MOTORS AND IMPOSSIBLE FIRING PATTERNS IN THE PARALLEL CHIP-FIRING GAME*

TIAN-YI JIANG†, ZIV SCULLY†, AND YAN X. ZHANG‡

Abstract. The *parallel chip-firing game* is an automaton on graphs in which vertices "fire" chips to their neighbors. This simple model contains much emergent complexity and has many connections to different areas of mathematics. In this work, we study *firing sequences*, which describe each vertex's interaction with its neighbors in this game. First, we introduce the concepts of *motors* and *motorized games*. Motors both generalize the game and allow us to isolate local behavior of the (ordinary) game. We study the effects of motors connected to a tree and show that motorized games can be transformed into ordinary games if the motor's firing sequence occurs in some ordinary game. Then, we completely characterize the periodic firing sequences that can occur in an ordinary game, which have a surprisingly simple combinatorial description.

Key words. chip-firing, parallel chip-firing, periodicity, fixed-energy sandpile

AMS subject classifications. 37B15, 82C20

DOI. 10.1137/130933770

1. Introduction.

Background. The *parallel chip-firing game*, also known as the *discrete fixed-energy sandpile model*, is an automaton on graphs in which vertices that have at least as many chips as incident edges "fire" chips to their neighbors. In graph theory, it has been studied in relation with the critical group of graphs [3]. In computer science, it is able to simulate any two-register machine and is thus universal [8]. As a specific case of the *abelian sandpile model*, which is itself a generalization of a sandpile model introduced by Bak, Tang, and Weisenfeld [1, 2] in the study of self-organized criticality, it has even more links with other fields.

The game. The parallel chip-firing game is played on a graph as follows:
- At first, a nonnegative integer number of chips is placed on each vertex of the graph.
- The game then proceeds in discrete turns. Each turn, a vertex checks to see if it has at least as many chips as incident edges:
  - If so, that vertex *fires*.
  - Otherwise, that vertex *waits*.
- To fire, a vertex passes one chip along each of its edges. All vertices that fire in a particular turn do so in parallel.
- Immediately after firing or waiting, every vertex receives any chips that were fired to it.

Here we will only consider games on finite, undirected, connected graphs, though the definition of the game can be easily generalized for arbitrary multidigraphs. An example game is illustrated in Figure 1. Given a parallel chip-firing game σ, we refer to the chip configuration, also called the position, at a particular time $t \in \mathbb{N}$ as σ_t.
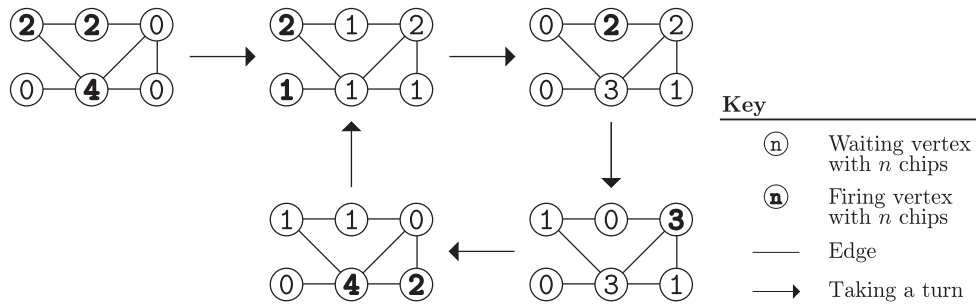
FIG. 1. *A parallel chip-firing game. From an initial position in the upper left, the game eventually enters a period of length* 4.

The total number of chips on all vertices of the graph is constant throughout a game, so there are finitely many possible positions in every game. Therefore, every game eventually reaches a position σ_t that is identical to a later position σ_{t+p} for some $t, p \in \mathbb{N}$ with $p > 0$. (We write $\sigma_t = \sigma_{t+p}$.) The game is deterministic, so $\sigma_{t+n} = \sigma_{t+n+p}$ for all $n \in \mathbb{N}$. Thus, every parallel chip-firing game is eventually periodic.

In this paper, we concern ourselves with both *firing sequences* and *periodic firing patterns* (PFPs) of vertices. Each is a binary string representing whether or not a particular vertex fires or waits each turn. The sequence covers all times from 0 to infinity, while the periodic pattern covers just one period.

Previous work. The periodicity of the parallel chip-firing game gives rise to two questions. First, what characteristics of a game and its underlying graph determine the length of a period? It is known exactly what periods are possible on certain classes of graphs, such as trees [4], simple cycles [6], the complete graph [13], and the complete bipartite graph [10]. For these graphs, the maximum period lengths are bounded by the number of vertices, but Kiwi et al. [11] constructed graphs on which the period of games can grow exponentially with polynomial increase in the number of vertices. There are also results regarding the total number of chips in a game. Kominers and Kominers [12] showed that games with a sufficiently large density of chips must have period 1. Dall'Asta [6] and Levine [13], in their respective characterizations of periods on cycles and complete graphs, related the total number of chips to a game's *activity*, the fraction of turns during which a vertex fires. The denominator of the activity must divide the period.

Second, we notice that some but not all positions σ_t are *periodic*, satisfying $\sigma_t = \sigma_{t+p}$ for some positive $p \in \mathbb{N}$. What characterizes periodic positions? This problem has not been as extensively studied. Dall'Asta [6] characterized the periodic positions of games on cycles.

Our results. We hope to advance the understanding of both of these questions through the study of firing sequences and PFPs.

After precisely defining the parallel chip-firing game in section 2, the first half of the paper develops a new tool for studying the chip-firing game: *motors*, vertices that fire with a regular pattern independent of normal chip-firing rules. Games with motors are called *motorized games*. Motors allow us to study the behavior of subgraphs in ordinary parallel chip-firing games. In section 3 we show that vertices always "follow" a motor in periodic motorized games on trees. In section 4, we prove that periodic

motorized games can be transformed into ordinary games as long as the firing sequence of each motor occurs in an ordinary game.

The second half of the paper characterizes the possible PFPs in parallel chip-firing games. Section 5 briefly steps away from the game to study certain signed sums of periodic binary sequences. The result is an inequality applicable to edges of the graph of a parallel chip-firing game. In section 6, we sum this inequality over all relevant edges to show that PFPs with both consecutive 0s and consecutive 1s cannot occur in a parallel chip-firing game. This, along with an already known construction, fully characterizes the PFPs possible in parallel chip-firing games. Finally, in section 7, we examine some implications of this theorem.

2. Preliminaries.

Definitions. A *parallel chip-firing game* σ on a graph $G = (V(G), E(G))$ is a sequence $(\sigma_t)_{t \in \mathbb{N}}$ of ordered tuples with natural number elements indexed by $V(G)$. Each tuple represents the chip configuration at a particular turn, where each element of the tuple is the number of chips on the corresponding vertex. We define the following for all $v \in V(G)$:

$$N(v) = \{w \in V(G) \mid \{v, w\} \in E(G)\},$$

$$d(v) = \#N(v),$$

$$\sigma_t(v) = \text{number of chips on } v \text{ in position } \sigma_t,$$

$$F_t(v) = \begin{cases} 0 & \text{if } \sigma_t(v) \leq d(v) - 1, \\ 1 & \text{if } \sigma_t(v) \geq d(v), \end{cases}$$

$$\Phi_t(v) = \sum_{w \in N(v)} F_t(w).$$

In a parallel chip-firing game, σ_t induces σ_{t+1}. For all $v \in V(G)$,

(2.1) $$\sigma_{t+1}(v) = \sigma_t(v) + \Phi_t(v) - F_t(v)d(v),$$

so an initial position suffices to define a game on a given graph. When $F_t(v) = 0$, we say v *waits* at t, and when $F_t(v) = 1$, we say v *fires* at t.

A position σ_t is *periodic* if and only if there exists $p \in \mathbb{N}$ such that $\sigma_t = \sigma_{t+p}$. The minimum such p for which this occurs is the *period* of σ and is denoted T. Abusing notation slightly, "a period" of a game σ may also refer to a set of times $\{t, t+1, \ldots, t+T-1\}$, where σ_t is periodic. The parallel chip-firing game is deterministic and there are finitely many possible positions on a given graph with a given number of chips, so for any game σ, there exists $t_0 \in \mathbb{N}$ such that σ_t is periodic for all $t \geq t_0$. If the initial position of a game is periodic, we may also call the game itself periodic.

Notation. Definitions for invented notation are given in the section indicated in the last column.

Parallel chip-firing		Defined in
$\sigma_t(v)$	Number of chips on vertex v in position σ_t.	Section 2
$F^\sigma(v)$	PFP of v.	Section 6
$F_t^\sigma(v)$	Indicates whether or not vertex v fires in σ_t.	Section 2
$\Phi_t^\sigma(v)$	Number of chips vertex v will receive in σ_t.	Section 2
T^σ	Period of σ.	Section 2
M^σ	Set of vertices that are motors in σ.	Section 3
Graphs		
$V(G)$	Vertex set of graph G.	
$E(G)$	Edge set of graph G.	
$N_G(v)$	Neighbors of vertex v.	
$d_G(v)$	The degree of vertex v in graph G.	
Other		
$[a, b]$	The integer interval $\{a, a+1, \ldots, b\}$.	

We leave out the subscript G or superscript σ if there is no ambiguity.

3. Motors. Let G be a graph. Suppose we wish to study the periodic behavior of games on G, focusing on a particular subgraph $H \subseteq G$. Consider

$$X = \{v \in V(G) \setminus V(H) \mid N(v) \cap V(H) \neq \emptyset\},$$

the boundary of H. Knowing the initial chip configuration on $V(H) \cup X$ is, in general, not enough to determine all subsequent configurations because vertices in X may have interactions with vertices outside of $V(H) \cup X$. However, we do know that every vertex assumes a pattern of firing and waiting that repeats periodically as soon as a game reaches a periodic position. Therefore, we can simulate the presence of the rest of G by having each vertex in X fire with a regular pattern regardless of the number of chips it receives.

The *firing sequence* of a vertex v in game σ is the sequence $(F_t(v))_{t \in \mathbb{N}}$. A *motorized parallel chip-firing game*, or simply "motorized game," on G is a game σ obeying (2.1) with a nonempty set of *motors* $M \subseteq V(G)$. Each motor follows a predetermined firing sequence, firing without regard for the normal rules of the parallel chip-firing game, which means, for example, that a motor may have a negative number of chips. Put another way, for each $m \in M$, $F_t(m)$ does not depend on $\sigma_t(m)$. The term "ordinary game" refers to a game with no motors when there is ambiguity. A motorized game is shown in Figure 2.

If a motorized game σ is eventually periodic (which is the case if every motor's firing sequence is eventually periodic), then just as in an ordinary game, every vertex fires the same number of times each period. The proof is identical to the proof of this fact for ordinary games [10]: all neighbors of the vertex that fires the most times each period must also fire that maximal number of times, and by induction, so do all vertices. (Recall that we consider in this paper only connected graphs.)
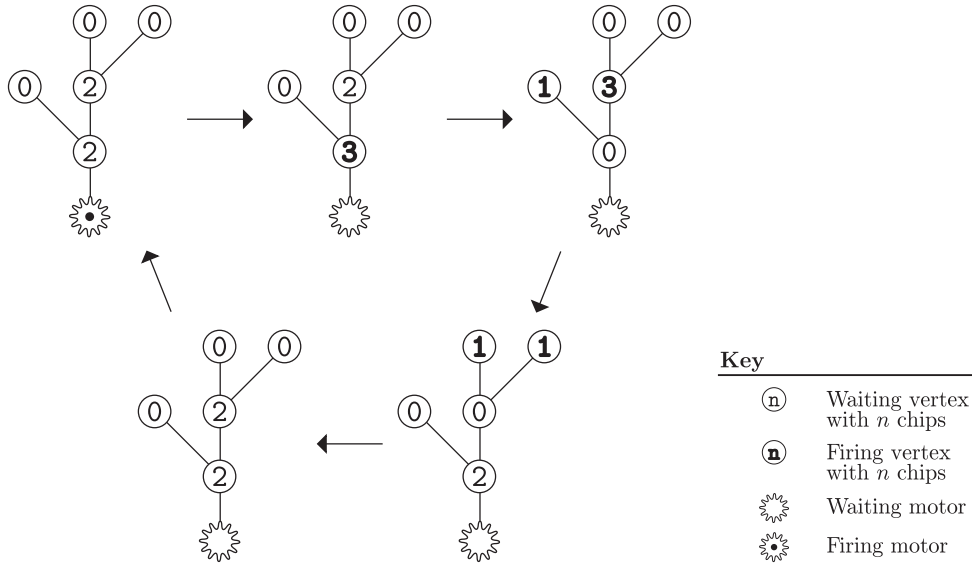
FIG. 2.   *A motorized parallel chip-firing game.   The motor has firing sequence* $(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, \dots)$.

Let $f \in \{0, 1\}$. Call an interval $[a, b]$ with $a < b$ an f-*clump* of $v \in V(G)$ if and only if $F_t(v) = f$ for all $t \in [a, b]$. We call $[a, b]$ an f-*max-clump* if, in addition, $F_{a-1}(v) = F_{b+1}(v) = 1 - f$. Given $v \in V(G)$, we can express $\mathbb{N}$ as the union of max-clumps of v and times during which v alternates between firing and waiting.

The proof of Theorem 3.2 follows the same structure as the proof that ordinary games on trees have period 1 or 2 [4]. In fact, we rely on a lemma originally introduced for that proof.

LEMMA 3.1 (see [4, Lemma 1]). *Let* σ *be a game on* G. *For all* $v \in V(G)$ *and* $f \in \{0, 1\}$, *if* $[a, b]$ *is an* f-*clump of* v, *then there exists a neighbor* $w \in N(v)$ *such that* $[a - 1, b - 1]$ *is an* f-*clump of* w.

Less technically, every clump of firing or waiting by a vertex must be supported by at least one of its neighbors. The lemma follows from the pigeonhole principle and Lemma 6.1, which we state and prove later.

THEOREM 3.2. *Let* σ *be a periodic motorized game on tree* T. *For all* $v \in V(T)$ *and* $f \in \{0, 1\}$, *if* $[a, b]$ *is an* f-*clump of* v, *then* $[a - D, b - D]$ *is an* f-*clump of* m *for some* $m \in M$, *where* D *is the distance from* m *to* v.

Proof. The result is clear if all vertices either always fire or always wait. In all other cases, each firing sequence has a max-clump, and the argument is roughly as follows. By Lemma 3.1, each clump of a vertex must be supported by a clump of a neighbor. Following the "chain of support" gives a sequence of vertices that either is infinite or ends with a motor. If we consider the containing max-clumps of clumps, we can guarantee a sequence with no backtracking. Trees have no cycles, so the sequence must end with a motor. The details follow.

Let $v_0 = v$ and $[a_0, b_0] \supseteq [a, b]$ be an f-max-clump of v_0. By Lemma 3.1, given a vertex $v_i \notin M$ with clump $[a_i, b_i]$, we can pick a supporting vertex $v_{i+1} \in N(v_i)$ and integers a_{i+1} and b_{i+1} such that $[a_{i+1}, b_{i+1}]$ is an f-max-clump of v_i and $[a_i - 1, b_i - 1] \subseteq [a_{i+1}, b_{i+1}]$. (The fact that σ is periodic means we need not worry about negative turn numbers.) If there is a maximum i for which v_i exists, that vertex must be a

motor, which would mean $[a - D, b - D] \subseteq [a_D, b_D]$, where D is the maximum i and $m = v_D \in M$. Thus, it suffices to show that the sequence $(v_0, v_1, \ldots)$ eventually terminates. There are finitely many vertices in the graph, so it suffices to show that the v_i are all distinct.

T has no cycles, so if $v_i \neq v_{i+2}$ for all i, then all v_i are distinct. Suppose that $v_i = v_{i+2}$ for some i. Then $[a_i, b_i] \cup [a_{i+2}, b_{i+2}]$ would be a clump of v_i. However, $[a_i - 2, b_i - 2] \subseteq [a_{i+2}, b_{i+2}]$, making $[a_i - 2, b_i]$ a clump of v_i. But $[a_i, b_i]$ is a max-clump for all i, so $v_i \neq v_{i-2}$ for all i. $\quad\square$

Call a firing sequence *clumpy* if it contains two consecutive 0s and two consecutive 1s; otherwise, call it *nonclumpy*.

COROLLARY 3.3. *Let σ be a periodic motorized game on tree T with a single motor m. If m has a nonclumpy firing sequence but has at least one clump, then $F_{t+D}(v) = F_t(m)$ for all $v \in V(T)$ and $t \in \mathbb{N}$, where D is the distance from v to m.*

Proof. The result is again clear in the always waiting and always firing cases. In all other cases, m has an f-max-clump, where $f \in \{0, 1\}$. Let $v \in V(T)$. By Theorem 3.2, v has a nonclumpy firing sequence because m does. All vertices fire the same number of times every period [10, Proposition 2.5], so v must have at least one max-clump, again because m does. For every f-max-clump $[a, b]$ of v, $[a - D, b - D]$ is an f-clump of m. The non-max-clump intervals of v's firing sequence are alternations between 0 and 1, starting and ending with $1 - f$. The same must be true of m for it to fire the same number of times as v each period. $\quad\square$

The reason we require the games in Theorem 3.2 and Corollary 3.3 to be periodic is to consider arbitrarily many past turns. We can likely weaken this condition if we require the statements to be true only after sufficiently many turns, though exactly how many turns that is could depend on the activity (firing frequency; see [13]) of the motor, the size of the tree, and the total number of chips in the initial position.

4. Simulating motors. In this section, to refer to multiple chip-firing games unambiguously, we include the subscripts and superscripts in, for example, $d_G(v)$ and $F_t^\sigma(v)$.

We call a firing sequence $(f_t)_{t \in \mathbb{N}}$ *possible* if there exists an ordinary game σ on some graph G such that $F_t^\sigma(v) = f_t$ for all $t \in \mathbb{N}$. Our next theorem states that we can simulate motorized games with ordinary games as long as every motor's firing sequence is possible. Figure 3 demonstrates the concept.

THEOREM 4.1. *Let σ be a motorized game on G. If every motor's firing sequence is possible and has the same activity (firing frequency; see [13]), then there exists an ordinary game σ' on a graph $H \supseteq G$ such that*
- *$F_t^{\sigma'}(u) = F_t^\sigma(u)$ for all $t \in \mathbb{N}$ and $u \in V(G)$,*
- *$d_H(v) = d_G(v)$ for all $v \in V(G) \setminus M^\sigma$, and*
- *the subgraph of H induced by $V(G)$ is G. (That is, H contains no edges between vertices of G that are not also in G.)*

Proof. Our approach will be, for each $m \in M^\sigma$, to attach many copies of a graph with a vertex with m's firing sequence to m. If sufficiently many copies are attached, the number of chips m has due to its neighbors in G becomes irrelevant as to whether or not it fires.

For each $m \in M^\sigma$, let A_m be a graph such that there exists a game σ^m and some vertex $u_m \in V(A_m)$ such that $F_t^{\sigma^m}(u_m) = F_t^\sigma(m)$ for all $t \in \mathbb{N}$. Let a_m and b_m be the minimum and maximum, respectively, of $\{\sigma_t(m) \mid t \in \mathbb{N}\}$. These bounds exist because all the motors have possible firing sequences with the same activity, which means the motorized game is eventually periodic. Let $k_m = b_m - a_m + 1$, and let H
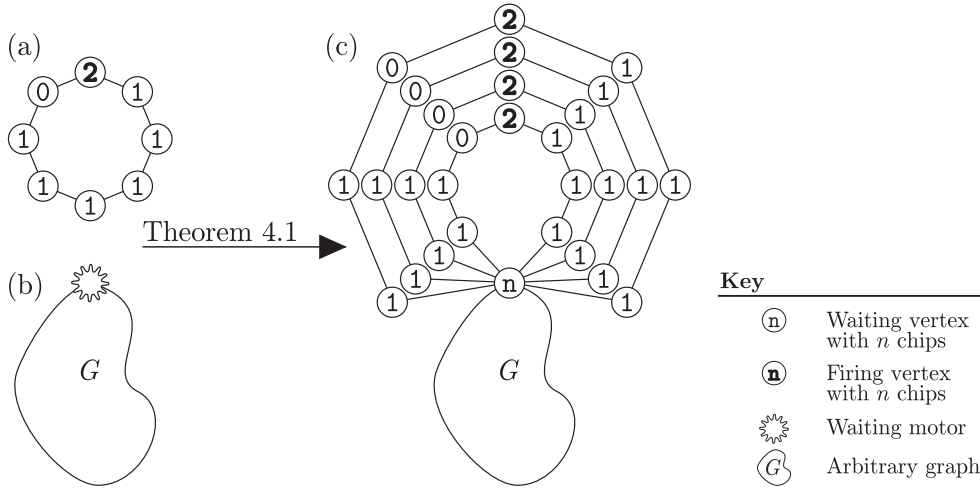
FIG. 3. *Suppose the motor in motorized game* (b) *has firing sequence* $(0, 0, 0, 0, 1, 0, 0, 0, \dots)$. *This occurs in ordinary game* (a). *By using sufficiently many copies of* (a) *and carefully choosing n, we construct* (c). *The behavior of G in* (c) *is identical to the behavior of G in* (b).

be the union of G and k_m copies of each A_m, with G and the copies of A_m disjoint except for $m = u_m$ for each $m \in M^\sigma$.

It is clear by construction that H contains no new edges between vertices of G and that

- $d_H(m) = k_m d_{A_m}(u_m) + d_G(m)$ for all $m \in M^\sigma$,
- $d_H(u) = d_{A_m}(u)$ for all $u \in V(A_m) \setminus \{m\}$ for each $m \in M^\sigma$, and
- $d_H(v) = d_G(v)$ for all $v \in V(G) \setminus M^\sigma$.

Suppose that for some $t \in \mathbb{N}$, σ'_t satisfies the following.

1. $\sigma'_t(m) = k_m \sigma^m_t(u_m) + d_G(m) + \sigma_t(m) - a_m$ for all $m \in M^\sigma$.
2. $\sigma'_t(u) = \sigma^m_t(u)$ for all $u \in V(A_m) \setminus \{m\}$ for each $m \in M^\sigma$.
3. $\sigma'_t(v) = \sigma_t(v)$ for all $v \in V(G) \setminus M^\sigma$.

We will show that σ'_{t+1} satisfies the above as well. We have $d_H(v) = d_G(v)$ for all $v \in V(G) \setminus M^\sigma$, so $F^{\sigma'}_t(v) = F^\sigma_t(v)$ for all $v \in V(G) \setminus M^\sigma$. Similarly, $F^{\sigma'}_t(u) = F^{\sigma^m}_t(u)$ for all $u \in V(A_m) \setminus \{m\}$ for each $m \in M^\sigma$. Finally, for all $m \in M^\sigma$, if $F^{\sigma^m}_t(u_m) = 0$, then

$$
\begin{aligned}
\sigma'_t(m) &\le k_m(d_{A_m}(u_m) - 1) + d_G(m) + \sigma_t(m) - a_m \\
&= k_m d_{A_m}(u_m) + d_G(m) + (\sigma_t(m) - b_m) - 1 \\
&\le d_H(m) - 1,
\end{aligned}
$$

and if $F^{\sigma^m}_t(u_m) = 1$, then

$$
\begin{aligned}
\sigma'_t(m) &\ge k_m d_{A_m}(u_m) + d_G(m) + (\sigma_t(m) - a_m) \\
&\ge d_H(m),
\end{aligned}
$$

so $F^{\sigma'}_t(m) = F^{\sigma^m}_t(u_m) = F^\sigma_t(m)$.

We now know that $F^{\sigma'}_t(v) = F^\sigma_t(v)$ for all $v \in V(H)$, so $\sigma'_{t+1}(v) = \sigma_{t+1}(v)$ for all $v \in V(G) \setminus M^\sigma$ and $\sigma'_{t+1}(u) = \sigma^m_{t+1}(u)$ for all $u \in V(A_m) \setminus \{m\}$ for each $m \in M^\sigma$.
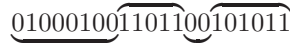
$$\overset{\frown}{01000100}\overset{\frown}{1101100}\overset{\frown}{101011}$$

Fig. 4. *A string's* 0-*sectors (marked below) and* 1-*sectors (marked above).*

Finally, we have

$$
\begin{aligned}
\sigma'_{t+1}(m) &= k_m\sigma_t^m(u_m) + d_G(m) + \sigma_t(m) - a_m + \Phi_t^{\sigma'}(m) - F_t^{\sigma'}(m)d_H(m) \\
&= k_m\sigma_t^m(u_m) + d_G(m) + \sigma_t(m) - a_m + \Phi_t^{\sigma}(m) - F_t^{\sigma}(m)d_G(m) \\
&\quad + k_m\Phi_t^{\sigma^m}(u_m) - k_mF_t^{\sigma^m}(u_m)d_{A_m}(u_m) \\
&= k_m(\sigma_t^m(u_m) + \Phi_t^{\sigma^m}(u_m) - F_t^{\sigma^m}(u_m)d_{A_m}(u_m)) + d_G(m) \\
&\quad + (\sigma_t(m) + \Phi_t^{\sigma}(m) - F_t^{\sigma}(m)d_G(v)) - a_m \\
&= k_m\sigma_{t+1}^m(u_m) + d_G(m) + \sigma_{t+1}(m) - a_m.
\end{aligned}
$$

for all $m \in M^\sigma$.

We can distribute chips in σ'_0 such that it satisfies (1), (2), and (3), in which case, by induction, σ'_t satisfies (1), (2), and (3) for all $t \in \mathbb{N}$, implying $F_t^{\sigma'}(u) = F_t^{\sigma}(u)$ for all $v \in V(G)$. □

In Theorem 3.2, motors were primarily a convenient intuition and terminology; we could have proved a similar theorem within the context of the ordinary parallel chip-firing game, though its statement would have been messier. Theorem 4.1 demonstrates another way in which the motor concept is useful. Its constructive power makes certain conjectures easy to prove or disprove by example. For instance, motors make it easy to construct games in which the period isn't bounded by the number of vertices.

5. Signed sums of binary sequences. We take a brief break from the parallel chip-firing game itself to consider binary strings. Throughout this section, p and q are length-n binary strings, $b \in \{0, 1\}$, and $\bar{b} = 1 - b$. We denote the ith element of a binary string p as p_i, and any integer equivalent to $i \bmod n$ may replace i.

The following definition formalizes the notion of part of a string being "mostly" 0s or 1s. A *b-sector* of p is a $\subseteq$-maximal integer interval $[x, y]$ such that $p_{y-1} = p_y = b$ and either $p_i = b$ or $p_{i+1} = b$ for all $i \in [x, y]$. Informally, *b*-sectors end with consecutive *b*s and extend back as far as possible. We have to make an exception for always-alternating strings, such as 01010101, that have neither 0 nor 1 twice in a row. We arbitrarily define $[0, n-1]$ to be a 0-sector of them. It is the transitions between 0- and 1-sectors that are ultimately important, so this exception is acceptable. It is simple to verify that the indices of every binary string have a unique decomposition into 0- and 1-sectors. An example is shown in Figure 4.

Let

$$
s_i(p) = \begin{cases} -1 & \text{if } i \text{ is in a 0-sector of } p, \\ 1 & \text{if } i \text{ is in a 1-sector of } p, \end{cases}
$$

$$
\delta_i(p) = \begin{cases} 0 & \text{if } i \text{ is in a } b\text{-sector of } p \text{ and } i+1 \text{ is in a } b\text{-sector of } p, \\ 1 & \text{if } i \text{ is in a } b\text{-sector of } p \text{ and } i+1 \text{ is in a } \bar{b}\text{-sector of } p, \end{cases}
$$

$$
M_i(p, q) = s_i(p)(p_i - q_{i-1}) + s_i(q)(q_i - p_{i-1}) - \delta_i(p) - \delta_i(q).
$$

Our main theorem in this section concerns what we refer to as the *mischief between*

$$p\colon \text{0100010011011100101011} \qquad p\colon \text{0100010011011100101011}$$

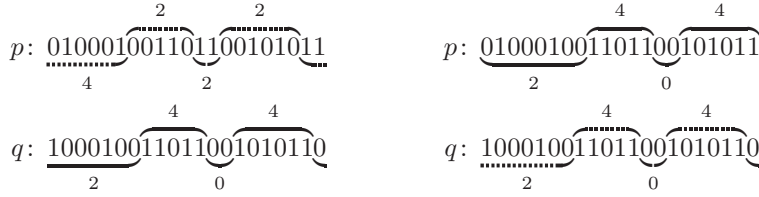$$q\colon \text{1000100110110010101010} \qquad q\colon \text{1000100110110010101010}$$

FIG. 5. *We calculate the mischief between two strings by dividing each into sectors. The misbehavior of p towards q is $-(2-4)+(4-2)-(0-2)+(4-2)=8$, as shown on the left. On the right, we see that the misbehavior of q towards p is 0. Each string has 4 sector switches, so $M(p,q)=8+0-4-4=0$.*

p and q,

$$M(p,q) = \sum_{i=0}^{n-1} M_i(p,q).$$

Mischief, superficially speaking, measures three things: how much p differs from q delayed one time step, which we call the *misbehavior of q towards p*; the reverse, and the total number of sector switches. An example calculation is shown in Figure 5. The rules of the parallel chip-firing game put a global upper bound on the total disagreement between vertices, yet the following theorem states that mischief is nonnegative, meaning that sector switches require disagreement. We show in section 6 that this implies that firing sequences with sector switches are impossible once a game has become periodic.

THEOREM 5.1. *The mischief between any two binary strings of equal length is nonnegative.*

Proof. Let

$$\mu_i(p,q) = (p_{i-1}, p_i, s_i(p), s_{i+1}(p), q_{i-1}, q_i, s_i(q), s_{i+1}(q)),$$

a tuple of all information required to calculate $M_i(p,q)$, and let $\mathscr{G}$ be a weighted digraph with

$$V(\mathscr{G}) = \{\mu_i(p,q) \mid p, q \text{ strings}, i \in \mathbb{N}\},$$
$$E(\mathscr{G}) = \{(\mu_i(p,q), \mu_{i+1}(p,q), M_i(p,q)) \mid p, q \text{ strings}, i \in \mathbb{N}\}.$$

(The third item of each edge is its weight.) Define the weight of a path to be the sum of the weights of its member edges, and call a path negative if it has negative weight. The mischief $M(p,q)$ is the weight of a path in $\mathscr{G}$ induced by the sequence of vertices $(\mu_0(p,q), \ldots, \mu_{n-1}(p,q), \mu_0(p,q))$. Therefore, it suffices to show that $\mathscr{G}$ has no negative cycles—in particular, no negative cycles of length n.

We discuss below the constraints that define what tuples can exist as some $\mu_i(p,q)$. These constraints yield a graph $\mathscr{G}$ with 64 vertices and 256 edges, which is impractical to analyze by hand. However, running the Bellman–Ford algorithm [5] on $\mathscr{G}$ shows it to be free of negative cycles. A Python program specifies both the graph and the algorithm in detail in the appendix. ∎

The only constraint on edges of $\mathscr{G}$ is that the state shared by the connected vertices be consistent. For example, both $\mu_i(p,q)$ and $\mu_{i+1}(p,q)$ contain p_i. There are two constraints on the vertices of $\mathscr{G}$ enforced by the definition of sectors in p. Analogous constraints apply for q.

- If $p_{i-1} = p_i = b$, then $s_i(p)$ indicates that i is in a b-sector of p.
- If $s_i(p) \neq s_{i+1}(p)$, then $p_{i-1} = p_i = b$ such that $s_i(p)$ indicates that i is in a b-sector of p.

To get a sense for how these force nonnegative mischief, consider the latter constraint. The sector switch reduces mischief, but it also makes $p_{i+1} = \bar{b}$. If $q_i = b$, then q misbehaves towards p at $i + 1$. If $q_i = \bar{b}$ and i is in a $\bar{b}$-sector of q, then p misbehaves towards q at i. However, we cannot finish the last case, $q_i = \bar{b}$ with i in a b-sector of q, with similar local reasoning. The former constraint limits the frequency with which that last case can happen, but there are many scenarios to consider before dispatching the case. Using the Bellman–Ford algorithm avoids this further casework.

6. Nonclumpiness of PFPs. We consider parallel chip-firing game σ on undirected graph G. The PFP of a vertex $v \in V(G)$ is the binary string

$$F_{t_0}(v) \dots F_{t_0+T-1}(v),$$

where t_0 is the smallest natural number such that σ_{t_0} is periodic.[1] We write the PFP of v as $F(v)$. For simplicity, we assume here that $t_0 = 0$ and index PFPs modulo T.

Let P be the set of all PFPs that are compatible with σ, which means they have the same number of 0s and 1s as a PFP that occurs in σ. Call a PFP with both consecutive 0s and consecutive 1s *clumpy*, and let Q be the set of clumpy PFPs that are compatible with σ. (Recall that the Tth and 0th entries of a PFP are the same, so, for example, 011010 is clumpy.) It is known that, given almost any[2] nonclumpy PFP, one can construct a parallel chip-firing game on a simple cycle in which every vertex has that PFP shifted by some number of steps [6]. We prove here that clumpy PFPs cannot occur in any parallel chip-firing game.

LEMMA 6.1. *In a periodic game on G, for all $v \in V(G)$ and $a, b \in \mathbb{N}$,*

$$(6.1) \qquad -d(v) + 1 \leq \sum_{t=a}^{b}(\Phi_{t-1}(v) - d(v)F_t(v)) \leq d(v) - 1.$$

Proof. We express $\sigma_b(v)$ in terms of $\sigma_{a-1}(v)$:

$$\sigma_b(v) = \sigma_{a-1}(v) + \sum_{t=a-1}^{b-1}(\Phi_t(v) - d(v)F_t(v)),$$

$$\sigma_b(v) - d(v)F_b(v) = \sigma_{a-1}(v) - d(v)F_{a-1}(v) + \sum_{t=a}^{b}(\Phi_{t-1}(v) - d(v)F_t(v)).$$

As mentioned in [12, section 2], the set of vertices v such that $\sigma_t(v) \geq 2d(v)$ chips is nonincreasing as t increases. In particular, in a periodic game, this set is empty unless the game has period 1, in which case the lemma is clear. Otherwise, simple

---

[1]The reason we introduce PFPs instead of continuing to reason with firing sequences is because a PFP is aware of the period of the game it occurs in. For instance, the PFPs 01 and 0101 result in the same periodic firing sequence, but while the latter, which has period 4, might occur in the same game as the PFP 0011, the former, which has period 2, cannot.

[2]The given construction requires that the PFP not be decomposable to a repeated substring. Using Theorem 4.1, one can expand the construction to any nonclumpy PFP other than those that are 01 or 10 repeated more than once. These PFPs turn out to be impossible, though the corresponding firing sequences are possible in games of period 2.

analysis of the waiting and firing cases shows that $\sigma_t(v) \leq 2d(v) - 1$ implies $0 \leq \sigma_t(v) - d(v)F_t(v) \leq d(v) - 1$, which gives the desired inequality. $\quad\square$

We define

$$\tau(p) = \{v \in V(G) \mid F(v) = p\},$$
$$\pi(p,q) = \{\{v,w\} \in E(G) \mid F(v) = p, F(w) = q\},$$
$$m_S(p,q) = \sum_{i \in S}(p_i - q_{i-1})$$

for binary strings p and q and sets of indices $S \subseteq \mathbb{N}$. Note that $m_S(p,q)$ is very closely related to the misbehavior of q towards p, missing only sector-dependent signs. We now have the tools to prove our main result.

THEOREM 6.2. *Clumpy PFPs do not occur in the parallel chip-firing game.*

Proof. The key is to examine the mischief between all pairs of neighbor vertices in which at least one neighbor has a clumpy PFP. Roughly speaking, summing an inequality derived from Lemma 6.1 over all vertices with clumpy PFPs bounds a negative sum of mischiefs below, and summing the inequality given by Theorem 5.1 over all edges incident with a vertex with a clumpy PFP gives an upper bound on the same sum of mischiefs. The lower bound relates positively with the number of vertices with clumpy PFPs, and the upper bound is 0.

Let $a, b \in \mathbb{N}$ and $v \in V(G)$. Grouping the sum in (6.1) by v's neighbors instead of time steps yields

$$-d(v) + 1 \leq -\sum_{w \in N(v)} m_{[a,b]}(F(v), F(w)) \leq d(v) - 1.$$

Reordering the inequality on the left, recalling that $d(v) = \#N(v)$, gives

$$1 \leq \sum_{w \in N(v)}(1 - m_{[a,b]}(F(v), F(w))),$$

and by considering the right inequality we obtain the same result with the $-$ switched to $+$. Let p be a PFP. By summing the above over $v \in \tau(p)$ for some choice of signs, we obtain

$$\#\tau(p) \leq \sum_{\substack{v \in \tau(p) \\ w \in N(v)}}(1 + r_v m_{[a,b]}(p, F(w))),$$

where each $r_v = \pm 1$ may be chosen dependent on v. (Notation: domains for outer sums are above domains for inner sums.) That each $m_{[a,b]}(p, F(w))$ term is preceded by a sign of our choice is a hint that we can relate this quantity to misbehavior.

For all PFPs p, let $\mathcal{X}(p)$ be the set of sectors of p. Because each sector of a PFP is of the form $[a,b]$ for some $a, b \in \mathbb{N}$, we can sum the above inequality over $X \in \mathcal{X}(p)$ and $p \in Q$ to get

(6.2) $$\sum_{\substack{p \in Q \\ X \in \mathcal{X}(p)}} \#\tau(p) \leq \sum_{\substack{p \in Q \\ v \in \tau(p) \\ w \in N(v) \\ X \in \mathcal{X}(p)}}(1 + r_{v,X} m_X(p, F(w))),$$

where each $r_{v,X} = \pm 1$ may be chosen dependent on v and X.

Let p be a clumpy PFP. If q is a clumpy PFP, then

$$(6.3) \qquad M(p,q) = \sum_{X \in \mathcal{X}(p)} (s_X(p)m_X(p,q) - 1) + \sum_{X \in \mathcal{X}(q)} (s_X(q)m_X(q,p) - 1).$$

(Abusing notation slightly, we write $s_X(p)$ instead of $s_i(p)$ if $i \in X \in \mathcal{X}(p)$.) The -1 in each sum accounts for the $-\delta_i(p) - \delta_i(q)$ term in $M_i(p,q)$. If instead q is not clumpy, then $\mathcal{X}(q) = \{[0, T-1]\}$, so q has no sector switches and

$$(6.4) \qquad M(p,q) = \sum_{X \in \mathcal{X}(p)} (s_X(p)m_X(p,q) - 1) + s_{[0,T-1]}(q)m_{[0,T-1]}(q,p).$$

However, $m_{[0,T-1]}(q,p) = 0$ because p and q have the same length and number of 1s. This means that the mischief between p and q only depends on the misbehavior of q towards p and sector switches in p. This is important because Lemma 6.1 bounds total misbehavior towards a vertex from all neighbors.

Let $W = \{v \in V(G) \mid F(v) \in Q\}$ be the set of vertices with clumpy PFPs. Choosing $r_{v,X} = -s_X(p)$ in (6.2) yields

$$\sum_{\substack{p \in Q \\ X \in \mathcal{X}(p)}} \#\tau(p) \leq \sum_{\substack{p \in Q \\ v \in \tau(p) \\ w \in N(v) \\ X \in \mathcal{X}(p)}} (1 - s_X(p)m_X(p, F(w)))$$

$$= \sum_{\substack{p \in Q \\ v \in \tau(p) \\ w \in N(v) \cap W \\ X \in \mathcal{X}(p)}} (1 - s_X(p)m_X(p, F(w))) + \sum_{\substack{p \in Q \\ v \in \tau(p) \\ w \in N(v) \setminus W \\ X \in \mathcal{X}(p)}} (1 - s_X(p)m_X(p, F(w)))$$

$$= \sum_{\substack{p,q \in Q \\ e \in \pi(p,q)}} \left(\sum_{X \in \mathcal{X}(p)} (1 - s_X(p)m_X(p,q)) + \sum_{X \in \mathcal{X}(q)} (1 - s_X(q)m_X(q,p)) \right)$$

$$+ \sum_{\substack{p \in Q \\ v \in \tau(p) \\ w \in N(v) \setminus W \\ X \in \mathcal{X}(p)}} (1 - s_X(p)m_X(p, F(w))).$$

Note that we consider p and q in the sum over $p, q \in Q$ to be unordered. (One alternative notation is a sum over $\{p, q\} \subseteq Q$.) We now substitute using (6.3) and (6.4) and apply Theorem 5.1 to get

$$\sum_{\substack{p \in Q \\ X \in \mathcal{X}(p)}} \#\tau(p) \leq - \sum_{\substack{p,q \in Q \\ e \in \pi(p,q)}} M(p,q) - \sum_{\substack{p \in Q \\ v \in \tau(p) \\ w \in N(v) \setminus W}} M(p, F(w)) \leq 0.$$

Sets have nonnegative sizes, so $\#\tau(p) = 0$ for all $p \in Q$. $\qquad\square$

7. Implications of nonclumpiness. It is a basic property of the parallel chip-firing game that every vertex fires the same number of times each period [10]. This means, roughly speaking, that every periodic game is either "mostly waiting" with bursts of firing or "mostly firing" with bursts of waiting. (In fact, there is a bijection between these two types of games. Each periodic game has a complement that inverts

firing and waiting [10].) This is because if a vertex waits twice in a row, then because it therefore never fires twice in a row, it fires less than half the time over the course of a period. Similarly, a vertex that fires twice in a row fires more than half the time. We cannot have a vertex that waits twice in a row and a vertex that fires twice in a row in the same periodic game because each vertex fires the same number of times each period.

COROLLARY 7.1. *Once a parallel chip-firing game is periodic, either no vertex fires twice in a row or no vertex waits twice in a row.*

That is, in periodic games, a firing sequence is possible if and only if it is non-clumpy.

The *interior* of a set of vertices W is $\{v \in W \mid N(v) \subseteq W\}$. Because a waiting (or firing) vertex with only waiting (or firing) neighbors will wait (or fire) the following turn as well, the above observation proves the following conjecture of Fey and Levine [7].

COROLLARY 7.2. *Once a parallel chip-firing game is periodic, the interior of the set of waiting vertices is always empty, the interior of set of firing vertices is always empty, or both interiors are always empty.*

Interestingly, Corollary 7.2 also implies Theorem 6.2. If clumpy PFPs were possible, then a leaf attached to a motor with a clumpy PFP would be at different times in both the waiting and firing interiors.

In one of the first papers on the parallel chip-firing game, Bitar and Goles characterized parallel chip-firing games on trees [4]. Corollary 3.3 and Theorem 6.2 allow us to characterize the behavior on tree-like subgraphs—subgraphs that, if an edge to a root vertex is cut, become a tree separated from the rest of the graph—by making the root vertex a motor.

COROLLARY 7.3. *Let σ be a periodic game on G with period at least 3 in which no vertex fires twice in a row, H be a tree-like subgraph of G, and $m \in V(H)$ be the root of H. Then for all $v \in V(H)$,*

$$\sigma_t(v) = \begin{cases} d(v) & \text{if } F_{t-D}(m) = 1, \\ 0 & \text{if } F_{t-D-1}(m) = 1, \\ d(v) - 1 & \text{otherwise,} \end{cases}$$

where D is the distance from m to v. An analogous result holds if no vertex waits twice in a row.

In some sense, tree-like subgraphs are passive in that their vertices fire only in response to their rootside neighbor firing. In a periodic game, we can completely remove tree-like subgraphs without affecting the PFPs of the other vertices.

COROLLARY 7.4. *Let σ be a periodic game on G, leaf $l \in V(G)$ have single neighbor m, and G' be G with l deleted. Then a game σ' exists on G' with the same firing behavior as σ.*

The starting position σ'_0 can agree with σ_0 completely except for possibly removing a chip from m. We consider the case where no vertex fires twice in a row. Compared to σ', vertex m has to have an extra chip to fire in σ. However, unless m fired the previous turn—which, because l is a leaf, is equivalent to saying l is firing this turn—m will have received the extra chip back from l, so removing both l and the chip has no effect on m as long as m does not fire while l has a chip, which doesn't happen due to nonclumpiness. The case where no vertex waits twice in a row is analogous. This corollary concerns a leaf, though the result generalizes to all tree-like subgraphs by repeated application, providing an alternate proof of Corollary 3.3.
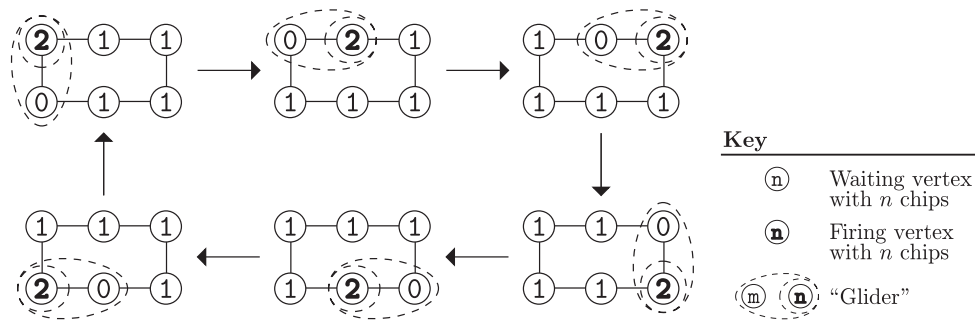
FIG. 6. *A game on a 6-cycle in which a glider orbits once each period.*

8. Discussion and directions for future work. We have introduced motors, studied motorized games on trees, and shown that motor-like behavior can be constructed in ordinary games, provided that each motor has a possible firing sequence. We then showed that PFPs are possible if and only if they are nonclumpy, which, among other things, allows classification of periodic games as "mostly waiting" or "mostly firing" and the removal of tree-like subgraphs without loss of generality.

We might expect the space of motorized games to be larger than that of ordinary games. Theorem 4.1 shows us that, as long as the firing sequences involved are possible, the parallel chip-firing game is in some sense just as "expressive" as its motorized variant. This allows, for example, the simulation of some aspects of the *dollar game*, a variant of the general chip-firing game discussed by Biggs [3]. In the dollar game, exactly one vertex, the "government," may have a negative number of chips and fires if and only if no other vertices can fire. We can construct a motorized parallel chip-firing game in which we replace the government with a motor that waits a sufficiently large number of steps between each firing such that it never fires in the same step as another vertex. Biggs showed that every dollar game tends towards a critical position regardless of the order of vertex firings, so this motorized parallel chip-firing game tends towards the same critical position. Theorem 4.1 may help reveal the extent to which the parallel chip-firing game can simulate additional aspects of the dollar game and other general chip-firing games.

Despite the expressiveness we get due to motors, the nonclumpiness of firing patterns tells us that the parallel chip-firing game is "easier" than its rules explicitly tell us it must be. In addition to results mentioned in section 7, Theorem 6.2 is a step towards reducing the parallel chip-firing game to one of interacting "gliders." For example, consider the situation in Corollary 3.3. Intuitively, we can think of this corollary as stating that each firing of the motor creates a wave of gliders that travels away from the motor. In fact, the corollary, together with Theorem 6.2, implies that every periodic position on tree-like subgraphs must be the result of such gliders, providing a new test that can diagnose some positions that are never repeated. Every game on a simple cycle with period at least 3 can be described by gliders [6]. (See Figure 6.) We believe that this approach could be used to analyze periodic behavior of games on further classes of graphs, such as those in which each vertex is in at most once cycle.

Nonclumpiness is essentially an unwritten rule of periods in the parallel chip-firing game, which is unusual because no local property of the firing mechanism disallows clumpiness. By contrast, in other graph automata that are more restrictive than the parallel chip-firing game, such as *source reversal* [9] (essentially a parallel chip-firing game with exactly one chip bound to each edge), nonclumpiness is obvious, even

locally. In the other direction, motors make it simple to show that certain stronger restrictions do not apply to the parallel chip-firing game. For example, a path where the leaves are motors can yield a game in which some chips cannot be bound to a single edge, which is a property of source reversal. We might ask which restrictions apply to which chip-firing-style games. Is the parallel chip-firing game on undirected graphs the most general game to which an analogue of Theorem 6.2 applies?

We hope that the intuition and constructive powers of motors and the reduction in the space of possible periodic games provided by nonclumpiness prove useful in further research.

Appendix: The Bellman–Ford Algorithm. The following Python program below constructs $\mathscr{G}$ from section 5 and shows it has no negative cycles. For a more detailed exposition of the Bellman–Ford algorithm and a proof of its validity, see [5].

```python
infty = float("inf")

class Vertex:
    def __init__(self):
        self.distance = infty

class Edge:
    def __init__(self, tail, head, weight):
        self.tail = tail
        self.head = head
        self.weight = weight

# Returns True if graph has a negative cycle or unreachable vertices.
def bellman_ford(verts, edges):
    # For our purposes, the start vertex is arbitrary.
    edges[0].tail.distance = 0
    # Find minimum length from the start vertex to each other vertex.
    for i in range(len(verts) - 1):
        for e in edges:
            new_distance = e.tail.distance + e.weight
            if new_distance < e.head.distance:
                e.head.distance = new_distance
    # Find triangle inequality failures caused by negative cycles.
    # Also confirms we searched entire graph (infty < infty == False).
    for e in edges:
        if e.tail.distance + e.weight < e.head.distance:
            return True
    return False

# Throughout, we use 0 instead of -1 for s_i(p), correcting when needed.

# Checks if mu_i(p,q) is compatible with the definition of a sector.
def valid_vert(mu1):
    (p0, p1, s1, s2, q0, q1, t1, t2) = mu1
    return (p0==s1 if p0==p1 else s1==s2) and (q0==t1 if q0==q1 else t1==t2)

# Checks if the first state can be followed by the second.
def valid_edge(mu1, mu2):
    (p0a, p1a, s1a, s2a, q0a, q1a, t1a, t2a) = mu1
    (p1b, p2b, s2b, s3b, q1b, q2b, t2b, t3b) = mu2
    return p1a==p1b and q1a==q1b and s2a==s2b and t2a==t2b

# Calculates M_i(p,q)
def weight(mu1):
    (p0, p1, s1, s2, q0, q1, t1, t2) = mu1
    return (2*s1-1)*(p1-q0) + (2*t1-1)*(q1-p0) - abs(s1-s2) - abs(t1-t2)
```

```
bits = {x : tuple((x // 2**i) % 2 for i in range(8)) for x in range(2**8)}
V = {bits[i]: Vertex() for i in range(2**8) if valid_vert(bits[i])}
E = [Edge(V[u], V[v], weight(u)) for u in V for v in V if valid_edge(u, v)]

# And Theorem 5.1 is...
print(not bellman_ford(V, E))
```

REFERENCES

[1] P. BAK, C. TANG, AND K. WIESENFELD, *Self-organized criticality: An explanation of the 1/f noise*, Phys. Rev. Lett., 59 (1987), pp. 381–384.

[2] P. BAK, C. TANG, AND K. WIESENFELD, *Self-organized criticality*, Phys. Rev. A (3), 38 (1988), pp. 364–374.

[3] N. L. BIGGS, *Chip-firing and the critical group of a graph*, J. Algebraic Combin., 9 (1999), pp. 25–45.

[4] J. BITAR AND E. GOLES, *Parallel chip firing games on graphs*, Theoret. Comput. Sci., 92 (1992), pp. 291–300.

[5] T. H. CORMEN, C. E. LEISERSON, R. L. RIVEST, AND C. STEIN, *Introduction to Algorithms*, 3rd ed., MIT Press, Cambridge, MA, 2009.

[6] L. DALL'ASTA, *Exact solution of the one-dimensional deterministic fixed-energy sandpile*, Phys. Rev. Lett., 96 (2006), 058003.

[7] A. FEY AND L. LEVINE, *private communication*, 2011.

[8] E. GOLES AND M. MARGENSTERN, *Universality of the chip-firing game*, Theoret. Comput. Sci., 172 (1997), pp. 121–134.

[9] E. GOLES AND E. PRISNER, *Source reversal and chip firing on graphs*, Theoret. Comput. Sci., 233 (2000), pp. 287–295.

[10] T.-Y. JIANG, *On the Period Lengths of the Parallel Chip-Firing Game*, preprint, arXiv:1003.0943, 2010.

[11] M. A. KIWI, R. NDOUNDAM, M. TCHUENTE, AND E. GOLES, *No polynomial bound for the period of the parallel chip firing game on graphs*, Theoret. Comput. Sci., 136 (1994), pp. 527–532.

[12] P. M. KOMINERS AND S. D. KOMINERS, *Candy-Passing Games on General Graphs,* II, preprint, arXiv:0807:4450v1 [math.CO], 2008.

[13] L. LEVINE, *Parallel Chip-Firing on the Complete Graph: Devil's Staircase and Poincaré Rotation Number*, preprint, arXiv:0811.2800, 2008.