On the Complexity of the Abelian Sandpile Model: Communication Complexity and Statistical Mechanics

J. Andres Montoya

Departamento de Matemáticas
Universidad Nacional de Colombia
Bogota, Colombia
jamontoyaa@unal.edu.co

In this paper, the complexity of recognizing the critical configurations of the two-dimensional Abelian sandpile model is studied, some known facts are reviewed, and a simplified proof of the burning test is presented. Then, the existence of sublinear time algorithms solving the aforementioned problem is studied, with a lower bound for the monotone complexity of the problem established by employing some tools of communication complexity.

1. Introduction

The Abelian sandpile model has been intensively studied in the physics milieu since its introduction by Bak et al. [1]. This model allows us to study some qualitative properties of dissipative dynamical systems such as forest fires, earthquakes, extinction events, the dynamics of the stock market, and avalanches [1].

Moore and Nilsson define the sandpile prediction problem [2], and they asked for a precise quantification of its algorithmic complexity. In this paper, we investigate, in some depth, a closely related problem, the so-called recurrence recognition problem. The algorithmic complexity of those two problems is well understood in any dimension other than two. We focus our attention on the two-dimensional case.

It is known that the two-dimensional recurrence recognition problem is logspace reducible to the prediction problem. We prove that the two-dimensional recognition problem requires $\Omega(n)$ depth on the monotone circuit model. It suggests that this problem is P-complete. It has been conjectured that the two-dimensional prediction problem is P-complete, but a proof (or a refutation) of this fact remains elusive. Our main result provides strong evidence in favor of the aforementioned conjecture. Our proof uses some tools and concepts of communication complexity.

Complex Systems, **21**
https://doi.org/10.25088/ComplexSystems.21.4.245

Discrete complex systems have been employed as models of natural phenomena. If a discrete complex system is to be used as a model, there has to be some control over the model: the model cannot be as complex as the phenomena being modeled, because the chart is not the territory. Researchers working in the field of discrete complex systems have paid some attention to the algorithmic hardness of their models. Most of the studies concerning the algorithmic hardness of complex systems are based on the classical notion of P completeness and the related notion of $\mathcal{NC}$ reducibility. Few works employ alternative tools and concepts. It should be clear that the very notion of $\mathcal{NC}$ reducibility is not sufficient to deal with all the complexity theoretical issues related to complex systems.

We prove some additional facts and discuss some alternative approaches to the complexity theoretic analysis of the two-dimensional Abelian sandpile model.

1.1 Previous Work and Contributions

Moore and Nilsson introduced the sandpile prediction problem and began the analysis of its algorithmic complexity [2]. They proved that the three-dimensional versions of the sandpile prediction and recurrence recognition problems are P-complete. Miltersen [3] studied one-dimensional sandpiles; he proved that the one-dimensional version of the sandpile prediction problem belongs to $\mathcal{NC}^2$ and is TC^0-hard. Those two papers left open the question concerning the algorithmic complexity of two-dimensional sandpiles (the best bounds are *ptime* computability and $\mathcal{NC}^1$ hardness, which are very far from being tight). In this paper, we begin a systematic analysis of the algorithmic complexity of the two-dimensional Abelian sandpile model. We focus our analysis on the recurrence recognition problem, which we denote with the symbol $RR[2]$. First, we derive, from a few basic principles, the algebraic theory of recurrent configurations. After that, we begin a systematic search for sublinear time algorithms solving the problem $RR[2]$. We prove that the elementary approaches to this goal are doomed to fail. We conclude with a theorem claiming that any uniform polynomial-size family of monotone circuits solving the problem $RR[2]$ requires depth $\Omega(n)$.

Remark 1. Given $i \geq 1$, the class $\mathcal{NC}^i$ is the class of problems that can be solved employing a uniform polynomial-size family of Boolean circuits of depth $O(\log^i)$. We can identify this class with the class of problems that can be solved in time $O(\log^i)$ employing a parallel random access machine (i.e., employing an idealized parallel computer with unit communication cost). Thus, the class $\mathcal{NC}$, defined as

https://doi.org/10.25088/ComplexSystems.21.4.245

$\bigcup_{i\geq 1} \mathcal{NC}^i$, corresponds to the class of problems that can be solved in polylogarithmic parallel running time.

2. The Abelian Sandpile Model

In this section, we introduce the basic definitions and some of the basic results concerning the Abelian sandpile model.

Given G, a finite connected graph, and given $s \in V(G)$, we say that the pair (G, s) is a *sandpile pair*. We use the symbol $V(G)^*$ to denote the set $V(G) \setminus \{s\}$ and we say that s is the *sink* of (G, s). A configuration of the pair (G, s) is a function $f : V(G)^* \to \mathbb{N}$. Given f, a configuration of (G, s), and given $w \in V(G)^*$, we say that w is f-stable if and only if $f(w) < \deg_G(w)$, where $\deg_G(w)$ is the degree of w as a node of G. We say that f is a stable configuration if and only if all the nodes in $V(G)^*$ are f-stable. The dynamics of the Abelian sandpile model on (G, s) are given by the *toppling rule* defined as follows.

Given $v \in V(G)^*$ such that $g(v) \geq \deg_G(v)$, $f \to f_v$ is a possible transition, where f_v is the configuration of (G, s), defined by

$$f_v(w) := \begin{cases} f_v(w) - \deg_G(w) & \text{if } v = w \\ f_v(w) + \#(\text{edges connecting } v \text{ and } w), & \text{otherwise.} \end{cases}$$

A transition $f \to f_v$ is called a *toppling* (or a *firing*), and if such a transition occurs we say that node v was toppled (fired). Note that when node v is toppled, it sends one grain of sand along each one of the edges that are incident to it. The sink never topples; we can think of a sandpile pair (G, s) as a table with the elements of $V(G)^*$ representing the sites on the table where piles of sand can be placed and the sink s representing the empty space surrounding the table. Once a grain falls off of the table, it cannot go back.

Given G, a sandpile lattice, and given, f an unstable configuration of (G, s), we can choose an unstable node, fire it, and obtain a new configuration. A sequence of firings $f_1 \to f_2 \to \cdots \to f_m$ is called an *avalanche* of length $m-1$ with initial configuration f_1, and we say that it is an avalanche from f_1 to f_m. If f_m is stable, we say that f_m is a *stabilization* or a *relaxation* of f_1. Given f, a configuration of (G, s), we use the symbol $\mathcal{ST}(G, f)$ to denote the set of relaxations of f. Furthermore, given G, f, and an avalanche $A = f \to f_1 \to \cdots \to f_m$, the *score vector* of A, which we denote with the symbol SC_A, is equal to $(t_v)_{v \in V(G)^*}$.

https://doi.org/10.25088/ComplexSystems.21.4.245

Remark 2. Given an avalanche $f_1 \to f_2 \to \cdots \to f_m$, we say that it is a maximal avalanche if and only if f_m is stable.

Theorem 1. The fundamental theorem of sandpiles.

Let (G, s) be a sandpile pair and let f be a configuration of (G, s):

1. Any avalanche beginning in f is finite.
2. $|\mathcal{ST}(G, g)| = 1$.
3. Given two maximal avalanches beginning in f, A and B, $SC_A = SC_B$.

A proof of this theorem can be found in [4]. Note that the configuration reached after a maximal avalanche only depends on the initial configuration; it does not depend on the order of topplings. This is the reason we call the model Abelian.

Remark 3. Given $\mathcal{C}(G) = \mathbb{N}^{V(G)^*}$, the set of all the configurations of (G, s), and $g \in \mathcal{C}(G)$, we use the symbol SC_g to denote the vector SC_A, where A is any maximal avalanche beginning in g.

Let $\mathcal{ST}(G)$ be the set of all the stable configurations of (G, s). We can define a function $st_G : \mathcal{C}(G) \to \mathcal{ST}(G)$ in the following way:

$st_G(g) :=$ the stabilization of g.

Note that, for all pairs (G, s), the function st_G is computable: given f, a configuration of (G, s), to compute $st_G(f)$, you only have to simulate the dynamics of the Abelian sandpile model determined by the triple (G, s, f).

Given a sandpile pair (G, s) and three configurations f_1, f_2, and f_3, we know that

$$st_G(f_1 + f_2 + f_3) = st_G(st_G(f_1 + f_2) + f_3). \tag{1}$$

We can associate a sandpile monoid with any sandpile pair. To this end, we define a binary operation $\oplus : \mathcal{ST}(G)^2 \to \mathcal{ST}(G)$ in the following way:

$$f \oplus g = st_G(f + g) = st_G(f) \oplus st_G(g).$$

Equation (1) implies that this algebraic operation is associative. Thus, the pair $(\mathcal{ST}(G), \oplus)$ is a finite commutative monoid. We use the name *sandpile monoid of* (G, s) to denote the pair $\mathcal{M}(G) = (\mathcal{ST}(G), \oplus)$.

2.1 The Two-Dimensional Abelian Sandpile Model

Given $n \geq 1$, we use the symbol $\mathcal{G}_n$ to denote the *two-dimensional square lattice* of order n, that is, we use the symbol $\mathcal{G}_n$ to denote the square lattice whose vertex set is the set $[n] \times [n]$. We use the symbol

https://doi.org/10.25088/ComplexSystems.21.4.245

$[n]$ to denote the set $\{1, \ldots, n\}$ and we use the symbol $\mathcal{L}_n$ to denote the *two-dimensional sandpile lattice* of order n, which is obtained from $\mathcal{G}_n$ by adding a special node s called the sink. Furthermore, given v, a node on the border of $\mathcal{G}_n$, there are $4 - \deg_{\mathcal{G}_n}(v)$ edges in $\mathcal{L}_n$ connecting v and s. We use the symbol $V(\mathcal{L}_n)^*$ to denote the set $V(L_n) - \{s\} = V(\mathcal{G}_n)$. Note that for all $v \in V(\mathcal{L}_n)^*$, $\deg(v) = 4$.

Remark 4. From now on, we use the symbol $\mathcal{L}_n$ to denote the sandpile pair $(\mathcal{L}_n, s)$.

In Section 2.2, we begin the analysis of the algorithmic complexity of the two-dimensional Abelian sandpile model.

2.2 The Prediction Problem

The two-dimensional sandpile prediction problem is the algorithmic problem defined by Problem 1.

Problem 1. *SPP*[2], two-dimensional sandpile prediction problem.

- Input: (n, f), where f is a configuration of $\mathcal{L}_n$.
- Problem: compute $st_n(f)$.

Remark 5. From now on, we use the symbol st_n to denote the function $st_{\mathcal{L}_n}$.

Definition 1. Given f, a configuration of $\mathcal{L}_n$, we use the symbol $\|f\|$ to denote the weight of f, which is equal to $\sum_{v \in V(\mathcal{L}_n)^*} f(v)$. Note that the weight of a configuration is simply the total amount of sand.

Tardos bound [5] implies that given f, a configuration of $\mathcal{L}_n$, the length of the avalanches triggered by f has an upper bound of $n^3 \|f\|$. It implies that the problem *SPP*[2] can be solved in polynomial time using a naive simulation algorithm.

On the other hand, it is known [2] that *SPP*[2] is NC^1-hard: the evaluation of monotone planar circuits is logspace reducible to *SPP*[2].

Those two bounds are the best upper and lower bounds for the algorithmic complexity of the problem *SPP*[2]. It is clear that those bounds are very far from being tight. Also, we are far away from a suitable quantification of the algorithmic hardness of the problem *SPP*[2].

In this paper, we begin a systematic analysis of the algorithmic hardness of the two-dimensional Abelian sandpile model. We focus our research on the recognition of two-dimensional recurrent configurations.

https://doi.org/10.25088/ComplexSystems.21.4.245

3. The Recognition of Recurrent Configurations

In this section we introduce the two-dimensional recurrent recognition problem.

We can associate a Markov chain with the Abelian sandpile model, which has been extensively studied as a simple model of self-organized criticality. Let (G, s) be a sandpile pair. We associate with (G, s) the Markov chain $\mathcal{CH}(G) = (\mathcal{ST}(G), \{X_i\}_{i\geq 1})$ defined by:

- $X_0 = Z_G$, where Z_G is the zero configuration of G, which is defined by: for all $v \in V(G)^*$, we have $X_0(v) = 0$.
- Given X_i, we choose uniformly at random a node $v \in V(G)^*$ and we make $X_{i+1} = X_i \oplus e_v$, where e_v is the configuration defined by

$$e_v(w) = \begin{cases} 1, & \text{if } v = w \\ 0, & \text{otherwise.} \end{cases}$$

Remark 6. We use the symbol $\mathcal{ST}(n)$ to denote the set $\mathcal{ST}(\mathcal{L}_n)$.

Definition 2. A configuration $f \in \mathcal{ST}(n)$ is recurrent if and only if

$$\Pr\left[|\{i : X_i = f\}| = \infty\right] = 1.$$

Recurrent configurations can be defined from an algebraic point of view. It is known that a configuration f is recurrent if and only if there exists a nonnull configuration g such that the equation $f \oplus g = f$ holds [4]. We have chosen to work with the above definition given that it implies that the set of recurrent configurations is the steady state of the system, that is, the set of recurrent configurations encodes the long-term behavior of the system. We use the symbol $\mathcal{K}(n)$ to denote the set of recurrent configurations of $\mathcal{L}_n$. Recurrent configurations are very important in the theory because they encode the long-term behavior of the system: the elements of $\mathcal{K}(n)$ constitute the stationary state of the chain $\mathcal{CH}(n)$.

We consider, in this section, the following problem: how can we recognize the recurrent configurations of a sandpile lattice $\mathcal{L}_n$? First, we introduce the formal definition of the two-dimensional recurrence recognition problem.

Problem 2. $RR[2]$, recognition of two-dimensional recurrent configurations.

- Input: (n, f), where $n \geq 1$ and f is a stable configuration of $\mathcal{L}_n$.
- Problem: decide if f is recurrent.

In the following, we prove that there exists a polynomial time algorithm solving the problem $RR[2]$; to this end, we exhibit a linear time

https://doi.org/10.25088/ComplexSystems.21.4.245

recognition algorithm for the set of two-dimensional recurrent configurations and prove that this algorithm is correct. Our algorithm is the burning test algorithm of Dhar [4]. We have included a full proof of this theorem given that we believe it is a simplified proof of the burning test.

Let δ_n be the *border configuration* of $\mathcal{L}_n$, which is the configuration defined by

$$\delta_n(v) = \#\text{ of edges connecting } v \text{ with the sink of } \mathcal{L}_n.$$

Lemma 1. Given f, a stable configuration of $\mathcal{L}_n$, and given a node v, we have $SC_{f+\delta_n}(v) \leq 1$.

Proof. We use the symbol M_n to denote the *maximal configuration* of $\mathcal{L}_n$, which is the configuration defined by

$$\text{given } v \in V(\mathcal{L}_n)^*,\ M_n(v) = 3.$$

Given two configurations f, g, we use the symbol $f \leq g$ to indicate that for all $v \in V(\mathcal{L}_n)^*$, the inequality $f(v) \leq g(v)$ holds. Note that for all $f \in \mathcal{ST}(n)$, we have $f \leq M_n$. Let v be a node of $\mathcal{L}_n$ and let $h \leq t$ be two configurations of $\mathcal{L}_n$. We have $SC_h(v) \leq SC_t(v)$. It implies that for all $f \in \mathcal{ST}(n)$ and for all $v \in V(\mathcal{L}_n)^*$,

$$SC_{f+\delta_n}(v) \leq SC_{M_n+\delta_n}(v).$$

It can be checked (using induction on n) that for all $n \geq 1$ and for all $v \in V(\mathcal{L}_n)^*$, the equality $SC_{M_n+\delta_n}(v) = 1$ holds. Thus, given f, a stable configuration of $\mathcal{L}_n$, and v, a node of $\mathcal{L}_n$, we have $SC_{f+\delta_n}(v) \leq 1$. □

Corollary 1. Given f, a stable configuration of $\mathcal{L}_n$, we have $f \oplus \delta_n = f$ if and only if for all $v \in V(\mathcal{L}_n)^*$ the equality $SC_{f+\delta_n}(v) = 1$ holds.

Remark 7. We have remarked that for all $v \in V(\mathcal{L}_n)^*$, the equation $SC_{M_n+\delta_n}(v) = 1$. It clearly implies that $M_n \oplus \delta_n = M_n$.

Theorem 2. There exists a linear time recognition algorithm for the set of recurrent configurations.

Proof. Given f, a configuration of $\mathcal{L}_n$, we say that f is a *critical configuration* if and only if there exists a configuration h such that $M_n \oplus h = f$. It is clear that a configuration f is recurrent if and only if it is critical. Also, to prove the theorem we only have to design a linear time algorithm recognizing the set of critical configurations.

https://doi.org/10.25088/ComplexSystems.21.4.245

Consider the sequence $\mathcal{S}_n = \{f_i\}_{i\geq 1}$ defined by:

- $f_1 = \delta_n$.
- $f_{i+1} = f_i \oplus \delta_n$.

It is known that for all $n \geq 1$, the sequence $\mathcal{S}_n$ reaches a fixed point [4]. Let e_n be the fixed point of the sequence $\mathcal{S}_n$. It follows, from the definition of e_n, that there exists $N_n \in \mathbb{N}$ such that for all $m \geq N_n$ the equality $st_n(m\,\delta_n) = e_n$ holds. Babai and Gorodezky [6] proved that there exists a positive constant C such that for all $n \geq 1$ and for all configurations of f, if $\|f\| \geq Cn^{30}$, then $st_n(f)$ is a recurrent configuration of the sandpile lattice $\mathcal{L}_n$. The last two facts imply that the configuration e_n is a recurrent one. Let f be a recurrent configuration and let h be a configuration such that $M_n \oplus h = f$. Note that

$$\delta_n \oplus f = \delta_n \oplus (M_n \oplus h) = (\delta_n \oplus M_n) \oplus h = M_n \oplus h = f.$$

That is, if f is recurrent, the equality $\delta_n \oplus f = f$ holds.

Now we pick a configuration g such that the equality $\delta_n \oplus f = f$ holds, and we ask: is g recurrent? Does the invariance under the action of δ_n characterize the set of recurrent configurations? Note that

$$e_n \oplus g = N_n\,\delta_n \oplus g = (N_{n-1}\,\delta_n) \oplus (\delta_n \oplus g) = (N_{n-1}\,\delta_n) \oplus g = (N_{n-2}\,\delta_n) \oplus g = (N_{n-3}\,\delta_n) \oplus g = \cdots = \delta_n \oplus g = g.$$

That is, if g is invariant under the action of δ_n, it is invariant under the action of e_n as well. Recall that the pair $\mathcal{M}(n) = (\mathcal{ST}(n), \oplus)$ is a finite commutative monoid. Let I be an ideal of $\mathcal{M}(n)$ and let f be an element of I. Note that $M_n = f \oplus (M_n - f)$; it implies that $M_n \in I$ and that the set of critical configurations is a subset of I. Let $\mathcal{K}(n)$ be the set of critical configurations; it follows, from the pure definition of $\mathcal{K}(n)$, that $\mathcal{K}(n)$ is an ideal of $\mathcal{M}(n)$. Then, $\mathcal{K}(n)$ is an ideal that is contained in any nonempty ideal of $\mathcal{M}(n)$, that is, $\mathcal{K}(n)$ is the *kernel* of $\mathcal{M}(n)$. Given $\mathcal{M}$, a finite commutative monoid, its kernel (i.e., the intersection of its nonempty ideals) constitutes an Abelian subgroup [7], then the pair $\mathcal{K}(n) = (C(n), \oplus)$ is an Abelian group. Recall that e_n is a critical configuration, and recall that the set of critical configurations is an Abelian group. It implies that there exists a configuration h_n such that $M_n \oplus h_n = e_n$. Let g be a configuration that is invariant under the action of e_n:

$$g = g \oplus e_n = g \oplus (M_n \oplus h_n) = M_n \oplus (g \oplus h_n).$$

Then we know that the configuration g is recurrent. Thus, we have proven that given f a configuration of $\mathcal{L}_n$, the configuration f is recurrent if and only if the equality $f \oplus \delta_n = f$ holds.

https://doi.org/10.25088/ComplexSystems.21.4.245

Lemma 1 and its corollary imply that given f, a configuration of $\mathcal{L}_n$, the configuration f is recurrent if and only if for all $v \in V(\mathcal{L}_n)^*$ the equality $SC_{f+\delta_n}(v) = 1$ holds.

We can derive from the last fact a linear time algorithm solving the problem $RR[2]$. This algorithm was discovered by Dhar [4] and is called the burning test algorithm. We use the symbol $\mathcal{B}$ to denote Dhar's algorithm. Algorithm $\mathcal{B}$ works on input (n, f) as follows:

1. $\mathcal{B}$ simulates one of the maximal avalanches triggered by $f + \delta_n$.
2. $\mathcal{B}$ counts the number of firings that occurred at each one of the nodes of $\mathcal{L}_n$.
3. If there exists $v \in V(\mathcal{L}_n)^*$ such that $SC_{f+\delta_n}(v) = 0$, algorithm $\mathcal{B}$ rejects the input; otherwise it accepts.

It should be clear that algorithm $\mathcal{B}$ solves the problem $RR[2]$ in linear time. □

Do there exist more efficient algorithms to solve the problem $RR[2]$? It can be argued that the burning test algorithm of Dhar is a real-time algorithm. To beat a real-time algorithm a sublinear time algorithm must be designed to solve the same problem. Also, we ask: do there exist sublinear time algorithms for the problem $RR[2]$?

4. Sublinear Time Algorithms for *RR*[2]

Does there exist a sublinear-time parallel algorithm for solving RR[2]? A first natural attempt is to optimize (parallelize) the burning test algorithm of Dhar. Let (n, f) be an input of $\mathcal{B}$. The core of $\mathcal{B}$ is the computation of $f \oplus \delta_n$. Algorithm $\mathcal{B}$ simulates the avalanche triggered by $f + \delta_n$, employing the sequential updating protocol (we pick an unstable node and we fire it, that is, we fire exactly one node per iteration). A naive improvement of $\mathcal{B}$ consists in employing the *parallel updating protocol*: we fire all the unstable nodes at once, instead of firing a single unstable node per iteration.

Is there an upper bound on the number of iterations that are $o(n^2)$, that is, less than cn^2 for any constant c? Unfortunately, the answer is no.

Theorem 3. The worst-case running time of the parallel updating simulation algorithm is $\Omega(n^2)$.

https://doi.org/10.25088/ComplexSystems.21.4.245

Proof. There exists a bijection between the set of critical configurations of $\mathcal{L}_n$ and the set of spanning trees of $\mathcal{L}_n$ rooted at s [4]. The bijection can be computed in the following way.

Let f be a critical configuration and compute a rooted tree (T_f, s) as follows.

1. Fix a linear ordering of $V(\mathcal{L}_n)^*$.
2. Introduce three variables E_T, V_T, and x.
3. Set $V_T = \{v : v \text{ is } f + \delta_n\text{-unstable}\}$, $x = s$ and $E_T = \{\{s, v\} : v \in V_T\}$.
4. Look for the first node in V_T, say v, and set $x = v$. Then, fire x and set
   $V_T = V_T \bigcup \{u : u \text{ becomes unstable just after the firing of } x\}$
   and
   $E_T = E_T \bigcup \{\{x, w\} : w \in V_T\}$.
5. Continue in this way until $|E_T| = n^2 - 1$.

Let f be a critical configuration and suppose that (T_f, s) is a Hamiltonian path. It is easy to check that the running time of the parallel updating protocol on input (n, f) has a lower bound of n^2. Thus, we know that the employment of the parallel updating protocol does not yield sublinear time algorithms for RR[2]. □

4.1 Some Facts Concerning the Hardness of *RR*[2]

There are many algorithmic problems associated with the Abelian sandpile model. The most important of those problems is the prediction problem defined by Problem 3.

Problem 3. *SPP*[d], d-dimensional sandpile prediction problem.

- Input: (n, d, f), where $n \geq 1$ and f is a configuration of the d-dimensional sandpile lattice of order n.
- Problem: compute $st_{\mathcal{L}_n^d}(f)$.

Remark 8. Let G be a d-dimensional lattice. If all sides of G have the same length, say n, we say that G is a lattice of *order* n. We use the symbol $\mathcal{L}_n^d$ to denote the d-dimensional sandpile lattice of order n.

The algorithmic complexity of the prediction problem has been studied in [2, 3, 7]. The case $d \neq 2$ is well understood:

- If $d > 2$, the problem *SPP*[d] is *P*-complete [2] (upper and lower bounds are tight).
- The problem *SPP*[1] is TC^0-hard and it belongs to log *CFL* [3] (the gap between upper and lower bounds is not large).

https://doi.org/10.25088/ComplexSystems.21.4.245

On the other hand, we know that the complexity of the two-dimensional sandpile prediction problem is far from being well understood. The gap between upper and lower bounds is still very large: the best upper bound is *ptime* computability, while the best lower bound is NC^1-hardness [2].

We consider that closing this gap is the most important open problem related to the analysis of the computational complexity of the Abelian sandpile model. We will argue that problems $RR[2]$ and $SPP[2]$ are closely related. First, an easy lemma.

Lemma 2. Given $d \geq 1$, $RR[d]$ is logspace reducible to $SPP[d]$.

We know from Lemma 1 that any lower bound for $RR[2]$ is a lower bound for $SPP[2]$. We consider that the relation between those two problems goes deeper. Consider the higher-dimensional case, for example, $d \geq 3$. It is known that $RR[d]$ and $SPP[d]$ are P-complete under logspace reductions [2], that is, if $d \geq 3$, the problems $SPP[d]$ and $RR[d]$ are logspace equivalent. Let $d \geq 1$; are $SPP[d]$ and $RR[d]$ equivalent under logspace reductions? Not necessarily. The equivalence does not hold in any dimension (it might not hold in dimension two). Consider the case $d = 1$.

It is known that $SPP[1]$ is TC^0-hard, which means that $SPP[1]$ cannot be solved using a uniform polynomial-size family of circuits of constant depth. On the other hand, $RR[1]$ can be solved using a uniform polynomial-size family of circuits of depth two. Given $n \geq 1$, a stable configuration of $\mathcal{L}_n^1$ is a function $f : [n] \to \{0, 1\}$. Given a configuration f, we can identify f with the string $w_f = f(1) \ldots f(n)$. Dhar's theorem (Theorem 4) implies that f is critical if and only if $\left|f^{-1}(0)\right| \leq 1$, that is, f is critical if and only if w_f belongs to the regular language $(1^* 01^*) \bigcup (1^*)$. It is easy to check that this language can be recognized using depth two circuits.

Also, we cannot prove that the problems SPP and RR are equivalent in any dimension (including dimension two), but we claim that:

1. If a sublinear time algorithm does not exist for $RR[2]$, then a sublinear time algorithm does not exist for $SPP[2]$.
2. The existence of a sublinear time algorithm for $RR[2]$ gives us strong evidence that sublinear time algorithms exist for $SPP[2]$.

So, analyzing the problem $RR[2]$ could be a good way of studying the complexity of the problem $SPP[2]$.

https://doi.org/10.25088/ComplexSystems.21.4.245

4.2 Solving *RR*[2] with a Small Amount of Memory

Critical configurations are stable configurations of high complexity, which are very close to being unstable (they are locally unstable). This point of view is supported by the following theorem [4].

Theorem 4. Given a sandpile lattice $\mathcal{L}_n$ and $f \in \mathcal{M}(n)$, we know that f is a critical configuration if and only if $A \subseteq V(\mathcal{L}_n)^*$ does not exist such that for any $u \in A$, the inequality $\deg_A(u) \ngeq f(u)$ holds.

Theorem 4 suggests an alternative approach to the problem of designing efficient algorithms for the problem $RR[2]$. Consider the nondeterministic algorithm given by the following.

1. Guess $A \subseteq V(\mathcal{L}_n)^*$.
2. Check that for all $v \in A$, the inequality $f(v) < \deg_A(v)$ is satisfied, where the symbol $\deg_A(v)$ denotes the number of neighbors of v inside A.

Can the algorithm be efficiently implemented? Let f be a noncritical configuration. We know of the existence of a set A that witnesses the noncriticality of A. The possible witnesses are the connected subsets of $\mathcal{L}_n$, which are exponential many; nevertheless, if we could bound the complexity (geometrical, information-theoretical) of those witnesses, then we could design an efficient algorithm recognizing the set of two-dimensional critical configurations. Can we bound the complexity of the possible witness? Unfortunately, the answer is no. Let A be a connected subset of $\mathcal{L}_n$ and let f_A be the configuration defined by

$$f_A(v) = \begin{cases} \deg_A(v) - 1 & \text{if } v \in \gamma \\ 3, & \text{otherwise.} \end{cases}$$

If $|A| \geq 2$, the set A is the unique witness of noncriticality for f_A. Then, we know that noncritical configurations exist such that all of their witnesses only have complex descriptions (which are hard to navigate using small memory).

Let C be a class of rectangular lattices; we use the symbol $RR[C]$ to denote the problem:

- Input: (G, f), where $G \in C$ and f is a configuration of G.
- Problem: decide if f is critical.

It is important to remark that all the characterizations of critical (recurrent) configurations studied in this paper, including Theorem 4, hold for general graphs.

https://doi.org/10.25088/ComplexSystems.21.4.245

Let $\mathcal{LOG}$ be the class of sandpile rectangular lattices of logarithmic height, that is, given $G \in \mathcal{LOG}$ there exists a positive integer n such that the underlying lattice of G is the lattice $[n]\times[\log(n)]$.

Remark 9. From now on, we use the symbol $\log(n)$ to denote the positive integer $\lceil \log_2(n) \rceil$.

Lemma 3 shows that $RR[\mathcal{LOG}]$ can be solved by employing a nondeterministic logarithmic space Turing machine, that is, problem $RR[\mathcal{LOG}]$ belongs to the class $\mathcal{NL}$. Recall that the class $\mathcal{NL}$ is included in $\mathcal{NC}^2$ [8].

Lemma 3. $RR[\mathcal{LOG}]$ belongs to $\mathcal{NL}$.

Proof. Let $n \geq 1$ and let f be a configuration of the lattice $[n]\times[\log(n)]$. We have to check if $A \subset [n]\times[\log(n)]$ exists such that for all $v \in A$, the inequality $f(v) \not\leq \deg_A(v)$ holds. Consider the $\mathcal{NL}$-machine $\mathcal{M}$ defined by the following.

Machine $\mathcal{M}$ works on input $([n]\times[\log(n)], f)$ as follows.

1. $\mathcal{M}$ guesses $v_l, v_m, v_r \in \{0, 1\}^{\log(n)}$.
2. $\mathcal{M}$ checks that for all $i \leq \log(n)$, the conditions

   if $v_l[i] = 1$, then $f(1, i) \not\leq v_l[i+1] + v_l[i-1] + v_m[i]$
   and
   if $v_m[i] = 1$, then $f(2, i) \not\leq v_l[i] + v_m[i+1] + v_m[i-1] + v_r[i]$

   hold. If this is the case, $\mathcal{M}$ goes to step 3; otherwise it rejects the input.
3. $\mathcal{M}$ sets $j = 2$, $X_l = v_l$, $X_m = v_m$, $X_r = v_r$, and $X =$ True.
4. While $j \leq n-1$ and $X =$ True, machine $\mathcal{M}$ does the following:
   - $\mathcal{M}$ guesses $X \in \{0, 1\}^{\log(n)}$.
   - $\mathcal{M}$ sets $X_l := X_m$, $X_m = X_r$, $X_r = X$, and $j := j + 1$
   - $\mathcal{M}$ checks that for all $i \leq \log(n)$, the condition

     if $X_m[i] = 1$, then $f(j, i) \not\leq X_l[i] + X_m[i+1] + X_m[i-1] + X_r[i]$

     holds. If this is the case, $\mathcal{M}$ sets $X =$ True; otherwise it sets $X =$ False.
5. If $j = n-1$, machine $\mathcal{M}$ goes to step 6.
6. $\mathcal{M}$ guesses $X \in \{0, 1\}^{\log(n)}$ and checks that for all $i \leq \log(n)$, the condition

   if $X[i] = 1$, then $f(n, i) \not\leq X[i+1] + X[i-1] + X_r[i]$

https://doi.org/10.25088/ComplexSystems.21.4.245

holds. If this is the case, $\mathcal{M}$ halts and accepts the input, otherwise it halts and rejects the input.

It is clear that $\mathcal{M}$ uses logarithmic space, and it is easy to check that $\mathcal{M}$ is correct. Let C be a column of $[n]\times[\log(n)]$; given $X \in \{0, 1\}^{\log(n)}$, we can think of X as the characteristic function of $A \bigcap C$. We use the term the *trace of A over C* to denote the characteristic function of $A \bigcap C$. Machine $\mathcal{M}$ simply guesses three consecutive traces, corresponding to three consecutive columns C_1, C_2, and C_3, and then it uses this information to check that for all $v \in V(C_2)$ (where C_2 is the column in the middle); the inequality $f(v) \nleqq \deg_A(v)$ holds.

Thus, the problem *co-RR*[$\mathcal{LOG}$] belongs to $\mathcal{NL}$. The theorem of Immerman–Szelepcsényi [8] implies that $RR[\mathcal{LOG}]$ belongs to $\mathcal{NL}$. □

Remark 10. We consider that the polylog-time computability of $RR[\mathcal{LOG}]$ is far from being obvious given that rectangular lattices of logarithmic height can support complex dynamics.

Let $\mathcal{F}:\mathbb{N}\to\mathbb{N}$ be a function such that for all n, the inequality $\mathcal{F}(n) \leq n$ holds. We use the symbol $RR[\mathcal{F}]$ to denote the *restriction* of $RR[2]$ to the class of rectangular lattices of height $\mathcal{F}$ (the definition is analogous to the definition of $RR[\mathcal{LOG}]$).

Lemma 4. *co-*$RR[\mathcal{F}] \in \mathcal{SPACE}\left[\mathcal{F}^2\right]$.

Proof. We can define a nondeterministic Turing machine $\mathcal{N}$ that solves the problem *co-*$RR[\mathcal{F}]$ and that uses $O(\mathcal{F})$ workspace; the definition of $\mathcal{N}$ is analogous to the definition of the machine $\mathcal{M}$ employed in the proof of Lemma 3. Machine $\mathcal{N}$ shows that $RR[\mathcal{F}]$ belongs to $\mathcal{NSPACE}[\mathcal{F}]$; Savitch's theorem [8] states that $\mathcal{NSPACE}[\mathcal{F}] \subseteq \mathcal{SPACE}\left[\mathcal{F}^2\right]$. □

Also, if we bound the height of the lattices, we can bound the workspace employed in the recognition of noncritical configurations. If the workspace employed by an algorithm is bounded above by the logarithm of the input size, then the algorithm can be efficiently parallelized (the classes $\mathcal{L}$ and $\mathcal{NL}$ are included in $\mathcal{NC}^2$); this fact allows us to show that $RR[\mathcal{LOG}]$ belongs to $\mathcal{NC}^2$. On the other hand, if the workspace employed by an algorithm is super-logarithmic, then we cannot claim that such an algorithm can be efficiently parallelized. Also, we cannot ensure polylog-time computability beyond logarithmic height.

Conjecture. A phase transition for feasibility.

If $\mathcal{F}$ belongs to $O(\log(n))$, then the problem belongs to $\mathcal{NC}^2$; otherwise it is P-complete.

https://doi.org/10.25088/ComplexSystems.21.4.245

Let $\mathcal{I}$ be the identity function (i.e., given $n \geq 1$, $\mathcal{I}(n) = n$). If the conjecture were true, the problem $RR[2] = RR[\mathcal{I}]$ would become *P*-complete.

5. Dealing with the Conjecture: On the Communication Complexity of *RR*[2]

We conjecture that $RR[2]$ is *P*-complete (the best lower bound is NC^1-hardness: the value problem for planar monotone Boolean circuits is logspace reducible to $RR[2]$ as shown in [2]). Also, we conjecture that $RR[2]$ cannot be parallelized. We support our belief with the following fact.

The dynamics of the Abelian sandpile model exhibit long-range correlations: given a sandpile graph G, a stable configuration f of G, and a node v, the node v is fired, along with the avalanche triggered by $f + \delta_G$, depending on the values taken by f at nodes of G that are placed far away from v. It makes it hard to find an efficient parallel algorithm solving problem $RR[2]$ given that we cannot split the graph into small independent pieces (there are no small independent pieces).

We know that this observation is not a conclusive argument. Consider the one-dimensional Abelian sandpile model: it exhibits long-range correlations as well, but the problem $RR[1]$ can be solved in parallel constant time.

In this section, we study an intermediate problem, the sandpile accessibility problem, which we denote with the symbol *SPA*.

Let d be a positive integer. We use the symbol $SPA[d]$ to denote the problem:

- Input: (m, f, v), where $m \in \mathbb{N}$, f is a configuration of the d-dimensional sandpile lattice of order m, denoted by $\mathcal{L}_m^d$, and v is a node of $\mathcal{L}_m^d$.
- Problem: decide if $SC_{f+\delta_m^d}(v) > 0$ (we use the symbol δ_m^d to denote the border configuration of the sandpile lattice $\mathcal{L}_m^d$).

Lemma 5. $RR[d] \preceq_{\mathcal{NC}} SPA[d] \preceq_{\mathcal{NC}} SPP[d]$.

Proof. Let d, m be two positive integers and let f be a configuration of the sandpile lattice $\mathcal{L}_m^d$. Note that f is recurrent if and only if for all $v \in V\left(\mathcal{L}_m^d\right)^*$, the inequality $SC_{f+\delta_m^d}(v) \ngeq 0$ holds. Thus, $RR[d]$ is $\mathcal{NC}$ reducible to $SPA[d]$.

https://doi.org/10.25088/ComplexSystems.21.4.245

Let $d, m \geq 1$. We suppose that we have fixed a linear ordering on $V\left(\mathcal{L}_m^d\right)^*$. Let $v_1, \ldots, v_{m^d}$ be the ordered list of the elements of $V\left(\mathcal{L}_m^d\right)^*$. Given a configuration f of $\mathcal{L}_m^d$, we can represent f as a vector $(k_i)_{i \leq m^d}$, where given $i \leq m^d$ the entry k_i is equal to $f(v_i)$. The *reduced Laplacian* of $\mathcal{L}_m^d$ is the matrix $L(d, m) = \left[l_{ij}\right]_{i,j \leq m^d}$ given by

$$l_{ij} = \begin{cases} -\deg(v_i), & \text{if } i = j \\ 1, & \text{if } v_i \neq v_j \text{ and } \{v_i, v_j\} \in E\left(\mathcal{L}_m^d\right) \\ 0, & \text{otherwise.} \end{cases}$$

It is easy to check (see [7]) that for all configurations of f the equation

$$st_{\mathcal{L}_m^d}(f) = f + L(d, m)\left(SC_f\right)$$

holds. The *Kirchhoff matrix theorem* [7] implies that $L(d, m)$ is nonsingular. Then,

$$SC_f = (L(d, m))^{-1}\left(st_{\mathcal{L}_m^d}(f) - f\right).$$

Let f be a stable configuration of $\mathcal{L}_m^d$. If we can compute in polylogarithmic time the vector $st_{\mathcal{L}_m^d}(f)$, then we can use it and the given equation to compute in time $O\left(\log^2(m)\right)$ the vector $SC_{f+\delta_m^d}$. Note that if we can compute $SC_{f+\delta_m^d}$ we can decide, given $v \in V\left(\mathcal{L}_m^d\right)^*$, if the inequality $SC_{f+\delta_m^d}(v) \not\geq 0$ holds. Thus, the problem $SPA[d]$ is $\mathcal{NC}$ reducible to $SPP[d]$. □

Recall that a $\mathcal{NC}$ reduction is a reduction that can be implemented in polylogarithmic parallel time. The class $\mathcal{NC}$ is closed under $\mathcal{NC}$ reductions, that is, if $L \in \mathcal{NC}$ and problem T is $\mathcal{NC}$ reducible to L, then T also belongs to $\mathcal{NC}$.

5.1 Some Facts Concerning the Communication Complexity of *SPA*[2]

Remark 11. Given a function $f : A \to B$ and given $C \subset B$, we use the symbol $f\restriction_C$ to denote the restriction of the function f to the set C.

Lemma 5 implies that it is worth it to pay some attention to the problem $SPA[d]$. Let $n \geq 1$, let v be a node of the one-dimensional sandpile lattice $\mathcal{L}_n^1 = ([n], s)$, and let N_v be a small neighborhood of

https://doi.org/10.25088/ComplexSystems.21.4.245

v, and suppose that $f\restriction_{N_v}$ is known, where f is some stable configuration of $\mathcal{L}_n^1$. How much additional information is needed in order to decide if node v is fired along the avalanche triggered by $f + \delta_{\mathcal{L}_n^1}$? If N_v contains two nodes $i \notin v \notin j$ such that $f(i) = f(j) = 0$, then $SC_{f+\delta_{\mathcal{L}_n^1}}(v) = 0$ and no further information is required. Suppose that for all $w \in N_v$, $f(w) = 1$. In this case one bit of information suffices.

Let Bob be a party who knows the definition of the configuration f out of N_v. We suppose that we can ask Bob to send us any information concerning the nodes located out of N_v. Then we can ask Bob to send us a bit of information, say a_v, where a_v has the following meaning: $a_v = 0$ if and only if there exist two nodes, say u and w, with one of them placed to the right of N_v and the second one placed to the left, such that $f(v) = f(w) = 0$. We know that $SC_{f+\delta_{\mathcal{L}_n^1}}(v) = 1$ if and only if $a_v = 1$. It is clear that the information encoded by a_v suffices. Also, the one-dimensional version of *SPA* can be solved using local information and one bit of advice.

Now, we consider the case of rectangular lattices of logarithmic height. We can solve the problem $SPA[\mathcal{LOG}]$ in parallel time $O\left(\log^2(n)\right)$. Let $n \geq 1$, let $\mathcal{L}_n^{\log}$ be the sandpile lattice $([n]\times[\log(n)], s)$, let v be a node of $\mathcal{L}_n^{\log}$, and let N_v be a small neighborhood of v, and suppose that $f\restriction_{N_v}$ is known. How much information is required in order to decide if the equality $SC_{f+\delta_{\mathcal{L}_n^1}}(v) = 0$ holds? Suppose $v = (i, j)$ and suppose that N_v is equal to the set

$$\{(l, r) : l \in \{i-k, \ldots, i+k\} \;\&\; r \leq \log(n)\}.$$

We ask Bob to compute the sets

$$L_1 = \left\{w \in \left\{(i-k-1, x) : SC_{f+\delta_{\mathcal{L}_n^1}}(w) = 0\right\}\right\}$$

and

$$L_2 = \left\{w \in \left\{(i+k+1, x) : SC_{f+\delta_{\mathcal{L}_n^1}}(w) = 0\right\}\right\}. \tag{2}$$

Then we ask Bob to send us the characteristic functions of those two sets. It is clear that this information suffices. Also, $SPA[\mathcal{LOG}]$ can be solved using local information and $O(\log(n))$ bits of advice.

We have seen that the tractable versions of *SPA* can be solved using local information and few bits of advice; it seems to be a fundamental feature of the tractable cases. Let us try a last mental experiment, this time considering the case of square lattices. Let $n \geq 1$, let

https://doi.org/10.25088/ComplexSystems.21.4.245

$(x, y) \in V(\mathcal{L}_n)^*$, and let $N_{(x,y)}$ be the strip

$$\{(x+i, y+j) : i \in \{-k, \ldots, k\} \ \& \ y + j \in [n]\}.$$

Let Ψ be the query: is $SC_{f+\delta_n}(v)$ equal to zero? It seems that the amount of advice that is required to solve the query Ψ depends linearly on the size of $\delta(N_v)$, the border of N_v. Thus, the amount of advice required seems to be equal to $\Omega(n) = \Omega\left(\sqrt{|\mathcal{L}_n|}\right)$, which is not polylogarithmic with respect to the size of the lattice.

Analyzing the communication complexity of $SPA[2]$ and related problems can give us some insight concerning the algorithmic complexity of those problems. Note that we could detect some differences between the tractable cases and the cases that are conjectured to be hard.

Now we will try to analyze the problem $RR[2]$.

5.2 The Communication Game: The Communication Complexity of *RR*[2]

In this section, we introduce the basic theory of communication games that can be used as a lower bound for the monotone depth required to compute a given sequence of monotone Boolean functions. We use this theory in Section 5.2.1 to prove that $RR[2]$ requires large monotone depth.

Let $n \geq 1$. We can associate $\{0, 1\}^n$ with the partial ordering given by the following: let $u, v \in \{0, 1\}^n$. We have

$$u \leq_n v \text{ if and only if for all } i \leq n, \text{ the inequality } u[i] \leq v[i] \text{ holds.}$$

Given a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$, we say that it is a *monotone function* if and only if the condition

$$\text{if } f(u) = 1 \text{ and } u \leq_n v \text{ then } f(v) = 1$$

holds for all $u, v \in \{0, 1\}^n$. Given $\{f_i\}_{i \geq 1}$, a sequence of Boolean functions, it is monotone if and only if for all $n \geq 1$ the function f_n is monotone. Given a monotone sequence $\{f_i\}_{i \geq 1}$, we say that it is a *normal sequence* if and only if there exists a polynomial $p(X)$ such that for all $n \geq 1$ the inequality $\# VAR(f_n) \leq p(n)$ holds, where $\# VAR(f_n)$ denotes the number of variables occurring in f_n. Given a normal sequence of Boolean functions $\mathcal{F} = \{f_i\}_{i \geq 1}$, it determines an algorithmic problem, denoted with the symbol $EVAL(\mathcal{F})$ and defined by Problem 4.

Problem 4. $EVAL(\mathcal{F})$: evaluating sequence $\mathcal{F}$.

https://doi.org/10.25088/ComplexSystems.21.4.245

- Input: (n, u), where n is a positive integer and $u \in \{0, 1\}^{\sharp\, VAR(f_n)}$.
- Problem: compute $f_n(u)$.

If $\mathcal{F}$ is a monotone sequence, the problem $EVAL(\mathcal{F})$ can be computed employing a family of monotone Boolean circuits (a circuit C is monotone if and only if all its Boolean gates are either conjunctions or disjunctions). Given $(C_i)_{i \leq n}$, a polynomial-size family of Boolean circuits, its depth is the function $d_C : \mathbb{N} \to \mathbb{N}$ defined by $d_C(n) = \text{depth}(C_n)$. We say that $\mathcal{F}$ requires $\Omega(g)$ *monotone depth* if and only if given $(C_n)_{n \geq 1}$, a uniform polynomial-size family of monotone Boolean circuits computing the problem $EVAL(\mathcal{F})$, it happens that $d_C \in \Omega(g)$. If the sequence $\mathcal{F}$ requires large depth, it does not imply that the problem $EVAL(\mathcal{F})$ is P-hard, but it indicates that the problem is hard (in some sense) and it suggests that the problem is P-hard.

We use the symbol $M\text{-depth}(\mathcal{F})$ to denote the monotone depth required by the problem $EVAL(\mathcal{F})$.

We can think of $RR[2]$ as if it were a sequence of Boolean functions. We show that $RR[2]$ is monotone and we prove that $RR[2]$ requires large depth. To this end, we use the theory of communication games.

Let $f : \{0, 1\}^n \to \{0, 1\}$ be a monotone function; a *maxterm* of f is an assignment u such that $f(u) = 0$ and for all $u \not\leq_n v$, $f(v) = 1$. A *minterm* is an assignment $u \in \{0, 1\}^n$ such that $f(u) = 1$ and for all $v \not\leq_n u$, $f(v) = 0$.

Consider the following game. There are two parties, say Alice and Bob. Suppose that Alice gets $u \in \{0, 1\}^n$, which is a minterm of f, and Bob gets $v \in \{0, 1\}^n$, which is a maxterm of f. Suppose that they are asked to compute a number $i \leq n$ such that $u[i] \geq v[i]$. Which is the minimum number of bits that they must communicate to each other in order to solve the above task? We use the symbol $\text{Max}(n)$ to denote the set $\{u \in \{0, 1\}^n : u \text{ is a maxterm}\}$ and we use the symbol $\text{Min}(n)$ to denote the corresponding set of minterms. Suppose we have fixed a communication protocol $\mathcal{P}$ that is employed by Alice and Bob on any possible pair (u, v). We use the symbol $CC_{\mathcal{F},\mathcal{P}}$ to denote the function defined by

$$CC_{\mathcal{F},\mathcal{P}}(n) = \min \left\{ CC_{\mathcal{F},\mathcal{P}}(u, v) : u \in \text{Max}(n) \text{ and } v \in \text{Min}(n) \right\}$$

where $CC_{\mathcal{F},\mathcal{P}}(u, v)$ is the number of bits that must be communicated when Alice gets u, Bob gets v, and they employ protocol $\mathcal{P}$. There must exist a protocol $\mathcal{P}_0$ such that given $\mathcal{P}$, any other protocol, it hap-

https://doi.org/10.25088/ComplexSystems.21.4.245

pens that $CC_{\mathcal{F},\mathcal{P}} \in \Omega\left(CC_{\mathcal{F},\mathcal{P}_0}\right)$. We use the symbol $CC_{\mathcal{F}}$ to denote the function $CC_{\mathcal{F},\mathcal{P}_0}$. Theorem 5 is one key fact of the theory; for a proof (and much more information concerning these issues), see [9].

Theorem 5. M-depth$(\mathcal{F}) \in \Omega\left(CC_{\mathcal{F}}\right)$.

5.2.1 The Theorem

We are ready to analyze the monotone complexity of RR[2].

Suppose there are two parties, say Alice and Bob, and suppose that each of them get a configuration of the sandpile lattice $\mathcal{L}_n$. Let f_1 be the configuration received by Alice and let f_2 be the one received by Bob. Suppose that f_1 is a minterm and suppose that f_2 is a maxterm. The existence of a node v for which the inequality $f_1(v) \not\leq f_2(v)$ holds is implied. Alice and Bob are asked to find such a node, that is, Alice and Bob are asked to compute $v \in V(\mathcal{L}_n)^*$ such that $f_1(v) \not\leq f_2(v)$. We want to measure the amount of communication bits that are required to carry out this task.

We can define an order relation over the set $\mathcal{ST}(n)$: given $f, g \in \mathcal{ST}(n)$, the inequality $f \leq g$ holds if and only if for all $v \in V(\mathcal{L}_n)^*$, then $f(v) \leq g(v)$. Let $\mathcal{R}_n : \mathcal{ST}(n) \to \{0, 1\}$ be the function

$$\mathcal{R}_n(f) = 1 \text{ if and only if } f \text{ is recurrent.}$$

Note that $\mathcal{R}_n$ is monotone, that is, if $f \leq g$ and $\mathcal{R}_n(f) = 1$, then $\mathcal{R}_n(g) = 1$.

We can identify $\mathcal{ST}(n)$ with the set $\{0, 1\}^{4n^2}$ if we think of the elements of $\{0, 1\}^{4n^2}$ as $2n \times 2n$ Boolean matrices. Given a $2n \times 2n$ Boolean matrix M, we identify the site $(i, j) \in V(\mathcal{L}_n)^*$ with the 2×2 minor of M constituted by the intersection of the rows $2i - 1$ and $2i$ with the columns $2j - 1$ and $2j$.

We use the symbol M_{ij} to denote this minor. Matrix M determines a configuration of $\mathcal{L}_n$ denoted with the symbol f_M. Configuration f_M is the function defined by

$$f_M(i, j) = \sum_{t \leq 2} \sum_{k \leq 2} M_{ij}[k, t].$$

On the other hand, given f, a stable configuration of $\mathcal{L}_n$, we say that M represents f if and only if the equation $f = f_M$ holds. Note that any stable configuration is representable in the sense given.

https://doi.org/10.25088/ComplexSystems.21.4.245

Let $n \geq 1$ and let $RR_n : \{0, 1\}^{n^2} \to \{0, 1\}$ be the Boolean function defined by

$$RR_n(M) = 1 \text{ if and only if } f_M \text{ is a recurrent configuration.}$$

The function RR_n is monotone. Given M and N two matrices, if $M \leq N$ (as Boolean strings) then $f_M \leq f_N$. We identify the problem $RR[2]$ with the monotone sequence $(RR_n)_{n\geq 1}$, which we denote with the symbol $\mathcal{RR}[2]$. The problem $RR[2]$ is essentially the same as the problem $EVAL(\mathcal{RR}[2])$. The latter problem can be computed employing a uniform polynomial-size family of monotone circuits. We prove that M-depth$(\mathcal{RR}[2])$ belongs to $\Omega(n)$.

Theorem 6. $CC_{\mathcal{RR}[2]}(n) \in \Omega(n)$.

Proof. Given λ, a simple cycle contained in $[n]\times[n]$, and given $v \in \lambda$, we use the symbol $\deg_\lambda(v)$ to denote the number of neighbors of v that belong either to the set enclosed by λ or to λ itself. It is easy to check that the configuration f_λ defined by

$$f_\lambda(v) = \begin{cases} 3 & \text{if } v \notin \lambda \\ \deg_\lambda(v) - 1, & \text{otherwise} \end{cases}$$

is a maxterm of $RR[2]$. Given f, a recurrent configuration of $[n]\times[n]$, and given λ, a simple cycle contained in $[n]\times[n]$, we say that λ is an f-critical cycle if and only if there exists a node $v \in \lambda$ such that $f_v = f - e_v$ is a nonrecurrent configuration. Moreover, we say that v is a critical node of λ. Suppose that there exists a cycle λ such that for all $v \in \lambda$ if $f(v) \neq 0$ then $f_v = f - e_v$ is a recurrent configuration. Then, f is not a minterm. Therefore, we can conclude that given a recurrent configuration f, if f is a minterm term of $RR[2]$, then any cycle contained in $[n]\times[n]$ is f-critical.

From now on, if f is a recurrent configuration that is a minterm of $RR[2]$, we say that it is a *minterm configuration*.

One key point of our proof is the choice of the minterm configuration denoted by f_1. We suppose without loss of generality that $n = 6\,m$ for some $m \geq 1$. Given $i \leq m$, the symbol C_i denotes the sublattice of $\mathcal{L}_n$ constituted by the set of sites

$$\{(x, y) : y \leq 6 \text{ and } x \in \{6\,i - 5, \ldots, 6\,i\}\}.$$

https://doi.org/10.25088/ComplexSystems.21.4.245

We use the symbol L_i to denote the horizontal path connecting the sites $(6\,(i-1), 4)$ and $(6\,i, 4)$. We use the symbol T_i to denote the path

$$(6\,i-6, 4), (6\,i-5, 4), (6\,i-5, 3), (6\,i-5, 2), (6\,i-4, 2),$$
$$\ldots, (6\,i-1, 2), (6\,i-1, 3), (6\,i-1, 4), (6\,i, 4).$$

We use the symbol P_i to denote the site $(6\,i-3, 4)$ and the symbol Q_i to denote the site $(6\,i-3, 2)$. Moreover, we use the symbol A to denote the vertical path connecting the sites $(6, 4)$ and $(6, n-2)$, the symbol B to denote the horizontal path connecting the sites $(6, n-2)$ and $(n-6, n-2)$, and the symbol C to denote the vertical path connecting the sites $(n-6, n-2)$ and $(n-6, 4)$.

Let $X, Y \in \{0, 1\}^{m-2}$. Suppose that $Z \in \{X, Y\}$. We define a cycle $\gamma(Z)$ in the following way:

$\gamma(Z)$ is the concatenation of C, B, A and the sequence $\{R_i : i \in \{2, \ldots, m-1\}\}$ of short paths, which is determined by the rule

$$R_i = \begin{cases} L_i & \text{if } Z[i] = 1 \\ T_i, & \text{otherwise.} \end{cases}$$

We set $f_2 = f_{\gamma(X)}$. It is clear that f_2 is a maxterm. Now, we define a second configuration that we denote with the symbol g_1. If $v \notin V(\gamma(Y))$, we set $g_1(v) = 3$. If $v \in V(\gamma(Y))$ but $v \notin \bigcup_{2 \le i \le m-1} V(C_i)$, we set $g_1(v) = \deg_{\gamma(Y)}(v) - 1$. If we suppose that $v \in V(\gamma(Y)) \bigcap V(C_i)$ for some $i \in \{2, \ldots, m-1\}$, then we set

$$g_1(v) = \begin{cases} \deg_{\gamma(Y)}(v) - 1, & \text{if } X[i] = Y[i] \\ \deg_{\gamma(Y)}(v) - 1, & \text{if } X[i] \neq Y[i] \text{ and } v \notin \{Q_i, P_i\} \\ 3 & \text{if } X[i] \neq Y[i] = 1 \text{ and } v = P_i \\ 3 & \text{if } X[i] \neq Y[i] = 0 \text{ and } v = Q_i. \end{cases}$$

We can suppose that there exists i such that $X[i] \neq Y[i]$. Then, we claim that g_1 is a recurrent configuration. Note that if $g_1(v) \ngeq f_2(v)$, there exists $i \in \{2, \ldots, m-1\}$ such that $X[i] \neq Y[i]$ and $v \in V(C_i)$.

Let f_1 be a minterm configuration satisfying the inequality $f_1 \le g_1$. Let us use the symbols $f_1(Y)$ and $f_2(X)$ to denote the configurations f_1 and f_2. Suppose that Alice and Bob can compute the requested v communicating no more than k bits. Then, if Alice were given string Y, Bob were given string X, and they were requested to compute an i such that $X[i] \neq Y[i]$, they could accomplish this task communicating no more than k bits. We know that the latter task requires, in the worst case, communicating $\Omega(m)$ bits. Then, Alice and Bob must communicate, in the worst case, $\Omega(n)$ bits. □

https://doi.org/10.25088/ComplexSystems.21.4.245

6. Concluding Remarks

Our basic conjecture is that the problem *SPP*[2] is *P*-complete; we conjecture that the problem *RR*[2] is *P*-complete as well. We know that the latter conjecture entails the former. The Holy Grail of our quest is a proof of the *P*-completeness of the problem *RR*[2]. Unfortunately, we could not find such a proof. We have constrained ourselves, in this paper, to look for partial results. We have tried different approaches (workspace bounds for different variations, communication complexity bounds), which in no sense exhausted the universe of possible approaches. Some other approaches can be tried as well; we conjecture, for instance, that the avalanche process is inherently sequential. It does not imply that the problem is *P*-complete, because it only shows that the simulation of avalanches (which is the core of the algorithms we use to compute stabilizations and recognize recurrent configurations) cannot be efficiently parallelized. Greenlaw has developed (see [10]) a formal framework that allows us to analyze the inherent sequentiality of a given problem.

Acknowledgments

Thanks to VIE-UIS. Thanks to Anahi Gajardo and CONICYT; part of this work was done while the author was visiting the department of mathematical engineering of the Universidad de Concepción. Thanks to DIF, Universidad Nacional, R.P. HERMES 16860.

References

[1] P. Bak, C. Tang, and K. Wiesenfeld, "Self-Organized Criticality," *Physical Review A*, **38**(1), 1988 pp. 364–374. doi:10.1103/PhysRevA.38.364.

[2] C. Moore and M. Nilsson, "The Computational Complexity of Sandpiles," *Journal of Statistical Physics*, **96**(1–2), 1999 pp. 205–224. doi:10.1023/A:1004524500416.

[3] P. B. Miltersen, "The Computational Complexity of One-Dimensional Sandpiles," *Theory of Computing Systems*, **41**(1), 2007 pp. 119–125. doi:10.1007/s00224-006-1341-8.

[4] D. Dhar, "Theoretical Studies of Self-Organized Criticality," *Physica A*, **369**(1), 2006 pp. 29–70. doi:10.1016/j.physa.2006.04.004.

[5] G. Tardos, "Polynomial Bound for a Chip Firing Game on Graphs," *SIAM Journal of Discrete Mathematics*, **1**(3), 1988 pp. 397–398. doi:10.1137/0401039.

https://doi.org/10.25088/ComplexSystems.21.4.245

[6] L. Babai and I. Gorodezky, "Sandpile Transience on the Grid Is Polynomially Bounded," in *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA07)*, New Orleans, LA (H. Gabow, ed.), Philadelphia: Society for Industrial and Applied Mathematics, 2007 pp. 627–636.

[7] C. Mejia and J. A. Montoya, "On the Complexity of Sandpile Critical Avalanches," *Theoretical Computer Science*, **412**(30), 2011 pp. 3964–3974. doi:10.1016/j.tcs.2011.02.029.

[8] M. Sipser, *Introduction to the Theory of Computation*, Boston: Thomson, 2006.

[9] E. Kushilevitz and N. Nisan, *Communication Complexity*, New York: Cambridge University Press, 1997.

[10] R. Greenlaw, "A Model Classifying Algorithms as Inherently Sequential with Applications to Graph Searching," *Information and Computation*, **97**(2), 1992 pp. 133–149. doi:10.1016/0890-5401(92)90033-C.

https://doi.org/10.25088/ComplexSystems.21.4.245