Computational Complexity of the Avalanche Problem on One Dimensional Kadanoff Sandpiles

Enrico Formenti[1], Kévin Perrot[1,2,3(✉)], and Éric Rémila[4]

[1] Laboratoire I3S (UMR 6070 - CNRS), Université Nice Sophia Antipolis,
2000 Route des Lucioles, BP 121, F-06903 Sophia Antipolis Cedex, France
enrico.formenti@unice.fr
[2] Université de Lyon - LIP (UMR 5668 - CNRS - ENS de Lyon - Université Lyon 1),
46 allé d'Italie, 69364 Lyon Cedex 7, France
[3] Universidad de Chile - DII - DIM - CMM (UMR 2807 - CNRS),
2120 Blanco Encalada, Santiago, Chile
kperrot@dim.uchile.cl
[4] Université de Lyon - GATE LSE (UMR 5824 - CNRS - Université Lyon 2) ,
Site stéphanois, 6 Rue Basse des Rives, 42 023 Saint-etienne Cedex 2, France
eric.remila@univ-st-etienne.fr

Abstract. In this paper we prove that the general *avalanche problem* **AP** is in NC for the Kadanoff sandpile model in one dimension, answering an open problem of [2]. Thus adding one more item to the (slowly) growing list of dimension sensitive problems since in higher dimensions the problem is P-complete (for monotone sandpiles).

Keywords: Sandpile models · Discrete dynamical systems · Computational complexity · Dimension sensitive problems

1 Introduction

This paper is about cubic sand grains moving around on nicely packed columns in one dimension (the physical sandpile is two dimensional, but the support of sand columns is one dimensional). The Kadanoff Sandpile Model is a discrete dynamical system describing the evolution of sand grains. Grains move according to the repeated application of a simple local rule until reaching a fixed point.

We focus on the avalanche problem (**AP**), namely the problem of deciding if adding a single grain of sand in the first column of a sandpile given as an input causes a series of topples which hit some position (also given as a parameter).

This is an interesting problem from several points of view. First of all, it is dimension sensitive. Indeed, it is proved to be P-complete for sandpiles in dimension 2 or higher [2] and we proved it in NC^1 in this paper. Roughly speaking the problem is highly parallelisable in dimension 1 but not in higher dimensions (unless P=NC, of course). Second, an efficient solution to this problem could be useful for practical applications. Indeed, one can use sandpile models for

implementing load schedulers in parallel computers [9]. In this context, answering to **AP** helps in forecasting the number of supplementary processors that are needed to satisfy one more load which is submitted to the system.

The paper is structured as follows. Next section introduces the basic notions and results about Kadanoff sandpiles. Section 3 gives the formal statement of **AP** and recalls known results about it. In Sect. 4, main lemma and notions that are necessary for the proof of the main result are introduced and proved. Section 5 contains the main result. Section 6 draws our conclusions and give some perspectives.

2   Kadanoff Sandpile Model

We present the definition of the model in dimension one. A *configuration* is a decreasing sequence of integers $h = {}^\omega h_1, h_2, \ldots, h_n^\omega$, where h_i is the number of stacked grains (*height*) on column i, and such that all the heights on the left of h_1 equal h_1, and on the right of h_n equal h_n. Note that all the configurations we consider are *finite*. According to a fixed parameter p, the transition rule is the following: if the difference of heights between two columns i and $i + 1$ is strictly greater than p, then p grains can fall from column i and one of them land on each of the p adjacent columns on the right (see Fig. 1).
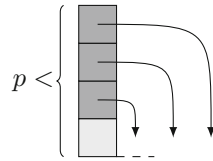

Fig. 1. Transition rule with parameter $p = 3$.

A more uniform and convenient representation of a configuration uses *slopes*. The *slope* at i is the height difference $s_i = h_i - h_{i+1}$. The transition rule thus becomes: if $s_i > p$, then

$$\begin{aligned}
s_{i-1} &\mapsto s_{i-1} + p \\
s_i &\mapsto s_i - (p+1) \\
s_{i+p} &\mapsto s_{i+p} + 1.
\end{aligned}$$

$|h| = |s| = n - 1$ is the *length* of the configuration, and the slope of an index i such that $i < 1$ or $n - 1 < i$ equals 0. The transition rule may be applied using different update policies (sequential, parallel, *etc.*), however we know from [8] that for any initial configuration, the orbit graph is a lattice, hence the stable configuration reached is unique and independent of the update policy. When, from the configuration s to s', the rule is applied on column i, we say that i is *fired* and we denote $s \xrightarrow{i} s'$ or simply $s \to s'$.

Notation 1. We denote ${}^\omega s_i$ (*resp.* s_i^ω) to say that all the slopes on the left (*resp.* right) of column i are equal to s_i.

Notation 2. For any $a, b \in \mathbb{Z}$ with $a \leq b$, let $[\![a, b]\!] = [a, b] \cap \mathbb{N}$ and $[\![a, b) = [a, b) \cap \mathbb{N}$. Finally, $s_{[\![a,b]\!]}$ denotes the subsequence $(s_a, s_{a+1}, \ldots, s_b)$.

A configuration s represented as a sequence of slopes is *monotone* if $s_i \geq 0$ for all $i \in [\![1, |s|)$. A configuration is *stable* if all its columns are *stable, i.e.,* $s_i \leq p$ for

all $i \in [\![1, |s|)$. A stable monotone configuration is therefore a finite configuration s of the form

$$^{\omega}0, s_1, s_2, \ldots, s_{n-1}, 0^{\omega}$$

Let $\mathbf{gSM}(n)$ be the set of all stable monotone configurations of length n (note that in [2], the authors added the restrictive condition $s_i > 0$ for all i, whereas we let $s_i \geq 0$ for all i and add the letter $\mathbf{g}$ standing for *general*). Finally, Let $\mathbf{gSM} = \bigcup_{n \in \mathbb{N}} \mathbf{gSM}(n)$.

3   Avalanche Problem AP

An *avalanche* is informally the process triggered by a single grain addition on column 1 (a formal definition is given at the beginning of Sect. 4). The size of an avalanche may be very small, or quite long, and is sensible to the tiniest change on the configuration. We are interested in the computational complexity of avalanches.

Avalanche Problem AP
A parameter $p \in \mathbb{N}$, with $p \geq 1$, is fixed.
Instance: a configuration $s \in \mathbf{gSM}$
         a column $k \in (|s|, |s| + p]\!]$
Question: does adding a grain on column 1 trigger a grain addition
         on column k?

For a fixed parameter p, the size of the input is in $\Theta(|s|)$. Thanks to the convergence, the answer to this question is well defined and independent of the chosen update strategy.

Let us give some examples. For $p = 2$, consider the instance

$$^{\omega}0, \underline{2}, 0, 2, 1, 1, 2, 1, 0, 2, 0^{\omega},$$

where the slope of column 1 is underlined. The question is "does adding a grain on column 1 increases the slope of column k equal to 10 or 11?" And the answer is negative in both cases. Here is a sequential evolution:

$$^{\omega}0, \underline{\mathbf{3}}, 0, 2, 1, 1, 2, 1, 0, 2, 0^{\omega} \rightarrow {}^{\omega}0, 2, \underline{0}, 0, 3, 1, 1, 2, 1, 0, 2, 0^{\omega}$$
$$\rightarrow {}^{\omega}0, 2, \underline{0}, 2, 0, 1, 2, 2, 1, 0, 2, 0^{\omega}$$

For $p = 3$, consider the instance $0^{\omega}, \underline{3}, 0, 2, 3, 1, 3, 1, 0^{\omega}$. We have to decide if column k equal to 8, 9 or 10 ends up with a strictly positive slope after a grain is added on column 1. The answer is positive, positive and negative, respectively. Here is a sequential evolution:

$$^{\omega}0, \underline{\mathbf{4}}, 0, 2, 3, 1, 3, 1, 0^{\omega} \rightarrow {}^{\omega}0, 3, \underline{0}, 0, 2, 4, 1, 3, 1, {}^{\omega}0$$
$$\rightarrow {}^{\omega}0, 3, \underline{0}, 0, 5, 0, 1, 3, 2, 0^{\omega} \rightarrow {}^{\omega}0, 3, \underline{0}, 3, 1, 0, 1, 4, 2, 0^{\omega}$$
$$\rightarrow {}^{\omega}0, 3, \underline{0}, 3, 1, 0, 4, 0, 2, 0, 1, 0^{\omega} \rightarrow {}^{\omega}0, 3, \underline{0}, 3, 1, 3, 0, 0, 2, 1, 1, 0^{\omega}$$

Known results on the dimension sensitive complexity of $\mathbf{AP}$ are the followings.

– In dimension one: the restriction of **AP** to the set of configurations s satisfying $s_i > 0$ for all i is known to be in NC^1 [2]. The key simplification induced by this restriction is the following: an avalanche goes forward if and only if it encounters a slope of value p at distance at most p from the previous one, and thus stops when there are p consecutive slopes strictly smaller than p. This condition is not sufficient anymore when we allow slopes of value 0, as shown for example by the instance $^\omega 0, \underline{2}, 0, 2, 2, 1, 2, 2, 0^\omega$ and $p = 2$:

$$^\omega 0, \underline{\mathbf{3}}, 0, 2, 2, 1, 2, 2, 0^\omega \rightarrow {}^\omega 0, 2, \underline{0}, 0, 3, 2, 1, 2, 2, 0^\omega$$
$$\rightarrow {}^\omega 0, 2, \underline{0}, 2, 0, 2, 2, 2, 2, 0^\omega$$

– In dimension two: there are two possible definitions of the model. One has two directions of grain fall, and a configuration is a tabular of sand content that is decreasing with respect to those two directions. In this model **AP** is P-complete for all parameter $p > 1$ [2]. The second definition follows the original model of Bak, Tang and Wiesenfeld [1], and it has been proved that information cannot cross (under reasonable conditions) when $p = 1$, a strong obstacle for a reduction to a P-complete circuit value problem [6].

– In dimension three or greater: sandpiles are capable of universal computation [7].

4   Avalanches, Peaks and Cols

This subsection partly intersects with the study presented in [11], but follows a new and hopefully clearer formulation. For a configuration $s \in \mathbf{gSM}$, an avalanche is the process following a single grain addition on column 1, until stabilization. We will consider avalanches according to the sequential update policy, and prove that it is formed by the repetition (not necessarily alternated) of the following two basic mechanisms:

– fire a column greater than all the previously fired columns;
– fire the immediate left neighbor of the last fired column.

An *avalanche strategy* for s is a sequence $a = (a_1, \ldots, a_T)$ of columns such that $s^+ \xrightarrow{a_1} \ldots \xrightarrow{a_T} s'$, where s^+ denotes the configuration $s \in \mathbf{gSM}$ on which a grain has been added on column 1, and s' is stable. Such a strategy is not unique, therefore we distinguish a particular one which we think is the simplest.

Definition 1. *The* avalanche *for s is the minimal avalanche strategy for s according to the lexicographic order, which means that at each step the leftmost column is fired.*

For example, let us consider $p = 2$ and the configuration $s = {}^\omega 0, \underline{2}, 2, 2, 2, 2, 0^\omega$, then $(0, 2, 4, 1, 3)$ is an avalanche strategy, but *the* avalanche for s is $(0, 2, 1, 3, 4)$ and leads to the same final configuration thanks to the lattice structure of the model [8].

Let us give two terms corresponding to the two basic mechanisms underlying the avalanche process, and prove the above mentioned description.

– a_t is a *peak* $\iff a_t > \max a_{[\![1,t[\![}$;
– a_t is a *col* $\iff a_t = a_{t-1} - 1$.

First, a simple Lemma.

Lemma 1. *An avalanche fires at most once every column.*

Proof. It is straightforward to notice that in order for a column to receive enough units of slope to be fired twice, another column must have been fired twice before, which leads to the impossibility of this situation when adding a single grain on column 1 of a stable configuration. $\square$

Now, the intended description.

Lemma 2. *The avalanche of a configuration $s \in \textbf{gSM}$ is a concatenation of peaks and cols.*

Proof. Let $a = (a_1, \ldots, a_T)$ be the avalanche for s. We prove the lemma by induction on the avalanche size. The first fired column is necessarily $a_1 = 1$, and we take as a convention that $\max \emptyset = 0$ thus a_1 is a peak. Suppose that the result is true until time t, we'll prove that a_{t+1} is either a peak or a col. It follows from Lemma 1 that $a_{t+1} \neq a_t$, and let us denote a_{t-j} with $j \geq 0$ the largest (rightmost) peak before time $t+1$. The induction hypothesis implies that columns a_t to $a_{t-j} - 1$ are cols.

$$a_{t-j+1} = a_{t-j+1} - 1$$
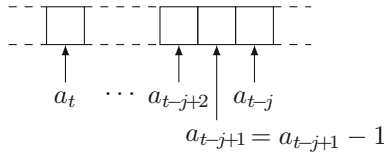
– If $a_{t+1} > a_t$, by induction on i from 0 to $j-1$, we have $a_{t+1} > a_t + i$ because $a_t + i$ has already been fired by hypothesis and a column cannot be fired twice (Lemma 1). As a consequence $a_{t+1} \geq a_{t-j}$ and for the same reason $a_{t+1} > a_{t-j}$, which was the greatest peak so far, therefore a_{t+1} is also a peak.
– If $a_{t+1} < a_t$, then, by contradiction, if $a_{t+1} \neq a_t - 1$ then the firing at a_t does not influence the slope at a_{t+1}, and firing this latter after a_t contradicts the minimality of the avalanche according the lexicographic order, because column a_{t+1} was already unstable at time t. Therefore, a_{t+1} is a col. $\square$

Interestingly, avalanches are local processes because they cannot fire a column too far (neither on the left nor on the right) from the last fired column, as it is proved in the following lemma.

Lemma 3. *Let a be the avalanche of a configuration $s \in \textbf{gSM}$, $q > 0$ is a peak of a implies that $s_q = p$ and there exists another peak q' satisfying $q - q' \leq p$.*

Proof. Let t be such that $q = a_t$. By definition of peak, at time t column q could only have received units of slope from columns on its left, that is, by Lemma 1 it received at most 1 unit of slope from column $q - p$. Since it was stable on configuration s, it has necessarily received this unique unit from column $q - p$ and became unstable thanks to it, which straightforwardly proves both claims.   □

Note that the converse implication is false. Figure 2 illustrates the results of this section.

Fig. 2. For $p = 4$, the arrow pictures the proceedings of an avalanche, which is a concatenation of peaks and cols (Lemma 2) where two consecutive peaks are at distance at most p (Lemma 3).

5   AP is in NC¹ in dimension one

We consider that the input configuration is represented as a sequence of slopes, since it is possible to efficiently transform a representation into another in parallel (for a configuration of size n, it requires constant time on n parallel processors). We consider the parameter p as a fixed constant, as it is part of the model definition

Remark 1. *In this paper, we consider the parameter p as a fixed constant which is part of the model definition. Indeed, if p would have been part of the input, which would therefore have size $(|s| + 2) \log p$, then comparing the height of a column to p (in order to know if the rule can be applied at this column) would not take a constant time anymore. This implies many low level considerations we want to avoid and inflate complexity.*

We recall that $\mathsf{NC} = \bigcup_{k \in \mathbb{N}} \mathsf{PT}/\mathsf{WK}(\log^k n, n^k)$, where $\mathsf{PT}/\mathsf{WK}(f(n), g(n))$ (Parallel Time / WorK) is the class of decision problems solvable by a uniform family of Boolean circuits with depth upper-bounded by $f(n)$ and size (number of gates) upper-bounded by $g(n)$, which is more conveniently seen for our purpose as solvable in time $\mathcal{O}(f(n))$ on $\mathcal{O}(g(n))$ parallel PRAM processors. We recall that $\mathsf{NC}^1 = \mathsf{PT}/\mathsf{WK}(\log n, \mathbb{R}[n])$ where $\mathbb{R}[n]$ denotes the set of polynomial functions.

As a consequence of Lemmas 2 and 3, the avalanche process is local. Moreover, if we cut the configuration into two parts, we can compute both parts of the avalanche independently, provided a small amount of information linking the two parts. This independency will be at the heart of our construction in order to compute the avalanche efficiently in parallel. Let us have a closer look at how to encode this "midway information", which we call *status* (a notion named *trace* has been defined in [12], which shares some of those ideas).

For a column $i > p$ of a configuration s, the *status* at i of the avalanche a for s is the boolean p-tuple $(b_0, \ldots, b_{p-1})$ such that $b_j = 1$ if column $i - p + j$ is fired within a, and 0 otherwise. For example, consider the avalanche of Fig. 2, its status at column 8 (the column where the avalanche starts has index 1) is $(0, 1, 0, 1)$.

We claim that given a column i, the incomplete configuration $s \cap [\![i, |s|)$ and the status at i of the avalanche a for s, we can compute the avalanche on the part of s that we have, that is, $a \cap [\![i, |s|)$.

Note that in the proof of Theorem 1 we use only simple instances of Lemma 4, but we still present it in a general form.

Lemma 4. *Given*
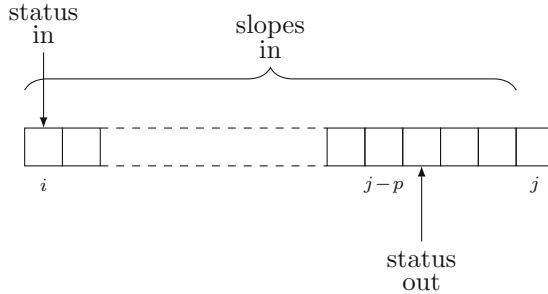
– *a part $s \cap [\![i, j)$ with $i + p < j$,*
– *the status at i of the avalanche a for s,*

one can compute

– *the avalanche on $a \cap [\![i, j - p]\!]$,*
– *the status of a at $j - p + 1$,*

in time $\mathcal{O}(j - i)$ on one processor.

Proof. We claim that given the status of the avalanche a at a column k, we can find the smallest (leftmost) peak after column k, let us denote it by $q = \min\{q \mid q \geq k \text{ and } q \text{ is a peak}\}$, and the part of a between k and q, i.e., $a \cap [\![k, q]\!]$. This will be done in constant time thanks to Lemma 3: $q - k < p$ so we have to check a constant number of columns. The result then follows an induction on the peaks within $[\![i, j)$: from the status at k (initialized for $k = i$), we find the next peak q and compute $a \cap [\![k, q]\!]$, append it to the previously computed $a \cap [\![i, k]\!]$, which also allows to construct the status at $q + 1$ in constant time. And this process is repeated at most a linear number of times:

– either the avalanche stops at some time,
– or the greatest peak encountered is between $j - p$ and $j - 1$,

and in both cases we can compute the intended objects by appending the previously computed parts of the avalanche (Lemma 2, recall that the status at $j - p + 1$ tells wether columns between $j - p + 1 - p$ and $j - p$ are fired or not).

Knowing the status of a at k, let us explain how to compute the smallest peak after column k, denoted q, and $a \cap [\![k, q]\!]$. Let $(b_0, \ldots, b_{p-1})$ be the status of a at k. From Lemma 3 the peak q has a value of slope equal to p in s and is at distance smaller than p from k. We will now prove that it is very easy to find q in constant time: q is the smallest column ℓ such that $0 \leq \ell - k < p$, and $s_\ell = p$ and $b_{\ell-k} = 1$.

– Such an ℓ is a peak: since $b_{\ell-k} = 1$, column $\ell - p$ is fired. When it is fired, it gives one unit of slope to column ℓ which can be fired since its slope is initially equal to p and becomes $p + 1$. It cannot be a col, which would mean that there is another peak q'', greater, which is fired before ℓ, but from Lemmas 2 and 3 this contradicts the minimality of the avalanche because when q'' is fired the column ℓ is also firable ($\ell - p$ has already been fired since it is at distance strictly greater than p of q'').
– The smallest peak greater or equal to k satisfies those three conditions: the two first conditions are straightforward from Lemma 3. The last condition can be proved by contradiction: suppose there is a peak q'' such that $b_{q''-k} = 0$, i.e., column $q'' - p$ is not fired in the avalanche, then q'' still needs to receive some units of slope to become unstable, which can only come from its left neighbor $q'' + 1$ thus this latter has to be fired before it, a contradiction.

As a consequence of the two above facts, the smallest such ℓ is indeed the intended peak q, and can be computed in constant time. There are $\mathcal{O}(j - i)$ peaks within $[\![i, j)$, and each step of the induction needs a constant computation time on one processor, thus the last part of the lemma holds.   □

Thanks to Lemma 4 we can perform the computation efficiently in parallel as follows.

Theorem 1. *For a fixed parameter p and in dimension one, $\boldsymbol{AP}$ is in NC^1.*

Proof. An input of $\mathbf{AP}$ is a configuration $s \in \mathbf{gSM}$ and a column $k \in (|s|, |s|+p]$. Let $k = |s| + \kappa$ with $\kappa \in (0, p]$.

The proof works in two stages: first, we compute for every position i the function that associates to each status at i, the corresponding status at $i + 1$, which we call "the function status at $i \to$ status at $i + 1$" (since status are elements of $\{0, 1\}^p$, the size of these functions is a constant). This can be done in constant time on $|s|$ parallel PRAM using Lemma 4; and in a second stage we compute in parallel the function status at $p + 1 \to$ status at $|s| + 1$ using $log|s|$ steps, by pairwise composing the functions as illustrated in Fig. 3.

One of the processors can then finish the job in constant time by first computing the status $\dot{b}$ at $p + 1$, which can easily be done in constant time using only $(s_1, \ldots, s_{p+1})$ since either s_1 is a peak or the avalanche stops, then either s_{p+1} is a peak or the avalanche stops, and finally the cols within s_1 and s_{p+1} are straightforwardly found thanks to Lemma 2. Then, it computes the status $\ddot{b} = \mu(\dot{b})$ at $|s| + 1$, and answers *yes* if and only if $\ddot{b}_{\kappa-1} = 1$, because columns on the right of $|s|$ cannot be fired but can only receive grains from their left neighbor at distance p, so does column $k = |s| + \kappa$.
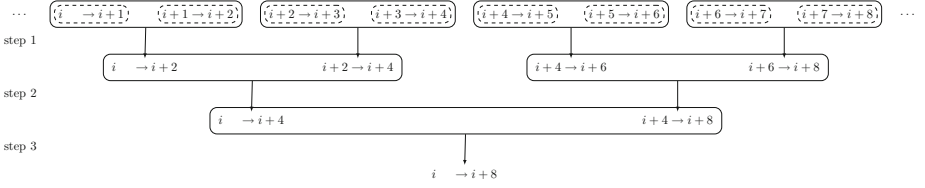
Fig. 3. Illustration of the parallel computation, each symbol $x \to y$ represents the function status at $x \to$ status at y. Dashed on the top are the functions computed during the first stage. Then, in $log|s|$ steps (each of them uses a polynomial number of parallel processors and a constant amount of time) we compose the circled functions in order to compute the function pointed out with an arrow. This composition is straightforwardly performed in constant time with the two processors: one of them transmits its function to the other one (a function of constant size is transmitted in constant time), and the latter composes two functions of constant size... in constant time. We perform those computations such that the resulting function μ has type: status at $p + 1 \to$ status at $|s| + 1$.

The complete procedure uses a logarithmic amount of time on a polynomial number of parallel processors (the input has size $\Theta(|s|)$), *i.e.*, the decision problem **AP** is in the complexity class NC^1. □

6   Conclusion and Open Problem

In this paper we proved that **AP** is in NC^1 in dimension 1 solving an open question of [2]. Going in the direction of [3], one might ask what is the complexity of **AP** when the constraint on monotonicity is relaxed. Clearly, by the results of [3], the problem is in P, but is it complete?

Another possible generalisation concerns symmetric sandpiles (see [4,5,10], for example). In this case, the lattice structure of the phase space is lost and therefore we cannot exploit it in the solving algorithms. This would probably direct the investigations towards non-deterministic computation and shift complexity results from P-completeness to NP-completeness.

It also remains to classify the computational complexity of avalanches in two dimensions when the parameter p equals 1 (that is, in the classical model introduced by Bak *et al.* [1]). As it is exposed in [6], this question interestingly emphasizes the links between NC, P-completeness, and information crossing.

References

1. Bak, P., Tang, C., Wiesenfeld, K.: Self-organized criticality: an explanation of the 1/f noise. Phys. Rev. Lett. **59**, 381–384 (1987)
2. Formenti, E., Goles, E., Martin, B.: Computational complexity of avalanches in the kadanoff sandpile model. Fundam. Inform. **115**(1), 107–124 (2012)
3. Formenti, E., Masson, B.: On computing fixed points for generalized sand piles. Int. J. on Unconventional Comput. **2**(1), 13–25 (2005)
4. Formenti, E., Masson, B., Pisokas, T.: Advances in symmetric sandpiles. Fundam. Inform. **76**(1–2), 91–112 (2007)
5. Formenti, E., Van Pham, T., Phan, H.D., Tran, T.H.: Fixed point forms of the parallel symmetric sandpile model. Theor. Comput. Sci. **322**(2), 383–407 (2014)
6. Gajardo, A., Goles, E.: Crossing information in two-dimensional sandpiles. Theor. Comput. Sci. **369**(1–3), 463–469 (2006)
7. Goles, E., Margenstern, M.: Sand pile as a universal computer. Int. J. Mod. Phys. C **7**(2), 113–122 (1996)
8. Goles, E., Morvan, M., Phan, H.D.: The structure of a linear chip firing game and related models. Theor. Comput. Sci. **270**(1–2), 827–841 (2002)
9. Laredo, J.L.J., Bouvry, P., Guinand, F., Dorronsoro, B., Fernandes, C.: The sand-pile scheduler. Cluster Comput. **17**(2), 191–204 (2014)
10. Perrot, K., Phan, H.D., Van Pham, T.: On the set of fixed points of the parallel symmetric sand pile model. Full Pap. AUTOMATA **2011**, 17–28 (2011)
11. Perrot, K., Rémila, E.: Avalanche structure in the kadanoff sand pile model. In: Dediu, A.-H., Inenaga, S., Martín-Vide, C. (eds.) LATA 2011. LNCS, vol. 6638, pp. 427–439. Springer, Heidelberg (2011)
12. Perrot, K., Rémila, E.: Transduction on kadanoff sand pile model avalanches, application to wave pattern emergence. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 508–519. Springer, Heidelberg (2011)