beginning with $ws(u)$. For a given u_i, this implies that if the chips in u_i are only fired with other chips in u_i, then they will stabilize in sorted order. However, since u_i is a single connected component, the support of $stab(u_i)$ does not overlap with the stabilizations of any other connected components, so there can be no firing moves involving chips from two different connected components. Thus, all chips in u_i end in sorted order for all i.

Furthermore, for any $i < j$, the chips in u_i must remain to the left of the chips in u_j throughout the process. Since the chips begin in weakly sorted order, all chips in u_i must have lower values than all chips in u_j for all $i < j$. Thus, no two chips in different connected components may be out of order after stabilization. Since the chips in each connected component also end in sorted order, this implies that u sorts if each of its connected components sorts. □

By Lemma 5.2.1, we can reduce the problem of determining whether a configuration sorts to the problem of determining whether a connected configuration sorts.

We say that two unlabeled configurations u_1 and u_2 are **weakly firing equivalent** if u_2 can be reached from u_1 through (possibly illegal) firing and un-firing moves. We say that u_1 and u_2 are **firing equivalent** if u_2 can be reached from u_1 through legal firing and un-firing moves.

We now prove results on classes of weakly firing equivalent and firing equivalent configurations.

Lemma 5.2.2. *Consider a finite unlabeled chip configuration u on the line. Then there exists a unique configuration u' such that*

(1) u' is weakly firing equivalent to u.

(2) $supp(u')$ is either a single site, or two adjacent sites.

Proof. We first show that such a u' exists. Let $k = \lfloor mean(u) \rfloor$. Note that k is not changed by firing and un-firing moves. We show that u' has all of its chips at k and (possibly) $k+1$.

We first show that it is possible through firing and un-firing moves to reach a configuration u_1 in which every site less than k has 0 chips. We prove this by induction on $k - min(c)$. If $k - min(c) = 0$, then we let $u_1 = u$. Otherwise, we perform $v_{min(u)}(u)$ un-firing moves at site $min(u) + 1$. This results in a configuration with 0 chips at site $min(u)$, so the smallest site with a nonzero number of chips is at least $min(u) + 1$. This means that we can get from u to another configuration u_2 such that $k - min(u_2) \leq k - min(u) - 1$. By our inductive assumption, it is possible through firing and un-firing moves to get from u_2 to a configuration u_1 with 0 chips at every site less than k. Thus, it is possible through firing and un-firing moves to get from u to that same u_1.

We can analogously show that it is possible through firing and un-firing moves to get from u_1 to a state u' in which every site greater than $k+1$ has 0 chips. We induct on $max(u_1) - (k+1)$. If $max(u_1) - (k+1) = 0$, then we let $u' = u_1$. Otherwise, we

perform $v_{max(u_1)}(u_1)$ un-firing moves at site $max(u_1) - 1$. This results in a configuration with 0 chips at site $max(u_1)$, so the largest site with a nonzero number of chips is at most $max(u_1) - 1$. This means that we can get from u_1 to another configuration u_3 such that $max(u_3) - (k+1) \leq max(u_1) - (k+1) - 1$. By our inductive assumption, it is possible through firing and un-firing moves to get from u_3 to a configuration u' with 0 chips at every site greater than $k+1$, and thus possible through firing and un-firing moves to get from u_1 to that same u'.

Further, since the proof that we can reduce $max(u)$ down to at most $k+1$ never requires firing or un-firing at a site less than $k+1$, the resulting configuration u' must still have no chips to the left of k. As a result, the final configuration u' has all of its chips at k or possibly $k+1$. Furthermore, since the mean chip position is not affected by firing moves, we must have $mean(u') = mean(u)$, which is between k and $k+1$. Thus, k and $k+1$ end with nonnegative numbers of chips.

Now, we show that this configuration is unique. The final configuration has $v_k(u')$ chips at site k and $v_{k+1}(u')$ chips at site $k+1$, where $v_k(u') + v_{k+1}(u') = n$. Furthermore, since the sum of all chip positions is not changed by firing and un-firing, we must have

$$\begin{aligned} & kv_k(u') + (k+1)v_{k+1}(u') \\ =& min(u)v_{min(u)}(u) + (min(u)+1)v_{min(u)+1}(u) + \cdots + max(u)v_{max(u)}(u) \\ =& n \cdot mean(u) \end{aligned}$$

This is a nonsingular linear system with two equations and two unknowns, and it thus has the single solution:

$$v_k(u') = n(1 + k - mean(u))$$

$$v_{k+1}(u') = n(mean(u) - k)$$

As no other pair of adjacent chip quantities could produce the same mean chip position, this is the only possible solution with chips at one site or two adjacent sites.

□

As a consequence of Lemma 5.2.2, we can define the **unstabilization** of an unlabeled chip-configuration u, denoted $unstab(u)$, as the single one- or two-pile configuration u' that is weakly firing equivalent to u.

Note that the proof of Lemma 5.2.2 does allow for illegal firing moves. There are chip configurations where illegal firing moves are necessary to get from u to $unstab(u)$. For example, when u consists of a chip at site -1 and a chip at site 2, no legal firing or un-firing moves can be performed, but u is weakly firing equivalent to a configuration $unstab(u)$ with 1 chip at site 0 and 1 chip at site 1.

The proof of Lemma 5.2.2 is much more straightforward (and does not rely on illegal firing moves), when $stab(u) = stab(unstab(u))$, which, as we show later, always occurs when u is connected.

Lemma 5.2.3. *If $stab(u) = stab(unstab(u))$, then u and $unstab(u)$ are firing equivalent.*

Proof. We find a legal firing sequence s from u to $stab(u)$, and a legal firing sequence s' from $unstab(u)$ to $stab(unstab(u))$. We can then get from u to u' by performing s, and then performing s' in reverse. Since both s and s' are sequences of legal moves, all firing and un-firing moves to get from u to $unstab(u)$ are legal. □

Our goal is to prove the following result:

Theorem 5.2.8. *Consider an unlabeled configuration u on the line. If u is connected, then u and $unstab(u)$ are firing equivalent.*

We will prove this result by showing that $stab(u) = stab(unstab(u))$, allowing us to use Lemma 5.2.3. We can do this by deriving the form of $stab(u)$ and $stab(unstab(u))$.

Consider a stable chip configuration u on the line. We define a **hole** in u to be a site k with $min(u) < k < max(u)$ such that u contains no chips at site k.

Given a site k, define configuration e_k to be the configuration consisting of a single chip at site k. We want to prove that connected configurations stabilize to configurations with at most one hole.

Lemma 5.2.4. *Consider a stable chip configuration u on the line. If $stab(u)$ has at most one hole, and $min(u) - 1 \leq k \leq max(u) + 1$, then $stab(u + e_k)$ also has at most one hole.*

Proof. Add a chip at site k to configuration u. There are three cases to consider.

Case 1: u has no chips at site k. Either $k = min(u) - 1$, $k = max(u) + 1$, or k is the site of the lone hole in u. In the first two cases, the max or min of the configuration is changed, but the number of holes is not. If k is the site of the lone hole in u, then the number of holes is reduced from 1 to 0.

Case 2: u has no holes, and $min(u) \leq k \leq max(u)$. Then by Proposition 2 of [27], the stabilization $stab(u + e_k)$ consists of a single chip at every site from $min(u) - 1$ to $max(u) + 1$, except for site $min(u) + max(u) - k$, which has 0. Thus, the resulting configuration still has at most one hole.

Case 3: u has a hole at site $i < k$. Let configuration u' consist of a single chip at every site from $min(u)$ to $i - 1$, and u'' consist of a single chip at every site from $i + 1$ to $max(u)$. Just as in Case 2, $stab(u'' + e_k)$ consists of a single chip at every site from i to $max(u) + 1$ except for $i + 1 + max(u) - k$. This does not overlap with configuration u', so $stab(u + e_k) = u' + stab(u'' + e_k)$, which consists of a chip at each site from $min(u)$ to $max(u) + 1$, except for a hole at site $i + 1 + max(u) - k$.

Case 4: u has a hole at site $i > k$. The same argument applies as in case 3, resulting in a single chip at each site from $min(u) - 1$ to $max(u)$, except for site $min(u) + i - 1 - k$.

□

Lemma 5.2.5. *Consider two connected unlabeled configurations u and u' on the line, such that $u + u'$ is also connected. If $stab(u)$ and $stab(u')$ each have at most one hole, then $stab(u + u')$ also has at most one hole.*

Proof. It is known that $stab(u + u') = stab(stab(u) + stab(u'))$ [30]. Thus, we only need to consider $stab(u)$ and $stab(u')$. $stab(u) + stab(u')$ consists of some sites with 1 chip, some sites with 2 chips, and up to two holes with 0 chips. We need to consider two cases.

Case 1: $stab(u) + stab(u')$ has at most one hole. If there are no sites with multiple chips, then $stab(u) + stab(u')$ has already stabilized, and the result holds. Otherwise,

let $k_1, k_2, \dots k_j$ be the sites with multiple chips in $stab(u) + stab(u')$. Let $u_1 = e_{k_1} + e_{k_2} + \cdots + e_{k_j}$, and let $u_2 = stab(u) + stab(u') - u_1$ consist of a single chip at every site that is nonempty in $stab(u) + stab(u')$. We have that

$$stab(stab(u) + stab(u')) = stab(u_2 + u_1)$$

$$= stab(\dots stab(stab(stab(u_2 + e_{k_1}) + e_{k_2}) + e_{k_3}) \dots)$$

In each case, we are adding a single chip in the middle of a stable configuration with at most one hole, so by Lemma 5.2.4, the resulting configuration has at most one hole, as desired.

Case 2: $stab(u) + stab(u')$ has two holes. Suppose without loss of generality that $min(u) \leq min(u')$. If $stab(u)$ has a hole at k and $stab(u')$ has a hole at k', the fact that $stab(u) + stab(u')$ has two holes implies that $max(u) < k'$ and $min(u') > k$, since the only way for the holes to remain after adding u and u' are if the holes coincide (which would only result in one hole rather than two), or if the hole of each configuration does not overlap with the support of the other configuration.

This implies that the configuration $stab(u) + stab(u')$ consists of some sites with one chip each to the left of k, some sites with one chip each to the right of k', and some sites with one or two chips between k and k'. Since $stab(u) + stab(u')$ is connected, we must have $max(u) \geq min(u')$, so there must be at least one site in $stab(u) + stab(u')$ with two chips.

Just as in case 1, we define $k_1, k_2, \dots k_j$ be the sites with multiple chips in $stab(u) + stab(u')$, $u_1 = e_{k_1} + e_{k_2} + \cdots + e_{k_j}$ and $u_2 = stab(u) + stab(u') - u_1$ consist of a single chip at every site that is nonempty in $stab(u) + stab(u')$. We again get

$$stab(stab(u) + stab(u')) = stab(u_2 + u_1)$$

$$= stab(\dots stab(stab(stab(u_2 + e_{k_1}) + e_{k_2}) + e_{k_3}) \dots)$$

$u_2 + e_{k_1}$ consists of a single chip at every site from $k + 1$ to $k' - 1$, except for site k_1,

which has 2 chips, as well as single chips at every site from $min(u)$ to $k-1$, and from $k+1$ to $max(u')$. Thus, $stab(u_2 + e_{k_1})$ consists of a single chip at every site from k to k', except for a hole at $k+k'-k_1$, as well as a chip at each site from $min(u)$ to $k-1$, and $k+1$ to $max(u)$. This includes exactly one chip at every site from $min(u)$ to $max(u')$, except for a single hole at $k+k'-k_1$. By Lemma 5.2.4, every subsequent stabilization will continue to produce a configuration with at most one hole, so $stab(stab(u)+stab(u'))$ has at most one hole. □

Lemma 5.2.6. *For every connected unlabeled configuration u on the line, $stab(u)$ has at most one hole.*

Proof. We proceed by strong induction on the number of piles in u. If u has one pile with an odd number of chips, then $stab(u)$ has no holes. If u has one pile with an even number of chips, then $stab(u)$ has 1 hole.

Otherwise, suppose that u has $p \geq 2$ piles. Then by the definition of connectedness, it must be possible to decompose u into two connected components u' and u'' with p' and p'' piles respectively, such that $p', p'' < p$ and $v(u') + v(u'') = v(u)$. By our inductive assumption, $stab(u')$ and $stab(u'')$ must both have at most one hole, so by Lemma 5.2.5, $stab(u)$ must also have at most one hole, as desired. □

Lemma 5.2.7. *For any two n-chip stable configurations $u_1 \neq u_2$ on the line, such that u_1 and u_2 each have at most one hole, $mean(u_1) \neq mean(u_2)$.*

Proof. Suppose that u_1 has no hole, and $min(u_1) = k$. Then

$$mean(u_1) = \frac{1}{n}\left(nk + \frac{(n-1)n}{2}\right) = k + \frac{n-1}{2}$$

If, instead, u_1 satisfies $min(u_1) = k$ and has a hole at site $k+j$, $1 \leq j \leq n$, then

$$mean(u_1) = \frac{1}{n}\left(nk + \frac{n(n+1)}{2} - j\right) = k + \frac{n+1}{2} - \frac{j}{n}$$

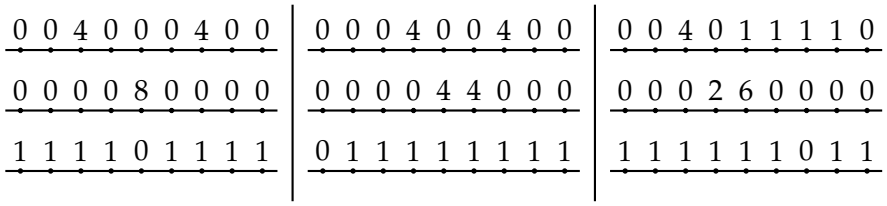


FIGURE 5.1: The first row contains three connected configurations of 8 chips. Below each configuration is its corresponding one- or two pile unstabilization, as well as its stabilization.

Note that this is distinct for each value of j, and it is strictly between the mean values of configurations with no holes with minima at k and $k+1$ respectively, so all such configurations have distinct mean chip positions. □

This brings us to the main result of this section. We can show that any connected configuration is firing equivalent to a unique configuration with one or two piles of chips.

Theorem 5.2.8. *For every connected unlabeled configuration u on the line, u is firing equivalent to $unstab(u)$.*

Proof. By Lemma 5.2.2, u is weakly firing equivalent to $unstab(u)$. Furthermore, since firing and un-firing do not change the mean chip position of a configuration, $mean(u) = mean(unstab(u))$. By Lemma 5.2.7, there is only one stable configuration with at most one hole with the same mean chip position as u and $unstab(u)$. Since u and $unstab(u)$ are both connected (the only disconnected configuration at two adjacent sites occurs if there is a single chip at each site, but this configuration is not firing equivalent to any connected configuration), we must have that $stab(u)$ and $stab(unstab(u))$ have at most one hole by Lemma 5.2.6. This implies that $stab(u) = stab(unstab(u))$, so $unstab(u)$ is firing equivalent to u. □

Some examples of connected configurations, along with their corresponding one- or two-pile unstabilizations, and their corresponding stabilizations, are shown in Figure 5.1.

We can show that these "unstabilizations" are special, in that they are the unique configurations in their firing equivalence classes that are "as far away as possible" from the corresponding stable configurations.

Lemma 5.2.9. *Given a connected unlabeled configuration u on the line, the number of firing moves required for u to stabilize is less than or equal to the number of firing moves required for $unstab(u)$ to stabilize, with equality if and only if $u = unstab(u)$.*

Proof. Without loss of generality, we assume that $0 \leq mean(u) < 1$. Otherwise, we can shift the coordinates so that this is the case, which would not affect the number of firing moves to stabilize.

In this case, $unstab(u)$ consists of at least 1 chip at site 0, and possibly some chips at site 1. Note that the sum of the squares of the coordinates of the chips increases by 2 with each firing move, while the sum of the coordinates of the chips remains the same throughout the firing process. In $unstab(u)$ the sum of the coordinates is equal to the sum of the squares of the coordinates, which is equal to the number of chips at site 1. Thus, this is equivalent to proving that if we have n chips with coordinate sum $k < n$, then the sum of the squares is minimized if k of the chips are at site 1, and the other $n - k$ are at site 0.

Consider an unlabeled chip configuration u with n chips, where the sum of the coordinates of the chips k satisfies $0 \leq k < n$. If the sum is 0, then the sum of squares is clearly minimized when all of the chips are at site 0. Otherwise, we have $0 < mean(u) < 1$, so there must be at least one chip at a site less than or equal to 0, and at least one chip at a site greater than or equal to 1.

Suppose there is a chip a at a site $s > 1$. There must be a chip b at site less than or equal to 0. If we move a to the left and b to the right, then the sum of squares changes by $(a-1)^2 - a^2 + (b+1)^2 - b^2 = 2(b - a + 1)$. Since $b - a$ is at most -2, this result is negative, so there exists another configuration with a lower sum of squares.

Similarly, suppose there is a chip a at site $s < 0$. There must be a chip b at site greater than or equal to 1. If we move a to the right and b to the left, then the sum of squares

changes by $(a+1)^2 - a^2 + (b-a)^2 - b^2 = 2(a-b+1)$. Since $a-b$ is at most -2, this result is negative, so there exists another configuration with a lower sum of squares.

Thus, no configuration with a chip that is not at site 0 or 1 can have the minimum sum of squares in its equivalence class, so $unstab(u)$ requires more firing moves to stabilize than any other configuration in its equivalence class, as desired. □

Stable configurations are associated with solutions to certain energy minimization problems [4] [24]. We conjecture that "unstabilizations" are solutions to a related energy maximization problem, although we do not consider that here.

The results so far in this section have merits on their own. We have shown that any connected configuration stabilizes to a configuration with one hole. This will help us work toward a more general version of the Diamond Lemma, in which the last move in the process must take place at the site with the hole, and the left and right corners of the diamond correspond to the single moves at the left- and rightmost sites that fire.

In addition to each firing equivalence class having a nice representative stable element, each class also has a representative "maximally unstable" element in which all chips are at one or two adjacent sites. This "unstabilization" of all configurations in the class corresponds to the unlabeled version of a fundamental weight of Type A root system chip-firing (see [21] and Chapter 7 of this thesis). Thus, for every connected configuration in a particular equivalence class, the end of the process looks very similar to the end of the process for the corresponding "unstabilizing configuration."

Firing equivalence classes are crucial to classifying which initial labeled configurations sort and which do not. Given an unlabeled configuration u with n chips, the location (or nonexistence) of the hole in $stab(u)$ determines the nature of the grid structure at the bottom of the move poset for chip-firing from u. The grid structure of the unlabeled move poset can be combined with bounds on chip positions throughout the process to determine whether a labeled configuration sorts, or whether specific chips have the potential to not end in their sorted positions. In section 5.4, we use these grid structures to provide a complete classification of which weakly sorted configurations sort and which do not.

While we are able to use the results of this section to classify sorting and non-sorting configurations, we originally introduced unstabilizing configurations with the intention of showing that this classification took a specific form. We conjectured that in any firing equivalence class, the "worst" connected configuration at sorting would be the unstabilizing configuration, and that in general, the more spread out the (weakly sorted) chips are at the beginning of the process, the better the configuration will be at sorting. The conjecture would look something like this:

Conjecture 5.2.10 (False)**.** *Consider a connected unlabeled configuration u on the line. Then the number of possible labeled stabilizations of $ws(u)$ is less than or equal to the number of possible labeled stabilizations of $ws(unstab(u))$.*

There is also a weaker version, which looks like this:

Conjecture 5.2.11 (Weaker, but still false)**.** *Consider a connected unlabeled configuration u on the line. If $ws(unstab(u))$ sorts, then $ws(u)$ sorts.*

Note that the second conjecture is strictly weaker than the first, as it is a special case of the first in which the number of possible labeled stabilizations of $ws(unstab(u))$ is equal to 1. We have the following counterexample to both conjectures:

Example 5.2.12. There exists a connected unlabeled configuration u on the line such that $ws(unstab(u))$ sorts, but $ws(u)$ does not sort. An example of such a configuration is shown in Figures 5.2 and 5.3.

Example 5.2.12 shows an unlabeled configuration u that is firing equivalent to a configuration with 12 chips at the origin, but while the configuration with 12 chips at the origin sorts (see Chapter 3), $ws(u)$ has 9 possible stabilizations (chip 5 can be inverted with chip 4, chip 6, or neither, and chip -5 can be inverted with chip -4, chip -6, or neither).

While example 5.2.12 shows that the classification of sorting configurations doesn't take as nice of a form as we initially hoped, we are still able to provide a complete classification of which weakly sorted initial configurations must produce sorted stable configurations. See Section 5.4.

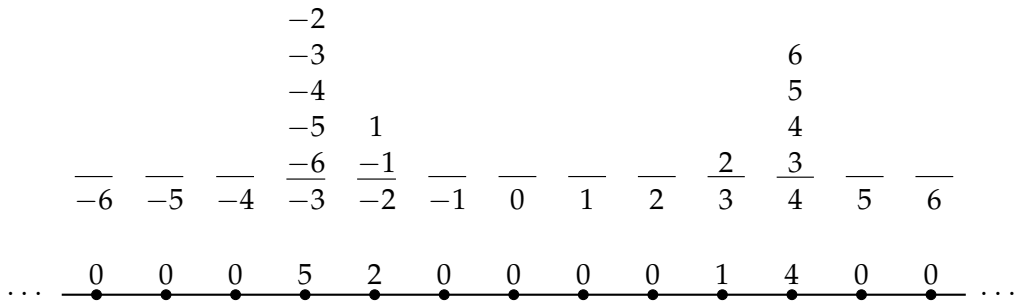


FIGURE 5.2: An example of a labeled configuration ℓ (and corresponding unlabeled configuration u shown below) with 12 chips. u is connected, the chips are in weakly sorted order, and $unstab(u)$ consists of 12 chips at the origin, but ℓ is not guaranteed to sort.

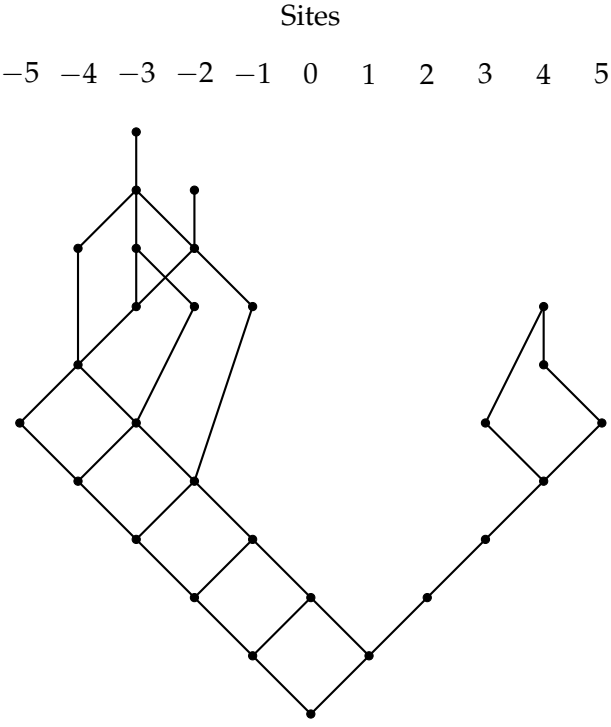


FIGURE 5.3: The move poset for the unlabeled chip configuration from Figure 5.2. Note that this configuration exhibits some of the same diamond structure as the configuration with 12 chips at the origin, but there are in some sense "not enough diamond moves" to guarantee sorting.

Remark 5.2.13. Connected configurations are discussed by Galashin et al. [21] in the context of root system central-firing. They define connectedness in terms of labeled chips, although their definition is equivalent to ours. Galashin et al. also discuss a few of the other concepts mentioned in this section, including holes (which they call *gaps*), and mean chip positions (which they call *centers of mass*). They also define a *pseudo-stabilization* of any unlabeled configuration u, which is equivalent to $stab(unstab(u))$ in our language.

They used these concepts to determine stabilizations of unlabeled configurations, and to determine which labeled chips could fire together in given labeled configurations. They also proved which configurations are connected in labeled chip-firing, as well as in central-firing on other root systems. They used gaps in stable configurations to prove that certain initial weights in Type A central firing do not sort.

5.3 The Generalized Diamond Lemma

In Chapter 3, we proved sorting for a configuration with $2m$ chips at the origin. The proof consisted of three main components:

(1) Prove the last moves in the process follow a locally confluent grid structure.

(2) Bound the positions that chips can reach throughout the firing process.

(3) Combine (1) and (2) to uniquely constrain a chip's final location.

By generalizing these results beyond the single case with $2m$ chips at the origin, we can prove several convergence results at once, and ultimately classify every weakly sorted labeled chip configuration as either "sorting" or "non-sorting." We begin with some additional notation.

Given a connected configuration u for which $stab(u)$ has a single hole, we define $h(u)$ to be the location of the hole in $stab(u)$. For a site k, we define $f_u(k)$ to be the number of firing moves needed at site k for u to stabilize. For a site k, we say that neighbor k' of k is an **outer neighbor** of k if k is between $h(u)$ and k', inclusive, and we say that k' is an **inner neighbor** of k if k' is between $h(u)$ and k, inclusive.

We use this notation to provide a more general version of the Diamond Lemma:

Lemma 5.3.1 (General Diamond Lemma on the Line)**.** *Consider a connected unlabeled configuration u on the line such that $stab(u)$ has a single hole $h(u)$. Consider a firing move k_j such that the following hold:*

- *For each site i such that $min(h(u),k) \leq i \leq max(h(u),k)$, $f_u(i) \geq j$.*
- *For each site i such that $max(h(u),k) < i < max(h(u),k)+j$, $f_u(i) \geq j+max(h(u),k)-i$.*
- *For each site i such that $min(h(u),k)-j < i < min(h(u),k)$, $f_u(i) \geq j+i-max(h(u),k)$.*

Then the following must hold:

- *For each outer neighbor k'' of k, if site k'' fires at least j times, then firing move k_j takes place after firing move k'_j.*
- *For each inner neighbor k' of k, if site k' fires at least $j+1$ times, then firing move k_j takes place after firing move k''_{j+1}.*
- *Firing move k_j takes place with exactly 2 chips present at site k.*

Proof. We first prove that move $h(u)_1$ must take place after all of the moves at its neighbors. The only required inequality is that $f_u(h(u)) \geq 1$, which must be true for move $h(u)_1$ to occur. Site $h(u)$ ends the process with 0 chips, so it cannot receive any additional chips after its last firing move. Thus, the last firing move at site $h(u)$ must occur after all firing moves at each of its neighbors. Since site $h(u)$ has no chips remaining after firing, it must have exactly 2 chips present immediately prior to move $h(u)_1$.

Now, we induct on $|k-h(u)|$ to show that the last move at each site other than $h(u)$ must take place after the last move at its outer neighbor, and after the second-to-last move at its inner neighbor. We assume that at least one move takes place at every site between $h(u)$ to k, inclusive.

Consider a site k with inner neighbor k' and outer neighbor k''. We assume that move k'_1 occurs after move k_1. Thus, site k receives 1 chip from site k' after the last firing move at site k. Since k terminates with 1 chip, this means that no other moves at neighbors of k may occur after move k_1. In particular, k_1 occurs after k'_2 and k''_1. Furthermore, since k receives 1 chip after firing, site k must have 0 chips left immediately after move k_1, so site k_1 must occur with exactly 2 chips present at site k. Thus, by induction on $|k - h(u)|$, we have that the result holds for the last move at each site.

Finally, we proceed by strong induction on $j + |k - h(u)|$, for firing move k_j. We want to show that move k_j takes place after move k'_{j+1} for each inner neighbor k' of k, and that k_j takes place after move k''_j for each outer neighbor k'' of k. We assume that move k_j is at the top of a j by $|k - h(u)| + j$ "diamond" of moves, each of which is assumed for induction to satisfy the conditions of this lemma.

Specifically, we assume that move k_j occurs before move k'_j for each inner neighbor k' of k, and move k''_{j-1} for each outer neighbor k'' of k, which in turn take place before move k_{j-1}. Between the time that k_j is about to occur and the time that move k_{j-1} is about to occur, site k must lose 2 chips due to firing move k_j, and gain at least 2 chips due to firing moves at its neighbors. Since there are assumed to be exactly 2 chips present at site k for move k_{j-1}, there must be at most (and thus exactly) 2 chips present at site k for move k_j. In particular, move k'_{j+1} must take place before move k_j, for every inner neighbor k' of k, and move k''_j must take place before move k_j, if such moves exist. The result follows by strong induction.

□

Lemma 5.3.1 is shown visually in Figure 5.4.

As a corollary, we can prove the existence of a diamond of firing moves in the grid from Lemma 5.3.1. Given a connected unlabeled configuration u such that $stab(u)$ has hole $h(u)$, let $h'(u) = min\big(stab(u)\big) + max\big(stab(u)\big) - h(u)$. $h'(u)$ represents the x-coordinate of the top of the diamond. Define the **diamond number** of a site k $d_u(k)$ as follows:

(1) For $min\big(stab(u)\big) + 1 \leq k \leq min\big(h(u), h'(u)\big)$, we have

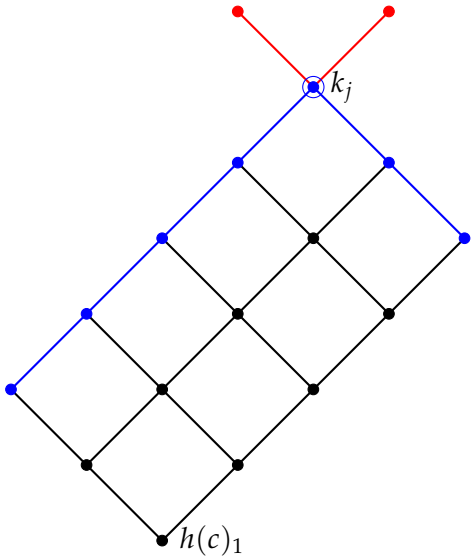


FIGURE 5.4: An example of a step in the proof of Lemma 5.3.1. The vertices on the top left and top right sides of the diamond are colored in blue, and the two vertices above the diamond are colored in red. If the chip firing process has enough moves at each site so that all of the blue moves occur, then move k_j must fit into the grid, firing with exactly 2 chips present and occurring after the two red moves if they occur.

$$d_u(k) = k - min\big(stab(u)\big).$$

(2) For $min\big(h(u), h'(u)\big) \leq k \leq max\big(h(u), h'(u)\big)$, we have

$$d_u(k) = min\big(h(u), h'(u)\big) - min\big(stab(u)\big).$$

(3) For $max\big(h(u), h'(u)\big) \leq k \leq max\big(stab(u)\big) - 1$, we have

$$d_u(k) = max\big(stab(u)\big) - k$$

(4) For all other k, we have $d_u(k) = 0$.

Corollary 5.3.2. *Consider a connected unlabeled configuration u on the line such that $stab(u)$ has a single hole $h(u)$. Suppose that for every site k, $f_u(k) \geq d_u(k)$.*

Then there exists a diamond of moves with lower vertex $h(u)_1$, left vertex $\big(min(stab(u)) + 1\big)_1$, and right vertex $\big(max(stab(u)) - 1\big)_1$, such that every move in the diamond satisfies the conditions of Lemma 5.3.1.

Proof. For every site k and every move $j \leq d_u(k)$, move k_j satisfies the conditions of Lemma 5.3.1. The moves k_j where $min(stab(u)) + 1 \leq k \leq max(stab(u)) - 1$ and

$j \leq d_u(k)$ form a rectangular diamond where one side consists of move k_1 for each k from $min(stab(u)) + 1$ to $h(u)$, and one side consists of move k_1 for each k from $h(u)$ to $max(stab(u)) - 1$. □

Lemma 5.3.1 shows the existence of a grid structure at the bottom of the move poset, and it shows precisely which firing moves are included in that grid structure. Corollary 5.3.2 shows when exactly that grid extends into a full diamond, based on whether enough firing moves occur at each site. In particular, Corollary 5.3.2 guarantees the existence of a diamond-shaped grid for every unstabilization, with the exception of two cases:

(1) $2m + 1$ chips at the origin.

(2) m chips at the origin, and m chips at site 1.

When the number of chips n is even, the unstabilization with the biggest diamond is the case of $2m$ chips at the origin, when the diamond is an $m \times m$ square. From there, each chip moved from site 0 to site 1 adds one row to the diamond and removes one column (so that the bottom left edge of the diamond is shorter and the bottom right edge is longer) until there are m chips at the origin and m chips at site 1, in which case there is no grid structure at all.

The opposite occurs in the case when n is odd. In the odd case, there is no grid structure in the case with $2m + 1$ chips at the origin. Moving 1 chip to site 1 produces a 1 by $2m + 1$ "diamond," and continuing to add chips to site 1 will grow the diamond, until it becomes an m by $m + 1$ diamond when there are $m + 1$ or m chips at the origin. Continuing to move chips to site 1 past that point results in the diamond shrinking again as it becomes closer to a line and then vanishes again.

One explanation for the difference between even and odd cases is due to what happens when a single chip is added. The cases without diamonds correspond to unlabeled configurations u where $stab(u)$ has no hole. Adding a chip to a configuration with no hole results in many constrained firing moves. On the other hand, the largest diamonds

correspond to configurations in which the hole is as close as possible to site 0. Adding another chip to site 0 does nothing when there is already a hole there.

The next step in proving more general labeled chip-firing results is to prove a more general form of our bounds on chip positions throughout the process. This is accomplished through the following lemma, analogous to Lemma 3.2.8:

Lemma 5.3.3. *Consider a weakly sorted connected labeled chip configuration ℓ with n chips and underlying unlabeled chip configuration u. Then the position of the k^{th} smallest chip must always be at least $\lceil mean(u) \rceil + k - n$, and the position of the k^{th} largest chip must always be at most $\lfloor mean(u) \rfloor + n - k$.*

Proof. We first prove that the starting position of the k^{th} largest chip must be at most $\lfloor mean(u) \rfloor + n - k$. Suppose that the k^{th} largest chip begins at position greater than $\lfloor mean(u) \rfloor + n - k$. As the chips are weakly sorted, this means that there are at least k chips at positions greater than $\lfloor mean(u) \rfloor + n - k$. If we consider a configuration with k chips at position $\lfloor mean(u) \rfloor + n - k + 1$, then the stabiliation of this configuration will have a chip at position at least $\lfloor mean(u) \rfloor + n - k + 1 + \lfloor \frac{k}{2} \rfloor$. By Corollary 9 from [HMP], any configuration with k chips at position greater than $\lfloor mean(u) \rfloor + n - k$ must also have a stabilization with a chip at position at least $\lfloor mean(u) \rfloor + n - k + 1 + \lfloor \frac{k}{2} \rfloor$, which is greater than the maximum possible chip position for a connected configuration with mean chip position $mean(u)$. This is a contradiction, so the k^{th} largest chip cannot be at a position greater than $\lfloor mean(u) \rfloor + n - k$ in configuration u. The lower bound follows by symmetry.

We now prove that the bounds continue to hold throughout the labeled chip-firing process. We prove the upper bound by strong induction on $n - k$. The initial position of the n^{th} largest chip is at most $\lfloor mean(u) \rfloor + n - k$. Furthermore, as this is the smallest chip, it cannot be fired to the right at any time throughout the process, so the n^{th} largest chip must never be at position greater than $\lfloor mean(u) \rfloor + n - k$. Now, suppose that the k'^{th} smallest chip is at position at most $\lfloor mean(u) \rfloor + n - k - 1$ for all $k' > k$. The initial position of the k^{th} smallest chip cannot exceed $\lfloor mean(u) \rfloor + n - k - 1$, and as no smaller chip can reach position $\lfloor mean(u) \rfloor + n - k$, the k^{th} smallest chip cannot be fired to the

right from position $\lfloor mean(u) \rfloor + n - k$. As a result, the position of the k^{th} smallest chip cannot exceed $\lfloor mean(u) \rfloor + n - k$ throughout the process. The result follows by strong induction, and the lower bound follows by symmetry. □

Note that while these bounds (analogous to the ones from [31] still apply, the stronger ones from [27] are too strong in this setting (In Example 5.2.12, chip 5 violates the upper bound, while chip -5 violates the lower bound). Our bounds are still probably not the strongest possible, but they are sufficient for our purposes.

Lemmas 5.3.1 and 5.3.3 allow us to provide a more general form of Lemma 3.2.9. If the number of firing moves satisfy the bounds of Lemma 5.3.1, and chip positions meet certain conditions, then sorting is guaranteed at the end of the process.

Lemma 5.3.4. *Consider a labeled configuration ℓ with connected underlying unlabeled configuration u on the line such that $stab(u)$ has a single hole $h(u)$, and such that for each site k, $f_u(k) \geq d_u(k)$. Let $k_0 = min(stab(u)) + max(stab(u)) - h(u)$. Suppose that for each $1 \leq i \leq k_0 - min(stab(u))$, the i^{th} largest chip is guaranteed to be at position greater than or equal to $k_0 - i + 1$ prior to move $(k_0 - i + 1)_{d_u(k_0 - i + 1)}$, and for each $1 \leq i \leq max(stab(u)) - k_0$, the i^{th} smallest chip is guaranteed to be at position less than or equal to $k_0 + i - 1$ prior to move $(k_0 + i - 1)_{d_u(k_0 + i - 1)}$. Then ℓ sorts.*

Proof. The proof proceeds like the proof of Lemma 3.2.9. The largest chip must be at or to the right of position k_0 prior to move $(k_0)_{d_u(k_0)}$. By Lemma 5.3.1, move $(k_0)_{d_u(k_0)}$ must occur with 2 chips present, so the largest chip must be at or to the right of position $k_0 + 1$ prior to the next move at site $k + 1$. Due to Corollary 5.3.2, move $(k_0)_{d_u(k_0)}$ is at the top of a diamond of 2-chip firing moves, so by the same logic as in Lemma 3.2.9, the largest chip must be at or to the right of the position of each move on the top right edge of the diamond before it occurs, so it is to the right of each corresponding move after it occurs. In particular, the largest chip must be to the right of position $max(stab(u)) - 1$ after move $\left(max(stab(u)) - 1\right)_1$ occurs, so it must end at position $max(stab(u))$.

Again using the same logic as in Lemma 3.2.9, the i^{th} largest chip for each $i \leq k_0 - min(stab(u))$ must appear at or to the right of the i^{th} line from the top right of

the diamond and end in the i^{th} largest position in $stab(u)$. A similar argument shows the same result for the $max(stab(u)) - k_0$ smallest chips, so the chips end in sorted order. □

Using Lemma 5.3.4, we can prove sorting for several initial configurations at once.

We need to make sure that the numbers of firing moves at each site satisfy the bounds of Corollary 5.3.2.

Lemma 5.3.5. *In the chip-firing process beginning with $2m-1$ chips at the origin and 1 chip at site 1, and running to completion, the number of firing moves at site k for $1 \leq |k| \leq m-1$ is equal to $\binom{m-|k|+1}{2}$. The number of firing moves at site 0 is equal to $\binom{m+1}{2} - 1$. No other sites fire throughout the process.*

Proof. By Lemma 5.2.6, $stab(u)$ has at most one hole. By 5.2.7, there can only be one stable configuration with at most one hole with the same mean chip position as u. We see that the configuration with a chip at every site from $-m$ to m except for -1 has one hole and the same mean chip position as u, so that configuration must be $stab(u)$.

We prove the result for $k \geq 1$, and $k \leq -1$ follows by symmetry. Since the final configuration has no chips past position m, there must be no firing moves at position m or greater throughout the process. For positions $1 \leq k \leq m-1$, the number of chips sent right by site k must be equal to the number of chips sent to the left by site $k+1$, plus the number of chips remaining at sites greater than k. Let $f_u(k)$ be the number of firing moves at site k. This gives the recurrence

$$f_u(k) = f_u(k+1) + m - k \text{ for } 1 \leq k \leq m-1$$

A similar argument gives a recurrence for $k = 0$.

$$f_u(0) = \frac{f_u(-1) + f_u(1)}{2} + \frac{(m-1) + (m-1)}{2}$$

Since there are $m-1$ chips to the left of 0 in the final configuration, and m chips to the right, 1 of chips is already to the right of 0 when the process begins.

Solving the recurrence gives a unique solution in which site k fires $\binom{m-|k|+1}{2}$ times for each $|k| > 0$, and site 0 fires $\binom{m+1}{2} - 1$ times.

□

Lemma 5.3.6. *In the chip-firing process beginning with $m+1$ chips at the origin and m chips at site 1, and running to completion, the number of firing moves at site k for $1 \leq k \leq m$ is equal to $\binom{m-k+2}{2}$. The number of firing moves at site k for $-m+1 \leq k \leq 0$ is equal to $\binom{m+k+1}{2}$. No other sites fire throughout the process.*

Proof. By Lemma 5.2.6, $stab(u)$ has at most one hole. By 5.2.7, there can only be one stable configuration with at most one hole and the same mean chip position as u. We see that the configuration with a chip at every site from $-m$ to $m+1$ except for 1 has one hole and the same mean chip position as u, so that configuration must be $stab(u)$.

We prove the result for $k \geq 1$, and $k \leq 0$ follows by symmetry. Since the final configuration has no chips past position $m+1$, there must be no firing moves at position $m+1$ or greater throughout the process. For positions $1 \leq k \leq m$, the number of chips sent right by site k must be equal to the number of chips sent to the left by site $k+1$, plus the number of chips remaining at sites greater than k. Let $f_u(k)$ be the number of firing moves at site k. This gives the recurrence

$$f_u(k) = f_u(k+1) + m + 1 - k \text{ for } 1 \leq k \leq m$$

Solving the recurrence gives a unique solution in which site k fires $\binom{m-k+2}{2}$ times for each $k \geq 1$. By the same argument, we get that for each $k \leq 0$, site k fires $\binom{m+k+1}{2}$ times.

□

This allows us to prove sorting for every major Type A class conjectured to sort. In fact, we can prove an even stronger result: that for any connected configuration u such that $unstab(u)$ is one of our three major Type A classes, and such that the number of firing moves at each site allows for a complete diamond, $ws(u)$ must sort.

Theorem 5.3.7. *Consider a connected unlabeled chip configuration u such that $unstab(u)$ takes on one of the following three forms:*

(1) $2m$ *chips at a single site*

(2) $2m-1$ *chips at one site and 1 chip at an adjacent site*

(3) $m+1$ *chips at one site and* m *chips at an adjacent site*

Suppose also that for all sites k, $f_u(k) \geq d_u(k)$. *Then* $ws(u)$ *sorts.*

Proof. By Lemma 5.3.3, the following must hold:

(1) In case 1, the i^{th} largest chip cannot have position less than $1-i$, and the i^{th} smallest chip cannot have position greater than $i-1$.

(2) In case 2, the i^{th} largest chip cannot have position less than $2-i$, and the i^{th} smallest chip cannot have position greater than $i-1$.

(3) In case 3, the i^{th} largest chip cannot have position less than $2-i$, and the i^{th} smallest chip cannot have position greater than $i-1$.

Thus, by Lemma 5.3.4, all configurations in any of the 3 cases sort. □

In particular, this allows us to prove that the Type A fundamental weights that are conjectured to sort actually do.

Theorem 5.3.8 ([27], Theorem 11 and [21], Problem 7.14, Parts (1) and (2))**.** *Consider a connected unlabeled chip configuration* u *that takes on one of the following three forms:*

(1) $2m$ *chips at a single site*

(2) $2m-1$ *chips at one site and 1 chip at an adjacent site*

(3) $m+1$ *chips at one site and* m *chips at an adjacent site*

Then $ws(u)$ *sorts.*

Proof. By Theorem 3.2.2 and Lemmas 5.3.5 and 5.3.6, $f_u(k) \geq d_u(k)$ for all k in each case. Thus, by Theorem 5.3.7, all 3 cases sort. □

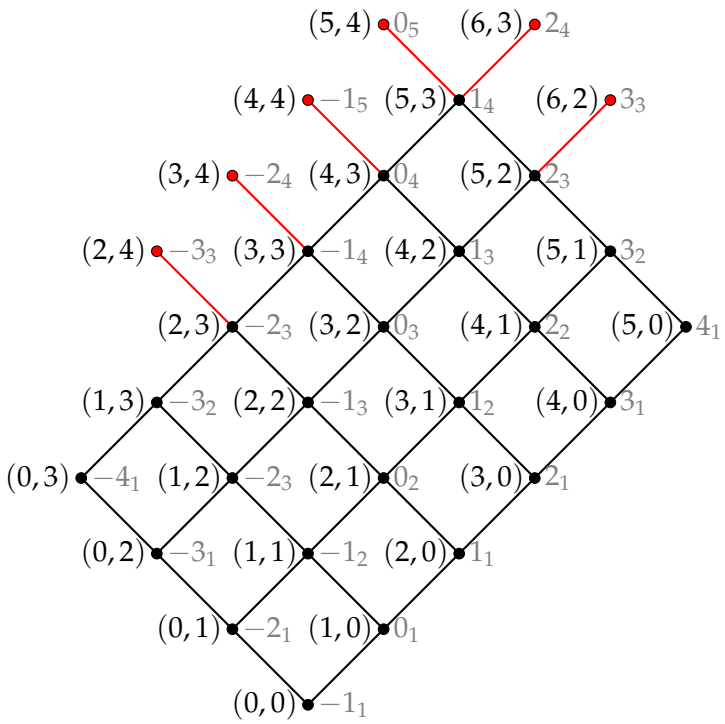


FIGURE 5.5: Bottom of the firing move poset, when u consists of 9 chips at the origin and 1 chip at site 1.

Theorem 5.3.7 can probably be strengthened (there are many weakly sorted configurations in the three equivalence classes mentioned that do not include all diamond moves and still sort), but this is a nice result showing that a mere $O(n^2)$ firing moves, out of a maximum possible $O(n^3)$, are enough to guarantee sorting if all are present. This allows our sorting proofs to extend from three special cases of configurations to some much broader classes of configurations.

We show the diamonds for cases 2 and 3 in Figures 5.5 and 5.6.

Example 5.3.9. Let u consist of 9 chips at the origin, and 1 chip at site 1. Figure 5.5 shows a bottom portion of the Haase diagram of the move poset.

Example 5.3.10. Let u consist of 6 chips at the origin, and 5 chips at site 1. Figure 5.6 shows a bottom portion of the Haase diagram of the move poset.

We can also use our methods to prove a more general version of our sorting result for the case with $m + 1$ chips at the origin and m chips at site 1. It turns out that while

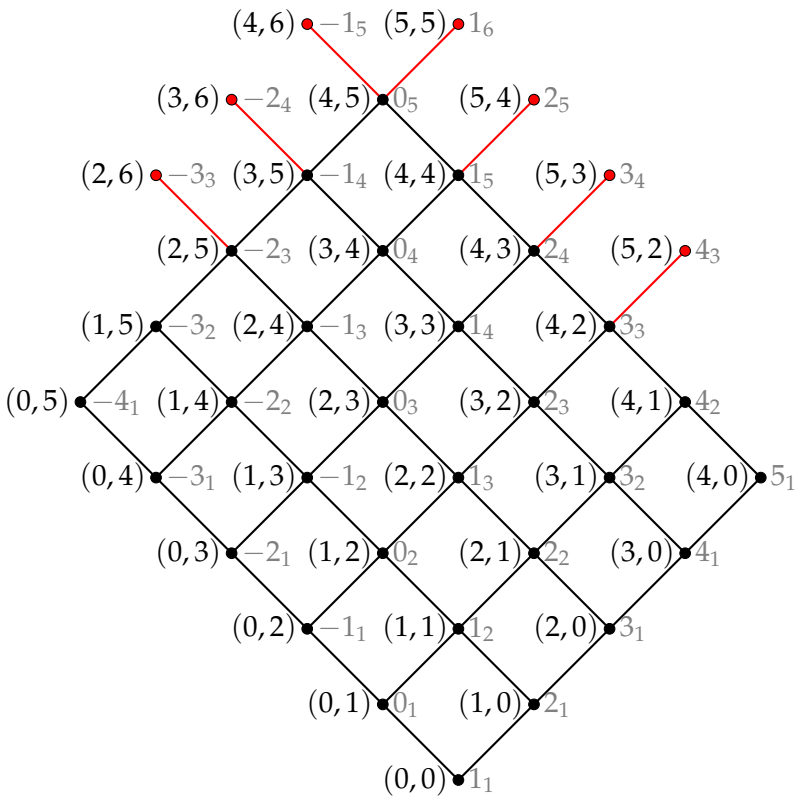


FIGURE 5.6: Bottom of the firing move poset, when u consists of 6 chips at the origin and 5 chips at site 1.

we assume that the chips must begin in weakly sorted order, the proof does not depend on the initial positions of most of the chips. This gives us the following result, in which the chips do not have to begin the process in sorted order.

Corollary 5.3.11. *Consider a labeled chip-firing process with $2m+1$ chips labeled from $-m$ to 0, and from 2 to $m+1$. The process begins with $m+1$ chips at the origin and m chips at site 1. Then this process is guaranteed to sort if and only if chip $-m$ begins at 0.*

Proof. First, suppose that the initial position of chip $-m$ is 0. Then the proof of Lemma 5.3.3 does not change. Chip $-m$ still cannot reach a site greater than 0, and we can show by induction that each subsequent chip cannot get more than 1 unit to the right of the rightmost possible position of the previous chip. Similarly, chip $m+1$ cannot get to the left of position 0, and all other lower bounds on chip positions remain the same. All other parts of the proof remain the same, so this is guaranteed to sort.

Suppose instead that the initial position of chip $-m$ is 1. We can construct a sequence of firing moves that result in chip $-m$ not ending up at site $-m$.

First, leave chip $-m$ fixed at site 1, and perform all available firing moves not involving chip $-m$ until every site other than site 1 has 1 chip or fewer. This is equivalent to firing from a configuration with $m+1$ chips at the origin and $m-1$ chips at site 1, so the resulting configuration has 1 chip at every site from $-m$ to m, except for site $-m+1$, which has 0 chips, and site 1, which has chip $-m$ and 1 other chip. Now we perform any available firing moves in any order until completion. The chip that ends the process at $-m$ is already at site $-m$ before any firing moves involving chip $-m$ occur, so chip $-m$ does not end up at site $-m$ after this firing sequence, so this initial configuration does not sort. □

Corollary 5.3.11 is a notable result, as it is (as far as we know) the only sorting result proven for chips that do not begin the process in weakly sorted order. In fact, sorting is guaranteed in $\frac{m+1}{2m+1}$ of all possible assignments of chip labels for processes beginning with $m+1$ chips at the origin and m chips at site 1. Sorting depends entirely on the location of the smallest chip, as the other chip labels can be assigned in any way, as long

as the smallest chip is at site 0, and the configuration still sorts. This provides a partial answer to question (2) presented at the beginning of this chapter.

Example 5.3.12. Lemma 5.3.4 can be used to prove sorting in cases where the "diamond" would otherwise not be big enough to guarantee sorting. We show a labeled configuration ℓ, with underlying unlabeled configuration u. u is not firing equivalent to one of the three classes of sorting configuration from Theorem 5.3.7, but ℓ still sorts.

				2						
				1						
				0						
				-1						
				-3				5		
				-4				4		
—	—	—	—	$\underline{-5}$	—	—	—	$\underline{3}$	—	—
-5	-4	-3	-2	-1	0	1	2	3	4	5

The unique final configuration is shown below:

$\underline{-5}$	$\underline{-4}$	$\underline{-3}$	—	$\underline{-1}$	$\underline{0}$	$\underline{1}$	$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$
-5	-4	-3	-2	-1	0	1	2	3	4	5

Figure 5.7 shows how the labeled chips' positions are restricted throughout the diamond. While the firing process from $ws\big(unstab(u)\big)$ would not be able to provide the necessary position bounds to guarantee sorting, $ws(u)$ does provide the necessary bounds, as the chips begin the process closer to their sorted positions.

Remark 5.3.13. Chip-firing on a path graph, and labeled chip-firing by extension, are usually considered to take place on an infinite graph. However, both can equivalently be seen as taking place on a finite graph, as long as the graph is large enough so that chips do not reach the edge over the course of firing. In particular, the equivalence classes of configurations considered here are closely related to the corresponding firing equivalence classes on certain cycle graphs.

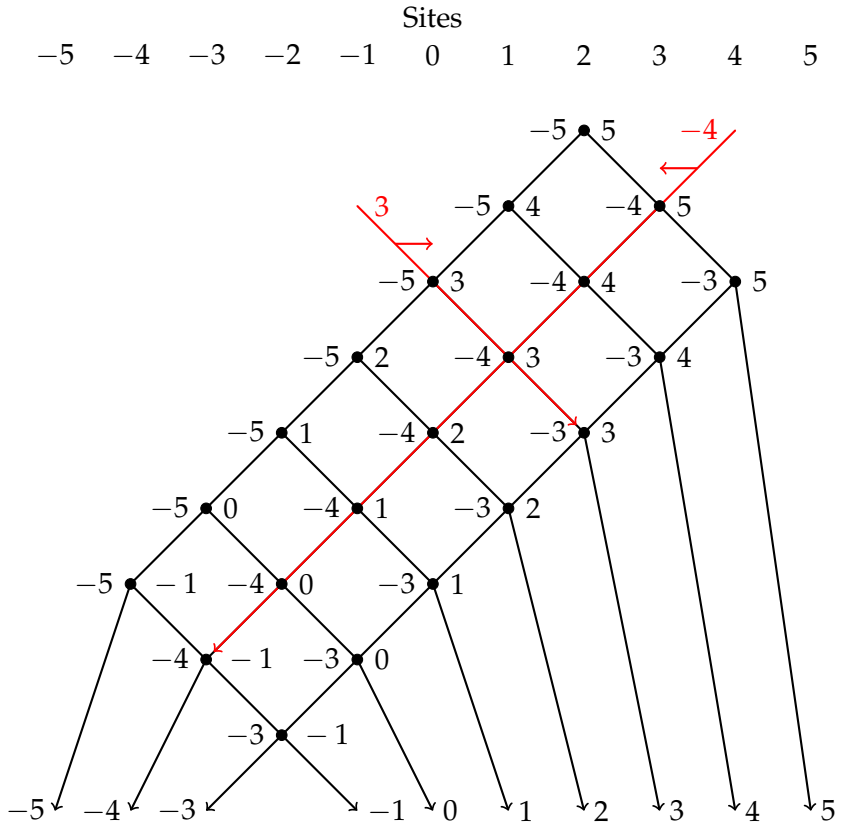


FIGURE 5.7: Bottom of the firing move poset for Example 5.3.12. For each firing move shown, the chip written to the left of the vertex must be at or to the left of that site when the firing move occurs, while the chip written to the right must be at or to the right of that site. The arrows at the bottom show how the last firing moves at each site send each of the chips to their final, sorted positions. As examples, the red arrows shown are the respective left and right bounds for the positions of chips 3 and −4 when given firing moves occur.

Consider a cycle graph with $n+2$ nodes. This graph has $n+2$ spanning trees. Thus, if we designate a single vertex as a sink, the resulting graph has $n+2$ critical configurations. The critical configurations consist of:

- The configuration with a single chip at each non-sink vertex.
- For each non-sink vertex v, a configuration with a single chip at each non-sink vertex except for v, which has no chips.

Thus, if we remove the sink and its incident edges to produce a path graph with $n+1$ vertices, we obtain:

- One configuration with a chip at each of the $n+1$ sites
- Two configurations with chips at n consecutive sites, corresponding to the same configuration on the infinite path graph if we are not concerned with shifting the entire configuration.
- $n-1$ configurations consisting of n chips spread out over $n-1$ sites, with a single hole in any possible location.

Other than the configuration with a chip at each of the $n+1$ sites, these correspond precisely to the stable configurations reached from connected configurations with n chips.

5.4 Classification of Sorting Configurations

We now have enough information to be able to classify which weakly sorted configurations on the line sort, and which ones do not. Because a configuration sorts if and only if each connected component sorts, we focus on connected configurations in this section. Because we need some negative results in addition to positive ones, we begin with a partial converse to the Diamond Lemma 5.3.1.

First, we exclude configurations whose stabilizations do not contain a hole.

Lemma 5.4.1. *Consider a connected unlabeled configuration u on the line with $n > 1$ chips such that $stab(u)$ does not contain a hole. Then $ws(u)$ does not sort.*

Proof. If $n > 1$, then u requires at least one firing move to stabilize. If $stab(u)$ does not contain a hole, then the last firing move in the stabilization of u must leave one chip at the firing site after firing. This implies that the last firing move of the process occurs with 3 chips present. There are 3 possible ways to perform a labeled firing move involving 3 chips, 2 of which produce an inversion. Thus, there must exist an instance of the labeled firing process beginning with $ws(u)$ that does not sort. □

Now, we show that every connected unlabeled configuration u has at least some grid structure at the bottom of the move poset, as long as $stab(u)$ contains a hole. In particular, there is a V-shape of 2-chip firing moves, where the bottom of the V corresponds to the last move at $h(u)$, and the two ends of the V correspond to the last moves at $min(stab(u)) + 1$ and $max(stab(u)) - 1$.

Lemma 5.4.2. *Consider a connected unlabeled configuration u such that $stab(u)$ has a hole at $h(u)$. Then every site from $min(stab(u)) + 1$ to $max(stab(u)) - 1$ must fire at least once. Furthermore for every k such that $min(stab(u)) < k < h(u)$, move k_1 occurs before move $(k+1)_1$, and for every k such that $h(u) < k < max(stab(u))$, move k_1 occurs before move $(k-1)_1$.*

Proof. Proceed by contradiction. Suppose that site k does not fire for some $min(stab(u)) + 1 \leq k \leq max(stab(u)) - 1$. Then either site $k+1$ also does not fire, or site $k-1$ also does not fire. If both sites were to fire, then site k would have 2 chips and would eventually fire. Suppose without loss of generality that site $k-1$ does not fire. Since $min(stab(u)) + 1 \leq k \leq max(stab(u)) - 1$, there must be some chip that ends at position less than k, and some chip that ends at position greater than or equal to k. However, since sites $k-1$ and k do not fire, all chips that begin at positions at most $k-1$ must remain at positions at most $k-1$ throughout the firing process, and all chips that begin at positions at least k must remain at positions at least k throughout the firing process. Thus, the configuration is not connected, which is a contradiction.

Because every site from $min(stab(u)) + 1$ to $max(stab(u)) - 1$ fires at least once, the relative orders of the firing moves follow by Lemma 5.3.1. □

From here, we use some of the notation from Chapter 4 that relates the move poset to the poset of join-irreducibles of the configuration poset. Let $J(k_j)$ be the join-irreducible configuration from which k_j is the only firing move available. We now prove a partial converse to the general Diamond Lemma: that any firing move outside of the grid at the bottom of the move poset has the potential to take place with at least 3 chips present.

Lemma 5.4.3. *Consider a connected unlabeled configuration u on the line such that $stab(u)$ has a single hole $h(u)$. Consider a firing move k_j such that at least one of the following does not hold:*

- *For each site i such that $min(h(u), k) \leq i \leq max(h(u), k)$, $f_u(i) \geq j$.*
- *For each site i such that $max(h(u), k) < i < max(h(u), k) + j$, $f_u(i) \geq j + max(h(u), k) - i$.*
- *For each site i such that $min(h(u), k) - j < i < min(h(u), k)$, $f_u(i) \geq j + i - max(h(u), k)$.*

Then there exists an instance of the chip-firing process beginning with u such that firing move k_j takes place with at least 3 chips present at site k.

Proof. We proceed by contradiction. Suppose that k_j must take place with 2 chips present. Then the join-irreducible $J(k_j)$ must have 2 chips at site k_j and at most 1 chip at every other site.

Starting with $J(k_j)$, fire site k, and then successively fire at sites $k-1, k-2, \ldots$ until no other firing moves are available. We consider 2 cases: either site $k+1$ has one chip after these moves are performed, or site $k+1$ has 2 chips.

In case 1, the process must have terminated after the specified firing moves are performed. By Lemma 5.4.2, the last firing move must be $h(u)_1$. Thus, k must be greater than or equal to $h(u)$, and there must be at least one firing move at every site from $h(u)$

to k, so the conditions on f_u from the statement of the lemma are all met. This is a contradiction, so case 1 cannot occur if the conditions of the lemma are not met.

Instead, suppose that site $k+1$ has 2 chips after the sequence of firing moves is performed. No other site can have 2 chips after this sequence of firing moves, so the resulting configuration is a join-irreducible corresponding to a firing move at site $k+1$.

In this case, we again start with $J(k_j)$, but instead fire site k and then successively fire at sites $k+1, k+2, \ldots$ until no more moves are available. As in the case above, this either leaves 1 chip at site $k-1$, which produces a contradiction, or it leaves 2 chips at site $k-1$.

Suppose that both sequences of firing moves leave 2 chips at sites $k+1$ and $k-1$ repectively. Then we consider another sequence of moves starting with $J(k_j)$. First fire site k, then successively fire at sites $k-1, k-2, \ldots$ until no moves are available. Then successively fire at sites $k+1, k+2, \ldots$ until no moves are available. The resulting configuration has 2 chips at site k and no more than 1 chip at any other site. Since site k has only fired once in this sequence, the resulting configuration must be $J(k_{j-1})$.

Thus, we have shown that k_j must occur before $(k+1)_{j'}$ and $(k-1)_{j''}$ for some j' and j'', and those two moves in turn take place before move k_{j-1}. Repeating this argument on moves $(k+1)_{j'}$ and $(k-1)_{j''}$ will continue until the last move at some site k' is reached, which leads to a contradiction. Thus, by contradiction, any firing move that fails the conditions of Lemma 5.3.1 must occur with at least 3 chips in some instance of the process.

□

This provides a nice, bidirectional result on unlabeled chip-firing. Any move in the grid takes place with 2 chips and thus must behave predictably, but any other move in the process has the potential to take place with 3 or more chips, thus allowing it to leave a chip in place.

Like most of our sorting results, this general sorting proof requires a diamond grid and some bounds on chip positions. Lemmas 5.3.1 and 5.4.3 provide us with most of what we need from the diamonds, so we proceed to figuring out how far each chip can

go to the left or right. Since we are trying to prove either sorting or non-sorting for every case, we need the strictest bounds possible for each chip.

These were obtained by Hopkins, McConville, and Propp in [27]. We adopt some of their notation here. Given labeled configuration ℓ, we define $\ell|_{\geq k}$ as the restriction of ℓ to chips with labels greater than or equal to k. Similarly define $\ell|_{\leq k}$. Given labeled configurations ℓ_1 and ℓ_2, we say that ℓ_2 is **reachable** from ℓ_1 if it is possible to get from ℓ_1 to ℓ_2 through some sequence of firing moves. Given unlabeled configurations u_1 and u_2, we say that u_2 is **right reachable** from u_1 if it is possible to get from u_1 to u_2 through some sequence of moves of the following two forms:

- Perform a legal chip-firing move.
- Send one chip one unit to the right.

Given a labeled configuration ℓ, we define $u(\ell)$ as the underlying unlabeled configuration of ℓ.

We have the following results from [27].

Lemma 5.4.4 ([27], Proposition 7)**.** *Consider two labeled configurations ℓ_1 and ℓ_2. If ℓ_2 is reachable from ℓ_1, then $u(\ell_2|_{\geq k})$ is right reachable from $u(\ell_1|_{\geq k})$*

Lemma 5.4.5 ([27], Corollary 9)**.** *Consider two unlabeled configurations c_1 and c_2. If c_2 is right reachable from c_1, then $min(stab(c_1)) \leq min(c_2)$.*

Combining these two results gives a generalization of Lemma 10 from [27]. Define $min_{\geq k}(\ell)$ as $min\Big(stab\big(u(\ell|_{\geq k})\big)\Big)$ and $max_{\leq k}(\ell)$ as $max\Big(stab\big(u(\ell|_{\leq k})\big)\Big)$.

Lemma 5.4.6. *Consider a labeled chip-firing process with weakly sorted initial configuration ℓ. Then chip k must always remain between positions $min_{\geq k}(\ell)$ and $max_{\leq k}(\ell)$, inclusive. Furthermore, there exists an instance of the process in which the lower bound is attained, and there exists an instance of the process in which the upper bound is attained.*

Proof. We prove the results for the lower bounds. For any configuration ℓ' reachable from ℓ, $u(\ell'|_{\geq k})$ is right reachable from $u(\ell|_{\geq k})$ by Lemma 5.4.4. By Lemma 5.4.5,

$min_{\geq k}(\ell) \leq min(\ell'|_{\geq k})$. In particular, $min_{\geq k}(\ell)$ is less than or equal to the position of chip k in configuration ℓ'.

Now, we can show that the bound is attained. Start with configuration ℓ and ignore all chips less than k. Perform all available firing moves with chips greater than or equal to k, subject to the condition that any time a firing move is performed at the site containing chip k, chip k must be fired. Since we are ignoring all chips less than k, all firing moves involving k send k to the left, and because we require k to be fired every time that a site containing k is fired, there can never be any chips with values greater than k to the left of k at any time throughout the process. Once all moves with chips greater than or equal to k are performed, we reach a configuration in which chip k is at position $min_{\geq k}(\ell)$, so the bound is attainable.

The upper bound follows by symmetry. □

Lemma 5.4.6 has a nice, intuitive explanation. If you want to know how far to the left chip k can go, then you can answer the question by removing all chips smaller than k. The lower bound for the position of chip k is the same before and after all of the smaller chips are removed, and the bound is much easier to compute when k is the smallest chip.

This brings us to a position in which we are ready to prove the main result of this chapter. Given labeled configuration ℓ with corresponding unlabeled configuration u, and given a value $k > h(u)$, define k' to be $min_{\geq k}(\ell) + h(u) - k$. Given a value $k < h(u)$, define k' to be $max_{\geq k}(\ell) + h(u) - k$.

Theorem 5.4.7. *Consider a connected unlabeled configuration u such that $stab(u)$ has a single hole at $h(u)$. Obtain labeled configuration ℓ by labeling the chips of u from $min(stab(u))$ to $h(u) - 1$ and from $h(u) + 1$ to $max(stab(u))$ in weakly sorted order. Then the process must terminate with the chips in sorted order if and only if the following conditions hold for each chip k:*

- *If $k > h(u)$, then*

(1) *For each site i with $k' + 1 \leq i \leq min\Big(min_{\geq k}(\ell), h(u)\Big)$, site i fires at least $i - k'$ times.*

(2) *For each site i with $min\Big(min_{\geq k}(\ell), h(u)\Big) \leq i \leq max\Big(min_{\geq k}(\ell), h(u)\Big)$, site i fires $k - max\Big(min_{\geq k}(\ell), h(u)\Big)$ times.*

(3) *For each site i with $max\Big(min_{\geq k}(\ell), h(u)\Big) \leq i \leq k - 1$, site i fires $k - i$ times.*

- *If $k < h(u)$, then*

(1) *For each site i with $k' + 1 \leq i \leq min\Big(max_{\leq k}(\ell), h(u)\Big)$, site i fires at least $i - k'$ times.*

(2) *For each site i with $min\Big(max_{\leq k}(\ell), h(u)\Big) \leq i \leq max\Big(max_{\leq k}(\ell), h(u)\Big)$, site i fires $k - max\Big(max_{\leq k}(\ell), h(u)\Big)$ times.*

(3) *For each site i with $max\Big(max_{\leq k}(\ell), h(u)\Big) \leq i \leq k - 1$, site i fires $k - i$ times.*

Proof. We first show that chip k ends at site k for each $k > h(u)$, if all of the conditions are met.

We induct downward on k starting with $k = max\big(stab(u)\big) - 1$. By Lemma 5.3.1, move $\big(min_{\geq k}(\ell)\big)_j$ for $j = k - max\Big(min_{\geq k}(\ell), h(u)\Big)$ must be part of the grid at the bottom of the poset and must take place with at most 2 chips present. Thus, chip k must remain on or to the right of the line connecting move $\big(min_{\geq k}(\ell)\big)_j$ for $j = k - max\Big(min_{\geq k}(\ell), h(u)\Big)$ and move $(k-1)_1$ in the diamond, as in the proof of Lemma 3.2.9. Thus, chip k is fired to the right by move $(k-1)_1$, and ends at site k.

Now, suppose that for some k, all chips greater than k remain at or to the right of the line connecting move $\big(min_{\geq k+1}(\ell)\big)_j$ for $j = k + 1 - max\Big(min_{\geq k+1}(\ell), h(u)\Big)$ and move $(k)_1$ in the grid. By Lemma 5.3.1, move $\big(min_{\geq k}(\ell)\big)_j$ for $j = k - max\Big(min_{\geq k}(\ell), h(u)\Big)$ must be part of the grid at the bottom of the poset and must take place with at most 2 chips present. Thus, since all chips greater than k are to the right of the line connecting move $\big(min_{\geq k}(\ell)\big)_j$ for $j = k - max\Big(min_{\geq k}(\ell), h(u)\Big)$ and move $(k-1)_1$ in the diamond, chip k must remain at or to the right of that line, being fired to the right at every move on that line on which it is involved, as in the proof of Lemma 3.2.9. Thus, chip k is fired to the right by move $(k-1)_1$, and ends at site k.

The result for $k < h(u)$ follows by symmetry.

Suppose instead that for some chip $k > h(u)$, the conditions of the lemma are not met. By Lemma 5.4.3, $\left(min_{\geq k}(\ell)\right)_j$ for $j = k - max\Big(min_{\geq k}(\ell), h(u)\Big)$ either does not exist, or is allowed to fire with at least 3 chips present. Consider the firing sequence used in the proof of Lemma 5.4.6, where chip k reaches position $min_{\geq k}(\ell)$.

Once chip k reaches position $min_{\geq k}(\ell)$, proceed to fire at all sites other than $min_{\geq k}(\ell)$ until only site $min_{\geq k}(\ell)$ can fire. The resulting configuration is $J(min_{\geq k}(\ell)_j)$ for some j. If $j \geq k - max\Big(min_{\geq k}(\ell), h(c)\Big)$, then there must be at least 3 chips present for this move by Lemma 5.4.3. Fire 2 chips other than k at site $min_{\geq k}(\ell)$. Repeat this process, firing at all other sites until a join-irreducible is reached, and then firing 2 chips other than k, until either the process terminates, or until a join-irreducible is reached with 2 chips at site $min_{\geq k}(\ell)$.

If the process has terminated, then either the last move at site $min_{\geq k}(\ell)$ occurred with 3 chips present, or site $min_{\geq k}(\ell)$ never fired after chip k arrived. If the last move at site $min_{\geq k}(\ell)$ occurs with 3 chips present, then the configuration cannot be connected by Lemma 5.4.2. If site $min_{\geq k}(\ell)$ never fires after chip k arrives, then either the last move involving chip k fired it to the left, leaving it at position less than $h(u)$, or chip k is never fired at all throughout the process. If chip k is never fired at all, then every chip to every share a site with k must have a smaller label. If any smaller chips ever reach the same site as chip k, then the last firing move involving those chips sends a chip smaller than k to the right, resulting in an unsorted configuration. If no chip ever reaches the same site as chip k, then the configuration is not connected. In any case, we either reach an unsorted configuration or a contradiction.

Suppose instead that a join-irreducible is reached with 2 chips at site $min_{\geq k}(\ell)$. The first such join-irreducible is $J(min_{\geq k}(\ell)_j)$ for some $j < k - max\Big(min_{\geq k}(\ell), h(u)\Big)$. From there, chip k must remain on or to the left of the line connecting move $\left(min_{\geq k}(\ell)\right)_j$ for $j < k - max\Big(min_{\geq k}(\ell), h(u)\Big)$ and move $(k-2)_1$ in the diamond, so chip k's final position must be at most $k-1$, and the final configuration is not sorted.

The case for $k < h(u)$ follows by symmetry.

□

This proves the classification theorem for a connected, weakly sorted configuration. We extend this result to any weakly sorted configuration.

Theorem 5.4.8. *Consider an unlabeled configuration u. Then $ws(u)$ sorts if and only if each connected component of u satisfies the conditions of Theorem 5.4.7, or consists of a single chip.*

Proof. By Lemma 5.2.1, $ws(u)$ sorts if and only if $ws(u')$ sorts for each connected component u'. Given a connected component u', if $stab(u')$ does not contain a hole, then $ws(u')$ sorts if and only if u' consists of 1 chip. If $stab(u')$ does contain a hole, then $ws(u')$ sorts if and only if it satisfies the conditions of 5.4.7. □

An example of a sorting configuration and a non-sorting configuration that produce the same diamond in the move poset are shown in Figures 5.8 and 5.9.

Remark 5.4.9. This general classification may seem a bit complicated, as it requires computations of minimum and maximum chip positions, as well as determining the exact nature of the grid at the bottom of the move poset. However, given a weakly sorted configuration, it is actually possible to determine fairly quickly whether it sorts.

Given a labeled configuration ℓ with unlabeled configuration u, computing $h(u)$ can be done very quickly. By Lemmas 5.2.6 and 5.2.7, $stab(u)$ it the unique stable configuration with at most one hole and the same mean chip-configuration as u, so the time it takes to compute is linear in the number of chips (and much faster if the chips are initially in a small number of piles).

Finding $min_{\geq k}(\ell)$ and $max_{\leq k}(\ell)$ can also be done efficiently for all k. When k is the value of the largest chip, $min_{\geq k}(\ell)$ is simply the value of the largest chip. Once we have $stab\Big(u(\ell|_{\geq k})\Big)$, we can calculate $stab\Big(u(\ell|_{\geq k-1})\Big)$ by simply adding a chip at the initial position of chip $k-1$ and letting it stabilize. This stabilization can be computed efficiently using Proposition 2 from [27], which quickly computes a final configuration when a single chip is added to a stable configuration and left to stabilize. While finding a data structure to allow these stabilizations to be computed in constant time may be

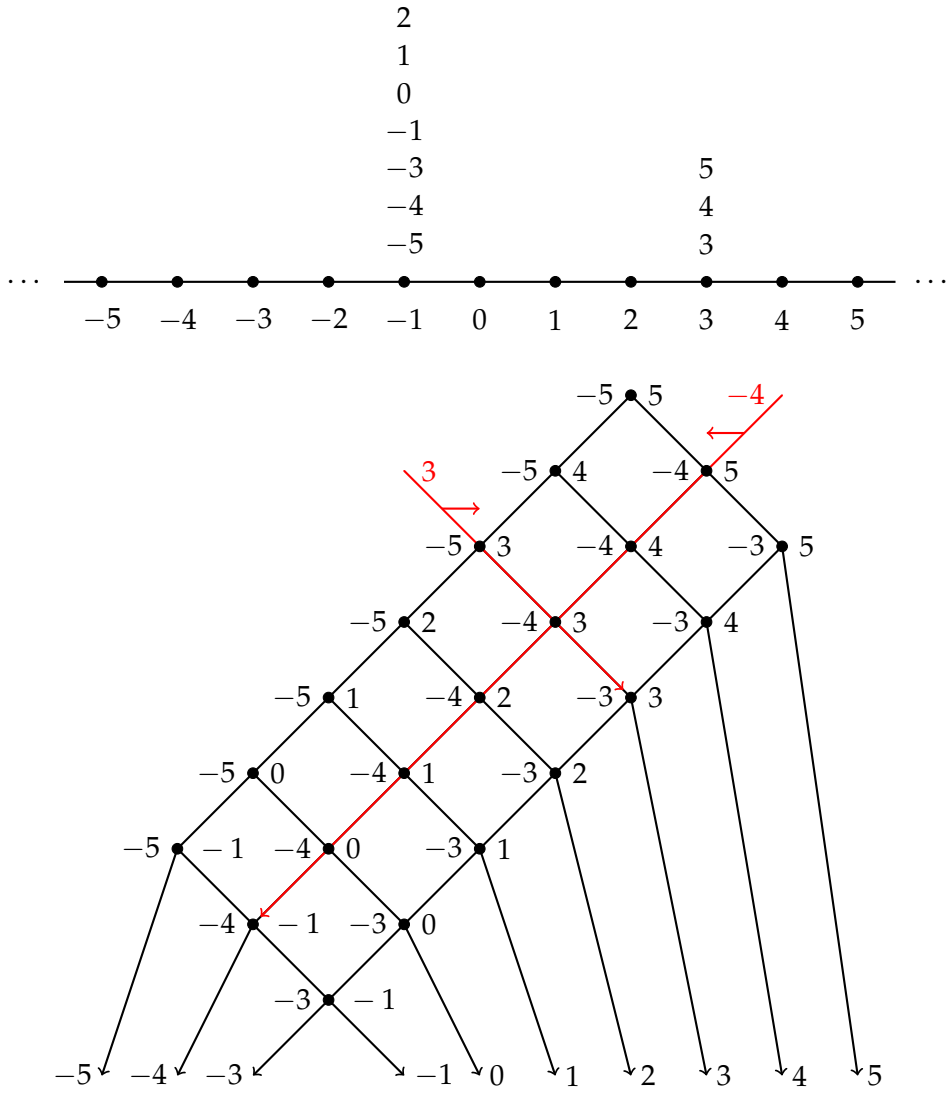


FIGURE 5.8: In the diamond from Figure 5.7, the chips begin the process close enough to their final positions that they are all sorted at the end of the process.

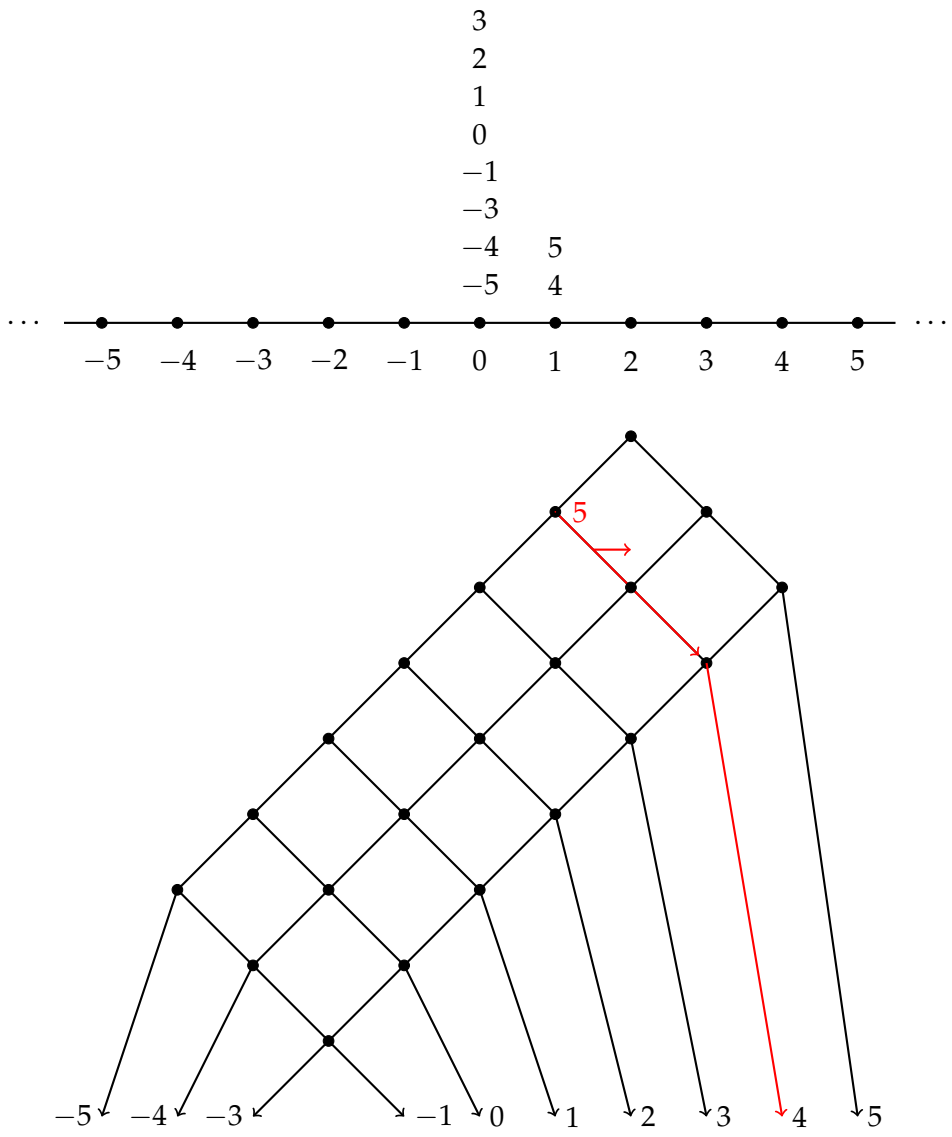


FIGURE 5.9: The configuration above produces the same diamond at the bottom of the move poset as the configuration in Figure 5.8. However, the chips may not end in sorted order. If chip 5 is not fired before move -1_3, then it ends at site 4, while a chip with a smaller label ends at site 5.

difficult, computing them in linear time is not a problem, so all of the $min_{\geq k}(\ell)$ and $max_{\leq k}(\ell)$ can be computed in $O(n^2)$ time, where n is the number of chips.

From there, all we need to know if the number of firing moves at each site. These can all be computed in linear time using recurrences like the ones in 3.3.8 or Theorem 5.3.2 from [30].

For each chip, numbers of firing moves need to be checked at each of $O(n)$ sites, so the total number of checks is $O(n^2)$.

Thus, it is possible to check whether a weakly sorted configuration sorts in $O(n^2)$ time, where n is the number of chips. The number of firing moves in a (labeled) chip-firing process on the line is potentially as large as $\Theta(n^3)$, so it is possible to determine whether a configuration sorts in less time than it takes to perform the sorting.

Again, we are not merely reducing the run time from $O(n^3)$ to $O(n^2)$. We are saying that in $O(n^2)$ time, we can determine whether or not *every single* sequence of labeled moves starting from ℓ sorts, or whether there are any such sequences that do not. Actually checking all of these sequences, even using more efficient dynamic programming based methods, has a super-exponential run time.

CHAPTER 6

Labeled Chip-Firing for n Odd

6.1 Introduction

It has been shown by Hopkins et al. [27] and Klivans and Liscio [31] that labeled chip-firing from a pulse at the origin sorts when the number of chips n is even. When n is odd, the behavior is a bit more complex. While much is known about the behavior that such a system tends to exhibit, most major results in the odd case are still open. In particular, there are two main conjectures.

First, in a labeled chip-firing setting, define an "inversion" as a pair of chips labeled i and j, such that $i < j$ and j is to the left of i. In the even case, the process must terminate with no inversions. In the odd case, we have the following conjecture, due to Hopkins, McConville, and Propp [27]:

Conjecture 6.1.1 ([27])**.** *Consider an instance of a labeled chip-firing process beginning with* $2m + 1$ *chips at the origin. The number of inversions in the final configuration is at most* m*.*

This bound is always achievable for any m. If we number the chips from $-m$ to m, then we can choose to keep chip m (or, alternatively, chip $-m$) at the origin throughout the chip-firing process. When the process finishes, all other chips will be sorted (because this is effectively performing labeled chip-firing with $2m$ chips), but chip m (resp. $-m$) will be inverted with all chips to its right (resp. left), resulting in m inversions.

The other conjecture deals with what happens when the chips are fired at random. We outline three procedures for random labeled chip-firing, originally presented by Hopkins et al. [27]:

(1) At each step, choose one pair of chips to fire uniformly at random from all pairs of chips that can fire.

(2) At each step, choose a site uniformly at random from the sites with at least two chips. Then, choose 2 chips uniformly at random from that site to fire.

(3) Choose one stabilizing sequence of firing moves from the beginning to the end of the process uniformly at random.

Define $\mathbb{P}_1$, $\mathbb{P}_2$, and $\mathbb{P}_3$ to be the probability measures over firing sequences produced by these three firing procedures.

Note that procedures (1) and (2) provide different methods for choosing a random firing move at each step, while procedure (3) provides a method for choosing a random firing sequence throughout the entire process.

This leads to what is known as the $\frac{1}{3}$ conjecture:

Conjecture 6.1.2 ([27])**.** *Begin a labeled chip-firing process with* $2m + 1$ *chips at the origin, and fire until stabilization. Under any of the 3 probability measures above, the probability of the chips ending in sorted order converges to* $\frac{1}{3}$ *(from below) as* m *goes to infinity.*

Unlike in the even case, the last move in the odd case must take place at a site with 3 chips. Of the 3 ways to choose 2 of those 3 chips, exactly one of them will result in those 3 chips being sorted. Thus, the $\frac{1}{3}$ conjecture essentially says that with high probability, this is the "only" thing that goes wrong in the sorting process, although it is possible not to end up with a sorted configuration even if the two chips are chosen "correctly" at the last move.

In Section 6.2, we look further at the $\frac{1}{3}$ conjecture, outlining what we think might be a good approach toward the solution. We prove a result that unifies all three probability distributions by showing that all possible sequences of firing chip pairs are equally likely, conditioned on the sequence of unlabeled moves.

In Section 6.3, we look at what causes inversions to form during a labeled chip-firing process. We prove a surprising result about the total number of chips present at all moves throughout the process. Corollary 6.3.4 states that for any instance of a chip-firing process beginning with $2m+1$ chips at the origin, the average number of chips present for each firing move is exactly equal to 3. This result is used to compute the expected number of inversions created throughout the process.

6.2 Probabilities in Labeled Chip-Firing

One surprising thing about the $\frac{1}{3}$ conjecture is that the result is conjectured to hold for three different probability distributions. Thus, in order to prove the conjecture, one would need to tailor three different proofs to the respective distributions, or otherwise to develop a way to unify the three distributions in a way that allowed the conjecture to be proved for all three distributions at once.

We provide a method to unify the three distributions. The unification comes from the relationship between labeled configurations and their underlying unlabeled configurations. Conditioned on a particular sequence of unlabeled firing moves, all three distributions assign the same probability to each sequence of labeled firing moves. The following lemma could provide a path toward resolving the conjecture for all three probability measures at once.

Consider a stabilizing sequence of firing moves $S = (s_1, s_2, \ldots s_l)$, where for each $1 \leq t \leq l$, $s_t = (k_t, i_t, j_t)$ means that the t^{th} firing move consists of the i_t^{th} and j_t^{th} smallest chips at site k_t being fired, for $i_t < j_t$. The corresponding unlabeled firing sequence is $K = (k_1, k_2, \ldots k_l)$. Given an initial unlabeled configuration u, an unlabeled firing sequence K, and an integer t with $1 \leq t \leq l$, define $c(u, K, t)$ to be the number of chips present at site k_t prior to the t^{th} firing move in K starting from configuration u.

Lemma 6.2.1. *Consider a positive integer n and the three probability distributions $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3$ on stabilizing labeled chip-firing sequences beginning at a configuration u of n chips at the origin. The following hold for $\mathbb{P} \in \{\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3\}$.*

(1) For any $1 \leq t \leq n$, any firing moves $s^1, s^2, \ldots s^{t-1}$, and any site k^t, we have that the probability

$$\mathbb{P}\big((i_t, j_t)|(s_1, s_2, \ldots s_{t-1}) = (s^1, s^2, \ldots s^{t-1}), k_t = k^t\big)$$

is uniform over all pairs (i, j), $1 \leq i < j \leq c$ where c is the number of chips at site k following moves $s^1, \ldots s^{t-1}$.

(2) For any unlabeled firing sequence $(k^1, k^2, \ldots k^l)$, we have that the probability

$$\mathbb{P}\big(S|K = (k^1, k^2, \ldots k^l)\big)$$

is uniform over all stabilizing labeled firing sequences with firing moves at sites $(k^1, k^2, \ldots k^l)$.

Proof. Part (1) is true by definition for $\mathbb{P} = \mathbb{P}_1, \mathbb{P}_2$, while part (2) is true by definition for $\mathbb{P} = \mathbb{P}_3$. We first prove that part (1) is true for $\mathbb{P} = \mathbb{P}_3$.

Fix an unlabeled firing sequence $K = (k^1, k^2, \ldots k^l)$, and fix the first t pairs of chips that get fired $\big((i_1, j_1), (i_2, j_2), \ldots (i_t, j_t)\big) = \big((i^1, j^1), (i^2, j^2), \ldots (i^t, j^t)\big)$. The number of complete sequences of labeled firing moves satisfying these properties is equal to the number of possible sequences $\big((i_{t+1}, j_{t+1}), (i_{t+1}, j_{t+2}), \ldots (i_l, j_l)\big)$, subject to the existing constraints. The number of possible pairs (i_r, j_r) for a particular $t < r \leq l$ depends only on $c(u, K, r)$, the number of chips present for the r^{th} firing move, with a number of pairs equal to $\binom{c(K,r)}{2}$.

Thus, the total number of possible sequences of chip pairs from step $t+1$ through step ℓ is equal to $\prod_{r=t+1}^{\ell} \binom{c(u,K,r)}{2}$. If pair (i^t, j^t) is replaced with pair $\big((i^t)', (j^t)'\big)$, the number of sequences of chip pairs from step $t+1$ through step ℓ is still $\prod_{r=t+1}^{\ell} \binom{c(u,K,r)}{2}$. Thus, the probability of any pair for (i_t, j_t), conditioned on the moves $(s_1, \ldots s_{t-1})$ and the site k_t, is a uniform distribution, as desired.

Now we show that part (2) is true for $\mathbb{P} = \mathbb{P}_3$.

First, note that if move s_t occurs at site k_t, then all pairs of chips at site k_t are equally likely to be fired at time t under both procedures (1) and (2). Specifically, the probability

of each pair of chips (i_t, j_t) at site k being chosen to fire at time t is uniform over all pairs of chips at site k prior to firing move t. Furthermore, this does not depend on which pairs of chips were chosen at previous time steps. Thus, if we condition on the sequence of unlabeled firing moves $K = (k^1, \ldots k^l)$, the probability of any possible sequence of (i, j) pairs occurring is equal to

$$\prod_{t=1}^{l} \frac{1}{\binom{c(K,t)}{2}}$$

This probability is the same for any legal sequence of (i, j) pairs, so all sequences are equally likely.

□

Informally, Lemma 6.2.1 says that for any of our 3 firing procedures, and any fixed sequence of unlabeled firing moves, all sequences of labeled firing moves are equally likely. Furthermore, at each time step, each pair of chips at the firing site is equally likely to fire. In other words, suppose that we think of a random firing procedure as a two step process:

(1) Choose a stabilizing sequence of unlabeled firing moves.

(2) Choose a stabilizing sequence of labeled firing moves, subject to the underlying unlabeled firing moves from step (1).

According to Lemma 6.2.1, all 3 firing procedures differ in how they deal with step (1), but they all select from the same uniform distribution in step (2).

This breakdown allows us to think of probabilities of sorting as

$$\mathbb{P}_i(\text{sorting}) = \sum_K \mathbb{P}_i(\text{unlabeled sorting sequence } K)\mathbb{P}(\text{sorting}|K)$$

Note that the last $\mathbb{P}$ does not have a subscript, because that conditional probability is the same for all three firing procedures. This breakdown provides two possible paths to the proof of the $\frac{1}{3}$ conjecture:

(1) Find the minimum value of $\mathbb{P}(\text{sorting}|K)$ over all unlabeled sequenced K for each value of n chips. Show that this converges to $\frac{1}{3}$ as n goes to ∞.

(2) For some sequences K, $\mathbb{P}(\text{sorting}|K)$ is precisely equal to $\frac{1}{3}$. Show for $i = 1, 2, 3$ that $\mathbb{P}_i\big(K \text{ is a sequence such that } \mathbb{P}(\text{sorting}|K) = \frac{1}{3}\big)$ converges to 1 as n goes to ∞.

Both of these statements appear experimentally to be true, and both imply the $\frac{1}{3}$ conjecture. However, it is possible for either or both to be false, even if the $\frac{1}{3}$ conjecture is true. The first approach appears more promising, as it does not require a separate proof for each firing procedure, and it has stronger experimental evidence to support it.

For values of n ranging from 3 to 13, we chose a series of random unlabeled firing sequences, and then exactly computed the probability of a labeled firing sequence sorting, conditioned on that unlabeled firing sequence. The random unlabeled sequences were chosen by choosing a site uniformly at random from those ready to fire at each step. For each value of n, we report:

(1) the minimum probability of sorting over all unlabeled sequences, if we were able to compute that value exactly.

(2) the maximum probability of sorting over all unlabeled sequences.

(3) the minimum probability of sorting found over all trials.

(4) the proportion of trials for which the probability of sorting was exactly equal to $\frac{1}{3}$.

We ran 1000 trials for $n = 3$ through 13:

The minimum probability of sorting over all unlabeled sequences checked does seem to be quickly converging to $\frac{1}{3}$. The proportion of unlabeled sequences that produce sorting probabilities of exactly $\frac{1}{3}$ is increasing with n, but data would be needed for much larger values of n to even guess whether this proportion converges to 1. Even if we were able to prove that this proportion converges to 1, this would prove the $\frac{1}{3}$ conjecture under probability measure P_2, but not necessarily under the other two measures.

n	(1)	(2)	(3)	(4)
3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1
5	0.2	0.2	0.2	0
7	0.2259	0.3037	0.2259	0
9	?	$\frac{1}{3}$	0.2573	0.004
11	?	$\frac{1}{3}$	0.2960	0.021
13	?	$\frac{1}{3}$	0.3203	0.126

TABLE 6.1: Selected probabilities and empirical probabilities of sorting for labeled chip-firing with n odd.

6.3 Inversions in Labeled Chip-Firing

We can also make progress toward the m inversion conjecture by studying the way that inversions form throughout the labeled chip-firing process. We work toward a result that the "average" number of inversions created by each move in the labeled chip-firing process with n odd is equal to $\frac{2}{3}$. We begin with the following lemma:

Lemma 6.3.1. *Consider two sequences of legal (unlabeled) firing moves $K_i = (k_{i,1}, k_{i,2}, \ldots k_{i,l})$ and $K_f = (k_{f,1}, k_{f,2}, \ldots k_{f,l})$ that each have the same starting configuration u and ending configuration v. Then there exists some sequence of legal firing sequences $K_i = K_0, K_1, K_2, \ldots, K_r = K_f$, such that for all j from 1 to r, K_j can be obtained from K_{j-1} by choosing two consecutive firing moves and switching the order in which those sites fire.*

Proof. This is a special case of Lemma 4.3.14. The configuration poset of an unlabeled chip-firing process is an LLD lattice by Theorem 4.3.16 and Corollary 4.3.17, so it must be S-colorable. Thus, this result holds by Lemma 4.3.14. □

Next, we prove results about the number of chips present for various moves throughout the process. Recall that for initial configuration u and sequence of unlabeled firing moves $K = (k_1, k_2, \ldots k_\ell)$, $c(u, K, t)$ is the number of chips present at site k_t prior to the t^{th} firing move in K from initial configuration u.

Lemma 6.3.2. *Consider a terminating chip-firing process on an undirected graph beginning with some configuration u. Let $K_i = (k_{i,1}, k_{i,2}, \ldots k_{i,l})$ and $K_f = (k_{f,1}, k_{f,2}, \ldots k_{f,l})$ be two stabilizing sequences of firing moves starting from u. Then $\sum_{t=1}^{l} c(u, K_i, t) = \sum_{t=1}^{l} c(u, K_f, t)$.*

Proof. By Lemma 6.3.1, it is possible to get from K_i to K_f through a sequence of swaps of consecutive firing sites. Thus, we only need to prove that swapping two consecutive sites does not change this sum.

Suppose that a firing sequence K has consecutive moves at sites k_t and k_{t+1}. If $k_t = k_{t+1}$, then swapping the sites of moves t and $t+1$ leaves the process exactly the same. Otherwise, if there is an edge between site k_t and k_{t+1}, then swapping the order of moves t and t_1 to obtain a new sequence K' results in $c(u, K', t) = c(u, K, t+1) - a_{k_t,k_{t+1}}$ and $c(u, K', t+1) = c(u, K, t) + a_{k_t,k_{t+1}}$, where $a_{k_t,k_{t+1}}$ is the number of edges between k_t and k_{t+1}. The numbers of chips present for all other firing moves remain the same, so the sum remains the same.

If k_t and k_{t+1} are not the same or adjacent, then swapping the sites of moves t and $t+1$ to obtain a new sequence K' results in $c(u, K', t) = c(u, K, t+1)$ and $c(u, K', t+1) = c(u, K, t)$. The numbers of chips present for all other firing moves remain the same, so the sum remains the same. □

Note that Lemma 6.3.2 applies to any terminating chip-firing process on an undirected graph, not just on a line.

In the case of chip-firing on the line beginning with n chips at the origin, a consequence of Lemma 6.3.2 is that the sum of the $c(u, K, t)$ values over the course of such a process depends only on n. We can now determine that sum as a function of n by adding up the $c(u, K, t)$ values for one stabilizing firing sequence beginning with n chips at the origin.

Lemma 6.3.3. *Consider a chip-firing process on the line, beginning with a configuration u of $2m+1$ chips at the origin. For any stabilizing sequence $K = (k_1, k_2, \ldots, k_l)$ of moves, $\sum_{t=1}^{l} c(u, K, t) = \frac{m(m+1)(2m+1)}{2}$.*

Proof. Consider the class of firing sequences considered by [2], which lead to the following intermediate configurations:

$$
\begin{array}{ccccccccc}
 & & & & 2m+1 & & & & \\
 & & & 1 & 2m-1 & 1 & & & \\
 & & 1 & 1 & 2m-3 & 1 & 1 & & \\
 & 1 & 1 & 1 & 2m-5 & 1 & 1 & 1 & \\
 & & & & \vdots & & & &
\end{array}
$$

To get from one intermediate configuration to the next, we follow the following pattern:

$$
\begin{array}{ccccccccccc}
 & 1 & 1 & 1 & 1 & 3 & 1 & 1 & 1 & 1 & \\
 & 1 & 1 & 1 & 2 & 1 & 2 & 1 & 1 & 1 & \\
 & 1 & 1 & 2 & 0 & 3 & 0 & 2 & 1 & 1 & \\
 & 1 & 2 & 0 & 2 & 1 & 2 & 0 & 2 & 1 & \\
 & 2 & 0 & 2 & 0 & 3 & 0 & 2 & 0 & 2 & \\
1 & 0 & 2 & 0 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \\
1 & 1 & 0 & 2 & 0 & 3 & 0 & 2 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 2 & 1 & 2 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
$$

There is a region of consecutive 1's, and a region of 2's and 0's, which expands and then retreats, leaving chips one site further out, but leaving two fewer chips at the origin. Note that since adjacent sites are never fired at the same time, the resulting number of chips at each firing site is the same as if the firing moves were viewed as happening simultaneously.

To simplify our calculations, we compute $\sum_{t=1}^{l} \big(c(u,K,t) - 2\big)$, and then add in 2 times the number of firing moves at the end.

All firing moves away from the origin leave 0 chips at their respective sites, so the moves at the origin are the only ones contributing to the sum. This leaves

$$\sum_{t=1}^{l} c(u,K,t) - 2 = 1(2m+1) + 2(2m-1) + 3(2m-3) + \cdots + m(1) \tag{6.1}$$

$$= \big((2m+1) + (2m-1) + \cdots + 1\big) + \big((2m-1) + (2m-3) + \cdots + 1\big) + \cdots + 1 \tag{6.2}$$

$$= m^2 + (m-1)^2 + \cdots + 1 \tag{6.3}$$

$$= \frac{m(m+1)(2m+1)}{6} \tag{6.4}$$

$$\tag{6.5}$$

as desired.

The total number of moves of the process is equal to $\frac{m(m+1)(2m+1)}{6}$ [2].

Thus, we get $\sum_{t=1}^{l} c(u,K,t) = \frac{m(m+1)(2m+1)}{6} + 2\frac{m(m+1)(2m+1)}{6} = \frac{m(m+1)(2m+1)}{2}$, as desired.

□

We define $c'(u,K,t)$ to be the number of chips present at k_t immediately after the t^{th} firing move in K starting from configuration u. It differs from $c(u,K,t)$ only in that the number of chips is counted after the t^{th} move instead of before, so $c'(u,K,t) = c(u,K,t) - 2$.

Corollary 6.3.4. *Consider a chip-firing process on the line, beginning with a configuration u of $2m+1$ chips at the origin. For any stabilizing sequence $K = (k_1, k_2, \ldots, k_\ell)$ of moves, the average value of $c(u,K,t)$ as t ranges from 1 to ℓ is equal to 3. The average value of $c'(u,K,t)$ is equal to 1.*

Proof. The total number of moves of the process is equal to $\frac{m(m+1)(2m+1)}{6}$ [2].

Since the sum of the $c(u,K,t)$ values for any stabilizing firing sequence is equal to $\frac{m(m+1)(2m+1)}{2}$, the average value of $c(u,K,t)$ over the course of the process must be 3.

As $c'(u,K,t) = c(u,K,t) - 2$, the average value of $c'(u,K,t)$ is equal to 1. □

Corollary 6.3.5. *Consider a chip-firing process on the line, beginning with a configuration u of $2m$ chips at the origin. For any stabilizing sequence $K = (k_1, k_2, \ldots, k_l)$ of moves, $\sum_{t=1}^{l} c(u, K, t) = m^2(m+1)$.*

Proof. The proof of Lemma 6.3.3 uses a sequence of firing moves in which there is always at least one chip at the origin. Thus, we can create a valid firing sequence beginning with $2m$ chips at the origin by simply removing 1 chip at the origin at every step in that sequence. Thus, the sum of the $c(u, K, t)$ values beginning with $2m$ chips is the same as the sum of the $c(u, K, t)$ values beginning with $2m+1$ chips, minus the number of firing moves at the origin. There are $\frac{m(m+1)}{2}$ firing moves at the origin, so the sum is equal to $\frac{m(m+1)(2m+1)}{2} - \frac{m(m+1)}{2} = m^2(m+1)$. □

We can also use our result about $c(u, K, t)$ values to get a few more useful results about the number of inversions. We say that an inversion of pair (i, j) of labeled chips $(i < j)$ is created by a firing move s at site k_t if the position of chip i is less than or equal to the position of chip j prior to move s, but the position of chip i is greater than the position of chip j immediately after move s.

Lemma 6.3.6. *Consider a labeled chip-firing process on the line, beginning with a configuration u, and a stabilizing sequence $K = (k_1, k_2, \ldots, k_l)$ of moves. Under any of our random firing procedures, the expected number of inversions produced by move t is equal to $\frac{2}{3}(c(u, K, t) - 2)$*

Proof. By Lemma 6.2.1, any pair of the $c(u, K, t)$ chips at site k_t is equally likely to be chosen to fire under any of the 3 firing procedures. If there are only 2 chips at site k_t, then both are fired, and no inversions are created. Otherwise, consider 3 chips h, i, and j at site k_t, with $h < i < j$. Any of the 3 pairs (h, i), (h, j), and (i, j) are equally likely to be chosen to fire. If h and j are chosen, no inversions are created among these 3 chips. If h and i are chosen, then i and j are inverted after the move is performed. If i and j are chosen, then h and i are inverted after the move is performed. Thus, for every triple of chips at site k_t containing the two chips that are fired by move t, there is a $\frac{2}{3}$ probability of a single inversion being created among those 3 chips. The number of triples that

include the 2 chips that are fired is equal to $c(u, K, t) - 2$, so the expected number of inversions created is equal to $\frac{2}{3}(c(u, K, t) - 2)$. □

Corollary 6.3.7. *Consider a labeled chip-firing process on the line, beginning with a configuration u of $2m + 1$ chips at the origin. The expected number of inversions created throughout the entire labeled chip-firing process is equal to $\frac{m(m+1)(2m+1)}{9}$.*

Proof. For any sequence $K = (k_1, k_2, \ldots k_l)$ of unlabeled moves, we have that $\sum_{t=1}^{l} c(u, K, t) = \frac{m(m+1)(2m+1)}{2}$ by Lemma 6.3.3. By the proof of Lemma 6.3.3, we also get that $\sum_{t=1}^{l}(c(u, K, t) - 2) = \frac{m(m+1)(2m+1)}{6}$, giving us that $\sum_{t=1}^{l} \frac{2}{3}(c(u, K, t) - 2) = \frac{m(m+1)(2m+1)}{9}$. This is equal to the expected number of inversions created throughout the entire labeled chip-firing process. □

This result says nothing about the number of inversions that are destroyed throughout the process (which must also be pretty substantial, since the number of inversions created is larger than the maximum number of inversions possible in a list of length $2m + 1$). However, we hope that studying inversions in this way can help to approach the m inversion conjecture. In addition, Lemma 6.3.2 may have some utility on other undirected graphs.

6.4 The Move Poset

In Chapter 3, we use the move poset of chip-firing on the line to prove that labeled chip-firing sorts when the number of chips n is even. While there is less defined structure in the case where n is odd, the move poset can still tell us a lot about the process, and may be relevant to the $\frac{1}{3}$ and m inversion conjectures. Figure 6.1 shows the move posets for $n = 10$ and $n = 11$.

As we can see, the structures are very similar. The number of firing moves at each site is the same in each case, but the diamond at the bottom of the Hasse diagram for $n = 10$ is replaced with a triangle. While this triangle is no longer structured enough to guarantee sorting, there is still plenty of structure on the odd case, some of which

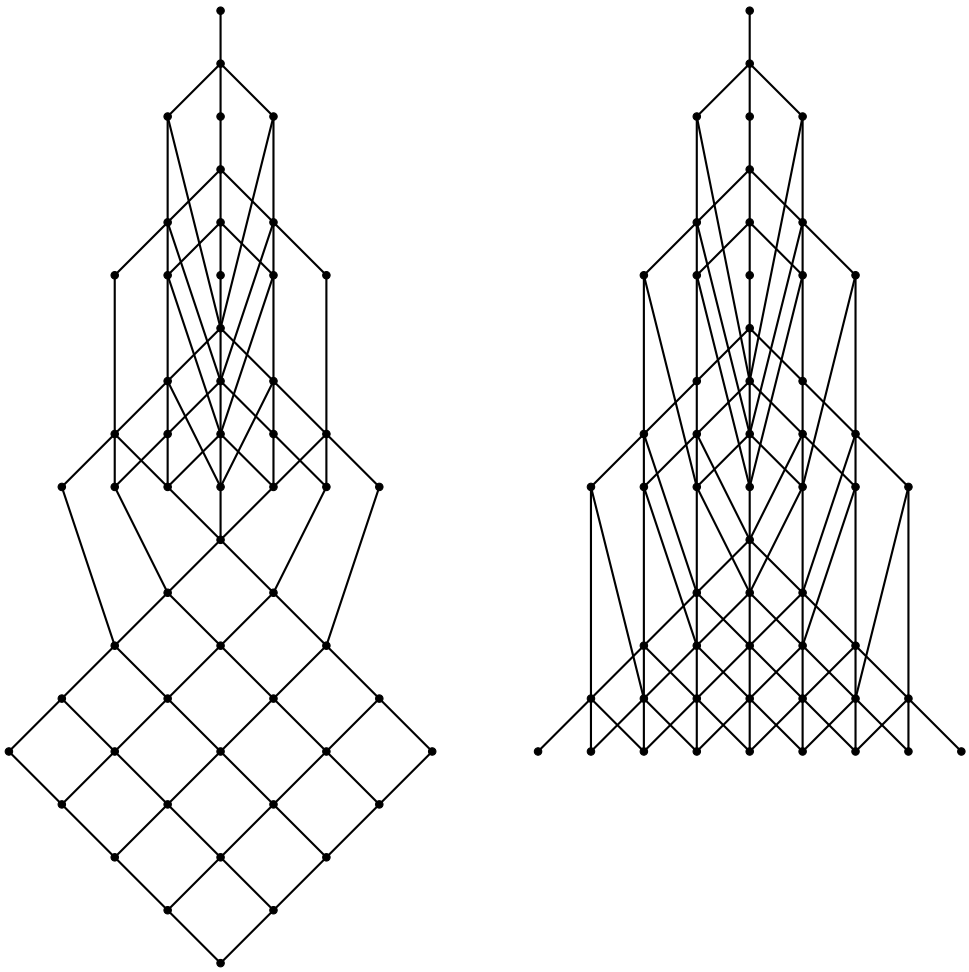


FIGURE 6.1: Firing order poset for $n = 10$ (left) and $n = 11$ (right).

cannot be fully captured by the Hasse diagram. We prove two results on the order of these firing moves and the number of chips involved in those moves.

Lemma 6.4.1. *Consider an instance of a chip-firing process beginning with $2m+1$ chips at the origin. Consider an integer j, $1 \leq j \leq m$, and consider moves k_j for all sites k such that $0 \leq |k| \leq m-j$. Then the following hold:*

(1) Let $k^{f(j)}$ be the value of k such that k_j occurs after k'_j for all $0 \leq |k'| \leq m-j$, $k' \neq j$. Then if $|k^{f(j)}| < m-j$, move $k_j^{f(j)}$ occurs with 3 chips present. If $|k^{f(j)}| = m-j$, then move $k_j^{f(j)}$ occurs with 2 or 3 chips present.

(2) For all k' such that $j-m \leq k' < k^{f(j)}$, move k'_j occurs before moves $(k'+1)_j$ and $(k'-1)_{j+1}$ if those moves exist, and move k'_j occurs with 2 chips present.

(3) For all k' such that $k^{f(j)} < k' \leq m-j$, move k'_j occurs before moves $(k'-1)_j$ and $(k'+1)_{j+1}$ if those moves exist, and move k'_j occurs with 2 chips present.

Proof. Induct on $|k' - k^{f(j)}|$, and then on j. First, let $j = 1$. Then $k_j^{f(j)}$ is the last move in the entire process. Since site $k^{f(1)}$ ends the process with 1 chip, and no firing moves occur after the last move at $k^{f(1)}$, move $k_1^{f(1)}$ must occur with 3 chips present.

Now induct on $|k' - k^{f(1)}|$. Consider firing move k'_1, for $2-m < k' < k^{f(1)}$. Assume for induction that move k'_1 occurs before move $(k'+1)_1$, so site k' must receive 1 chip from site $k'+1$ after move k'_1 occurs. Since site k' ends the process with 1 chip, its last move must occur after moves $(k'-1)_1$ and $(k'+1)_2$ if those moves exist, and move k'_1 must occur with 2 chips present. The result follows for $k' > k^{f(1)}$ by symmetry.

Now induct on j. Suppose that the result holds for all k_{j-1} moves for a particular value of j. Then move $k_j^{f(j)}$ must occur after moves $(k^{f(j)}-1)_j$ (if $k^{f(j)} > j-m$) and $(k^{f(j)}+1)_j$ (if $k^{f(j)} < m-j$), but before moves $(k^{f(j)}-1)_{j-1}$ and $(k^{f(j)}+1)_{j-1}$. If $k^{f(j)} = k^{f(j-1)}$, then move $k_{j-1}^{f(j)}$ occurs with 3 chips present, and site $k^{f(j)}$ loses 2 chips and gains 2 chips (if $|k^{f(j)}| < m-j$, or $|k^{f(j)}| = m-j$ and move $k_j^{f(j)}$ does not occur before move j at an outer neighbor of $k^{f(j)}$) or 3 chips (if $|k^{f(j)}| = m-j$ and move $k_j^{f(j)}$ does occur before move j at an outer neighbor of $k^{f(j)}$) between the start of move $k_j^{f(j)}$ and the start

of move $k_{j-1}^{f(j)}$. If $k^{f(j)} \neq k^{f(j-1)}$, then move $k_{j-1}^{f(j)}$ occurs with 2 chips present, and site $k^{f(j)}$ loses 2 chips and gains 1 chip (if $|k^{f(j)}| < m - j$, or $|k^{f(j)}| = m - j$ and move $k_j^{f(j)}$ does not occur before move j at an outer neighbor of $k^{f(j)}$) or 2 chips (if $|k^{f(j)}| = m - j$ and move $k_j^{f(j)}$ does occur before move j at an outer neighbor of $k^{f(j)}$) between the start of move $k_j^{f(j)}$ and the start of move $k_{j-1}^{f(j)}$. Either way, move $k_j^{f(j)}$ must take place with 3 chips present (if $|k^{f(j)}| < m - j$) or 2-3 chips present (if $|k^{f(j)}| = m - j$).

Now induct on $|k' - k^{f(j)}|$. Consider firing move k'_j, for $j - m + 1 < k' < k^{f(j)}$. Assume for induction that move k'_j occurs before moves $(k' + 1)_j$ and $(k' - 1)_{j-1}$. If $k' = k^{f(j-1)}$, then move k'_{j-1} occurs with 3 chips present, and site k' loses 2 chips and gains at least (and thus exactly) 3 chips between the start of move k'_j and the start of move k'_{j-1}. If $k' \neq k^{f(j-1)}$, then move k'_{j-1} occurs with 2 chips present, and site k' loses 2 chips and gains at least (and thus exactly) 2 chips between the start of move k'_j and the start of move k'_{j-1}. Either way, move k'_j must occur after move $(k' - 1)_j$ and move $(k' + 1)_{j+1}$, and it must occur with exactly 2 chips present. □

Lemma 6.4.1 allows us to consider the triangle at the bottom of the move poset in terms of its rows. The last move in each row occurs with 3 chips present (except for an exception where the last move is on the end of the row, in which case that move may take place with only 2 chips present). The rest of the moves in each row occur from the outside in, and all with 2 chips present. Thus, each row corresponds to a pair of partial "bubble sort passes:" one starting from the left, and one starting from the right. At the end of these bubble sort passes is a single firing move with 3 chips, which has the potential to create one inversion (although this inversion gets "fixed" in the next pass).

This provides a possible approach to the m inversion conjecture. Maybe the maximum of m inversions has something to do with the fact that each of the m rows in the triangle has a single 3-chip firing move with the potential to create a single inversion.

The next lemma further reinforces this idea of looking at the triangle in terms of its rows. We can show that moves in later rows cannot "affect" moves in earlier rows, essentially allowing us to think of the rows as happening in order, even if it is possible for some moves from later rows to occur before some moves from earlier rows.

Lemma 6.4.2. *Consider a labeled chip-firing process on the line beginning with* $2m+1$ *chips. Consider a chip* i*, and two firing moves* $(k_1)_{j_1}$ *and* $(k_2)_{j_2}$ *such that* $j_2 < j_1 \leq m$*,* $|k_1| + j_1 \leq m$*, and* $|k_2| + j_2 \leq m$*. Then chip* i *cannot be fired by move* $(k_2)_{j_2}$*, and then fired later by move* $(k_1)_{j_1}$*.*

Proof. Since each firing move involving chip i moves it 1 unit to the left or right, consecutive firing moves involving chip i must occur at adjacent sites. By Lemma 6.4.1, for any $1 \leq j \leq m$ and $0 \leq |k| \leq m - j$, move k_j must occur after moves $(k-1)_{j+1}$ and moves $(k+1)_{j+1}$. Thus, if chip i is involved in firing move $(k_2)_{j_2}$, the next move it is involved in must have subscript at most j_2. By induction, we can show that every subsequent move involving chip i must have subscript at most j_2, so chip i cannot later be fired by move $(k_1)_{j_1}$ as desired. □

Lemma 6.4.2 tells us that we can think of the firing moves in the triangle as happening "one row at a time" for the purposes of labeled chip-firing. Even though it is possible for a move in one row to happen before a move in a previous row, it is impossible for a single chip to be involved in both moves, so the order of these moves does not affect the outcome of a labeled chip-firing process (although it may affect the probabilities of certain sequences of moves occurring). Thus, we can think of the triangle at the bottom of the move poset as constituting a series m passes of bubble sort affecting more and more chips, until all chips are involved in the last pass.

CHAPTER 7

Root System Chip-Firing

7.1 Introduction

In [22] and [21], Galashin et al. view labeled chip-firing as a "Type A" case of a more general type of root system chip-firing. They consider two extensions of labeled chip-firing onto root systems: interval-firing, which allows for chips at nearby sites to fire together, and central-firing, which imposes additional firing rules for root systems of Types B, C, and D. This chapter focuses on root system central-firing.

In central-firing, chips are numbered from 1 to n and fired according to various combinations of the following types of moves:

(a) For $i < j$, if chips i and j are in the same position, move chip i one step to the left and chip j 1 step to the right.

(b) For $i \in [n]$, if chip i is at the origin, move it one step to the right.

(c) For $i \in [n]$, if chip i is at the origin, move it two steps to the right.

(d) For $i < j$, if chips i and j are in opposite positions, move both chips one step to the right.

Type A central-firing uses moves (a), which are precisely the moves allowed in labeled chip-firing. Type B central-firing uses moves (a), (b), and (d). Type C central-firing uses moves (a), (c), and (d). Type D central-firing uses moves (a) and (d).

The addition of new types of moves creates longer processes. As moves (b), (c), and (d) all send chips to the right without sending chips to the left, the final configurations tend to have more chips further to the right than in the Type A case.

However, while other root system central-firing processes differ in behavior from traditional labeled chip-firing, many initial configurations in all four types of central-firing are either known or conjectured to sort. Our methods involving move posets and join-irreducible configurations allow us to prove and provide insight into open problems in central-firing.

In Section 7.2, we provide some preliminaries on root system central-firing, and introduce the main conjectures of Galashin et al. [21] on which initial configurations sort in each type of root system. In Section 7.3, we prove the remaining major open problem in Type B central-firing (also an open problem in Type D) that central-firing sorts from an initial configuration with all chips at site $\frac{1}{2}$. This conjecture is proved by using S-colorings and the join-irreducibles of the configuration poset to analyze the move poset of Type B central-firing. In Section 7.4, we address the remaining open problems in central-firing of Types C and D. We disprove and modify a conjecture about Type D central-firing based on our computational results, and we show a contrast in the move posets of configurations that are conjectured to sort and not sort.

7.2 Preliminaries

Here, we present some preliminaries on root system central-firing, and we again state the sorting conjectures from [21] that we presented in Chapter 2. For an introduction to root systems and root system central-firing, see Section 2.5.

7.2.1 Unlabeled Central-Firing

Just as in labeled chip-firing, every central-firing process has an underlying unlabeled process. Given a central-firing process R on root system Φ, define the corresponding **unlabeled central-firing process** R' by taking every configuration of R modulo the Weyl group of Φ.

Given a configuration λ, taking λ modulo the Weyl group of Φ yields:

- the multiset of chip locations when Φ is of type A_{n-1}.
- the multiset of the absolute values of the chip locations when Φ is of type B_n or C_n.
- the multiset of the absolute values of the chip locations, along with the sign of the product of the chip locations, when Φ is of type D_n.

Note that when Φ is of type A_{n-1}, taking a labeled configuration modulo the Weyl group yields the unlabeled configuration in traditional labeled chip-firing.

Galashin et al. show that any central-firing process modulo the Weyl group of Φ is locally confluent, although they use a slightly weaker version of local confluence than what has been used throughout this thesis [21].

For the remainder of this chapter, we say that a process is **locally confluent** if for any three configurations u, v, and w, such that v and w are reachable from u, there must exist a configuration z that is reachable from v and w. This is weaker than our previously discussed definition of local confluence in that none of these configurations are required to be reachable in a single move. If the process is finite, local confluence still does imply global confluence, but the process is no longer required to take a fixed number of moves to complete.

7.2.2 Sorting Conjectures

Galashin et al. outline all initial weights conjectured to sort for each type of root system central-firing. Their confluence conjecture is as follows:

Conjecture 7.2.1 ([21], Conjecture 7.1)**.** *Let Φ be of Type A, B, C, or D, and let $\omega \in \Omega \cup \{0\}$ be a fundamental weight or zero. Then central-firing is confluent from ω if and only if $\omega \notin Q + \rho$, unless one of the four exceptional cases happens:*

(1) $\Phi = A_n$, in which case central-firing is confluent from ω if and only if

$$\begin{cases} \omega = 0, \omega_1, \omega_n & n \text{ odd} \\ \omega = \omega_{n/2}, \omega_{n/2+1} & n \text{ even} \end{cases}$$

(2) $\Phi = B_n$, *in which case central-firing is confluent from* $\omega = \omega_n$ *despite the fact that* $\omega_n \in Q + \rho$.

(3) $\Phi = D_{4n+2}$ *for* $n \geq 1$ *in which case central-firing is not confluent from* $\omega = 0$ *even though* $0 \notin Q + \rho$.

They prove several results from Conjecture 7.2.1, but 6 problems are left open.

Conjecture 7.2.2 ([21], Problem 7.14)**.**

(1) *Central-firing is confluent from* ω_1 *and* ω_{2n+1} *for* $\Phi = A_{2n+1}$.

(2) *Central-firing is confluent from* ω_n *and* ω_{n+1} *for* $\Phi = A_{2n}$.

(3) *Central-firing is confluent from* ω_n *for* $\Phi = B_n$ *(equivalently, for* $\Phi = D_n$*).*

(4) *Central-firing is confluent from* $\omega \in \Omega \cup \{0\}$ *if and only if* $\omega \not\equiv \rho$ *in* P/Q *for* $\Phi = C_n$.

(5) *Central-firing is not confluent from 0 for* $\Phi = D_{4n+2}$.

(6) *Central-firing is confluent from* $\omega \in \Omega \cup \{0\}$ *for all* $\omega \not\equiv \rho$ *in* P/Q *for* $\Phi = D_n$*, except for the case (5) above.*

In terms of chips, The open problems are as follows:

(1) Type A central-firing is confluent from a weakly sorted configuration with 1 chip at the origin and $2n-1$ chips at site 1, or with $2n-1$ chips at the origin and 1 chip at site 1.

(2) Type A central-firing is confluent from a weakly sorted configuration with n chips at the origin and $n+1$ chips at site 1, or $n+1$ chips at the origin and n chips at site 1.

(3) Type B and Type D central-firing are confluent from a configuration with all chips at site $\frac{1}{2}$.

(4) Type C central-firing is confluent from any weakly sorted configuration with an even number of chips at site 1, and the rest at the origin, if the number of chips $n \equiv 1$ or 2 mod 4, or an odd number of chips at site 1, and the rest at the origin, if $n \equiv 0$ or 3 mod 4.

(5) Type D central-firing is not confluent from any configuration with n chips at the origin, for $n \equiv 2$ mod 4.

(6) Type D central-firing is confluent from any weakly sorted configuration with an even number of chips at site 1, and the rest at the origin, if the number of chips $n \equiv 2$ or 3 mod 4, or an odd number of chips at site 1, and the rest at the origin, if $n \equiv 0$ or 1 mod 4, with the exception of case (5).

Parts (1) and (2) are shown in [35] and [31], although their proofs are not discussed in detail. The proofs are shown in full in Chapter 5 of this thesis. We prove (3) in Section 7.3 (see also [35]). (4), (5), and (6) are discussed in Section 7.4. We discuss our computational counterexample to (5) and modify conjecture (6) accordingly.

7.3 Type B

We will now prove part (3) of Conjecture 7.2.2. In initial weight ω_n for B_n and D_n, all chips start at site $\frac{1}{2}$. Define a move at location k for $k \geq 0$ to be a firing move involving chips at sites $\pm(k + \frac{1}{2})$. Thus, a type (a) move that sends two chips from $k + \frac{1}{2}$ to $k - \frac{1}{2}$ and $k + \frac{3}{2}$, and a type (d) move that sends chips from sites $\pm(k + \frac{1}{2})$ to sites $k + \frac{3}{2}$ and $-k + \frac{1}{2}$ are both considered to be moves at location k. Note that both types of moves have the same effect on the absolute values of the two chips. If $k \geq 1$, then both send chips from absolute value $k + \frac{1}{2}$ to absolute value $k - \frac{1}{2}$ and $k + \frac{3}{2}$. If $k = 0$, then both send chips from absolute value $\frac{1}{2}$ to absolute value $\frac{1}{2}$ and $\frac{3}{2}$. Note that no type (b) moves may occur in this process, because no chips ever reach the origin throughout the process.

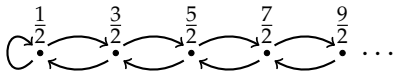


FIGURE 7.1: The unlabeled version of Type B and D central-firing with initial weight ω_n corresponds to chip-firing on this graph.

The unlabeled version of this problem corresponds to chip-firing on the graph shown in Figure 7.1. There is one vertex corresponding to each positive half-integer. From each $k > 0$, there is an edge from $k + \frac{1}{2}$ to $k - \frac{1}{2}$, and an edge from $k - \frac{1}{2}$ to $k + \frac{1}{2}$. There is also a self-loop at $\frac{1}{2}$.

We first need the final unlabeled configuration of this process.

Lemma 7.3.1. *For every i from 0 to $n-1$, there is exactly one chip in the final configuration whose position has absolute value $i + \frac{1}{2}$.*

Proof. We modify the proof of Theorem 2 from [31]. As shown in [21], the chip configuration modulo the Weyl group of B_n is locally confluent, and thus globally confluent. This implies that the absolute values of the chip positions in any final chip-configuration must be fixed, regardless of the sequence of moves performed. Thus, if we can show a single sequence of moves leading to our desired final configuration, then the same must be true for all possible sequences of moves. We will deal with a sequence that leads to the following intermediate configurations:

$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$	$\frac{9}{2}$	
n					
$n-1$	1				
$n-2$	1	1			
$n-3$	1	1	1		
$n-4$	1	1	1	1	
$\vdots$					
1	1	1	1	1	$\cdots$

In each step, we remove a chip with absolute value $\frac{1}{2}$ and replace it with a chip at the next closest value that is currently empty. Each step only requires the use of 2 chips at

$\frac{1}{2}$, so we treat the problem as if there are only 2 chips present there initially. All possible firing moves are made simultaneously in the following pattern:

$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$	$\frac{9}{2}$
2	1	1	1	
1	2	1	1	
2	0	2	1	
1	2	0	2	
2	0	2	0	1
1	2	0	1	1
2	0	1	1	1
1	1	1	1	1

There is a line of alternating 2's and 0's (or 1's at site $\frac{1}{2}$) that expands until an extra chip is deposited at the farthest point, and then contracts again. Starting with n chips and then repeating the above process until there is only one chip left at $\frac{1}{2}$ yields a configuration of the desired form.

□

Now, define move k_j to be the j^{th} to last move at location k, where a move at location k is again defined to be any move involving chips moving from sites $\pm(k+\frac{1}{2})$. We want to prove the existence of a similar diamond to what exists in the type A case.

Define P_c to be the poset of unlabeled configurations. Here, we mod out by the Weyl group, so that each element in the configuration poset corresponds to the multiset of absolute values of the chip positions. We would like to use join-irreducibles of P_c to provide information about the move poset, so we first need to prove that P_c is an LLD lattice.

Lemma 7.3.2. *Consider the unlabeled configuration poset P_c of a central-firing process of Type B_n with initial weight ω_n. Color P_c such that each color corresponds to the location of each firing move. This coloring is a valid L-coloring.*

Proof. Given an unlabeled configuration u and location k, there is at most one way to perform a firing move at k from configuration u (note that type (a) and (d) firing moves at k have the same effect on u). Thus, this coloring satisfies condition (1) of L-coloring.

Furthermore, as each unlabeled firing move corresponds to a chip-firing move on the graph shown in Figure 7.1, the process is locally confluent (in the stronger sense discussed in Chapter 2), so the coloring satisfies condition (2), and is thus an L-coloring. □

Lemma 7.3.3. *Consider the unlabeled configuration poset P_c of a central-firing process of Type B_n with initial weight ω_n. Then each color in the S-coloring of P_c consists of all edges corresponding to moves of the form "the j^{th} firing move at location k, for fixed location k and integer j.*

Proof. By Lemma 7.3.2, there exists a valid L-coloring in which each color corresponds to a single location being fired. By Lemma 4.3.15, each color in this L-coloring is a union of colors of the S-coloring which form a chain in the color poset. As a result, given a location k, all colors of the S-coloring corresponding to firing moves at k must occur in the same order in any complete downward path in P_c. Thus, each color in the S-coloring corresponding to location k must consist of all moves of the form "the j^{th} move at k" for some fixed j. □

Becuase the colors of the S-coloring correspond to moves of the move poset, the move poset is isomorphic to the poset of join-irreducibles of P_c. Define $J(k_j)$ to be the join-irreducible configuration corresponding to move k_j. The rest of the proof is similar to the proof of sorting for labeled chip-firing with $2m$ chips. We prove the existence of a locally confluent diamond of firing moves, and we show that the diamond forces chips into their final sorted positions.

Lemma 7.3.4. *For $1 \leq j \leq n-k-1$, move k_j must take place before $(k+1)_{j-1}$ and $(k-1)_j$, if sites $k+1$ and $k-1$ fire at least $j-1$ times and j times, respectively. Further, move k_j must take place with exactly 2 chips present at sites $\pm(k+\frac{1}{2})$.*

Proof. The proof closely follows Lemmas 4.4.3 and 4.4.4. We identify the join-irreducible configurations associated with each of the moves in question, and we can show reachability relations between the relevant join-irreducibles.

By Theorem 4.3.12, a move $(k_1)_{j_1}$ being required to take place before $(k_2)_{j_2}$ is equivalent to $J\big((k_2)_{j_2}\big)$ being reachable from $J\big((k_1)_{j_1}\big)$. As a result, we will show that for $1 \leq j \leq n-k$, join-irreducibles $J\big((k+1)_{j-1}\big)$ and $J\big((k-1)_j\big)$ are reachable from $J(k_j)$, if those elements exist.

We will show that for $k \geq 0$, $0 \leq j \leq n-k-1$, $J(k_j)$ consists of 2 chips at location k, 0 chips at location $k+j$, and 1 chip at all other locations from 0 to $n-1$.

A configuration with the specified numbers of chips at each location is reachable through the following firing moves at each location:

- all moves at locations $k+j$ through $n-1$
- all but the last j moves at locations 0 through k
- all but the last $k+j-x$ moves at locations $x = k+1$ through $x = k+j-1$

We can show that these firing moves produce the desired configuration by comparing the number of firing moves at each location and its neighbors to the total number of firing moves throughout the process.

- At locations $x > k+j$, all moves have occurred at location x and its neighbors, so the number of chips at location x in configuration $J(k_j)$ is equal to the number of chips (1 chip) at location x in the final configuration.
- At location $x = k+j$, all moves have occurred at location x and one of its neighbors, but one move has not yet occurred at its other neighbor. As a result, location x has one fewer chip in configuration $J(k_j)$ (0 chips) than location x has in the final configuration.
- The number of firing moves yet to occur at location k exceeds the average number of remaining firing moves at its neighbors by $\frac{1}{2}$, so location k has one more chip in configuration $J(k_j)$ (2 chips) than location k has in the final configuration.

- At all other locations x, the number of firing moves yet to occur at location x is equal to the average number of remaining firing moves at each neighbor of location x, so location x has as many chips in configuration $J(k_j)$ (1 chip) as location x has in the final configuration.

As a result, each join-irreducible takes the desired form. Now consider join-irreducibles k_j and $(k+1)_{j-1}$, for $k > 0$. $J(k_j)$ consists of 2 chips at location k, 0 chips at location $k+j$, and 1 chip at all other locations from 0 to $n-1$. $J\big((k+1)_{j-1}\big)$ consists of 2 chips at location $k+1$, 0 chips at location $k+j$, and 1 chip at all other locations from 0 to $n-1$. It is possible to get from $J(k_j)$ to $J\big((k+1)_{j-1}\big)$ by firing successively at every location from k down to 0, so $J\big((k+1)_{j-1}\big)$ is reachable from $J(k_j)$. Similarly, it is possible to get from $J(k_j)$ to $J\big((k-1)_j\big)$ by firing successively at every location from k up to $k+j-1$, so $J\big((k-1)_j\big)$ is reachable from $J(k_j)$.

Because the corresponding join-irreducibles satisfy the desired reachability constraints, the moves of the poset satisfy the corresponding order, as desired.

For each k_j considered, $J(k_j)$ has 2 chips at location k. Because all firing moves at neighboring locations that can occur before $J(k_j)$ must do so to reach configuration $J(k_j)$, the number of chips at location k in configuration $J(k_j)$ must be the maximum number of chips that can be available for move k_j. Since move k_j cannot take place with fewer than 2 chips at location k, it must take place with exactly 2 chips present.

□

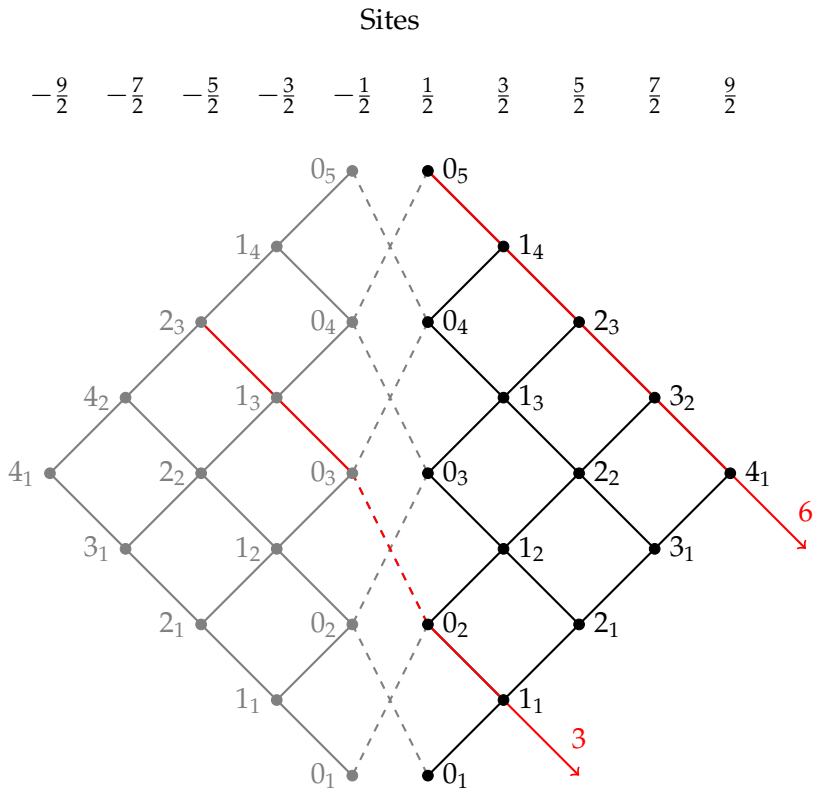


FIGURE 7.2: The Hasse diagram of the firing moves at the end of the labeled chip-firing process on B_6 (in which all chips begin at site $\frac{1}{2}$), and consequently, the join-irreducibles of the unlabeled configuration poset. The moves are duplicated in gray on the left side of the figure to represent that some moves may involve a chip initially at a negative site. The "paths" followed by chips 3 and 6 are shown in red. Neither chip may be involved in a firing move to the left of its corresponding red line, and must be fired to the right any time it is involved in a firing move on the red line.

Now that we have results on the unlabeled system, number the chips from 1 to n. We can establish lower bounds on the positions of various chips throughout the process.

Lemma 7.3.5. *For any k from 1 to n, the position of chip k must never be less than $\frac{1}{2} + k - n$ at any point in the chip-firing process.*

Proof. Proceed by strong induction on k, going downward from n. $k = n$ is the largest possible label that a chip can have. Thus, it can never be fired to the left from its initial position at $\frac{1}{2}$.

Now suppose that for each chip k' with label greater than k, the position of k' may never be less than $\frac{3}{2} + k - n$. Then consider the chip with label k. If the chip ever reaches

the position $\frac{1}{2} + k - n$, then by our inductive assumption, it must be the smallest chip at that site. Thus, it cannot be sent further to the left from that site, and its position can never be less than $\frac{1}{2} + k - n$ as desired. The result follows by induction. □

The next lemma tracks the locations of chips as they "move through the diamond." Given a chip's position throughout the process, we can use the relationships between the various firing moves to guarantee that specific chips must be sufficiently far to the right at specific times.

Lemma 7.3.6. *For each chip k with $2 \leq k \leq n$, and each x with $0 \leq x \leq n-1$, the position of chip k must be at least $\frac{1}{2} + k - n + x$ immediately preceding firing move $(n-k-x-1)_{k-1}$ for $x < n-k$, or $(k-n+x)_{n-x}$ for $x \geq n-k$.*

Proof. Proceed by induction, first on x, and then on k. First, let $k = n$ and $x = 0$. By Lemma 7.3.5, the position of chip n may never be less than $\frac{1}{2}$. In particular, chip n must be at position at least $\frac{1}{2}$ prior to move 0_n. Now suppose that chip n is at position at least $\frac{1}{2} + x$ immediately preceding move x_{n-x}. If chip n is at position $\frac{1}{2} + x$, then because there can only be two chips present at sites with absolute value $\frac{1}{2} + x$ prior to move x_{n-x}, chip n must be fired by move x_{n-x}, and since it is the largest chip at site $\frac{1}{2} + x$, it must be fired to the right. Thus, chip n must be at position at least $\frac{1}{2} + x + 1$ immediately following move x_{n-x}. Since move x_{n-x} must occur between move $(x+1)_{n-x}$ and move $(x+1)_{n-x-1}$, chip n must be at position at least $\frac{1}{2} + x + 1$ by the time the move $(x+1)_{n-x-1}$ occurs. Thus, the bounds on chip n follow by induction.

Now, induct on k. Assume that the position bounds hold for all chips $k' > k$ and $0 \leq x \leq n-1$. We show that the bounds hold for chip k and all values for x from 0 to $n-1$. Chip k can never reach a position less than $\frac{1}{2} + k - n$ at any point in the firing process, and in particular, chip k must be at position at least $\frac{1}{2} + k - n$ immediately preceding move $(n-k-1)_{k-1}$. Thus, the bound holds for $x = 0$. Now, assume that chip k is at position at least $\frac{1}{2} + k - n + x$ immediately preceding firing move $(n-k-x-1)_{k-1}$ for $x < n-k$. By our inductive assumption, all chips with value greater than k must already be at positions greater than $\frac{1}{2} + k - n + x$ when move $(n-k-x-1)_{k-1}$ occurs. Thus, if

chip k is at position $\frac{1}{2}+k-n+x$, then it must be the greatest chip at that position, and because there are at most two chips present for firing move $(n-k-x-1)_{k-1}$, chip k must be fired to the right. Chip k thus must be at position at least $\frac{1}{2}+k-n+x+1$ by the time the next move occurs at that site. If $x < n-k-1$, the next move at site $(n-k-x)$ is move $(n-k-x)_{k-1}$. If $x = n-k-1$, then the next move at site $(n-k-x)$ is 0_k. The same proof applies for $x \geq n-k$, so these position bounds hold for all $2 \leq k \leq n$ and all $0 \leq x \leq n-1$, as desired. □

Lemma 7.3.6 allows us to complete our final result.

Theorem 7.3.7 ([21], Problem 7.14, Part (3))**.** *When the labeled chip-firing process terminates on a system of type B_n (or D_n), with initial configuration ω_n, each chip k, $2 \leq k \leq n$, is at position $k-\frac{1}{2}$.*

Proof. By Lemma 7.3.6, chip k must be at position at least $k-\frac{3}{2}$ by the time move $(k-2)_1$ occurs. Using the same argument as in Lemma 7.3.6, chip k must be at position at least $k-\frac{1}{2}$ immediately following move $(k-2)_1$. Since no firing moves may occur at positions greater than $k-\frac{3}{2}$ after firing move $(k-2)_1$, the final position of chip k must be greater than or equal to $k-\frac{1}{2}$. The only way for every chip to satisfy this condition is if every chip k, $2 \leq k \leq n$ is at position $k-\frac{1}{2}$ in the final configuration, as desired. □

Note that chip 1 may be at position $\pm\frac{1}{2}$, depending on the value of n, but as chip 1 is the smallest chip, all chips will be in sorted order regardless.

Theorem 7.3.7 on labeled chip-firing actually allows us to prove a previously open problem about unlabeled chip-firing. Local confluence modulo the Weyl Group is enough to guarantee that the final chip-configuration consists of 1 chip at $\pm\frac{1}{2}$, 1 chip at $\pm\frac{3}{2}$, ... and 1 chip at $\pm n-\frac{1}{2}$. However, this alone does not guarantee whether those chip locations are positive or negative. Our result, however, does make this guarantee. Other than possibly the chip at $\pm\frac{1}{2}$, all chips must terminate at positive final positions.

Corollary 7.3.8. *Consider a collection of n indistinguishable chips on the line, beginning with all chips at position $\frac{1}{2}$. Let the system evolve using two types of moves:*

(1) *If two chips are at the same site, send one of the chips at that site one unit to the left, and one of the chips at that site one unit to the right.*

(2) *If two chips are at sites with opposite coordinates, send both chips one unit to the right.*

Then the process terminates with 1 chip at every half-integer site from $\frac{3}{2}$ to $n - \frac{1}{2}$. Furthermore, if $n \equiv 0$ or 1 mod 4, then the final position also has a chip at site $-\frac{1}{2}$. If $n \equiv 2$ or 3 mod 4, then the final position also has a chip at site $\frac{1}{2}$.

Proof. Assign labels 1 through n to the chips. Whenever a move of type (2) is performed, any chips at the two firing sites may be chosen to be sent to the right. Whenever a move of type (1) is performed, any two chips at the firing site may be chosen to fire, with the chip with the larger label sent to the right, and the chip with the smaller label sent to the left. This process is identical to root system central-firing of type B_n (or D_n), so by Theorem 7.3.7, the process terminates with 1 chip at every half-integer site from $\frac{3}{2}$ to $n - \frac{1}{2}$. As the addition of labels does not restrict which chips can move at which times, the original unlabeled process must also terminate with chips in positions $\frac{3}{2}$ to $n - \frac{1}{2}$.

Now, note that every firing move of type (1) does not change the sum of the chip positions, while every firing move of type (2) increases the sum of the chip positions by 2. Thus, the difference between the final sum of the chip positions and the initial sum must be even. By local confluence of unlabeled chip-firing, the last chip must be at either site $-\frac{1}{2}$ or site $\frac{1}{2}$.

The initial sum of the chip-positions is equal to $\frac{n}{2}$, and the final sum is equal to $\frac{n^2}{2} - 1$ (if the remaining chip is at $\frac{1}{2}$) or $\frac{n^2}{2}$ (if the remaining chip is at $\frac{1}{2}$). Now, consider 4 cases:

$$\begin{aligned}
n \equiv 0 \bmod 4 \qquad & \tfrac{n^2}{2} - \tfrac{n}{2} = \tfrac{n}{2}(n-1) \text{ (even)} \\
n \equiv 1 \bmod 4 \qquad & \tfrac{n^2}{2} - \tfrac{n}{2} = \tfrac{n-1}{2}(n) \text{ (even)} \\
n \equiv 2 \bmod 4 \qquad & \tfrac{n^2}{2} - 1 - \tfrac{n}{2} = \tfrac{n-2}{2}(n+1) \text{ (even)} \\
n \equiv 3 \bmod 4 \qquad & \tfrac{n^2}{2} - 1 - \tfrac{n}{2} = \tfrac{n+1}{2}(n-2) \text{ (even)}
\end{aligned}$$

Note that as $\frac{1}{2}$ and $-\frac{1}{2}$ differ by 1, the difference involving $\frac{1}{2}$ is even if and only if the difference involving $-\frac{1}{2}$ is odd. Thus, the only way to produce an even difference is if the last chip is at $\frac{1}{2}$ ($n \equiv 0$ or 1 mod 4) or at $-\frac{1}{2}$ ($n \equiv 2$ or 3 mod 4). □

Note that while the final location of the smallest chip depends on n, it does not depend on the choices of firing moves throughout the process. Thus, the final chip positions do not depend on the choices of moves.

In addition to resolving a conjecture about labeled chip-firing, we have used a labeled chip-firing result to prove an unlabeled chip-firing result. This makes Corollary 7.3.8 particularly notable, as it shows the value of labeled chip-firing to the broader field of chip-firing.

7.4 Types C and D

Here, we attempt to provide some insight into root system chip-firing of types C and D, and why the conjectures on sorting based on the parity of the number of chips seem natural. In Figure 7.3, we show the bottom of the move poset for type D_9, with initial weights ω_7 (2 chips at site 0, and 7 chips at site 1) and ω_6 (3 chips at site 0, and 6 chip at site 1). In Figure 7.4, we show the bottom of the move poset for type C_9, with initial weights ω_8 (1 chip at site 0, and 8 chips at site 1), and ω_9 (9 chips at site 1).

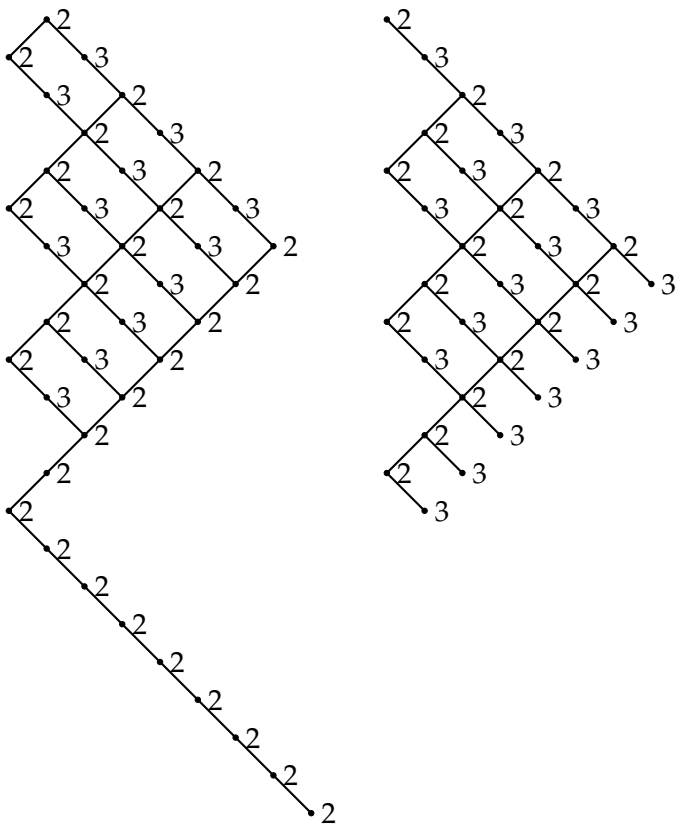


FIGURE 7.3: Pictured are the bottom of the move posets for two different initial configurations on D_9. The left is initial weight ω_7, which sorts, while the right is initial weight ω_6, which does not. Next to each move is the maximum number of chips that can be present for that move. The 3's at the bottom of the picture on the right guarantee that the system does not sort from ω_6, as the last move in the entire firing process always has the potential to produce an inversion. The row of 2's at the bottom of the left picture act as a sort of "bubble sort" pass, fixing the few inversions that remain after the rest of the process is completed.

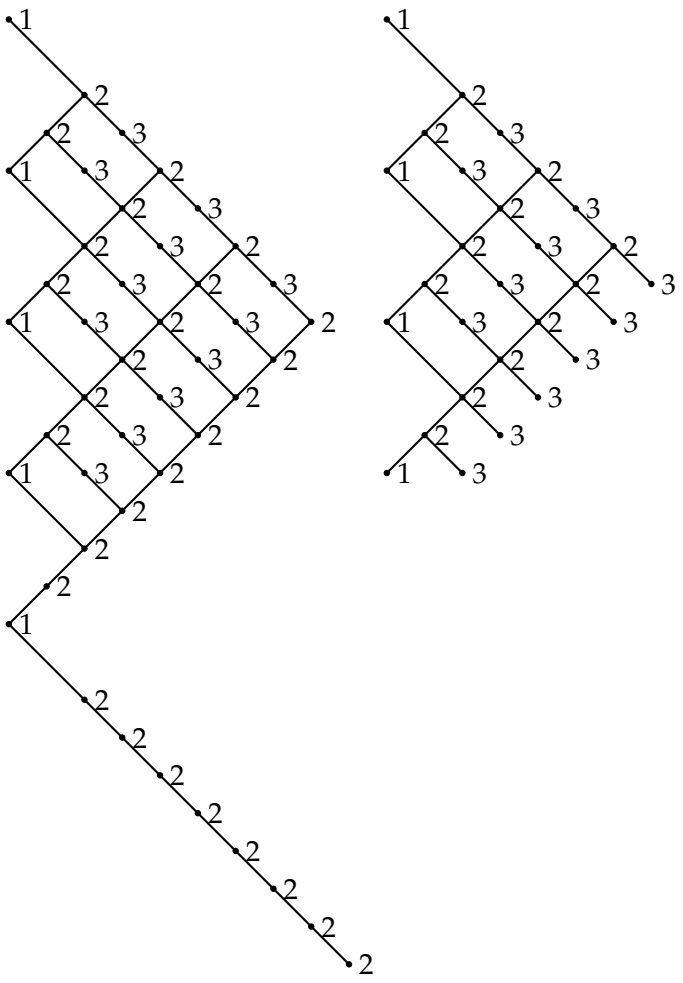


FIGURE 7.4: Pictured are the bottom of the move posets for two different initial configurations on C_9. The left is initial weight ω_8, which sorts, while the right is initial weight ω_9, which does not. Next to each move is the maximum number of chips that can be present for that move. The 3's at the bottom of the picture on the right guarantee that the system does not sort from ω_9, as the last move in the entire firing process always has the potential to produce an inversion. The row of 2's at the bottom of the left picture act as a sort of "bubble sort" pass, fixing the few inversions that remain after the rest of the process is completed. Note that near the end of the process, there can never be more than 1 chip at site 0, so all moves at site 0 must be (c) moves.

In most of the non-sorting cases, as in the examples on the right of Figures 7.3 and 7.4, the last firing move must take place with 3 chips present, thus guaranteeing that at least one possible choice of chips to fire will result in an unsorted final configuration. The sorting cases, as in the examples on the left, have much more structure. The tail at the bottom represents a sort of "bubble sort" pass in which pairs of chips are sorted

correctly two at a time. The other lines of 2's in the half-diamond shape above act in similar ways, while the lines of 3's act in a similar way to type-A labeled chip-firing with an odd number of chips (see Chapter 6). They occur from the outside in, with all but the last taking place with 2 chips present, and the last taking place with 3 chips present. Thus, this structure at the bottom of the Hasse diagram acts to "fix" any unsorted chips that still remain, although the author has not yet proven that the process must result in a sorted final configuration.

We have resolved one of the open problems on Types C and D. Part (5) of Conjecture 7.2.2 is not true. We have shown computationally that central-firing on D_{10} sorts from initial weight 0 (this is also true for central-firing on D_2). Central-firing on D_6 does not sort from initial weight 0, but a specific sequence of things needs to happen in order for the chips to end up out of order. It appears that the diamond-like structure at the bottom of the move poset is not large enough to "fix" every inversion that may arise throughout the central-firing process.

It seems to us that in some sense, the structure at the bottom of the move poset is growing faster than the maximum "unsortedness" of the list, so we conjecture instead that initial weight $\omega = 0$ for $\Phi = D_6$ is actually the only exception to part (6) of Conjecture 7.2.2. Note that for any even $m \leq n-2$, it is possible to reach ω_m from ω_0 through $\frac{m}{2}$ type (d) firing moves. Thus, if central-firing on D_n sorts from initial weight 0, then it must also sort from initial weight ω_m for every $m \leq n-2$ such that m is even.

The bottom of the move posets for the Type C sorting and non-sorting cases look similar to the Type D cases. Note that while unlabeled Type D central-firing is locally confluent in the stronger sense discussed in Chapter 2, unlabeled Type C central-firing is only locally confluent in the weaker sense discussed earlier in this chapter. This is due to the presence of (c) moves. Thus, the configuration poset for Type C central-firing is actually not LLD. However, near the end of the process, only one chip can be at the origin at a time, so any firing move at the origin must be a (c) move. This means that unlabeled Type C central-firing actually is locally confluent (in the stronger sense) near the end of the process, which provides a similar structure in the move poset to the Type

D case. Thus, we suspect that it could be possible to prove sorting for Types C and D at the same time.

This leaves us with the following conjecture for the remaining open problems in central-firing.

Conjecture 7.4.1. *(1) Central-firing is confluent from $\omega \in \Omega \cup \{0\}$ if and only if $\omega \not\equiv \rho$ in P/Q for $\Phi = C_n$.*

(2) Central-firing is confluent from $\omega \in \Omega \cup \{0\}$ for all $\omega \not\equiv \rho$ in P/Q for $\Phi = D_n$, except for the case $\Phi = D_6$, $\omega = 0$.

CHAPTER 8

Mining for Diamonds

8.1 Introduction

One nice property of chip-firing on the 1-dimensional grid graph that is extremely useful in proving results about labeled chip-firing is that the move poset has a simple diamond grid structure at the bottom. In higher dimensional grid graphs, this is generally not the case. Final configurations in higher dimensional grids are much more complex (see [42], [16], and Chapter 5 of [30] for pictures of some of these pictures in 2 dimensions). As a result, their firing move posets are also more complex. However, there are some nonlinear graphs that still have move posets exhibiting a grid structure.

The goal of this chapter is to find a broader set of sufficient conditions on a graph G and unlabeled initial configuration u such that chip-firing on G from configuration u produces a move poset with a diamond grid structure at the bottom. We find a sufficient condition that exists on a much larger class of bipartite graphs.

The class of graphs with the desired diamond property includes two notable types of graphs. The first case involves trees with constant branching number, of which the 1-dimensional grid graph is a special case. The other involves grids in higher dimensional graphs. While an ordinary grid graph does not allow for such a nice poset structure, there are ways to assign multiplicities to each edge in the graph so that the necessary properties hold.

In order to show the existence of these grid structures, we prove a final generalization of the Diamond Lemma from Chapter 3. Consider an undirected bipartite graph with a single vertex denoted as the origin. Each vertex aside from the origin must have an integer number of edges directed away from the origin for every edge directed toward the origin, where that integer is a function only of the distance from the origin.

Ideally, it would be nice to come up with some notion of labeled chip-firing that can be done on either of these classes of graphs, but no such notion is known at this time. Any other globally confluent system that is not locally confluent would also be a nice result of this work.

8.2 Distance-Balanced Graphs

Our first goal is to try to generalize the property that gives the diamond structure to the 1D grid graph. In particular, we are looking for some collection of firing moves at the end of a chip-firing process, in which each move in the collection must occur before certain moves at neighboring sites, and after all other moves at neighboring sites.

In particular, the 1D grid has the following property. Aside from the origin, every vertex has the same number of edges (1) directed toward the origin and away from the origin. We formalize this concept here.

Consider an undirected graph G with a distinguished vertex x, known as the origin. For each nonnegative integer i, let level ℓ_i be the set of vertices at a distance i away from x. As in Chapter 5, given vertices $v \in \ell_i$ and $v' \in \ell_{i+1}$ such that v' is adjacent to v, we say that v' is an **outer neighbor** of v, and v is an **inner neighbor** of v'.

For any nonegative integer i and any vertex v in ℓ_i, define $in(v)$ to be the number of edges from v to ℓ_{i-1}, $out(v)$ to be the number of edges from v to ℓ_{i+1}, and $same(v)$ to be the number of edges from v to other vertices in ℓ_i. Note that there cannot be any edges between v and any other levels, as that would imply that v and one of its neighbors had mutually inconsistent distances from x. Define $c(v)$ to be the number of chips at a given vertex, which may change throughout the process.

We say that G is **integer-distance balanced of order** n if for every $1 \leq i \leq n-1$, there exists a positive integer p_i, referred to as the **edge ratio** of level i, such that for every vertex $v \in \ell_i$, $same(v) = 0$ and $out(v) = p_i \cdot in(v)$. In the special case where all of the p_i values are equal to 1, we say that G is **1-distance balanced of order** n. G is n**-extended** if for every vertex v in ℓ_n, $same(v) = 0$ and $out(v) > 0$.

Given an undirected graph G that is integer-distance balanced of order n and n-extended, define a quantity of chips s_n as

$$s_n = \sum_{i=1}^{n} \sum_{v \in \ell_i} in(v)$$

This is enough chips to give each site v in levels ℓ_1 through ℓ_n a number of chips equal to the number of edges from v to its inner neighbors. We would like to show that this system behaves in very similar ways to a traditional chip-firing system on the line with an even number of chips. We will first show that an initial configuration u consisting of s_n chips at the origin leads to a simple final configuration.

Our proofs make use of a slightly modified version of the firing process. As chip-firing is locally confluent, the final configuration does not depend on the order in which sites are fired. As a result, if we can reach a stable configuration after any terminating sequence of firing moves, then the resulting stable configuration must be the unique final configuration of the process. We perform the process in such a way that whenever a node is fired at level i, all nodes at level i must simultaneously be fired. Since any two nodes in the same level must always receive chips from the same direction at the same time, fire at the same time, and have the same ratio of $in(v)$ to $out(v)$, with no firing moves sending chips within the same level, we ensure that for any two nodes v_1 and v_2 in the same level i ($1 \leq i \leq n-1$), we must always have $\frac{c(v_1)}{in(v_1)} = \frac{c(v_2)}{in(v_2)}$, $\frac{c(v_1)}{out(v_1)} = \frac{c(v_2)}{out(v_2)}$ and $\frac{c(v_1)}{deg(v_1)} = \frac{c(v_2)}{deg(v_2)}$ throughout the process.

Since any two nodes at the same level have the same ratio of chips to outgoing edges throughout the process, this process only terminates when no individual site can fire, so the resulting configuration is $stab(u)$. In particular, this implies that the process terminates with all sites in a given level having the same ratio of chips to outgoing edges.

Let r_i be the ratio $\frac{c(v)}{in(v)}$ for each vertex v in level ℓ_i, and let

$$d_i = \frac{\sum_{j \geq i} \sum_{v \in \ell_j} c(v)}{\sum_{v \in \ell_i} in(v)}$$

be the number of chips in all levels greater than or equal to i, divided by the total number of edges from ℓ_{i-1} to ℓ_i. Let r_i^{max} and d_i^{max} be the maximum values that r_i and d_i, respectively, can take on throughout the process.

Lemma 8.2.1. *For all $i \geq 2$, we have that $d_i^{max} \leq \frac{d_{i-1}^{max}-1}{k_{i-1}}$.*

Proof. Consider the first time that $d_i = d_i^{max}$. in order for this to occur, there must be a firing move that sends chips from level ℓ_{i-1} to each site at level ℓ_i. Prior to these firing moves, we must have $d_{i-1} \leq d_{i-1}^{max}$. Performing these firing moves must send at least $in(v)$ chips from each site v at level ℓ_{i-1} back to level ℓ_{i-2}, so after these firing moves, we must have $d_{i-1} \leq d_{i-1}^{max} - 1$. Thus, the number of chips at levels at least $i-1$ (and thus, at levels at least i, must be at most $(d_{i-1}^{max} - 1) \sum_{v \in \ell_{i-1}} in(v)$ when $d_i = d_i^{max}$. This implies that

$$\begin{aligned} d_i^{max} &\leq \frac{(d_{i-1}^{max} - 1) \sum_{v \in \ell_{i-1}} in(v)}{\sum_{v \in \ell_i} in(v)} \\ &= \frac{(d_{i-1}^{max} - 1)}{p_{i-1}} \end{aligned}$$

as desired. □

Lemma 8.2.2. *For a process beginning with*

$$s_n = \sum_{i=1}^{n} \sum_{v \in \ell_i} in(v)$$

chips at the origin, and any m satisfying $1 \leq m \leq n$, we must have

$$d_m^{max} \leq 1 + \sum_{i=m-1}^{n-1} \prod_{j=m}^{i} p_j$$

Proof. We have

$$s_n = \sum_{i=1}^{n} \sum_{v \in \ell_i} in(v)$$

$$= \sum v \in \ell_1 in(v) \Big(\sum_{i=0}^{n-1} \prod_{j=1}^{i} p_j \Big)$$

Now, we can proceed by induction. We must have $d_1^{max} \leq \frac{s_n}{\sum_{v \in \ell_1} in(v)}$, which is equal to $\sum_{i=0}^{n-1} \prod_{j=1}^{i} p_j$.

Now, if the formula holds for a particular value of m, then we can apply the inequality from Lemma 8.2.1.

$$d_{m+1}^{max} \leq \frac{d_m^{max} - 1}{p_m}$$

Plugging in the assumed bound for d_m^{max} and applying the inequality removes the 1 from the beginning of the expression, and removes a factor of k_m from each product, leaving us with the expression

$$d_{m+1}^{max} \leq 1 + \sum_{i=m}^{n-1} \prod_{j=m+1}^{i} p_j$$

as desired.

□

Note that the bound determined above is equal to the number of shortest paths from v to ℓ_n for any vertex v in ℓ_m. We are now ready to prove our result on the final configuration of the chip-firing process.

Lemma 8.2.3. *Consider an integer-distance balanced, n-extended graph G. Consider an initial chip configuration consisting of*

$$s_n = \sum_{i=1}^{n} \sum_{v \in \ell_i} in(v)$$

chips at the origin, and 0 chips elsewhere. Then the stabilization of this chip configuration consists of $in(v)$ chips at every site v from ℓ_1 through ℓ_n.

Proof. Induct downward from n, proving that each site v at each level from 1 to n must have $in(v)$ chips.

By Lemma 8.2.2, no chips may ever reach a level greater than n. Suppose for contradiction that the final configuration also has $r_n = 0$. Then levels 0 through $n-1$ have s_n chips at $s_{n-1}+1$ sites with a total of $s_n + s_{n-1}$ outgoing edges, so at least one site is ready to fire. This contradicts that this is the final configuration, so the final configuration must have $r_n \geq 1$. By Lemma 8.2.2, we also have $r_n \leq r_n^{max} \leq d_n^{max} \leq 1$, so we must have $r_n = 1$ in the final configuration.

Now, suppose that each site v at levels ℓ_i, $m < i \leq n$ has exactly $in(v)$ chips. This implies that the number of chips at all levels less than or equal to m is

$$s_n - \sum_{i=m+1}^{n} \sum_{v \in \ell_i} in(v) = s_m$$

By the same argument above, we get that $r_m \geq 1$. Again by Lemma 8.2.2, we have that

$$d_m^{max} \leq 1 + \sum_{i=m-1}^{n-1} \prod_{j=m}^{i} p_j$$

However, we already have that

$$d_m \geq \sum_{i=m-1}^{n-1} \prod_{j=m}^{i} p_j$$

from our inductive assumption. r_m must be at most 1, so it must be exactly 1.

Thus, by induction, the final configuration must have $in(v)$ chips at each site v from level ℓ_1 to level ℓ_n, as desired.

□

Before deriving the number of firing moves at each site, define s_n^m:

$$s_n^m = \sum_{i=m}^{n} \prod_{j=m}^{i-1} p_j$$

Just as s_n provides a number of chips that can fill levels 1 through n with one chip per inward edge, s_n^m has a similar interpretation in terms of inward edges. s_n^m is the number of inward edges at levels $m+1$ through n, divided by the number of inward edges at level m.

Also define f_n to be the number of firing moves to completion of a node in ℓ_n.

Lemma 8.2.4. *Consider an integer-distance balanced, n-extended graph G. Consider an initial chip configuration consisting of*

$$s_n = \sum_{i=1}^{n} \sum_{v \in \ell_i} in(v)$$

chips at the origin, and 0 chips elsewhere. Then for each m, the number of firing moves to stabilization at each site in ℓ_m is equal to

$$\sum_{i=m+1}^{n} s_n^i$$

Proof. The number of chips that end up in levels m through n is equal to the number of chips received by level m from level $m-1$, minus the number of chips sent from level m back to level $m-1$. This gives us the recurrence relation

$$s_n^m \sum_{v \in \ell_m} in(v) = f_{m-1} \sum_{v \in \ell_m} in(v) - f_m \sum_{v \in \ell_m} in(v)$$

$$f_{m-1} = f_m + s_n^m$$

Furthermore, since no sites at level n may fire during the process, we have $f_n = 0$. Starting with this initial condition and applying the recurrence relation gives the following expression for f_m:

$$f_m = \sum_{i=m+1}^{n} s_n^i$$

as desired. □

While a closed form for the number of firing moves at each site is generally difficult to define (as the s_n^i values depend on each of the p_j's), there are closed form expressions in a few of the cases that we care about. When all of the p_j values are equal to a constant p (as would be the case in a tree with fixed degree), this summation follows the generating function $g(x) = \frac{x}{(1-px)(1-x)^2}$, producing a number of firing moves $f_j = \frac{l^{j+1}-(p-1)j-p}{(p-1)^2}$ where $j = n - m$ is the distance between a vertex and the greatest nonempty level.

Now that we have information about the final configuration and number of firing moves, we can move on to proving our main result about the firing moves at the end of the process. As in the case of firing in one dimension, let k_j denote the j^{th} to last firing move at site k.

This brings us to our newest generalization of the Diamond Lemma, which provides a similar grid structure to graphs that do not exist on the line.

Theorem 8.2.5 (Diamond Lemma for nonlinear graphs)**.** *Consider an integer-distance balanced, n-extended graph G. Consider an initial chip configuration consisting of*

$$s_n = \sum_{i=1}^{n} \sum_{v \in \ell_i} in(v)$$

chips at the origin, and 0 chips elsewhere. Then the following must hold.

(1) Given site k at level ℓ_i, $0 \leq i \leq n-1$, outer neighbor k'' of k, and each j such that $1 \leq j \leq n - i$, move k_j must take place after move k''_j, if site k'' fires at least j times.

(2) Given site k at level ℓ_i, $1 \leq i \leq n$, inner neighbor k' of k, and each j such that $1 \leq j \leq n - i$, move k_j must take place after move k'_{j+1}.

Furthermore, all such moves must take place with exactly enough chips to fire.

Proof. We first show that move x_1, where x is the origin, must take place after all of the moves at its neighbors. The origin ends with 0 chips, so it cannot receive any additional chips after its last firing move. Thus, the last firing move at x must be after all firing moves at each of its neighbors, and since it has no chips remaining after firing, it must have exactly enough chips to fire.

Now, induct on the level to show that the last move at each non-root site must take place after the last move at each of its outer neighbors, and after the second-to-last move at each of its inner neighbors. Consider a site k at level ℓ_i, such that we assume k_1 occurs before k'_1 for each inner neighbor k'. Thus, k receives $in(k)$ chips from its inner neighbors after its last firing move. Since k terminates with $in(k)$ chips, this means that no other moves at neighboring sites may occur after k_1. In particular, k_1 occurs after k''_1 for outer neighbor k'' of k, and k_1 occurs after k'_2 for each neighbor k' of k. Furthermore, since k receives $in(k)$ chips after firing, it must have 0 chips left when k_1 is complete, so k_1 must occur with exactly enough chips to fire. Thus, by induction on i, the result holds for the last move at each site.

Finally, induct on $j + i$, for firing move k_j at level ℓ_i. Consider a move k_j, which we assume occurs before move k'_j at each inner neighbor k', and before move k''_{j-1} at each outer neighbor k'' of k, which in turn all take place before move k_{j-1}. Between the time that k_j is about to occur and the time that k_{j-1} is about to occur, site k must lose $in(k)(1 + p_i)$ chips due to firing move k_j, and gain at least $in(k)(1 + p_i)$ chips due to firing move k'_j for each inner neighbor k', and firing move k''_{j-1} for each outer neighbor k''. Since move k_{j-1} has exactly enough chips to fire, move k_j must also have at most (and thus exactly) enough chips to fire. Furthermore, no other moves may take place at any of k's neighbors between moves k_j and k_{j-1}. In particular, the moves k'_{j+1} for each inner neighbor k' and k''_j for each outer neighbor k'' must occur before k_j. The result follows by induction. □

8.3 Bipartite Graphs

Our condition that vertices in the same level of a graph cannot be connected forces us to consider only bipartite graphs. While most bipartite graphs are not integer-distance balanced, it turns out that a large class of bipartite graphs can be modified to become integer-distance balanced, with edge ratios of our choosing, simply by modifying the multiplicities of each edge.

Lemma 8.3.1. *Consider a bipartite graph G with distinguished vertex x, and a sequence of positive integers $p_1, p_2, \ldots p_{n-1}$. If $\ell_1, \ell_2, \ldots \ell_n$ are finite, and every vertex in $\ell_1, \ell_2, \ldots \ell_{n-1}$ is on a path of length n from x to ℓ_n, then it is possible create a graph G' such that*

(1) G' has the same vertices as G.

(2) For every edge $e = (u, v)$ in G, there is a nonzero integer number of edges in G' between u and v.

(3) G' is integer-distance balanced of order n.

(4) For each i from 1 to $n - 1$, the edge ratio at level ℓ_i in G' is equal to p_i.

Proof. Given G, we want to reassign nonzero multiplicities to each edge in the first n levels in order to provide the integer distance-balance condition with the correct edge ratios. We first find an assignment of rational edge multiplicities, which we then convert to integer multiplicities by multiplying by an appropriate constant.

We provide a construction of such a set of rational multiplicities. First, assign 1 edge from x to each vertex in ℓ_1. Then, for each i from 1 to $n - 1$ and each v in ℓ_i, assign an edge multiplicity of $\frac{p_i in'(v)}{out(v)}$ to each edge from v to ℓ_{i+1}, where $in'(v)$ is the number of from v to ℓ_{i-1} in G', and $out(v)$ is the number of edges from v to ℓ_{i+1} in G.

This construction ensures that each edge from x has equal weight, and that for each i and each vertex v in ℓ_i, the ratio of $out(v)$ to $in(v)$ in G' is equal to p_i. Furthermore, as each number of edges is derived by multiplying finitely many ratios of integers, and since there are finitely many such edge values, it is possible to multiply each edge multiplicity by the greatest common denominator of all of the values to obtain a new graph G'' with integer values satisfying the same edge ratios. Thus, G" is integer-distance balanced with the correct edge ratios, as desired. □

While Theorem 8.2.5 helps us to find a diamond-like grid structure on a broader class of graphs with certain edge ratios, Lemma 8.3.1 allows us to extend that to much larger classes of bipartite graphs. As long as a bipartite graph does not have "leaf nodes"

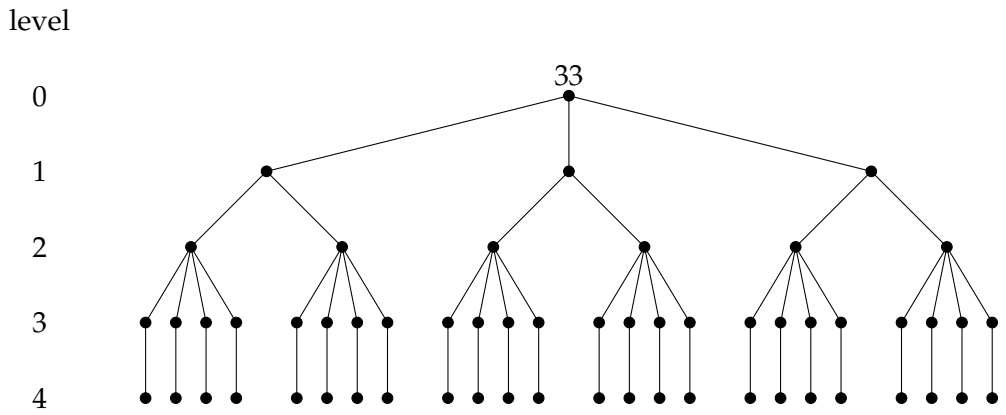


FIGURE 8.1: An example of a tree that is integer-distance balanced of order 3 and 3-extended.

with no outward edges at distance less than n from the origin, we can adjust the edge multiplicities of the graph to produce a new graph that is integer-distance balanced.

We can now consider two important classes of graphs, that are either already integer-distance balanced, or can be made so by Lemma 8.3.1

8.3.1 Trees

Perhaps the most straightforward examples of graphs satisfying the conditions of this chapter are certain types of trees. In particular, any rooted tree in which the number of children is constant across each level is an example of an integer-distance balanced graph. The 1D grid is a special case of this in which the origin has 2 children and each other vertex has a single child. If the initial number of chips at the root is chosen carefully, Theorem 8.2.5 guarantees a certain grid structure to the firing moves at the end of the process.

Figure 8.1 shows an example of a tree that is integer-distance balanced of order 3 and 3-extended. If we place 33 chips at the root of the tree and fire until completion, a few things are guaranteed. First, by Lemma 8.2.3, the process terminates with 1 chip at every site at levels 1 through 3 in the tree. By Lemma 8.2.4, the origin fires 17 times, every site in level ℓ_1 fires 6 times, and every site in level ℓ_2 fires once.

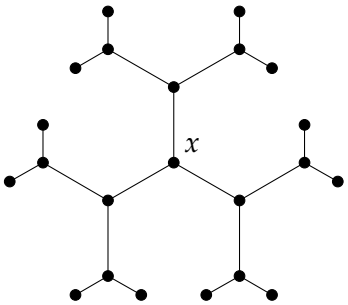


FIGURE 8.2: A neighborhood of an infinite tree in which every site has degree 3. The origin x is labeled.

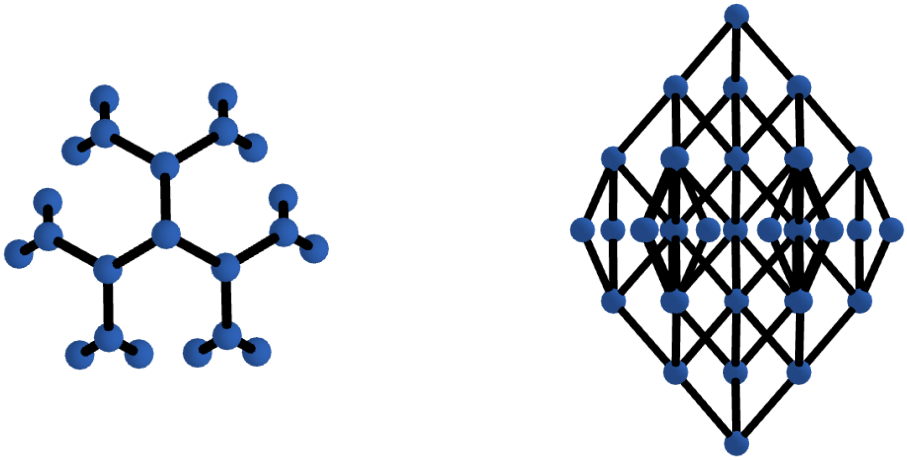


FIGURE 8.3: Place 21 chips at the origin of the graph in Figure 8.2, and fire until completion. Left: a top view of the grid at the bottom of the move poset. Right: a side view of the grid.

Theorem 8.2.5 imposes a grid structure on the last moves in the process. The last 3 moves at the origin, the last 2 moves at each site in ℓ_1, and the last move at each site in ℓ_2 must each occur with exactly enough chips to fire. Furthermore, the last move at the origin must occur after the last move at each site in ℓ_1, which must occur after the last move at each site in ℓ_2 and the second to last move at the origin, etc.

An example of a tree with degree 3 at every vertex is shown in Figure 8.2. We assume that the tree is infinite, but levels ℓ_0 through ℓ_3 are shown. If 21 chips are placed at the origin, and chip-firing is performed to completion, then by Theorem 8.2.5, there is a grid structure at the bottom of the move poset for the firing process. Two views of this grid

are shown in Figure 8.3. Because the graph exists in 2 dimensions, 3 dimensions are needed to show the grid structure. The top view of the grid, which looks like the graph itself, is shown to the left of Figure 8.3, while a side view is shown to the right of the figure.

8.3.2 Higher Dimensional Grid Graphs

While graphs without cycles are often much nicer to analyze and have nicer structures in their move posets, our results on bipartite graphs allow us to find grid structures in more complicated graphs. In particular, any d-dimensional grid graph is a bipartite graph satisfying the conditions of Lemma 8.3.1. Thus, every d-dimensional grid graph can be given edge multiplicities to make it integer-distance balanced of order n for any n (it can be done for all n if we remove the restriction that edge multiplicities and chip quantities have to be integers). In particular, we can assign edge multiplicities so that a given grid graph is 1-distance balanced.

In Figure 8.4, Lemma 8.3.1 is applied to change the edge multiplicities on all edges within 5 units of the origin of a 2D grid graph. At each vertex v other than the origin, the number of edges from v to vertices closer to the origin is the same as the number of edges from v to vertices farther from the origin.

This graph is 1-distance balanced of order 5, and 5-extended, so the conditions of Lemma 8.2.3, Lemma 8.2.4, and Theorem 8.2.5 are met for n up to 5. If we place 3240 chips at the origin (equal to $\sum \left(in(v)\right)$ over all vertices v shown other than the origin), then the final configuration will have $in(v)$ chips at each site v shown other than the origin, and a collection of firing moves at the end of the process will follow a grid structure, with each such move occurring with exactly enough chips to fire. Two views of the move at the bottom of the move poset are shown in Figure 8.5.

8.3.3 Labeled Chip-Firing

The purpose of the original diamond lemma was to aid in the proof that labeled chip firing on the line sorts from certain initial configurations. In this chapter, we extend the

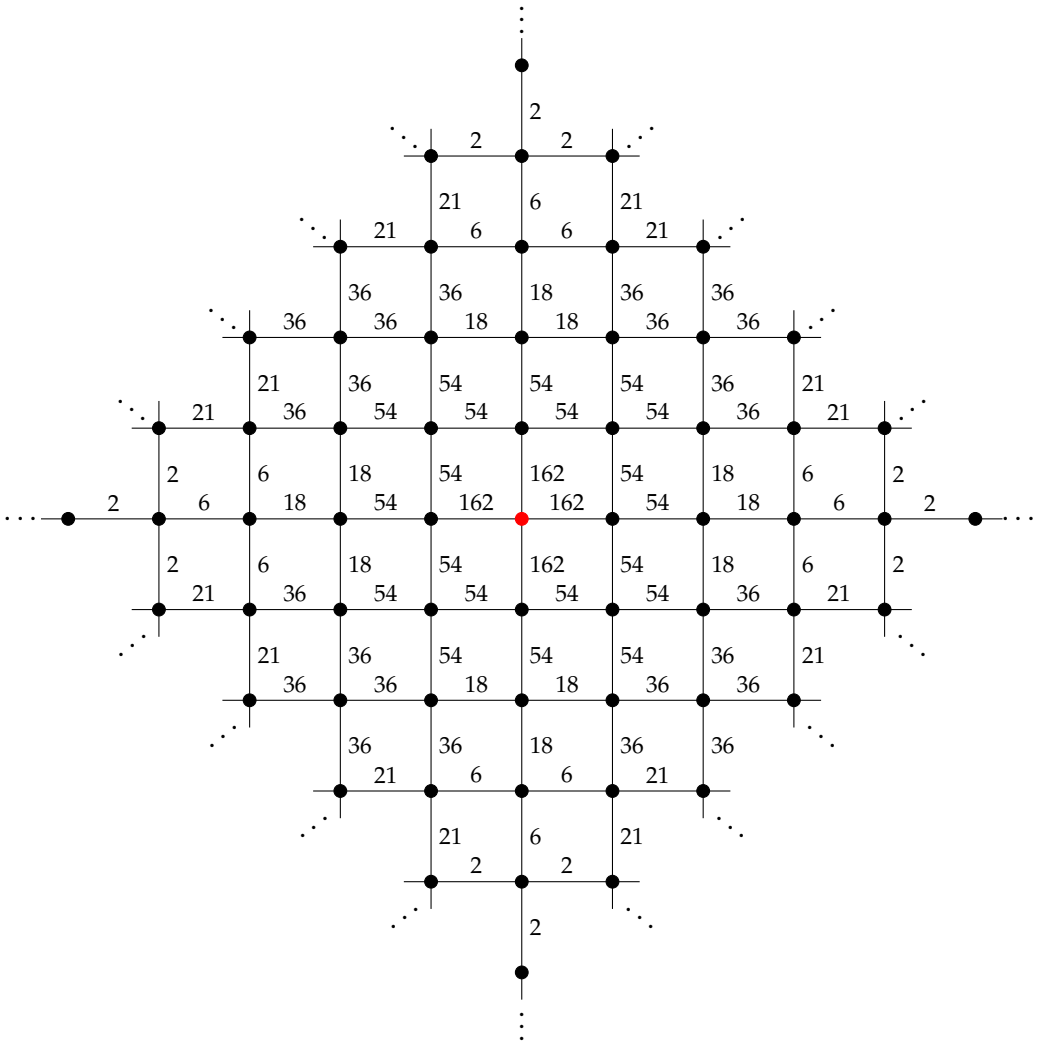


FIGURE 8.4: A collection of edge multiplicities generated by Lemma 8.3.1 for a 2D grid graph. The origin is marked in red, and all edges are labeled with their multiplicities. Note that for every vertex v other than the origin, the number of edges from v to vertices closer to the origin is the same as the number of edges from v to vertices farther from the origin. We assume that the grid extends to infinity with multiplicity 1 at every edge not shown.

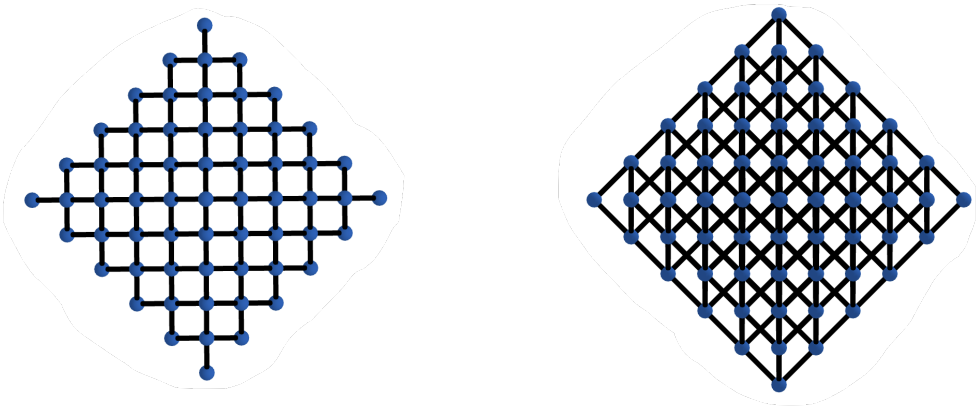


FIGURE 8.5: Place 3240 chips at the origin of the graph in Figure 8.4, and fire until completion. Left: a top view of the grid at the bottom of the move poset. Right: a side view of the grid.

diamond lemma to apply to graphs other than the line, but it is unclear whether these techniques might be able to apply to a problem resembling labeled chip-firing.

If we choose any procedure for labeling chips and then distributing those labels via firing, we have some assurance about local confluence near the end of the process. Because the "diamond" firing moves must occur with exactly the right number of chips to fire, it is impossible for two neighbors to be able to fire at the same time during that stage of the firing process. As a result, there is no concern that changing the order of the firing moves would change the way that labeled chips are fired during the diamond stage of the process, so local confluence is assured.

However, it is not clear if there is a natural labeled chip-firing procedure on a tree or a higher-dimensional grid. We leave open the question of how to construct a labeled chip-firing system on any of the classes of graphs covered in this chapter. As a broader question, it would be interesting if there were any way of using the nonlinear diamond lemma to construct a process that is globally confluent but not locally confluent.

CHAPTER 9

Asymptotic Run Time

9.1 Introduction

Labeled chip-firing is a notoriously slow sorting algorithm. Starting with $n = 2m$ chips that the origin, the number of firing moves required to stabilize (and thus, to sort the configuration) is $O(n^3)$.

In a 2016 talk [26], Hopkins provides a method for improving the run time of labeled chip-firing by performing labeled chip-firing on a slightly modified version of the line graph.

This modified graph is shown in Figure 9.1. The graph exists on the nonnegative integers. Each vertex has a self-loop and an edge directed to the right, so each firing move involves 2 chips. When 2 labeled chips are fired, the smaller chip is sent along the self-loop back to the original vertex, while the larger chip is sent one unit to the right. An example of a labeled chip-firing process on this graph is shown in Figure 9.2.

This process leaves the n chips in sorted order after $\binom{n}{2}$ moves. This improves the asymptotic run time of labeled chip-firing from $O(n^3)$ to $O(n^2)$. Hopkins leaves open the question of whether there is a method using labeled chip-firing to sort chips in

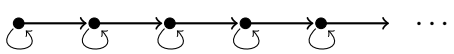


FIGURE 9.1: A graph for more efficient labeled chip-firing.

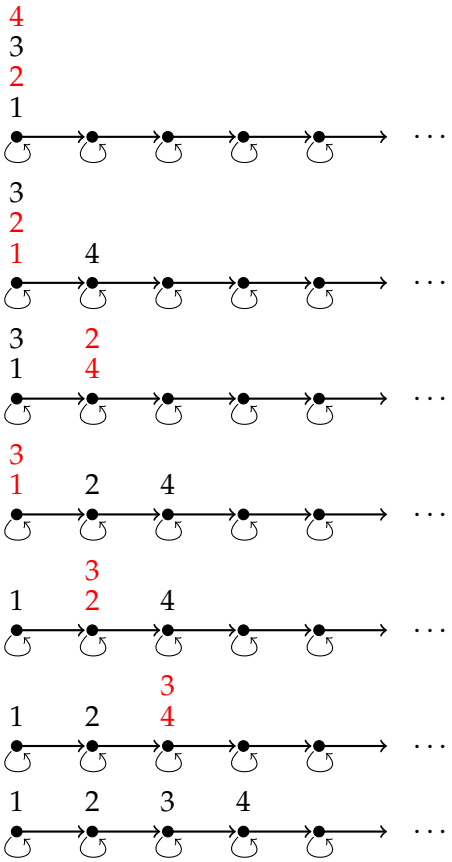


FIGURE 9.2: An example of labeled chip-firing with 4 chips on the modified graph. In each picture, the 2 chips about to be fired are written in red.

$O(n \log(n))$ time, which is the theoretical minimum asymptotic run time for a comparison-based sorting algorithm.

In [31] (as well as Chapter 3 of this thesis), Klivans and Liscio prove that labeled chip-firing on a line sorts by analyzing a collection of $\frac{n^2}{4}$ moves near the end of the process, and showing that some weak bounds on the positions of the chips prior to those moves are sufficient to guarantee sorting. In Section 9.3, this proof is used to show that labeled chip-firing on an undirected 1D grid graph can also guarantee $O(n^2)$ sorting time, if we choose a different initial configuration and do a small amount of preprocessing.

In order to achieve $O(n \log(n))$ sorting, we use a new graph, or more accurately, an infinite family of graphs. After defining what it means to have a labeled chip-firing process on a non-linear graph, we are able to construct graphs with the goal of sorting more efficiently.

Chip-firing is limited in its capacity to sort due in part to the fact that elements can only be compared if they appear in specific, predetermined locations. In most popular sorting algorithms, the elements being compared are chosen throughout the process based on things that have happened so far.

Fortunately, there is a large class of sorting algorithms that fix which elements are to be compared ahead of time: sorting networks. By showing that every sorting network can be implemented using a general labeled chip-firing process, we can use existing sorting networks to produce general labeled chip-firing processes with desired run time bounds.

Many well-known sorting algorithms, including Bubble Sort and Insertion Sort, can be implemented using sorting networks [32], although these both require $O(n^2)$ comparators to sort n elements. Several practical sorting networks have $O(n \log^2(n))$ comparators at a depth of $O(\log^2(n))$, allowing for sorting in $O(n \log^2(n))$ time on a single processor, or $O(\log^2(n))$ time in parallel. These sorting networks include Bitonic mergesort, Batcher odd-even mergesort [5], Shell sort [47], and the pairwise sorting network [43].

In 1983, Ajtai, Komlós, and Szemerédi achieved a theoretically optimal sorting run

time using a sorting network now known as the AKS network [1]. It uses $O(n\log(n))$ comparators, and can thus run in $O(n\log(n))$ time on a single processor, or $O(\log(n))$ time in parallel. While this algorithm is theoretically optimal, it is much slower in practice than some of the $O(n\log^2(n))$ algorithms, and so is not commonly used.

In Section 9.4 we show that any sorting network can be implemented using general labeled chip-firing with the same asymptotic run times on a single processor and in parallel.

9.2 Preliminaries

We first need to define what it means to perform labeled chip-firing on a more general class of graphs.

9.2.1 General Labeled Chip-Firing Processes

Define a **general labeled chip-firing process** R with n chips as follows. The process is performed on a directed graph $G = (V, E)$, where V is finite or countable. There is a finite subset $V_i \subseteq V$ of **input vertices** where chips are placed at the beginning of the process, and a subset $V_f \subseteq V$ of **output vertices** where chips are at the end of the process. V_i may or may not overlap with V_f. R has an **initial unlabeled configuration** $c_i : V_i \to \mathbb{Z}^+$ representing the number of chips at each site at the beginning of the process, and a **final unlabeled configuration** $c_f : V_f \to \mathbb{Z}^+$ representing the number of chips at each site at the end of the process. As there are n chips in R, we require $\sum_{v\in V_i} c_i(v) = \sum_{v\in V_f} c_f(v) = n$.

Each vertex $v \in V$ has finitely many outgoing edges, labeled $e(v,1), e(v,2), \ldots$ $e(v, \text{outdeg}(v))$. The outgoing edges from v are ordered according to a relation $\leq_v$ such that $e(v,i) \leq_v e(v,j)$ if $i \leq j$. Furthermore, the vertices in V_f are totally ordered by some relation $\leq_f$. The process is **bounded** if there exists a constant k such that for all $v \in V$, $\text{outdeg}(v) < k$.

At the beginning of the process, n chips, labeled with real numbers, are placed at vertices in V_i, such that $c_i(v)$ chips are placed at each vertex $v \in V_i$. At any time, a vertex

v can fire if the number of chips at v is greater than or equal to outdeg(v). When v fires, outdeg(v) of the chips at v are chosen to fire. Of these outdeg(v) chips, the chip with the smallest label is sent along edge $e(v, 1)$, the chip with the second smallest label is sent along $e(v, 2), \ldots$ and the chip with the largest label is sent along $e(v, \text{outdeg}(v))$, to the respective neighbors of v at the opposite ends of the edges. The process continues until no site can fire, with all chips at sites in V_f (we are not concerned with non-terminating labeled chip-firing processes, or with processes that terminate with chips not in V_f).

Given a general labeled chip-firing process R, R **sorts** if for any two sites $v_1, v_2 \in V_f$ such that $v_1 <_f v_2$, any chip $x_1 \in v_1$, and any chip $x_2 \in v_2$, we must have $x_1 \leq x_2$, regardless of the choices of move order or which chips are fired by each move. R **strongly sorts** if R sorts, and, in addition, $c_f(v) \leq 1$ for all $v \in V_f$.

We focus on bounded, strongly sorting processes to ensure that all of the "sorting work" is done by the chip-firing, and not by some other loophole in the system. Without the boundedness constraint, it would be possible to simply have one firing move with n chips that sorts all of the chips. Without strong sorting, it would be possible to have all of the chips end at a single site in V_f and technically be sorted by our definition. Both would technically be general labeled chip-firing processes that sort, but neither is really in the spirit of what we are trying to find.

Define a **labeled chip-firing family** to be a function R that maps each positive integer n to a sorting general labeled chip-firing process with n chips. This family is **bounded** if there exists a constant k such that for every positive integer n and every vertex v in $R(n)$, the outdegree of v is less than or equal to k. We are specifically concerned with families in which $k = 2$, which align most closely with the labeled chip-firing processes that we usually consider.

9.2.2 Sorting Networks

In Section 9.4, we use general labeled chip-firing processes to represent sorting networks. A **sorting network** is a network consisting of n **wires**, which carry values, and

m **comparators**, which compare the values on pairs of wires and sometimes switch elements between the wires. The wires are labeled from 1 to n, and initially a value is placed on each wire. Each comparator is connected to 2 predetermined wires, and the comparators act in a predetermined order. When a comparator acts, the comparator compares the values along its 2 incident wires. If the value on the smaller numbered wire is larger than the value on the larger numbered wire, the comparator switches the values between the two wires. Otherwise, the comparator does nothing.

The comparators are placed in such a way so that after all of the comparators act on the wires, the values are guaranteed to be sorted from the smallest value on wire 1, to the largest value on wire n. If all comparators act sequentially, the run-time of implementing a sorting network is $O(m)$, where m is the number of comparators. If comparators are allowed to act in parallel, the run time of implementing the network is on the order of the **depth** of the network, which measures the maximum number of comparators that an input can encounter when traveling through the network [15].

A sorting network is often drawn by drawing the wires from left to right, and drawing the comparators as vertical line segments connecting pairs of wires. The comparators act one at a time from left to right. An example of a sorting network drawing, as well as the result of that sorting network being applied to a list of numbers, is shown in Figure 9.3.

9.3 $O(n^2)$ Sorting on the Line

While we certainly do not intend to recommend labeled chip-firing as an efficient sorting algorithm, there are some improvements that we can make. First, our proof of sorting for labeled chip-firing provides a natural suggestion for a much more efficient, $O(n^2)$ sorting algorithm.

We define an algorithm, which we call the Diamond Sorting Algorithm (DiamondSort) as follows. Given a list L of n numbers:

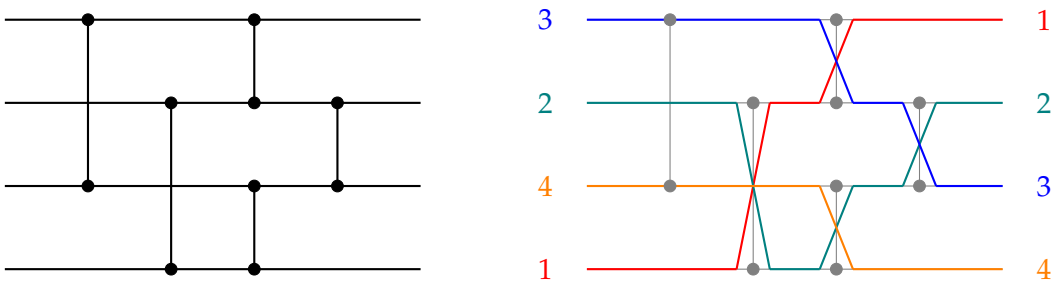


FIGURE 9.3: **Left:** An example of a 4-element sorting network. The 4 wires are represented by horizontal lines, while the 5 comparators are represented by vertical line segments between pairs of wires. **Right:** The same sorting network applied to the elements 1, 2, 3, and 4. The first comparator does not switch elements 3 and 4, which are already in order, but the rest of the comparators switch out-of-order elements. Figure adapted from [17].

(1) Find the median of the n numbers.

(2) Divide L into two sublists L_1 and L_2, such that L_1 contains the smallest $\lfloor \frac{n}{2} \rfloor$ elements, and L_2 contains the largest $\lceil \frac{n}{2} \rceil$ elements.

(3) Label n chips with the elements of L_1 and L_2.

(4) Place one chip with an element from L_1 at each site from $-\lfloor \frac{n}{2} \rfloor + 1$ to 0. Place one chip with an element from L_2 at each site from 0 to $\lceil \frac{n}{2} \rceil - 1$.

(5) Perform labeled chip-firing on the resulting chip configuration until completion.

Our first theorem of this chapter is to prove the correctness of this sorting algorithm.

Theorem 9.3.1. *DiamondSort terminates with all chips in sorted order, and runs in time $O(n^2)$, where n is the number of chips.*

Proof. We check the run time first. Steps 1-4 can all be done in $O(n)$ time.

Now, let $m = \lfloor \frac{n}{2} \rfloor$. If n is even, the number of firing moves at each site k is equal to $m - |k|$ for each k from $-m+1$ to $m-1$, for a total of $m^2 = \frac{n^2}{4}$ firing moves. If n is odd, the number of firing moves at each site k is equal to $m - k + 1$ for each $1 \leq k \leq m$, and $m + k$ for each $-m + 1 \leq k \leq 0$, for a total of $m(m+1) = \frac{n^2-1}{4}$ firing moves. Each firing move can be performed in constant time, so step 5 takes $O(n^2)$ time, and the whole process takes $O(n^2)$ time.

We can now show that this process sorts. If n is even, the process terminates with chips at every site from $-m$ to m except for the origin. If n is odd, the process terminates with chips at every site from $-m$ to $m+1$ except for 1. By Lemma 5.3.1, every firing move k_j throughout the entire process has to satisfy the following:

- For each outer neighbor k' of k, if site k' fires at least j times, then firing move k_j takes place after firing move k'_j.
- For each inner neighbor k' of k, if site k' fires at least $j+1$ times, then firing move k_j takes place after firing move k'_{j+1}.
- Firing move k_j takes place with exactly 2 chips present at site k.

Now, for each $1 \leq k \leq \lceil m \rceil$, the k^{th} smallest chip begins at position less than or equal to $k-1$ (in fact, it begins at position less than or equal to 0). For each $1 \leq k \leq \lfloor m \rfloor$, the k^{th} largest chip begins at position greater than or equal to $1-k$ (in fact, it begins at position greater than or equal to 0). Thus, by Lemma 3.2.9 and Theorem 3.2.10 (n even) and Lemma 5.3.4 and Theorem 5.3.8 (n odd), the chips must end the process in sorted order, as desired. □

This method improves the run time by a factor of n by using the fact from our proof of Theorem 3.2.10 that most of the work in sorting via chip-firing is accomplished during a relatively small collection of moves near the end of the process. This algorithm is also capable of running in $O(n)$ time when the firing moves are performed in parallel.

However, this method leaves a bit to be desired. First, the run time of $O(n^2)$ is better, but it still fails to achieve the theoretical lower bound of $O(n\log(n))$ for comparison sort algorithms.

Second, the approach also requires some preprocessing. While this preprocessing is efficient (it requires $O(n)$ time to find the median, use it to separate the list into two halves, and then place chips on the line accordingly), it still requires a decent amount of non-chip-firing work. We address these issues in Section 9.4.

9.4 $O(n\log(n))$ Sorting Using Sorting Networks

We provide a method for expressing any sorting network as a general labeled chip-firing process. Furthermore, the process that we create is strongly sorting and bounded, with maximum outdegree equal to 2.

Consider a sorting network N with n wires (labeled w_1 through w_n) and m comparators (labeled p_1 through p_m). Create the process $R(N)$ on graph $G(N)$ as follows. For each wire w_i, assign a vertex $v_{i,0}$ to the beginning of wire w_i, and a vertex $v_{i,f}$ to the end of wire w_i. Furthermore, for every pair of consecutive comparators acting on wire w_i, assign an additional vertex representing the stretch of wire between those two comparators. Label these vertices $v_{i,1}, v_{i,2}, \ldots$. Also assign a vertex to each comparator, which we label $v_{p_1}, v_{p_2}, \ldots v_{p_n}$. Let $V_i = \{v_{i,0}, i \in [n]\}$ and $V_f = \{v_{i,f}, i \in [n]$, where $\leq f$ is defined such that $v_{i,f} \leq v_{j,f}$ if $i \leq j$.

For each vertex corresponding to a stretch of wire (with the exceptions of $v_{i,f}$ for each i), draw a directed edge from that vertex to the vertex corresponding to the next comparator acting on that wire. For each comparator p_i, draw directed edges $e_1(p_i)$ and $e_2(p_i)$ to the two wires acted on by that comparator, with the smaller edge $e_1(p_i)$ sent to the vertex corresponding to the smaller wire, and the larger edge $e_2(p_i)$ sent to the vertex corresponding to the larger wire. For each wire w_i with initial value x, create a chip with label x and place the chip at vertex $v_{i,0}$. Then fire until completion.

Note that a sorting network with n wires (and thus n values) generates a labeled chip-firing process with n chips.

An example of a sorting network expressed as a general labeled chip-firing process is shown in Figure 9.4.

We can prove that the output vertices in $R(N)$ produce the same output as the ends of the wires in N.

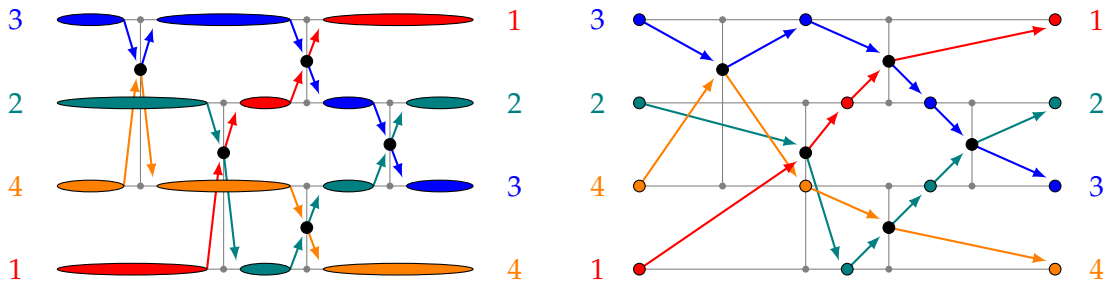


FIGURE 9.4: **Left:** the sorting network from Figure 9.3 expressed as a general labeled chip-firing process. Each comparator and strech of wire is given a vertex. The vertices corresponding to wires are stretched out to make the picture more clear. Each stretch of wire (except for the last stretch on each wire) has an edge to the next comparator, while each comparator has two edges to the next stretches of the two wires being compared. The element with the smaller label is sent to the smaller wire (drawn higher in the picture), while the greater label is sent to the larger (lower) wire. **Right:** The same picture from the left, but drawn with smaller vertices to look more like a graph.

Lemma 9.4.1. *Let N be a sorting network. Then $R(N)$ terminates with one chip on each vertex $v_{i,f}$ for $i \in [n]$. Furthermore, for each $i, j \in n$ with $i < j$, the value of the chip at $v_{i,f}$ is less than or equal to the value of the chip at $v_{j,f}$.*

Proof. We prove by structural induction that for each comparator and stretch of wire in the sorting network N, the values that reach those wires and comparators are the same as the values that reach the corresponding vertices in $G(N)$. Specifically:

(1) The value at of the chip at $v_{i,0}$ is equal to the value that starts at wire w_i for each $i \in n$.

(2) For each comparator c_i, there are two chips that reach v_{c_i}, and their values are equal to the two values compared by comparator c_i.

(3) For each stretch of wire w_i between comparators c_a and c_b, corresponding to vertex $v_{i,j}$, there is a single chip that reaches vertex $v_{i,j}$, which has a value equal to the value that travels along the corresponding stretch of wire.

(4) For the final stretch of each wire w_i, a single chip reaches vertex $v_{i,f}$, which has a value equal to the final value at wire w_i.

Condition (1) is true by definition, and acts as a base case to the induction. For condition (2), consider comparator c_i and assume that condition (3) is satisfied for the two streches of wire leading into comparator c_i. Then the vertices corresponding to the stretches of wire leading into c_i each receive single chips with labels equal to the values on the stretches of wire leading into c_i. Thus, v_{c_i} receives two chips with values equal to the values that reach comparator c_i, as desired.

For condition (3), consider a vertex $v_{i,j}$ corresponding to a stretch of wire w_i immediately after comparator c_a. Assume that v_{c_a} satisfies condition (2). Thus, v_{c_a} receives two chips whose values are equal to the values that reach comparator c_a. If w_i is the smaller numbered wire connected to c_a, then w_i receives the smaller value from c_a, and $v_{i,j}$ receives the smaller chip from v_{c_a}. Otherwise, w_i receives the larger value from c_a, and $v_{i,j}$ receives the larger chip from v_{c_a}. Either way, v_{i_j} receives a single chip with a label that is equal to the value that reaches the corresponding stretch of wire w_i.

Similarly, for condition (4), Consider a vertex $v_{i,f}$ corresponding to the end of wire w_i, immediately after comparator c_a. Assume that v_{c_a} satisfies condition (2). Thus, v_{c_a} receives two chips whose values are equal to the values that reach comparator c_a. If w_i is the smaller numbered wire connected to c_a, then w_i receives the smaller value from c_a, and $v_{i,f}$ receives the smaller chip from v_{c_a}. Otherwise, w_i receives the larger value from c_a, and $v_{i,f}$ receives the larger chip from v_{c_a}. Either way, $v_{i,f}$ receives a single chip with a label that is equal to the value that ends on wire w_i.

Since the label of the chip at each $v_{i,f}$ is equal to the value at the end of the corresponding wire w_i, and since the values on the wires end in sorted order, the values at the vertices $v_{i,f}$ must also end in sorted order, as desired.

□

Since we can construct a general labeled chip-firing process that reproduces any sorting network, we can prove the existence of general labeled chip-firing families with the same run-time bounds as sorting networks. In particular, this allows us to construct a general labeled chip-firing process that sorts n chips in time $O(n \log(n))$.

Lemma 9.4.2. *There exists a bounded general labeled chip-firing family consisting entirely of strongly sorting general labeled chip-firing processes that sorts n chips in time $O(n\log(n))$.*

Proof. Given a sorting network N, $R(N)$ requires 3 firing moves for each comparator c_a of N: one firing move at v_{c_a}, and one firing move at each of the vertices of $G(N)$ corresponding to the stretches of wire leading into c_a. Thus, the run time of $R(N)$ is bounded by a constant multiplied by the run time of N.

The AKS network with n values and n wires uses $O(n\log(n))$ comparators, and thus runs in $O(n\log(n))$ time [1]. As a result, the general labeled chip-firing family with n chips, generated by the AKS network on n values, must also run in $O(n\log(n))$ time, as desired. □

Note that this family also allows for parallel sorting in time $O(\log(n))$. If the impracticality of the AKS network is a concern, the same methods can be used to reproduce any sorting network with $O(n\log^2(n))$ run time on a single processor, or $O(\log^2(n))$ run time in parallel.

It is also worth noting that, unlike traditional labeled chip-firing and the modified version introduced by Hopkins [26], chip-firing with sorting networks cannot sort arbitrary numbers of chips on a single graph. For each value of n, a special graph is chosen to sort exactly n chips. The question of whether a single graph can sort any number of chips in $O(n\log(n))$ time is still open.

CHAPTER 10

M-Matrices

10.1 Introduction

One major attempt to generalize chip-firing is as a sequence of matrix operations. Given a graph G with n vertices $v_1, v_2, \ldots v_n$, a **chip configuration** can be defined as a vector $u = (u_1, u_2, \ldots u_n)$, in which each u_i represents the number of chips at vertex v_i. A firing move at site v_i is represented by reducing the value of u_i by the degree of v_i, and for every neighbor v_j of v_i, increasing the value of u_j by the number of edges from v_i to v_j.

Given a graph G with n vertices, define the **graph Laplacian** of G as the $n \times n$ matrix L such that for each $i, j \in [n]$

$$L_{i,j} = \begin{cases} outdeg(v_i) & i = j \\ -a_{i,j} & \text{otherwise} \end{cases}$$

Where $a_{i,j}$ is the number of edges from i to j.

Thus, firing vertex v_i is the same as subtracting the i^{th} row of the graph Laplacian from the chip configuration. If we remove the i^{th} row and i^{th} column from the graph Laplacian, this **reduced graph Laplacian** corresponds to chip-firing on graph G with a sink at vertex v_i. If we start with an initial configuration u and subtract rows of the reduced graph Laplacian from u (not including the entry corresponding to site i) without

allowing any entries to become negative, then the process is guaranteed to converge to a fixed final configuration, regardless of the order of the moves performed.

This formulation introduces possibilities for generalization of the problem. Rather than requiring "configurations" to be vectors of nonnegative integers, configurations can be any n-vectors (although we usually still require configurations to have nonnegative real entries in order to ensure that the firing process terminates). Rather than requiring "moves" to consist of subtracting rows of a reduced graph Laplacian, the graph Laplacian can be replaced with any n-column matrix (although we usually restrict the matrix to be a square matrix).

Subject to the above constraints, we can ask about which types of matrices guarantee certain properties that we care about in chip-firing. Which matrices give local or global confluence? Which matrices are guaranteed to stabilize in a finite number of steps? Which matrices maintain the concepts of critical and superstable configurations?

In [20], Gabrielov introduces **avalanche finite** matrices. Avalanche finite matrices generalize the class of graph Laplacians, in that reducing an avalanche finite matrix and subtracting its rows from a nonnegative vector eventually leads to a fixed final configuration. The class of avalanche finite matrices thus generalizes the concept of global confluence beyond the class of matrices produced by chip-firing processes.

In [24], Guzmán and Klivans identify avalanche finite matrices as precisely **M-matrices**, a class of matrices that is well-known from other contexts, including economics, operations research, and finite element analysis (see [10], [11], [14], [34], [29], [50]). They show that M-matrices generalize other properties of chip-firing, including the concepts of critical and superstable configurations, as well as energy minimization [24].

In [25], Guzmán and Klivans generalize chip-firing to any invertible matrix. They pair an invertible matrix with an M-matrix, and change the space of valid configurations from nonnegative vectors to a different cone that depends on the matrices chosen.

While any invertible matrix can be paired with any M-matrix in this generalized chip-firing process, there is a question of whether a given invertible matrix has a "natural" choice of M-matrix to pair with it. One possible guess for a choice of M-matrix

would be the "closest" M-matrix to a given invertible matrix. However, there is no reason a priori to believe that such a "closest" M-matrix is guaranteed to exist.

The question of finding a closest M-matrix to a given invertible matrix leads us to questions about the geometry of the space of M-matrices. In particular, if the space of M-matrices is convex, then any $n \times n$ matrix must have a closest M-matrix as measured using Euclidean distance. We focus this chapter on questions about the convexity of the space of M-matrices.

We find that the set of $n \times n$ M-matrices is not convex, although it is star convex with respect to every diagonal matrix with all positive diagonal entries. The set of $n \times n$ M-matrices is not star convex with respect to any other matrices.

We also find large convex subsets of the set of $n \times n$ M-matrices. By fixing certain elements to be 0 and allowing other elements to vary freely (while forcing diagonal entries to be positive and off-diagonal entries to not be positive), we can obtain convex subsets with as many as $\binom{n}{2}$ nonzero elements. One such convex subset is the set of upper triangular M-matrices.

While these convex subsets may not be relevant to chip-firing on an undirected graph (if entry (i, j) is allowed by vary, then entry (j, i) must be set to 0), they could be useful in finding "closest" M-matrices in a restricted class.

10.2 Results on M-matrices

Define a **Z-matrix** as a square matrix M such that $M_{i,j} \leq 0$ for all $i \neq j$, and $M_{i,i} > 0$ for all i. Given a Z-matrix M, then M is an **M-matrix** if and only if it is non-singular and any of the following equivalent conditions holds [44]:

(1) All entries of M^{-1} are non-negative.

(2) There exists $x \in \mathbb{R}^n$ such that $x \geq 0$ and Mx has all positive entries.

(3) All principal minors of M are positive.

Given a natural number n, define $S(n)$ to be the set of $n \times n$ M-matrices.

Lemma 10.2.1. *For any n, $S(n)$ is a cone. If $M \in S(n)$, then $kM \in S(n)$ for all $k > 0$.*

Proof. We have $(kM)^{-1} = \frac{1}{k}M^{-1}$. Thus, by condition (1), if M is an M-matrix, then kM is an M-matrix for any $k > 0$. □

A set S on a vector space is **star-convex** with respect to a point $x \in S$ if for all $s \in S$, $\lambda \in [0,1]$, we have $\lambda s + (1-\lambda)x \in S$. Note that S is convex if and only if it is star-convex with respect to every point in S.

Lemma 10.2.2. *Let M be a diagonal $n \times n$ matrix with positive diagonal entries. Then $S(n)$ is star-convex with respect to M.*

Proof. Let v be a vector consisting of n 1's. Mv has strictly positive entries, so by condition (2), M is an M-matrix. Suppose that M' is another $n \times n$ M-matrix. Then by condition (2), there exists a vector $x \geq 0$ such that $M'x$ has all positive entries. Mx must also have all nonnegative entries, so $\big(\lambda M' + (1-\lambda)M\big)x$ must have all positive entries for any $\lambda \in (0,1]$. As a result, $\big(\lambda M' + (1-\lambda)M\big)x$ is an M-matrix for any $\lambda \in [0,1]$, so $S(n)$ is star convex with respect to each diagonal matrix M in $S(n)$ with positive diagonal entries. □

While $S(n)$ is a star-convex set, it is not convex. We can actually show that the diagonal matrices are the only matrices with respect to which $S(n)$ is star-convex.

Lemma 10.2.3. *Consider an $n \times n$ M-matrix M such that $M_{i,j} < 0$ for some $i \neq j$. Then $S(n)$ is not star-convex with respect to M.*

Proof. Construct a matrix M' as follows. Let $M'_{i,i} = 1$ for all i from 1 to n, let $M'_{j,i} = d$, such that $d < 0$ and $(d + M_{j,i})M_{i,j} > (M_{i,i}+1)(M_{j,j}+1)$ (the left side goes to ∞ as d goes to $-\infty$, so such a d is guaranteed to exist), and let all other entries of M' be equal to 0. Then all eigenvalues of M' and its principal minors are equal to 1, so by condition (3), M' is an M-matrix. However, if we let $M'' = M + M'$, then the principal minor:

$$\begin{pmatrix} M''_{i,i} & M''_{i,j} \\ M''_{j,i} & M''_{j,j} \end{pmatrix}$$

has determinant $(M_{i,i}+1)(M_{j,j}+1)-(d+M_{j,i})M_{i,j}$, which is negative by the definition of d. As a result, $M+M'$ is not an M-matrix, and since $S(n)$ is a cone, this implies that $\frac{1}{2}M+\frac{1}{2}M'$ is also not an M-matrix. Therefore, $S(n)$ is not star-convex with respect to M.

□

However, while $S(n)$ is not convex for $n>1$, it contains large convex subsets based on the locations of the nonzero entries. In general, there are convex subsets containing as many as $\frac{n(n+1)}{2}$ nonzero entries, including all diagonal entries and $n!$ possible combinations of other entries.

Let $X(n)=\{(i,j):i,j\in[n],i\neq j\}$ be the set of off-diagonal coordinates in an $n\times n$ matrix, and let T be a subset of $X(n)$. Define $S'_T(n)$ be the set of all $n\times n$ Z-matrices such that all entries at coordinates not in T are equal to 0, and define $S_T(n)$ to be the set of all M-matrices in $S'_T(n)$.

Lemma 10.2.4. *If $|T|>\binom{n}{2}$, then $S_T(n)$ is not convex.*

Proof. By the pigeonhole principle, if $|T|>\binom{n}{2}$, then there exists some $i<j$ such that $(i,j)\in T$ and $(j,i)\in T$. Then define matrices M and M' as follows. Let all diagonal entries of M and M' be equal to 1. Let $M_{i,j}=M'_{j,i}=-4$, and let all other entries of the two matrices be equal to 0. M is upper triangular and M' is lower triangular, so all eigenvalues of M,M', and their principal minors are equal to 1. Thus, by condition 3, M and M' are both M-matrices. However, $M''=\frac{1}{2}M+\frac{1}{2}M'$ has the principal minor

$$\begin{pmatrix} M''_{i,i} & M''_{i,j} \\ M''_{j,i} & M''_{j,j} \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix}$$

which has determinant -3. Thus, by condition (3), M'' is not an M-matrix, so S_T is not convex.

□

Now, given a matrix M and permutations σ and τ, define M_σ to be the result of applying permutation σ to each column of M and M^τ to be the result of applying τ to each row of M. Define M^τ_σ to be the result of performing both permutations. Similarly

define $S_{T,\sigma}(n)$, $S_T^\tau(n)$ and $S_{T,\sigma}^\tau(n)$ to be the results of applying permutations σ and τ to all elements of $S_T(n)$.

Lemma 10.2.5. *If $|T| = \binom{n}{2}$. The following four conditions are equivalent:*

(1) All Z-matrices in $S'_T(n)$ are M-matrices.

(2) $S_T(n)$ is convex.

(3) For all permutations τ of $(1, 2, \ldots n)$ such that $\tau \neq I$, every matrix $M \in S_T^\tau(n)$ has at least one diagonal entry equal to 0.

(4) There exists some permutation σ of $(1, 2, \ldots n)$ such that every matrix $M \in S_{T,\sigma}^\sigma(n)$ is upper triangular.

Proof. Since $S'_T(n)$ is convex, (1), implies (2).

Suppose that σ is as defined in condition (4). Consider a Z-matrix $M \in S'_T(n)$, and let M_σ^σ be the result when σ is applied to the rows and columns of M. M_σ^σ is upper triangular, so the eigenvalues of M_σ^σ and all of its principal minors are equal to the diagonal elements of M_σ^σ, which must be positive. Applying σ to the rows and columns of M transforms each principal minor of M into a principal minor of M_σ^σ with the same eigenvalues, so each eigenvalue of each principal minor of M must also be an eigenvalue of a principal minor of M_σ^σ. Thus, the principal minors of M are positive, so M must also be an M-matrix. As this is true for all $M \in S_T$, (4) implies (1).

Now, suppose that (3) does not hold. Then there exists some matrix $M \in S_T(n)$ and permutation τ such that M^τ has all nonzero diagonal entries, including at least two negative entries. Consider a single cycle τ' of τ. Then define matrices M_1 and M_2 as follows. Let all diagonal entries of M_1 and M_2 be equal to 1. Now, define set T to be $\{(i, j) : i \neq j \text{ and } \tau' \text{ sends } i \text{ to } j\}$. Then partition T into nonempty subsets T_1 and T_2. Let $(M_1)_{t_1} = -4$ for all $t_1 \in T_1$, and let $(M_2)_{t_2} = -4$ for all $t_2 \in T_2$. Set all other entries of M_1 and M_2 equal to 0.

Since the cycle of (i, j) pairs is split between T_1 and T_2, matrices M_1 and M_2 have no nonzero contributors to their determinants other than their main diagonals. Thus,

M_1, M_2, and all principal minors of M_1 and M_2 must have determinant 1, so M_1 and M_2 are both M-matrices. However, $\left(\frac{1}{2}M_1 + \frac{1}{2}M_2\right)^{\tau'}$ has at least two diagonal entries equal to -2. The number of negative diagonal entries is odd if and only if τ' is an even permutation, so the determinant of $\frac{1}{2}M_1 + \frac{1}{2}M_2$ is equal to 1 minus some power of two that is at least 2^2. Thus, $\frac{1}{2}M_1 + \frac{1}{2}M_2$ is not an M-matrix, so the set of M-matrices in S_T is not convex. Thus, (2) implies (3).

Now, Consider a set T such that $|T| = \binom{n}{2}$ and $S_T(n)$ satisfies (3). Suppose that for all permutations τ of $(1, 2, \ldots n)$ such that $\tau \neq I$,, every matrix $M \in S_T^\tau(n)$ has at least one diagonal entry equal to 0.

For simplicity, consider a single matrix $M \in S_T^\tau$ such that all diagonal entries of M are equal to 1, and all off-diagonal entries of M at elements of T are equal to -1. If we can find a permutation σ such that M_σ^σ is upper triangular, then all matrices in $S_{T,\sigma}^\sigma$ are upper triangular.

We construct a permutation σ such that M_σ^σ is upper triangular, as follows. Initially set $\sigma = I$, and then express σ as a series of swaps. Consider the first column of M. Either $M_{1,1}$ is the only nonzero entry, or there exists some $M_{i,1} = -1$ for index $i > 1$. Then compose σ with the swap $(1\ I)$. Repeat this process as long as there are elements equal to -1 in the first column of M.

This leads to one of two possible outcomes: either the process terminates with no elements equal to -1 below the diagonal entry $(1, 1)$ of M_σ^σ, or one column is switched with the first column multiple times. This occurs if there exists some cycle of indices $i_0, i_1, \ldots i_k = i_0$ such that $M_{i_j, i_{j+1}} = -1$ for all j from 0 to $k - 1$. However, if we let $\tau = (i_0, i_1, \ldots i_{k-1})$, then M^τ has diagonal entries equal to -1 in every column in cycle τ, and every other diagonal entry equal to 1. However, M^τ is in S_T^τ, so this contradicts condition (3). Thus, we must reach a matrix M_σ^σ such that all entries below the diagonal in column 1 are equal to 0.

Repeat this process with each subsequent column, composing σ with swaps involving column 2 until there are no nonzero entries below the diagonal in the second column, then the third column, etc. The process terminates with no nonzero entries below

the diagonal for some M_σ^σ, so there exists a σ such that M_σ^σ is upper triangular. Because matrix M was set to be equal to -1 at every pair of indices in T, every matrix in $S_{T,\sigma}^\sigma$ is upper triangular. Thus, (3) implies (4), so the proof is complete. □

While a lack of convexity means that the concept of a "closest" M-matrix to a given matrix does not necessarily exist, star-convexity and certain large convex subsets can at least provide a concept of a closest M-matrix in a particular class. It is worth noting, however, that using the concept of a "closest" M-matrix in some class to bridge the gap between arbitrary invertible matrix firing and traditional chip-firing on unlabeled graphs may be difficult.

Lemma 10.2.5 requires that for one of the convex subsets from Lemma 10.2.5, allowing entry (i, j) to be nonzero forces entry (j, i) to be 0, and vice versa. In particular, no graph Laplacian for an undirected graph with at least one edge can appear in one of these convex subsets. However, there may still be some useful interpretations to some of the convex classes from Lemma 10.2.5, most notably the class of upper triangular matrices, which could be useful to the study of chip-firing on more general classes of matrices.

Bibliography

[1] M. Ajtai, J. Komlós, and E. Szemerédi. An $o(n \log n)$ sorting network. *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 1–9, 1983.

[2] R. Anderson, L. Lovász, P. Shor, J. Spencer, E. Tardos, and S. Winograd. Disks, balls, and walls: Analysis of a combinatorial game. *The American Mathematical Monthly*, 96:481–493, 1989.

[3] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: An explanation of the 1/f noise. *Phys. Rev. Lett.*, 59:381–384, 1987.

[4] M. Baker and F. Shokrieh. Chip-firing games, potential theory on graphs, and spanning trees. *J. Combin. Theory Ser. A*, 120:164–182, 2013.

[5] K. Batcher. Sorting networks and their applications. *AFIPS*, 32:307–314, 1968.

[6] G. Birkhoff. Rings of sets. *Duke Mathematical Journal*, 3:443–454, 1937.

[7] A. Björner and F. Brenti. *Combinatorics of Coxeter groups*. Springer, New York, 2005.

[8] A. Björner, L. Lovász, and P. Shor. Chip-firing games on graphs. *European Journal of Combinatorics*, 12:283–291, 1991.

[9] N. Bourbaki. *Lie groups and Lie algebras. Chapters 4-6*. Springer-Verlag, Berlin, 2002. Translated from the 1968 French original by Andrew Pressley.

[10] J. H. Bramble and B. E. Hubbard. On a finite difference analogue of an elliptic boundary value problem which is neither diagonally dominant nor of nonnegative type. *J. Math. and Phys*, 43:117–132, 1964.

[11] E. Burman and A. Ern. Stabilized galerkin approximation of convection-diffusion-reaction equations: discrete maximum principle and convergence. *Math. Comp*, 74:1637–1652, 2005.

[12] Robert Burton and Robin Pemantle. Local characteristics, entropy and limit theorems for spanning trees and domino tilings via transfer-impedances. *The Annals of Probability*, 21:1329–1371, 1993.

[13] A. Church and J. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, 1936.

[14] P. G. Ciarlet and P.-A. Raviart. Maximum principle and uniform convergence for the finite element method. *Comput. Methods Appl. Mech. Engrg*, 2:17–31, 1973.

[15] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, Cambridge, MA, 2009.

[16] M. Creutz. Plaing with sandpiles. *Physica A: Statistical Mechanics and its Applications*, 340:521–526, 2004.

[17] Wikipedia editors. Sorting network. https://en.wikipedia.org/wiki/Sorting_network. Accessed: 2022-03-15.

[18] S. Felsner and K. Knauer. Uld-lattices and δ-bonds. *Combin. Probab. Comput.*, 18:707–724, 2009.

[19] P. Felzenszwalb and C. Klivans. Flow-firing processes. *Journal of Combinatorial Theory, Series A*, 175, 2020.

[20] A. Gabrielov. Asymmetric abelian avalanches and sandpiles, preprint 93-65. *MSI, Cornell University*, 1993.

[21] P. Galashin, S. Hopkins, T. McConville, and A. Postnikov. Root system chip-firing II: Central-firing. *International Mathematics Research Notices*, 2017.

[22] P. Galashin, S. Hopkins, T. McConville, and A. Postnikov. Root system chip-firing I: Interval-firing. *Mathematische Zeitschrift*, 292:1337–1385, 2019.

[23] E. Goles and M. A. Kiwi. Games on line graphs and sand piles. *Theoretical Computer Science*, 115:321–349, 1993.

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

[25] J. Guzmán and C. Klivans. Chip-firing on general invertible matrices. *SIAM Journal on Discrete Mathematics*, 30, 2016.

[26] S. Hopkins. Sorting via chip-firing. AMS Fall Central Sectional Meeting, 2016.

[27] S. Hopkins, T. McConville, and J. Propp. Sorting via chip-firing. *Electronic Journal of Combinatorics*, 24, 2016.

[28] J. Humphreys. *Introduction to Lie algebras and representation theory*. Springer-Verlag, New York-Berlin, 1972.

[29] I. Kanenko. Linear complementarity problems and characterizations of minkowski matrices. *Linear Algebra and Appl.*, 20:111–129, 1978.

[30] C. Klivans. *The Mathematics of Chip-Firing*. Taylor and Francis Group, Boca Raton, FL, 2018.

[31] C. Klivans and P. Liscio. Confluence in labeled chip-firing. *Journal of Combinatorial Theory, Series A*, 186, 2022.

[32] D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, Boston, MA, 1997.

[33] M. Latapy and H. D. Phan. The lattice structure of chip firing games and related models. *Physica D: Nonlinear Phenomena*, 155:69–82, 2001.

[34] W. Leontief. *The Structure of the American Economy*. Harvard U.P., Cambridge, MA, 1941.

[35] P. Liscio. Lattices in chip-firing, 2020. arXiv:2010.15650.

[36] D. Lorenzini. Smith normal form and laplacians. *Journal of Combinatorial Theory, Series B*, 98:1271–1300, 2008.

[37] László Lovász. *Combinatorial problems and exercises*, volume 361. American Mathematical Soc., North Holland, 1993.

[38] C. Magnien, H. D. Phan, and L. Vuillon. Characterization of lattices induced by (extended) chip firing games. *Discrete Mathematics and Theoretical Computer Science Proceedings*, pages 229–244, 2001.

[39] S. N. Majumdar and Deepak Dhar. Equivalence between the abelian sandpile model and the $q \to 0$ limit of the potts model. *Physica A*, 185:129–145, 1992.

[40] P. McNamara. El-labelings, supersolvability and 0-hecke algebra actions on posets. *J. Combin. Theory Ser. A*, 101:69–89, 2003.

[41] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43:223–243, 1942.

[42] S. Ostojic. Patterns formed by addition of grains to only one site of an abelian sandpile. *Physica A Statistical Mechanics and its Applications*, 318:187–199, 2003.

[43] I. Parberry. The pairwise sorting network. *Parallel Processing Letters*, 2:205–211, 1992.

[44] R. J. Plemons. M-matrix characterizations. i. nonsingular m-matrices. *Linear Algebra and Appl.*, 18:175–188, 1977.

[45] J. Propp. Lattice structure for orientations of graphs, 1993. arXiv:math/0209005.

[46] N. C. Saldanha, C. Tomei, M. A. Casarin Jr., and D. Romualdo. Spaces of domino tilings. *Discrete Comput. Geom.*, 14:207–233, 1995.

[47] D. L. Shell. A high-speed sorting procedure. *Communications of the ACM*, 2:30–32, 1959.

[48] R. Stanley. Supersolvable lattices. *Algebra Universalis*, 2:197–217, 1972.

[49] H. N. V. Temperley. In combinatorics (london math. soc. lecture note series# 13), 1974.

[50] J. Xu and L. Zikatanov. A monotone finite element scheme for convection-diffusion equations. *Math. Comp.*, 68:1429–1446, 1999.