Understanding the Pentagon Game: Chip-Firing on the R_{10} Matroid

Implementation Guide

October 1, 2025

Abstract

This document explains the mathematical foundations of our Pentagon Game implementation, connecting the playable game to the deep mathematics of chip-firing on the R_{10} matroid. We explain *why* this particular game structure matters, how it relates to Alex McDonough's research, and what the code is actually computing.

1 The Big Picture: Why Study Chip-Firing?

1.1 What is Chip-Firing?

Chip-firing is a discrete dynamical system that appears in multiple areas of mathematics:

- **Combinatorics**: Counting spanning trees and graph structures
- **Algebra**: Understanding finite abelian groups
- **Physics**: Modeling self-organized criticality (sandpiles, earthquakes)
- **Algebraic Geometry**: Discrete analogs of divisors on Riemann surfaces

Core Idea: Simple local rules (firing chips between neighbors) lead to complex global behavior (group structure, equivalence classes).

1.2 The Sandpile Group

For a connected graph G with n vertices, the **sandpile group** $S(G)$ is a finite abelian group with a remarkable property:

Theorem 1 (Matrix-Tree Theorem Connection)**.** *The order of the sandpile group equals the number of spanning trees:*

$$|S(G)| = \textit{number of spanning trees of } G$$

The sandpile group captures the *algebraic essence* of the graph's structure.

2 Why R_{10}? Seymour's Decomposition

2.1 Matroids: Generalized Independence

A **matroid** is an abstraction of the concept of "linear independence." While graphs give us one type of matroid (graphic matroids), there are others.

Definition 1 (Regular Matroid)**.** *A* ***regular matroid*** *is one that can be represented by a totally unimodular matrix over* $\mathbb{R}$*, where all minors are in* $\{-1, 0, 1\}$*.*

2.2 Seymour's Fundamental Theorem

Theorem 2 (Seymour 1980)**.** *Every regular matroid can be built from three basic building blocks using sums:*

1. *Graphic matroids (from graphs)*
2. *Cographic matroids (from dual graphs)*
3. R_{10} *(a specific 10-element, rank-5 matroid)*

Analogy: Just as every integer is built from prime numbers, every regular matroid is built from these three types. R_{10} is like a "prime matroid."

Why this matters: Understanding chip-firing on R_{10} helps us understand chip-firing on *all* regular matroids!

3 The Pentagon Structure: From 10D to 5D

3.1 Standard Representation

The matroid R_{10} is standardly represented as a 5×10 matrix:

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Chip configurations live in $\mathbb{Z}^{10}$ - that's 10 dimensions!

3.2 The Gaussian Integer Trick

Alex McDonough's key insight: The matrix $\mathcal{D}$ (the second half of $\mathcal{A}$) is *symmetric*, which allows a clever representation using **Gaussian integers** $\mathbb{Z}[i] = \{a + bi : a, b \in \mathbb{Z}\}$.

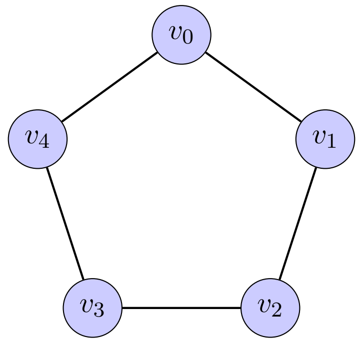
Define the bijection:

$$\varphi : \mathbb{Z}^{10} \to \mathbb{Z}[i]^5, \quad (v_0, \ldots, v_9) \mapsto (v_0 + v_5 i,\ v_1 + v_6 i,\ v_2 + v_7 i,\ v_3 + v_8 i,\ v_4 + v_9 i)$$

This groups each pair of coordinates into a single complex number!

3.3 The Pentagon Emerges

In the $\mathbb{Z}[i]^5$ representation, the firing moves act on a **pentagon graph**:



Each vertex is connected to exactly 2 neighbors (cycle graph C_5).

4 The Four Moves: A, B, C, D

4.1 Move Definitions

The complex number representation gives us four fundamental firing moves:

Definition 2 (The Four Moves)**.** *[(A)]*

1. ***Move A:*** *Add $1+i$ to the selected vertex, add $-i$ to each of its 2 neighbors*
2. ***Move B:*** *Add $-1+i$ to the selected vertex, add 1 to each of its 2 neighbors*
3. ***Move C:*** *Add $-1-i$ to the selected vertex, add i to each of its 2 neighbors*
4. ***Move D:*** *Add $1-i$ to the selected vertex, add -1 to each of its 2 neighbors*

4.2 Relationship Between Moves

Notice the beautiful pattern:

$$\text{Move C} = -(\text{Move A})$$
$$\text{Move D} = -(\text{Move B})$$

This is why our UI only shows buttons for A and B - the negatives are accessed via right-click!

4.3 Why These Specific Moves?

These moves come from the matrix $\overline{\mathcal{K}} = I_5 - \mathcal{D}i$:

$$\overline{\mathcal{K}} = \begin{bmatrix} 1+i & -i & 0 & 0 & -i \\ -i & 1+i & -i & 0 & 0 \\ 0 & -i & 1+i & -i & 0 \\ 0 & 0 & -i & 1+i & -i \\ -i & 0 & 0 & -i & 1+i \end{bmatrix}$$

Firing at vertex v_j corresponds to subtracting the j-th row of $\overline{\mathcal{K}}$ from the configuration vector.

5 The 162 Equivalence Classes

5.1 The Sandpile Group Structure

Theorem 3 (Structure of $S(R_{10})$). *The sandpile group of R_{10} is isomorphic to:*

$$S(R_{10}) \cong (\mathbb{Z}/3\mathbb{Z})^3 \oplus (\mathbb{Z}/6\mathbb{Z})$$

Counting: $|S(R_{10})| = 3 \times 3 \times 3 \times 6 = 162$

5.2 What Are These Equivalence Classes?

Two chip configurations c and c' are **firing equivalent** (written $c \sim c'$) if one can be reached from the other by a sequence of A, B, C, D moves.

The sandpile group is the quotient:

$$S(R_{10}) = \frac{\mathbb{Z}[i]^5}{\mathrm{im}(\overline{\mathcal{K}})}$$

Physical Interpretation: The 162 equivalence classes represent the 162 "fundamentally different" chip configurations. All others are just these 162 repeated.

5.3 The Unique Element of Order 2

The group $S(R_{10})$ has exactly **one element of order 2**, denoted H:

$$H + H = 0 \text{ (in the group)}$$

In Alex's paper, he identifies which configurations satisfy $c \sim H$.

6 Our Implementation: Code Meets Mathematics

6.1 Data Structures

Complex Number:

```
interface ComplexNumber {
  real: number;
  imag: number;
}
```

Game State:

```
interface GameState {
  vertices: ComplexNumber[5];     // Pentagon configuration
  currentMoveType: 'A'|'B'|'C'|'D';
  goalVertices: ComplexNumber[5]; // Always [0,0,0,0,0]
  isWon: boolean;
}
```

6.2 Move Implementation

The move definitions match the paper exactly (after bug fixes!):

```
const moves = {
  'A': { vertex: {real: 1, imag: 1}, adjacent: {real: 0, imag: -1} },
  'B': { vertex: {real: -1, imag: 1}, adjacent: {real: 1, imag: 0} },
  'C': { vertex: {real: -1, imag: -1}, adjacent: {real: 0, imag: 1} },
  'D': { vertex: {real: 1, imag: -1}, adjacent: {real: -1, imag: 0} },
};

const adjacency = {
  0: [1, 4],  // v0 connects to v1 and v4
  1: [0, 2],  // v1 connects to v0 and v2
  2: [1, 3],  // etc.
  3: [2, 4],
  4: [3, 0],
};
```

Critical Bug Fixed: Initially, we had the real and imaginary parts swapped for adjacent vertices. The paper specifies move A adds $-i$ (imaginary) to neighbors, not -1 (real)!

6.3 The Matrix Solver

We use $\overline{\mathcal{K}}^{-1}$ to find optimal moves:

$$\overline{\mathcal{K}}^{-1} = \frac{1}{6}\begin{bmatrix} 3-i & 1+i & -1+i & -1+i & 1+i \\ 1+i & 3-i & 1+i & -1+i & -1+i \\ -1+i & 1+i & 3-i & 1+i & -1+i \\ -1+i & -1+i & 1+i & 3-i & 1+i \\ 1+i & -1+i & -1+i & 1+i & 3-i \end{bmatrix}$$

Hint Algorithm:

1. Compute difference vector: $d = 0 - c$ (goal minus current)
2. Apply inverse: $s = \overline{\mathcal{K}}^{-1} \cdot d$
3. Test all 20 moves (4 types $\times$ 5 vertices)
4. Return move that minimizes distance to zero

Note: This is a *greedy* approach (one-step lookahead), not guaranteed optimal for all puzzles.

7 What's Different from Classic Chip-Firing?

7.1 Classic Chip-Firing (Graphs)

Rules:

- Vertex fires when $\text{chips}(v) \geq \deg(v)$

- Firing is automatic/required
- Process continues until stable

Goal: Understand which configurations stabilize

7.2 Our Game (R_{10} Variant)

Rules:

- Player chooses which move (A/B/C/D) and where
- Moves have fixed effects (not threshold-based)
- Complex number arithmetic

Goal: Reach the zero configuration $[0, 0, 0, 0, 0]$

7.3 Key Insight

Our game is not about *automatic stabilization* - it's about exploring the **equivalence classes** of the sandpile group!

Each puzzle starts in one of the 162 orbits. The question is: which moves bring you to the zero configuration (if possible)?

8 Open Questions & Future Work

8.1 What We Know

- The move definitions are mathematically correct ✓
- The matrix solver uses the correct $\overline{\mathcal{K}}^{-1}$ ✓
- The pentagon adjacency is correct ✓
- Puzzles are generated using random coefficient combinations ✓

8.2 What We Don't Show (Yet)

1. **Orbit Number:** Which of the 162 equivalence classes is the current configuration in?
2. **Representatives:** Display all 162 representative configurations
3. **Solvability:** Is the current puzzle actually solvable, or are we in the wrong orbit?
4. **Optimal Path:** The greedy solver may not find the shortest solution

8.3 Computing the Orbit

To determine which equivalence class a configuration c belongs to, we need to compute:

$$c \mod \mathrm{im}(\overline{\mathcal{K}})$$

This requires:

- Computing a Hermite normal form of $\overline{\mathcal{K}}$
- Finding a set of 162 canonical representatives
- Reducing c to one of these representatives

Challenge: This is computationally complex and not yet implemented!

9 Conclusion: Why This Game Matters

9.1 For Mathematics

This game provides a **concrete, interactive demonstration** of:

- Chip-firing on a non-graphic matroid
- The 162-orbit structure of $S(R_{10})$
- The Gaussian integer representation trick
- Matrix-based solver approaches

9.2 For Learning

The game makes abstract algebra **tangible**:

- See group equivalence classes in action
- Understand why certain moves "cancel out"
- Experience the difference between solvable and unsolvable configurations
- Visualize complex number arithmetic geometrically

9.3 The Big Takeaway

Key Insight: What looks like a simple number puzzle is actually a playable demonstration of deep mathematics connecting matroid theory, algebraic topology, and group theory.
The pentagon isn't random - it's the natural structure that emerges when you represent R_{10} using Gaussian integers!

Acknowledgments

This implementation is based on the mathematical framework developed by Alex McDonough in his paper "Chip-Firing and the Sandpile Group of the R_{10} Matroid." All mathematical credit goes to Alex and the broader chip-firing research community.