

The Computational Complexity of One-Dimensional Sandpiles

Peter Bro Miltersen¹

Dept. of Computer Science, University of Aarhus. bromille@brics.dk

Abstract. We prove that the one-dimensional sandpile prediction problem is in \mathbf{AC}^1 . The previously best known upper bound on the \mathbf{AC}^i -scale was \mathbf{AC}^2 . We also prove that it is not in $\mathbf{AC}^{1-\epsilon}$ for any constant $\epsilon > 0$.

1 Introduction

In physics, a *complex dynamical* system can be informally defined as a system operating on the “*edge of chaos*” (a phrase coined by Langton [7]). Thus, complexity is neither rigid order, with the future development of the system being easily and completely understood, nor chaos, with the system being virtually unpredictable and exhibiting “quasi-random” behavior. For the computer scientist, complex dynamical systems are interesting as they are exactly the physical systems capable of performing non-trivial computation.

In an attempt to put the above considerations on more rigorous ground, Machta, Moore and collaborators have, in a series of papers [5, 8, 9, 12, 11, 13], put forward a general program seeking to capture and measure the physicists’ informal notion of the “complexity” of a system by the rigorous notion of the *computational* complexity of the problem of predicting the behavior of the system. This approach gives us the opportunity of getting a fine-grained view of the complexity of a system by using the hierarchy of complexity classes of computational complexity theory. As a simple example, if the prediction problem for a system is complete for \mathbf{P} , we can say that the system is capable of efficiently emulating general sequential computation, while, if the prediction problem is in, say, \mathbf{NL} , the system can only perform computations obeying a severe space bound and thus not all computations, assuming $\mathbf{P} \neq \mathbf{NL}$.

An important model in complex systems theory is the Bak-Tang-Wiesenfeld *sandpile* model [1, 2]. The prediction problem for this model was considered from the computational complexity point of view by Moore and Nilsson [10]. The problem is defined for any dimension $d \geq 1$, but in this paper we concentrate on the one-dimensional case.

The one-dimensional sandpile prediction problem can be described as follows (the description of Moore and Nilsson is slightly different but can be easily seen to be equivalent to the following). A one-dimensional sandpile is a map $h : \mathbf{Z} \rightarrow \{0, 1, 2, \dots\}$. The value $h(i)$ is interpreted as the height of a pile of sand at position i . A pile of height 2 or more is *unstable*, and may *topple*, distributing sand evenly to the two neighboring positions. Formally, if map $h' : \mathbf{Z} \rightarrow \{0, 1, 2, \dots\}$

satisfies that for some i , $h'(i) = h(i) - 2$, $h'(i - 1) = h(i - 1) + 1$, $h'(i + 1) = h(i + 1) + 1$, and $h'(x) = h(x)$ for all $x \notin \{i - 1, i, i + 1\}$, we'll say that h' is a possible successor to h , or $h \rightarrow h'$. If there is no possible successor to h , we'll say that h is *stable*. If h is zero outside some interval I , say, $I = \{1, 2, \dots, n\}$, it can be shown that there is a unique stable g , so that $h \rightarrow^* g$. We will denote this unique g by h^* . The one-dimensional sandpile prediction problem is the following: Given $h : \{1, 2, \dots, n\} \rightarrow \{0, 1, 2\}$, find h^* . A similar definition can be made for the d -dimensional sandpile problem for any $d \geq 1$, but in this paper, we concentrate on the one-dimensional case.

Following their general program, Moore and Nilsson showed that the d -dimensional sandpile prediction problem is in \mathbf{P} for all d , that it is \mathbf{P} -complete for $d \geq 3$ and that the 1-dimensional problem is in $\mathbf{AC}^1[\mathbf{NL}] \subseteq \mathbf{AC}^2 \subseteq \mathbf{NC}^3$. They also present an efficient sequential algorithm for the 1-dimensional problem with a time bound of $O(n \log n)$.

The purpose of this paper is to present two pieces of information about the complexity of the one-dimensional sandpile prediction problem, narrowing down the possible range of its exact complexity considerably.

First we show the following improvement of the parallel upper bound of Moore and Nilsson.

Theorem 1. *The one-dimensional sandpile prediction problem is in $\mathbf{AC}^1 \subseteq \mathbf{NC}^2$.*

Conceptually, the algorithm is much simpler than the parallel algorithm of Moore and Nilsson. It is based on refining their *sequential* algorithm, and then noticing that the refined algorithm can be carried out by a deterministic poly-time logspace Turing machine with access to an auxiliary pushdown store. It is known that the class of languages so computed is $\mathbf{LOGDCFL}$ [15], the class of languages logspace reducible to a deterministic context-free language, and as $\mathbf{LOGDCFL} \subseteq \mathbf{AC}^1$ [14], the result follows.

Second, we prove the following hardness result.

Theorem 2. *The one-dimensional sandpile prediction problem is hard for \mathbf{TC}^0 with respect to constant depth reductions.*

That the one-dimensional sandpile prediction problem is hard for \mathbf{TC}^0 means that the one-dimensional sandpile is sufficiently complex to carry out at least some slightly non-rudimentary computation, such as computing the parity or majority of n bits. This provides formal justification for the statement of Moore and Nilsson that the dynamics of the one-dimensional sandpile problem is “non-trivial”. It also implies the following lower bound:

Corollary 1. *The one-dimensional sandpile prediction problem is not in $\mathbf{AC}^{1-\epsilon}$ for any constant $\epsilon > 0$.*

As \mathbf{AC}^i is the class of problems solvable by CRCW (concurrent-read, concurrent-write) PRAMs (parallel random access machines) with polynomially many processors in time $O((\log n)^i)$, we thus have fairly tight upper and lower bounds for solving the one-dimensional sandpile problem in this model of computation.

1.1 Organization of the paper

In Section 2, we sketch the proof of Theorem 1 and in Section 3, we sketch the proofs of Theorem 2 and corollary 3. We conclude with some open problems in Section 4.

2 The upper bound

In this section, we sketch the proof of Theorem 1.

In their paper, Moore and Nilsson show that the one-dimensional sandpile problem can be solved by the following sequential algorithm. Given an input sandpile $h : \{1, \dots, n\} \rightarrow \{0, 1, 2\}$, extend it to \mathbf{Z} by making zero the values of h outside the interval from 1 to n . We maintain two subsets T and N of the integers. Initially T is the set $\{i \in \mathbf{Z} | h(i) = 2\}$ and N is the set $\{i \in \mathbf{Z} | h(i) = 0\}$. Note that while N is an infinite set, it has an obvious finite representation. Now, while T is not empty repeat the following two steps, a *round*:

1. Pick an $t \in T$. Find $z_1, z_2 \in N$ so that $z_1 \leq t < z_2$.
2. Let $N = (N - \{z_1, z_2\}) \cup \{z_1 + z_2 - t\}$. Let $T = T - \{t\}$.

When T is empty, h' defined to be 0 on N and 1 on $\mathbf{Z} - N$ is the unique stable successor of h . We refer the reader to the argument for this in Moore and Nilsson.

Moore and Nilsson suggest implementing the above algorithm directly by maintaining the “finite part” of the set N in sorted order, and, for each $t \in T$, do a binary search in N to find z_1 and z_2 . Our first observation is that this binary search can be eliminated when going through the list T in increasing order. Indeed, we can maintain the following *invariant*. Between rounds, if $T = \{t_1 < t_2 < \dots < t_m\}$, we can maintain a partition of N into N_1 and N_2 in such a way that the following properties are true:

1. All elements in N_1 are smaller than each element in N_2 .
2. The elements z_1, z_2 so that $z_1 \leq t_1 < z_2$ are either
 - (a) The largest element of N_1 and the smallest element of N_2 or
 - (b) Both in N_2 .

First note that we can easily establish the invariant at the beginning of the computation. Now given the invariant, we want to perform one round of the algorithm of Moore and Nilsson. We should find an appropriate z_1, z_2 so that $z_1 \leq t_1 < z_2$. If case (a) applies, we remove t_1 from T (so t_2 will be the “new” t_1 in the next round), z_1 from N_1 and z_2 from N_2 . Note that $z_1 < z_1 + z_2 - t_1 \leq z_2$ and also note the t_2 is greater than the new largest element of N_1 (since t_1 was). Thus, if we insert $z_1 + z_2 - t_1$ as the new largest element of N_1 if $z_1 + z_2 - t_1 \leq t_2$ and we insert $z_1 + z_2 - t_1$ as the new smallest element of N_2 if $t_2 < z_1 + z_2 - t_1$, we have maintained the invariant. If case (b) applies, we move elements from N_2 to N_1 until case (a) applies, reducing it to this case. This completes the description of the algorithm.

Because of case (b), the reader may conclude that we have replaced binary search with linear search which does not sound like such a grand idea, but note that the size of N_2 is never increased during a round. Thus, the total cost of moving elements from N_2 during the entire execution of the algorithm is linear.

Concerning the complexity of the refined algorithm, viewed as a sequential algorithm we have to be somewhat careful about the exact model of computation. On a log cost RAM, the unrefined algorithm of Moore and Nilsson has complexity $O(n \log n)$ because of

1. The binary searches which cost $O(\log n)$ time each.
2. Adding and subtracting $O(\log n)$ -bit integers, each operation costing $O(\log n)$ time.

The refined algorithm also has complexity $O(n \log n)$ in the log-cost model. However, it is common practice to *only* use the log-cost time measure when integers of bit length much bigger than the log of the input are involved (e.g., as in case of the problem of multiplying two n -bit integers). When only $O(\log n)$ -bit length integers are involved, the *unit cost* RAM model is much more commonly used. In the unit cost RAM model, the refined algorithm has complexity $O(n)$ while the Moore-Nilsson version keeps having complexity $O(n \log n)$ because of the binary searches.

More important than its sequential complexity in various RAM models is the consequences of the refined algorithm for the parallel complexity of the sandpile prediction problem. Indeed, we next note that the refined algorithm can be implemented by a polynomial time, logspace Turing machine with access to an auxiliary pushdown store (i.e., an extra “free” tape, where a tape cell is erased when the head moves from the cell to its left neighbor). Because of the robustness of logarithmic space, we just have to argue that the algorithm can be implemented by a while-program using $O(1)$ variables, each holding an integer of $O(\log n)$ bits and an auxiliary object STACK where such $O(\log n)$ bit integers can be pushed and popped. We show how each variables in the refined algorithms can be represented using such objects.

N_1 will be represented by the STACK object and a single integer variable v . The invariants of the representation are the following:

1. N_1 is exactly $\{\dots, -v - 2, -v - 1, -v\} \cup \{\text{the elements in STACK}\}$.
2. The elements of STACK are sorted, with the largest at the top.

With this representation, we can remove the largest element from N_1 , and add an element to N_1 larger than the largest one. These are the only operations we need to perform on N_1 when running the refined algorithm. We can also easily initialize the representation to the correct content in the beginning of the refined algorithm.

N_2 will be represented by three integer variables l , i and w . The invariant of the representation is that $N_2 = \{l\} \cup \{j \geq i \mid H[i] = 0\} \cup \{j \in \mathbf{Z} \mid j \geq w\}$. Here $H[1..n]$ is the structure holding in the input, representing the map $h : \{1, \dots, n\} \rightarrow \{0, 1, 2\}$. With this representation, we can replace the smallest

element of N_2 by another element and remove the smallest element from N_2 . These are the only two operations the refined algorithm uses.

T is represented in a way similar to N_2 .

We have now shown that the refined algorithm for one-dimensional sandpile prediction can be implemented by a deterministic, polynomial time, logspace Turing machine with access to an auxiliary pushdown store. At the end of the algorithm, the output is contained in the representation of N_1 and N_2 . If we are interested in the i 'th bit of the output for some i , we can find it by either inspecting the structure for N_2 or the structure for N_1 , in the last case popping a sufficient number of elements from the stack. Thus the language $1\text{SANDPILE} = \{\langle h, i \rangle \mid \text{the } i\text{'th bit in } h^* \text{ is } 1\}$ can be decided by a deterministic logspace Turing machine with an auxiliary pushdown store. By a result of Sudborough [15], this means that the language is in **LOGDCFL** which, by a result of Ruzzo [14] is a subset of **AC**¹. The functional version of the problem, i.e. the map $h \rightarrow h^*$ itself is therefore computable in the same class.

3 The lower bound

In this section, we sketch the proofs of Theorem 2 and Corollary 3.

Moore and Nilsson showed **P**-completeness of higher-dimensional versions of the sandpile prediction problem; here we show hardness for much lower complexity classes of the one-dimensional problem. When considering **P**-completeness, logspace reductions are most often used. However, these are not meaningful for classes below **L**, as those classes are not closed under logspace reductions. Here, we use a weaker notion of reductions, DLOGTIME-uniform constant depth reductions [4, 3]. All classes considered here are closed under those reductions. A language π_1 reduces to an language π_2 by such reductions if we can build DLOGTIME-uniform, constant depth, polynomial size circuit for π_1 , using unbounded fan-in AND-gates, unbounded fan-in OR-gates, NEGATION-gates, and *oracle*-gates computing π_2 .

Let MAJORITY be the problem of deciding whether a string of n input bits (n odd) have more 1's than zeros. We shall show that MAJORITY reduces to 1SANDPILE. Since MAJORITY is complete for **TC**⁰ by constant depth reductions (indeed, the *definition* of **TC**⁰ is that it is the closure of MAJORITY under constant depth reductions), it follows that 1SANDPILE is hard for **TC**⁰ under constant depth reductions.

Let PARITY be the problem of deciding whether a string of n input bits have an odd number of ones. As PARITY is in **TC**⁰, PARITY is not in **AC**^{1- ϵ} for any $\epsilon > 0$ [6], and **AC**^{1- ϵ} is closed under constant depth reductions, we get the corollary that 1SANDPILE is not in **AC**^{1- ϵ} for any constant $\epsilon > 0$.

It remains to show that MAJORITY constant depth reduces to 1SANDPILE. We have to construct a uniform circuit for MAJORITY using unbounded fan-in AND gates, unbounded fan-in OR gates and 1SANDPILE oracle-gates.

Given an input $x_1 x_2 \dots x_n$ of the MAJORITY problem, we can assume that n is odd. Our circuit first constructs an instance $1y_{11}y_{12}y_{13}y_{21}y_{22}y_{23} \dots y_{n1}y_{n2}y_{n3}$

of the sandpile problem where $y_{i1} = x_i$, $y_{i2} = 1 - x_i$ and $y_{i3} = 2$, except for $y_{n3} = 1$. For example, we reduce 0101011 to 1 012 102 012 012 012 101. Applying the Moore-Nilsson algorithm, we see that this instance reduces to a stable value with exactly one zero with an index between 1 and $3n$, this index being $n + 1 + \#ones$ in $x_1x_2 \dots x_n$. Thus, to find the majority of $x_1x_2 \dots x_n$, we just need to check if the index of the unique one in the reduced pile is bigger than $\frac{3n}{2} + 1$. This is easily checked with a uniform constant depth circuit.

4 Discussion and open problems

In general, to carry out the program of making formal the informal notion of the “complexity” of a complex dynamical system by identifying “complexity” with the computational complexity of the prediction problem, it seems appropriate to use the finest scale available in the theory of computational complexity.

We have shown a \mathbf{TC}^0 lower bound and a $\mathbf{LOGDCFL}$ upper bound for the one-dimensional sandpile prediction problem. Obviously, getting tighter bounds would be desirable. In particular, is the one-dimensional problem hard for \mathbf{NC}^1 ? Is it in \mathbf{L} or \mathbf{NL} ?

Moore and Nilsson classified the d -dimensional sandpile prediction problem as \mathbf{P} -complete for $d \geq 3$ and left open whether the 2-dimensional sandpile prediction problem is also hard for \mathbf{P} . We may note that the reduction used by Moore and Nilsson to show that the 3-dimensional problem is hard for \mathbf{P} also establishes that the 2-dimensional problem is hard for \mathbf{NC}^1 : Their reduction is a reduction from the monotone circuit value problem and the third dimension is only used when implementing crossovers of wires. Thus, the monotone *planar* circuit value problem reduces to 2-dimensional sandpile prediction, and this problem is hard for \mathbf{NC}^1 . Getting a better lower bound than \mathbf{NC}^1 or a better upper bound than \mathbf{P} for the 2-dimensional sandpile prediction problem would be most interesting.

Acknowledgements

Peter Bro Milteren is supported by BRICS, Basic Research in Computer Science, a centre of the Danish National Research Foundation and by a grant from the Danish Research Council. He would like to thank Cris Moore for helpful discussions.

References

1. P. Bak, C. Tang and K. Wiesenfeld. Self-organized criticality: an explanation of $1/f$ noise. *Physical Review Letters* **59** (1987) 381–384.
2. P. Bak, C. Tang and K. Wiesenfeld. Self-organized criticality. *Physical Review A* **38** (1988) 364–374.
3. D. A. M. Barrington, N. Immerman and H. Straubing. On uniformity within \mathbf{NC}^1 . *Journal of Computer and System Sciences*, 41(3):274–306.

4. A.K. Chandra, L. Stockmeyer and U. Vishkin. Constant depth reducibility. *SIAM Journal on Computing* **13** (1984) 423–439.
5. D. Griffeath and C. Moore. Life without death is P-complete. *Complex Systems* **4** (1990) 299–318.
6. J. Håstad. *Computational Limitations of Small-Depth Circuits*. ACM doctoral dissertation award 1986. MIT Press, 1987.
7. C. G. Langton. Computation at the edge of chaos. *Physica D* **42** (1990).
8. J. Machta. The computational complexity of pattern formation. *Journal of Statistical Physics* **77** (1994) 755.
9. J. Machta and R. Greenlaw. The computational complexity of generating random fractals. *Journal of Statistical Physics* **82** (1996) 1299
10. C. Moore and M. Nilsson. The Computational Complexity of Sand-piles. *Journal of Statistical Physics* **96** (1999) 205–224. Also available on <http://www.santafe.edu/~moore/>.
11. C. Moore and M.G. Nordahl. Predicting lattice gasses is P-complete. *Santa Fe working Paper* 97-04-034.
12. C. Moore. Majority-vote cellular automata, Ising dynamics, and P-completeness. *Journal of Statistical Physics* **88** (1997) 795–805.
13. K. Moriarty and J. Machta. The computational complexity of the Lorentz lattices gas. *Journal of Statistical Physics* **87** (1997) 1245.
14. W. Ruzzo. Tree-size bounded alternation. *Journal of Computer and Systems Sciences* 21:218-235, 1980.
15. I. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the Association for Computing Machinery*, 25:405-414, 1978.